

Understanding Images via Visual Similarity and Deep Feature Representations

DISSERTATION

to obtain the Degree of Doctor rerum naturalium

submitted by

M.Sc. Mai Lan HA

submitted to the School of Science and Technology
of the University of Siegen
Siegen 2020

gedruckt auf alterungsbeständigem holz- und säurefreiem Papier

Supervisor and first appraiser
Prof. Dr. rer. nat. Volker Blanz
University of Siegen

Second appraiser
Prof. Dr. Terence SIM Mong Cheng
National University of Singapore

Date of the oral examination
18. August 2020

Declaration of Authorship

I, **MSc. Mai Lan HA**, declare that this thesis titled, “Understanding Images via Visual Similarity and Deep Feature Representations” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Zusammenfassungen

Maschinelles Lernen und Sehen werden oft so verstanden, als beträfen sie nur Maschinen und die Entwicklung von Algorithmen, welche einem Computer verschiedenste Aufgaben ermöglichen. Jedoch haben auch das menschliche Sehen und die menschliche Wahrnehmung einen Einfluss auf die Zielvorgaben, wie solche Algorithmen funktionieren sollen und wie ein Computer "sehen" soll. Die zwei Ziele dieser Dissertation sind daher die Untersuchung von visueller Ähnlichkeit (visual similarity) nach Gesichtspunkten menschlicher Wahrnehmung und die Untersuchung von Feature-Darstellungen in tiefen Faltungnetzwerken (DCNNs).

Die Bewertung von visueller Ähnlichkeit zwischen Bildern ist zwar eine Kernaufgabe der menschlichen Wahrnehmung, aber eine große Herausforderung für Maschinelles Sehen in echten Anwendungen, aufgrund der Subjektivität und Mehrdeutigkeit der Problemdefinition. Daher ist das erste Ziel dieser Arbeit eine grundlegende Studie von visueller Ähnlichkeit. Wir untersuchen eine Abgrenzung verschiedener Aspekte von Ähnlichkeit, die eine handhabbare und realisierbare Untersuchung ermöglicht. Wir diskutieren Ähnlichkeit auf Basis von Farbkomposition im Detail, anfangend bei menschlicher Evaluierung, bis hin zur Modellierung mithilfe von DCNNs. Wir benutzen diese Modelle zur Schaffung sowohl einer neuer Metrik für global Farbähnlichkeit als auch eines Farbtransfermodells. Weiterhin verknüpfen wir Farbkomposition und Objektähnlichkeit, um ein neues Modell für visuelle Ähnlichkeit zu definieren. Diese Kombination führt in der Anwendung, etwa im Bereich der Bildsuche auf einer feineren Unterscheidungsebene (fine-grained image retrieval) zu verbesserten Ergebnissen. Unser Ansatz ist ein Prototyp, der zeigt, wie subjektive Wahrnehmung, für Maschinen greifbar gemacht werden kann. Schliesslich entwickeln wir auch ein wahrnehmungs-inspirierte Metrik zur Evaluierung von intrinsischen bildgebenden Verfahren, die im Gegensatz zu bisherigen Metriken, genauere Methodikvergleiche ermöglicht.

Das zweite Ziel dieser Dissertation bildet die Untersuchung von Merkmalsrepräsentationen in verschiedenen Teilbereichen eines DCNNs, und stellt die Frage, wie diese Merkmale verstanden, effizient benutzt und verändert werden können. Merkmale aus den unteren und mittleren Schichten eines DCNNs, angefangen bei den Bildpixeln hin zu den Faltungsergebnissen der frühen Schichten, können gut als Metriken für Wahrnehmung und visuelle Ähnlichkeit verwendet werden. Bei Analyse der späteren Schichten eines DCNNs für Klassifizierung finden wir jedoch heraus, dass in diesen Gestaltsinformationen "versteckt" sind. Die Extraktion dieser Informationen führt uns zu einer schwach beaufsichtigten (weakly-supervised) Segmentierungsmethodik, die sogar jenseits der Klassen funktioniert, mit denen das DCNN trainiert wurde. Außerdem betrachten wir die diskriminativen Fähigkeiten

dieser späteren Klassifikationsmerkmale und diskutieren eine Verbesserung der Separation durch Verwendung von Methoden der linearen Diskriminationsanalyse während des Trainings. Unsere vorgeschlagene Optimierungsmethode führt zu verbesserten Klassifikationsergebnissen, besonders bei der Klassifikation auf einer feineren Unterscheidungsebene, die sogar für menschliche Experten schwierig ist.

Diese Untersuchungen von visueller Ähnlichkeit und von tiefen Merkmalsrepräsentationen in DCNNs zeigen neue Wege zu einer Theorie von Bildverständnis auf, die verschiedenste Aspekte von Bildern, wie Farben, Gestalt und Kategorien einschließt.

Abstract

Machine Learning and Computer Vision are often thought to relate only to machines, involving the development of algorithms and teaching computers to perform various tasks. However, human vision and perception are hidden aspects that influence how an algorithm should function, or how we would want a computer to "see". The two goals of this thesis are the study of perceptual visual similarity and that of feature representations from Deep Convolutional Neural Networks (DCNNs).

Assessing visual similarity in-the-wild, a core ability of the human visual system is a challenging problem for Computer Vision because of its subjective nature and its ambiguity in the problem definition. Therefore, the first goal of the thesis is to study the fundamental problems of visual similarity. We raise the question if we could break down different aspects of similarity that make their study more tractable and computationally feasible. We study color composition similarity in-depth, from human evaluation to its modeling using DCNNs. We apply the models to create a new global color similarity descriptor and color transfer method. We then couple color composition and category similarities to define a new model for visual similarity. The combination leads to better results in fine-grained image retrieval. Our approach is a proof of concept, showing that we can make subjective phenomena scientifically tractable. We also developed a perceptual inspired metric to evaluate intrinsic imaging methods resulting in a fairer evaluation compared to previous metrics.

The second goal of the thesis focuses on investigating what features are embedded in different parts of a DCNN, how we could use them efficiently, and how we can improve these features. On the one hand, the low to mid-level features, ranging from image pixels to different layers of convolutional responses in a DCNN, are used in perceptual metrics and visual similarity. On the other hand, we discover shape information "hidden" in the high-level features of a DCNN trained for classification. The shapes extracted from the DCNN are used to perform weakly supervised semantic segmentation that works well beyond the classes on which the DCNN was trained. We also find a way to improve the discriminative ability of deep classification features by incorporating Linear Discriminant Analysis objectives into a DCNN training optimization. Our proposed optimization method leads to better classification results, especially for fine-grained classification, which is challenging even for non-expert humans.

The studies on perceptual visual similarity and deep feature representations in the thesis shed new light on image understanding, which covers different aspects of images such as color, shape, and category.

Acknowledgements

I would like to express my deepest gratitude to my thesis advisor, Prof. Dr. Volker Blanz, for his guidance, support, helpful critiques, and insightful discussions throughout my Ph.D. study. I am sincerely grateful for Prof. Dr. Terence Sim for being supportive and creating opportunities for me to pursue my research. I would like to acknowledge Prof. Dr. Andreas Kolb, Prof. Dr. Michael Möller, and Prof. Dr. Thomas Vetter for their advice and feedback.

I would also like to extend my thanks to my colleagues, collaborators, and friends for their help, collaboration, and encouragement, in particular: Dr. Vlad Hosu, Dr. Gianni Franchi, Dr. Shida Beigpour, Roberto Cespi, Dr. Nadine Hoffmann, Dr. Davoud Shahlaei, Willi Gräfrath and Jonas Geiping.

Finally, I wish to thank my parents for their love, patience, support, and encouragement throughout my study and beyond.

In memory of my loving Nicky

Contents

Declaration of Authorship	v
Zusammenfassungen	vii
Abstract	ix
Acknowledgements	xi
Contents	xv
List of Figures	xix
List of Tables	xxiii
1 Introduction	1
2 Convolutional Neural Networks	9
2.1 Artificial Neural Network (ANN)	9
2.1.1 McCulloch-Pitts neuron (M-P neuron)	11
2.1.2 Perceptron	12
2.1.3 Neural Networks	13
Backpropogation	14
Activation functions and nonlinearity	15
2.2 Convolutional Neural Networks (CNN)	17
2.2.1 Fundamentals of CNN	17
Convolution	17
Bias and Nonlinearity	18
Pooling	20
Convolution and Neural Network	20
Training	23
2.2.2 CNN Architectures	23
I Perceived Similarity	27
3 Pixel-Based Perception-Inspired Metric for Intrinsic Imaging	33
3.1 Intrinsic Imaging	33
3.1.1 Intrinsic Imaging Formulation	33

3.1.2	Multi-View Multi-Illuminant Intrinsic Dataset	34
3.2	Existing Evaluation Metrics	35
3.3	Point-wise Consistency Metric (PCM)	36
3.3.1	Overview of the Algorithm	37
3.3.2	Point-wise Consistency Error	37
3.3.3	Point Selection Strategy	38
3.3.4	Sampling And Results	40
3.3.5	Evaluation	41
3.3.6	Chapter Summary	43
4	Perceptual Color Composition Similarity	45
4.1	Related Work For Perceptual Color Similarity	48
4.1.1	Hand-crafted Features for Color Similarity	48
4.1.2	Learned Features for Visual Similarity	48
4.1.3	Datasets for Perceptual Similarity	49
4.2	Process to Define Perceptual Color Composition Similarity	49
4.2.1	Binary Dataset and Network	54
4.2.2	Rating Dataset	56
4.2.3	Study on Rating Strategies	57
4.2.4	Quality of the Rating Dataset	60
4.3	Computational Model of Color Composition Similarity	62
4.4	Application 1: Global Color Descriptor	66
4.5	Application 2: Fine-Grained Image Retrieval	68
4.5.1	Related Work	68
4.5.2	Content versus Color Retrieval	69
4.5.3	Features and Training Model for Fine-Grained Retrieval	71
4.5.4	Results Analysis and Discussion	73
4.5.5	Analogy Image Retrieval - An Inspiration	75
4.6	Application 3: Neural Style Transfer with Perceptual Color Similarity	78
4.6.1	Related Work	78
4.6.2	Perceptual Color Transfer	81
4.6.3	Combining Style Transfer and Perceptual Color Transfer	81
4.6.4	Results and Analysis	81
4.6.5	More Style-Color Transfer Results	84
4.6.6	More color transfer results	87
4.7	Chapter Summary	88
II	Deep Features	91
5	Shape Extraction and Semantic Segmentation	95
5.1	Image Feature Representation	95
5.1.1	Deep Feature Extraction	96

5.1.2	Retrieval with Spatial Constraint	96
5.1.3	Results	98
5.2	Shape Extraction And Weakly Supervised Segmentation	100
5.2.1	Related Work	102
	Understanding DCNNs and Class Activation Maps	102
	Weakly Supervised Object Segmentation	103
	Class Activation Map (CAM)	103
5.2.2	High Resolution Class Activation Map (rCAM) and Shape Extraction .	105
	Multi-scale CAMs Extraction	105
	Shape Extraction from GoogLeNet-GAP	107
	Upsampling Using Guided Filters	110
	Fusion and Refinement	111
5.2.3	Evaluations	112
	Evaluation Datasets	112
	Evaluation Metrics	113
	Numerical results for various CAM methods	113
	Weakly Supervised Segmentation Comparison	115
5.2.4	Chapter Summary	119
6	Neural Discriminant Analysis	121
6.1	Introduction To Discriminant Analysis For Fine-Grained Visual Classification	121
6.2	Related Work	124
6.2.1	Fine-Grained Visual Classification (FGVC)	124
6.2.2	Linear Discriminant Analysis (LDA)	125
6.3	Two-phase Neural Discriminant Analysis (NDA)	126
6.3.1	Pre-optimized Features Extracted from Pre-trained DCNNs	126
6.3.2	Feature Discriminant Analysis	126
6.3.3	Discriminant Analysis Optimization with Neural Networks	127
6.4	Experiments with Two-phase NDA	130
6.4.1	Datasets and Feature Extraction	130
6.4.2	Implementation	131
6.4.3	Results	132
6.4.4	Discussion	135
6.5	Combined Optimization NDA	137
6.6	Chapter Summary	139
7	Conclusion	141
	Publications	145
	External Bibliography	147

List of Figures

1.1	Thesis Overview	6
2.1	Artificial Neural Network: Biological Neuron Model	10
2.2	Artificial Neural Network: McCulloch-Pitts Neuron Model	11
2.3	Artificial Neural Network: Perceptron Model	12
2.4	Artificial Neural Network: Data With XOR Function	13
2.5	Artificial Neural Network: Examples	14
2.6	Artificial Neural Network: Single Neuron Illustration	15
2.7	Artificial Neural Network: activation functions	16
2.8	Convolutional Neural Network: 2D Convolution Examples	18
2.9	Convolutional Neural Network: AlexNet Convolution Kernels	19
2.10	Convolutional Neural Network: Linear Regression	19
2.11	Convolutional Neural Network: Max Pooling	20
2.12	Convolutional Neural Network: AlexNet	21
2.13	Convolutional Neural Network: 2D Global Max Pooling	21
2.14	Convolutional Neural Network: Convolutional Feature Visualization	22
2.15	Convolutional Neural Network: VGG16 Architecture	23
2.16	Convolutional Neural Network: Residual Block	24
2.17	Convolutional Neural Network: 1x1 Convolution	25
2.18	Convolutional Neural Network: Inception Module	25
2.19	Perception Similarity: Checker Shadow Illusion	29
2.20	Perception Similarity: Cornsweet Illusion	30
3.1	Intrinsic Imaging: An Example	34
3.2	Intrinsic Imaging: Multi-View Multi-Illuminant Dataset	35
3.3	Point-wise Consistency Error Illustration	36
3.4	Point-wise Consistent Metric (PCM): Pairs Of Points Selection	38
3.5	Masks Illustration	39
3.6	Point-wise Consistent Metric (PCM): Error Analysis	41
3.7	Point-wise Consistent Metric (PCM): PCM Versus LMSE Results	42
3.8	Point-wise Consistent Metric (PCM): Reflectance Results From Existing Methods	42
4.1	Perceptual Color Composition Similarity: Different Aspects of Visual Similarity	45
4.2	Perceptual Color Composition Similarity: Different Ratings Examples	46

4.3	Perceptual Color Composition Similarity: Binary Dataset	50
4.4	Perceptual Color Composition Similarity: Instruction on Fined Rating	51
4.5	Perceptual Color Composition Similarity: Contributor Funnel	51
4.6	Perceptual Color Composition Similarity: Example of 5 Different Similarity Scores	52
4.7	Perceptual Color Composition Similarity: Test Questions Setting	53
4.8	Perceptual Color Composition Similarity: Active Learning Framework	54
4.9	Perceptual Color Composition Similarity: Examples of Similar Images	54
4.10	Perceptual Color Composition Similarity: Similar vs Dissimilar Images	55
4.11	Perceptual Color Composition Similarity: Crowdsourcing for Similar vs Dissimilar	56
4.12	Perceptual Color Composition Similarity: Binary Network Results	56
4.13	Perceptual Color Composition Similarity: Color Similarity Group Rating	57
4.14	Perceptual Color Composition Similarity: Examples of Triplet Comparison	58
4.15	Perceptual Color Composition Similarity: Rating Strategies: Pairs vs. Triplet	58
4.16	Perceptual Color Composition Similarity: Example of Different Rating Distribution	59
4.17	Perceptual Color Composition Similarity: PLCC Correlation of MOS	61
4.18	Perceptual Color Composition Similarity: Histogram of MOS for Color Similarity Dataset	61
4.19	Perceptual Color Composition Similarity: Siamese Architecture	62
4.20	Perceptual Color Composition Similarity: t-SNE MOS Embedding 3K Images	64
4.21	Perceptual Color Composition Similarity: t-SNE MOS Embedding 5K Images	65
4.22	Perceptual Color Composition Similarity: t-SNE Embedding: Color Features vs Category Content Features	65
4.23	Perceptual Color Composition Similarity: Mean Absolute Error	68
4.24	Fine-Grained Image Retrieval: Color versus Content Retrieval	70
4.25	Fine-Grained Image Retrieval: Color and Content Retrieval Example 1	71
4.26	Fine-Grained Image Retrieval: Color and Content Retrieval Example 2	71
4.27	Analogy Image Retrieval: Results from Google Keyword Search	75
4.28	Analogy Image Retrieval: Results from Pinterest Example Based Search	76
4.29	Analogy Image Retrieval: Color versus Category	77
4.30	Analogy Image Retrieval: Appearance Versus Category	77
4.31	Perceptual Neural Transfer: Style Versus Color Transfer	78
4.32	Perceptual Neural Transfer: Gram Matrix in Texture Synthesis	79
4.33	Perceptual Neural Transfer: Results of the Texture Synthesis using the Gram Matrix	79
4.34	Perceptual Neural Transfer: A Style Transfer Model	80
4.35	Perceptual Neural Transfer: Style and Color Transfer Result.	82
4.36	Perceptual Neural Transfer: Color Transfer Result.	83
4.37	Perceptual Neural Transfer: Style and Color Transfer Result Extra 1.	84
4.38	Perceptual Neural Transfer: Style and Color Transfer Result Extra 2.	85

4.39	Perceptual Neural Transfer: Style and Color Transfer Result Extra 3.	86
4.40	Perceptual Neural Transfer: Color Transfer Result Extra 1.	87
4.41	Perceptual Neural Transfer: Color Transfer Result Extra 2.	88
5.1	Image Feature Representation: Multiple Input Image Retrieval Example . . .	95
5.2	Image Feature Representation: Deep Feature Extraction Process	96
5.3	Image Feature Representation: Image Retrieval From Region Proposals	97
5.4	Image Feature Representation: Spatial Constraint	97
5.5	Image Feature Representation: Retrieval Results with a Spatial Constraint on Texture Images	98
5.6	Image Feature Representation: Retrieval Results with Spatial Constraint . . .	99
5.7	Shape Extraction And Semantic Segmentation: Fully Versus Weakly Super- vised Learning	100
5.8	Shape Extraction And Semantic Segmentation: CAM Results	101
5.9	Shape Extraction And Semantic Segmentation: Normal CAM Process	103
5.10	Shape Extraction And Semantic Segmentation: CAM For Different Classes . .	104
5.11	Shape Extraction And Semantic Segmentation: CAM For Action Classification	105
5.12	Shape Extraction And Semantic Segmentation: High Resolution CAM Extrac- tion Process	106
5.13	Shape Extraction And Semantic Segmentation: PCA	108
5.14	Shape Extraction And Semantic Segmentation: Shape Dataset Examples . . .	109
5.15	Shape Extraction And Semantic Segmentation: Shape With Different Colors And Textures	109
5.16	Shape Extraction And Semantic Segmentation: CAM Multiple Objects Results	113
5.17	Shape Extraction And Semantic Segmentation: Weakly Supervised Segmen- tation Results 1	117
5.18	Shape Extraction And Semantic Segmentation: Weakly Supervised Segmen- tation Results 2	118
6.1	Neural Discriminant Analysis: Inter- And Intra-Variance	122
6.2	Neural Discriminant Analysis: Before And After NDA Optimization	128
6.3	Neural Discriminant Analysis: Two-Phase Optimization Overview	129
6.4	Neural Discriminant Analysis: Convergence	132
6.5	Neural Discriminant Analysis: Standard Deviations Comparison	136
6.6	Neural Discriminant Analysis: NDA Combined Loss	137

List of Tables

3.1	Point-wise Consistent Metric (PCM) Versus LMSE: Evaluation Results	42
4.1	Perceptual Color Composition Similarity: Pairwise vs. Triplet Rating Performances	60
4.2	Perceptual Color Composition Similarity: Global Color Descriptors Evaluation	66
4.3	Fine-Grained Image Retrieval: The State of the Art Comparison	72
4.4	Fine-Grained Image Retrieval: Color Descriptors Comparison	72
5.1	Shape Extraction And Semantic Segmentation: CAM Results On Multiple Datasets	114
5.2	Shape Extraction And Semantic Segmentation: Weakly Supervised Segmentation Results	115
6.1	Neural Discriminant Analysis: Results And Comparison	133
6.2	Neural Discriminant Analysis: NDA versus Siamese	134
6.3	Neural Discriminant Analysis: Training Stability Results	135
6.4	Neural Discriminant Analysis: NDA versus CDA	138

Chapter 1

Introduction

“Research is formalized curiosity. It is poking and prying with a purpose.”

Zora Neale Hurston

Computer Vision is a scientific research field with a quest for computers to understand visual information such as images and videos, to form decisions and perform tasks as humans do. Image understanding is a process of analyzing and transforming data from the form of image signals to symbolic information that helps to interpret images. Image understanding covers a wide range of topics from objective studies such as object recognition, tracking, segmentation to subjective studies such as image aesthetics, saliency, visual similarity, to name a few. Algorithms and techniques for processing images span a broad spectrum of information levels, from low-level signal processing in early vision to high-level symbolic features for complex cognitive tasks. This thesis aims to study visual similarity and deep feature representations provided by Deep Convolutional Neural Networks (DCNNs), and apply them in different Computer Vision tasks.

The thesis comprises two main parts. The first part includes work focused on perceptual similarities such as perception-inspired evaluation metric for intrinsic imaging methods, and a complete framework and study on color composition similarity. The second part is the study of high-level features for shape extraction, semantic segmentation, and improving discriminative features for classification.

Pixel-Based Perception-Inspired Metric for Intrinsic Imaging (Part I - Chapter 3)

Intrinsic imaging is a process to decompose a color image into intrinsic reflectance, shading, and specular components. The reflectance is an intrinsic surface property, the spectral distribution of diffuse reflection of the surface. It does not change when lighting conditions change. Unlike reflectance, the shading component changes due to the angular distribution of incoming light, self-shadowing, and cast shadows. Similar to shading, specular reflection is created by strong lights' reflections at specific places on the objects' surfaces; however, it

is an effect that occurs on the metallic or non-metallic surface and not on scattering particles inside the material [Gro+09]. The reflectance component of an image is important for a realistic scene reconstruction in Computer Graphics. Therefore, many intrinsic imaging methods aim to create reliable reflectance results. To evaluate the quality of the reflectance results from intrinsic imaging methods, Mean Squared Error (MSE) or Local Mean Squared Error (LMSE) are often used. While MSE penalizes heavily for large errors from outliers, LMSE cannot guarantee the global consistency of the evaluation [Gro+09]. Bell *et al.* [BBS14] introduced a weighted human disagreement rate (WHDR) metric, but the metric seems to be less discriminative among different methods' results. Inspired by human perception where distinctions between colors are only observable above some certain thresholds that are studied in the Just Noticeable Difference (JND), we develop a perception-inspired metric called Point-wise Consistency Metric (PCM) that measures the reflectance consistency with respect to the ground-truth throughout different illuminations using perceptual color distance CIE DE2000 [LCR01; SWD05]. PCM is designed based on the main principles that if two points are perceptually similar in the ground-truth reflectance, they should be similar in the estimated reflectance, and the brightness differences between those two points should also be similar in the ground-truth and the estimated reflectance. Our proposed metric reflects the perceptual characteristics of human vision, but at the same time, avoids the visual illusion caused by strong shadows and specularities [OCS05]. The metric is presented in Chapter 3. We compare the performance of our metric against LMSE on three state-of-the-art intrinsic imaging methods. It shows that our metric evaluation is more stable across different illumination conditions and gives fairer judgments than LMSE. This work is described in Chapter 3 and published at the BMVC conference in 2016 [Bei+16].¹

Perceptual Color Composition Similarity (Part I - Chapter 4)

A large portion of part one of the thesis contains significant work on visual similarity, which highlights color composition similarity in Chapter 4. Visual similarity is a long-standing Computer Vision research subject that is highly related to human vision. It is a challenging field of study due to its subjective measure. There is no clear and concise definition of what it means for two images to be considered visually similar. Two images can be similar, for example, if they contain the same object, express the same photography style, or have the same color layout. Our goal is to break visual similarity down to different aspects for which the study is more well-defined.

While category and image type similarities are easy to handle by classification-based machine learning model, the global color similarity is more complicated, especially for images in-the-wild. We define color composition similarity between pairs of images based on human judgments on the similarities in hues, shades, their distributions, overall layouts, foreground colors versus background colors, and independent of the category. It goes beyond pixel-based approaches, to which the CIE DE2000 metric [LCR01; SWD05] can be applied, or patch-based approaches. Due to the complexity of natural images, hand-crafted color

¹References in red are my publications, references in blue are external bibliography.

descriptors [AF06; BZM08; BG09; Geu+01; Kha+13; LS13; Man+01; SGS10; WGB06] cannot predict color composition similarity well. Therefore, we are motivated to solve the color composition similarity problem that will lead to a better model for visual similarity.

We develop an active learning approach for collecting meaningful pairs of images to evaluate color composition similarity. It faces a cold start problem because there are many more pairs of images that are visually different than similar if we were to sample images randomly. We start with a small manual selection of similar pairs of images, use them for training a model to select more similar images for human evaluation, and repeat the process. At the final step, we ask participants to rate the color similarity using a 5-point Likert-type scale. We carefully design several crowd-sourced experiments, analyze their results, and choose the best format of the experiments that lead to a high quality and consistently rated dataset. As a result, we build a large-scale, high quality, and the first of its kind dataset for color composition similarity.

The color composition similarity dataset provides the groundwork for us to train color similarity metrics and descriptors, and learn global color similarity features. Our descriptor outperforms existing hand-crafted color descriptors by a large margin when applied to image retrieval based on visual color similarity. We also set a benchmark for evaluating global color similarity by providing a testing dataset, which is a part of our color composition dataset, and the similarity ranking measure using Spearman Rank Order Correlation Coefficient (SROCC) between the predicted results and ground-truth ratings.

In fine-grained image retrieval, existing techniques use general visual similarity features [BB15; Che+10; PM15; Wan+14; WKH17]. We propose to combine color similarity with category similarity to form novel features for fine-grained image retrieval. Our model shows superior results compared to the best of the existing methods, DeepRanking [Wan+14], with three orders of magnitude less training data.

We also apply our global color composition similarity features to color transfer. The work is based on neural style transfer [GEB16]. Gatys *et al.* discover that correlations of mid-level features in a Deep Convolutional Neural Network (DCNN) can represent textures, structures, and styles of images [GEB15; GEB16]. We train a DCNN to predict color composition similarity using the VGG19 architecture [SZ15] on our color similarity dataset. The responses from different convolutional layers of our trained VGG19 capture color features at different levels, from the lowest patch-based to the highest global representation. We successfully use these mid-level features of our color VGG19 network in color transfer. To go an extra step, we incorporate our color network with the neural style transfer network in [GEB16] to create a new algorithm for style transfer, in which color and style are extracted from different sources of input images. The results show that our color transfer method creates aesthetic and pleasant images, both in color transfer alone and in combination with style transfer. The work on color composition similarity is presented in Chapter 4 and published at the WACV conference in 2020 [HHB20].

Shape Extraction and Semantic Segmentation (Part II - Chapter 5)

In the second part of the thesis, we present different studies on high-level features from classification DCNNs. When a network is trained for image classification, its task is mainly to predict images' labels. However, when trained on a large-scale dataset such as ImageNet [Den+09] with 1,000 classes and millions of images, it raises the question of which types of features the network captures while learning the classification task. Our work on studying deep features started with the work from Zhou *et al.* [Zho+16] where the authors tried to identify which regions in an input image that the classification network "looks at" to decide on its prediction. In other words, it looks for regions in images that contain discriminative features for a predicted class. The results are called Class Activation Maps (CAMs), which can be loosely translated as a map showing regions that activate a class prediction. However, these CAMs are constructed at the size 14×14 pixels, which is the size of the last convolutional layer in the network, whereas the input image size is 224×224 . Therefore, the CAM resolution is very low when projected back to the input image.

In a separate study, we find that a classification CNN trained on ImageNet dataset [Den+09] can embed the shape information inside its high-level convolutional features. The study was done by analyzing convolutional responses from the network on a small dataset designed for shape detection. The dataset contains 200 images of different objects' shapes. We use data augmentation to add different colors and textures to the shapes. We look for a convolutional layer that has the least variance to the color and texture changes. We find that the shape information resides in one of the convolutional responses towards the end of the CNN. We combine the shape information with the existing CAMs on two resolutions of an input image to produce a high-resolution CAM (rCAM). Unlike existing CAM methods, our rCAM technique can discover multiple instances in multiple resolutions of the predicted class. Our high-resolution CAM results are very good such that we can apply them in semantic segmentation. We achieve state-of-the-art results in weakly supervised semantic segmentation without any retraining or fine-tuning the classification network. The complete work and results are described in Chapter 5 and published at the WACV conference in 2018 [Ha+18].

Neural Discriminant Analysis (Part II - Chapter 6)

The last work, but not least, in part two of the thesis is about learning discriminant analysis for deep classification features. Instead of just training on the loss function that measures errors, it is possible and makes sense to strengthen the signal on the feature layers by rewarding a high inter-class variation relative to the intra-class variance of the features. One of the effects is that the system not only just barely classifies the training examples correctly, but that it does so by a large margin. We have reasons to believe that classification by a large margin generalizes better to new data. For example, Support Vector Machines (SVMs) do exactly this, and they have a theoretical motivation using Vapnik Chervonenkis (VC) dimension.

To improve the discriminant of classification features, we take the objectives of Linear Discriminant Analysis (LDA) and incorporate it into DCNN training. LDA is a supervised dimension reduction technique that transforms a feature space into a lower-dimensional space that satisfies two conditions. The first one is to maximize the variance between classes. The second is to minimize the variance within each class. We put these criteria in the form of two-phase optimization for deep features using Neural Networks, hence the name Neural Discriminant Analysis (NDA). In the first phase, NDA optimizes the Mean loss, whose goal is to minimize the intra-class variance. In the second phase, we use a Siamese network that receives two input images. The two images are passed through a shared weight DCNN architecture to extract their deep features for classification. The optimization for the Siamese is to push the two inputs further apart if they are from different classes and pull them together if they are from the same class. The NDA optimization alternates between these two phases. We test our two-phase NDA on the fine-grained classification, where the system needs to classify sub-classes that often have high intra-class variances such as the same object with different colors, poses, or lighting conditions. At the same time, it can also have low inter-class variances, such as sub-classes of birds. We achieve good improvement results on several popular fine-grained classification datasets over the baselines, which is the classification results without using NDA. We also propose another version of NDA, where all the losses are combined. Combining all the loss functions into one optimization is advantageous in its ease of use for implementation and training. We conduct experiments with the combined loss on CIFAR-10 dataset, a general classification dataset known to be a miniature version of ImageNet [Den+09]. It shows that the combined loss NDA improves the classification results on various networks and surpasses the results of the competing method by a large margin. The details of this work are described in Chapter 6, and the work is published at the ICIP conference in 2020 [HB20].

Thesis Overview Relating To Feature Representations

Deep features are extracted at different layers inside a DCNN. The features are presented from the most fundamental to the highest abstract level in the direction from the input to the output. The lowest level of features corresponds to engineered features by using various designed kernels such as edge detection, corner detection and color blob detection, to name a few. Mid-level features encode the texture and complex patterns of the input image. The highest abstract features present discriminative features of input images that are very efficient for classification or regression. The thesis includes work covering the whole range of features, from pixel-based to high-level abstract features. An overview of the thesis is illustrated in Figure 1.1.

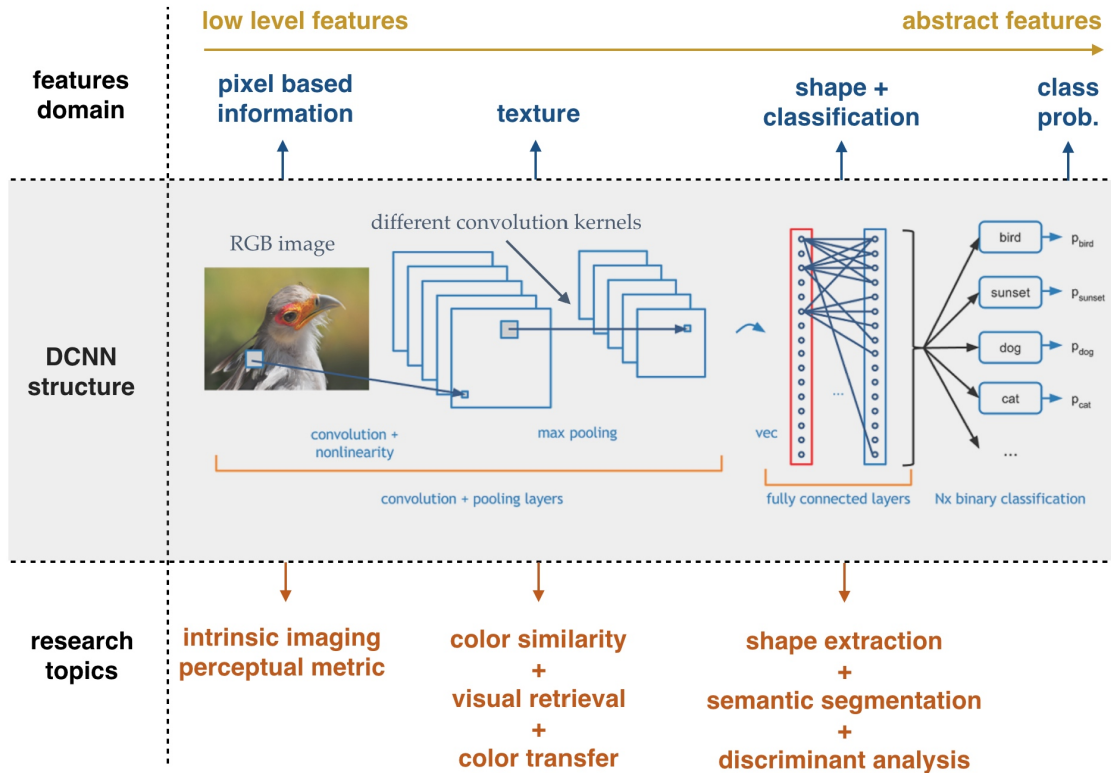


Figure 1.1: An overview of the thesis based on different levels of deep features extracted at different parts of a Deep Convolutional Neural Network (DCNN). At an image pixel-based level, we developed a perceptual metric to evaluate intrinsic imaging methods (in Chapter 3). For mid-level features that represent textures and structures of images, we innovate a global color composition similarity measure that is successfully applied in visual image retrieval and color transfer (in Chapter 4). At high-level abstract features, we produce high-resolution Class Activation Maps (rCAM), extract shape features, and perform weakly supervised semantic segmentation (in Chapter 5). We also introduce a Neural Discriminant Analysis (NDA) optimization to transform the high-level features into more discriminative features that result in better classification in general and fine-grained classification specifically (in Chapter 6). (Source image for the CNN structure: <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>)

Summary of Contributions

The main contributions of this thesis and the benefits that result from the thesis are summarized as follows:

- The first contribution is a new perception-inspired metric for evaluating intrinsic imaging methods. The metric evaluates intrinsic reflectance results using the perceptual color distance CIE DE2000. The evaluations for intrinsic imaging methods using this metric are more consistent across different illuminations than previous metrics.
- The second one is a color composition similarity dataset, the first of its kind, and an active learning framework for studying subjective topics. The benefits derived from this work are many folds. The active learning framework is proven to be an effective methodology to formulate and solve subjective problems. The dataset is a valuable data source for creating different global color similarity descriptors that yield better image retrieval and fine-grained image retrieval results. Deep color features, trained on the color similarity dataset using a DCNN, lead to a successful neural color transfer technique with satisfying results. Furthermore, combining these color features and content features from a classification DCNN produces a whole new type of style transfer technique that allows the final result to have the color composition and image content from separate image sources.
- Thirdly, new insights into properties of different layers in classification DCNNs are discovered, especially shape, and a method to extract shape information from a pre-trained DCNN for image classification. As a result, a "weaker" than weakly supervised learning method for semantic segmentation uses the extracted shape information is developed with far more generalization ability than existing weakly supervised segmentation methods.
- Finally, another significant contribution is an optimization implemented for DCNNs based on the classical Linear Discriminant Analysis principles to learn discriminative features. The optimization leads to improvements of the "black-box" results from classification DCNNs, especially for fine-grained classification.

As the majority of the work in this thesis is related to deep features embedded inside DCNNs, an overview of DCNNs and their fundamental components are described in Chapter 2. We also demonstrate the links of different feature levels to different topics of the thesis at the end of Chapter 2.

Chapter 2

Convolutional Neural Networks

This chapter presents some foundations and evolutions of Artificial Neural Networks (ANN) and Deep Convolutional Neural Networks (DCNNs). The second half will focus on studying the features produced by DCNNs and demonstrate the links to my research work.

2.1 Artificial Neural Network (ANN)

“The journey of a thousand miles begins with a single step.”

Lao Tzu

The human brain is extraordinary because it is capable of hosting a unique intelligence, that can be self-aware, study itself, and move forward to re-create its own intelligence using machines. In 1943, the neurophysiologist Warren McCulloch and the mathematician Walter Pitts marked the birth of Artificial Intelligence by creating electrical circuits that simulated how neurons in the human brain were believed to work [MP43]. The McCulloch-Pitts single neuron model (known as the *M-P neuron*) used linear separable logic functions (such as AND and OR) applied on the logical input signals to produce binary classification outputs.

In 1958, Rosenblatt proposed the *perceptron* model that was an improvement of the *M-P neuron* [Ros58]. Instead of using logical inputs and logic functions, the *perceptron* neuron produces binary classifications by a weighted sum of real numerical inputs. Combining multiple perceptrons into one layer extends the classification from binary to multiple classes. It forms the single-layer perceptron, also known as a single-layer Artificial Neural Network (ANN). However, from the *M-P neuron* to single-layer perceptron, they all share the same limitation, which is the inability to perform non-linear classification, for example, to learn the XOR function [MP69].

An extension of the single-layer perceptron, a multi-layer perceptron (known as ANN) has boosted the computational power to the extent that it can theoretically solve any logic functions, or be able to classify multi-class non-linear data.

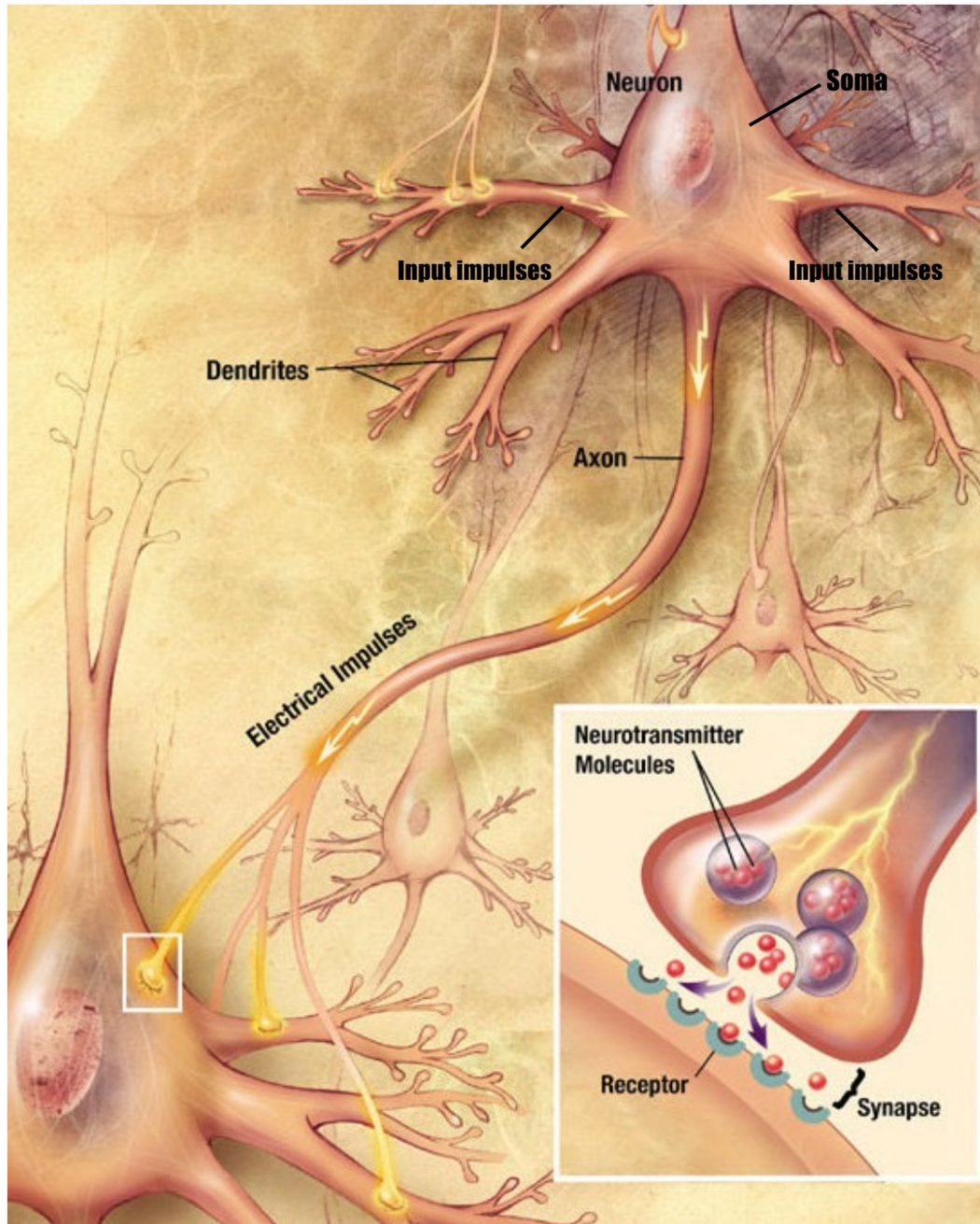


Figure 2.1: An illustration of a simplified and accepted structure of a biological neuron and its signal transmission mechanisms. A neuron consists of a soma which is the body of the cell, dendrites that receive and process input signals from other neurons, an axon to transmit the electrical impulses out of the soma, and axon terminals that form synapses to other neurons. A synapse is the point of “contact” of one neuron to another. The process of signal transmission is: a neuron receives input signals and processes the information in both the dendrites and the soma. In certain conditions, if the electrical polarization near the axon hillock surpasses a certain threshold, the neuron will produce output impulses (action potentials) that travel through the axon and transmit the signal to other connected neurons via synapses. (Source image: <https://en.wikipedia.org/wiki/Neuron>)

2.1.1 McCulloch-Pitts neuron (M-P neuron)

Loosely inspired by the biological brain neuron model, Warren McCulloch and Walter Pitts invented the first computational neuron called McCulloch-Pitts neuron (or M-P neuron in short). Even though neurology research continues to discover more about our brain and neural functioning, the widely accepted simplified neuron model until today is illustrated in Figure 2.1.

A neuron consists of several parts. It has a main body called *soma* which processes information. The information propagates from one neuron cell to another in the form of electrical impulses. A neuron receives electrical signals predominantly via its *dendrites*. When input signals are received, and certain conditions are met, the neuron produces electrical impulses (action potentials) that enable a fast transmission of information to another neuron via its *axon*. At the end of an *axon* are *axon terminals* that form synapses, which allow the stimulation to be passed along to other neurons.

M-P neuron model

Similar to a biological neuron, an M-P neuron also receives input signals in the form of binary values, processes the inputs, and generates an output signal (Figure. 2.2). The processing function comprises two steps: the first step is to aggregate the input, and the second step is to decide on the output based on the result of the first step.

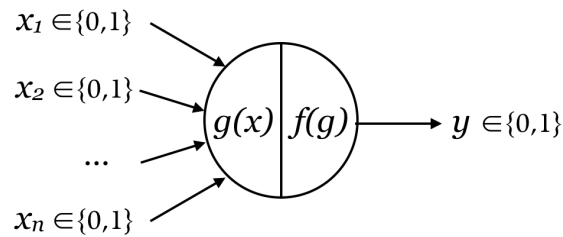


Figure 2.2: McCulloch-Pitts neuron model: the neuron receives inputs x_1, x_2, \dots, x_n that are boolean numbers. Function $g(x)$ aggregates the inputs and function $f(g)$ produces the output y , which is also a boolean value that can be used for binary classification.

Let $x_i \in 0, 1, i = 1..n$ be the binary inputs. g_x is the function to aggregate the inputs and $f(g)$ is the function that decides the output y , depends on the value produced by g .

$$g(x) = \sum_{i=1}^n x_i \quad (2.1)$$

$$f(g(x)) = \begin{cases} 1, & \text{if } g(x) \geq \theta \\ 0, & \text{if } g(x) < \theta \end{cases} \quad (2.2)$$

$$y = f(g(x)) \quad (2.3)$$

The M-P neuron was designed for processing logical circuits. Let's take an example of an M-P neuron that represents the AND function and has two inputs x_1 and x_2 . The output y can only be 1 if the values of both input x_1 and x_2 are 1. Therefore, the functions f and g are defined as: $g(x) = x_1 + x_2$ and $f(g(x)) = 1$ if $g(x) \geq 2$.

2.1.2 Perceptron

Improving the limitation in the input type of M-P neuron, Rosenblatt proposed a perceptron model [Ros58] that processes real value inputs.

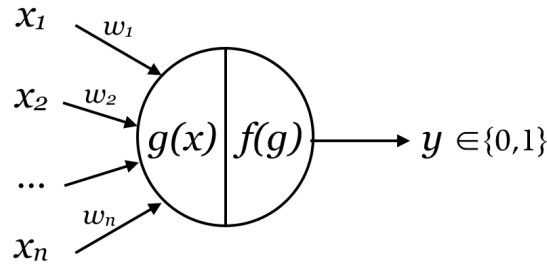


Figure 2.3: A perceptron model: the neuron receives inputs x_1, x_2, \dots, x_n that are real numbers. w_1, w_2, \dots, w_n are the weights of each input, respectively. Function $g(x)$ is the weighted sum of the inputs and function $f(g)$ produces the output y , which is a boolean value that can be used for binary classification.

Let $x_i \in \mathbb{R}$ be the real number inputs and $w_i \in \mathbb{R}$ be the weights respectively, $i = 1..n$. g_x is a function to aggregate the inputs and $f(g)$ is the function that decides the output y , depending on the value produced by g .

$$g(x) = \sum_{i=1}^n w_i * x_i \quad (2.4)$$

$$f(g(x)) = \begin{cases} 1, & \text{if } g(x) \geq \theta \\ 0, & \text{if } g(x) < \theta \end{cases} \quad (2.5)$$

$$y = f(g(x)) \quad (2.6)$$

The equation 2.5 can be rewritten as:

$$f(g(x)) = \begin{cases} 1, & \text{if } g(x) + b \geq 0 \\ 0, & \text{if } g(x) + b < 0 \end{cases} \quad (2.7)$$

where $b = -\theta$ and b is commonly referred to as "bias". In modern terms, $f(g(x))$ is called an activation function.

With the perceptron model, the inputs and weights are real values, the weights and the threshold θ are learned. Thus, it can be applied in many types of data with fewer restrictions

than the M-P neuron. The Rosenblatt perceptron model is also known as a single-layer neural network where the neuron itself is the single link between the inputs and the output signals.

However, both models share the same limitation, which is the inability to classify non-linear data. One example is the data in two-dimensional space, and the classification is the XOR function [MP69] (Figure. 2.4).

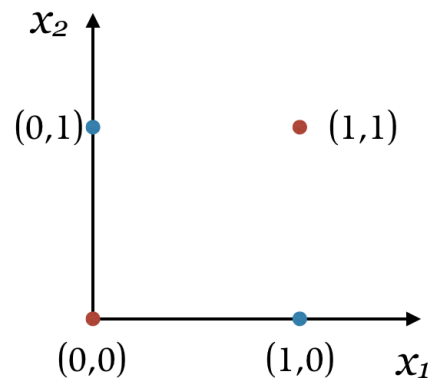


Figure 2.4: Data with XOR function (\oplus): XOR function creates non-linear data. $y = x_1 \oplus x_2$, $y = 0$ if x_1 and x_2 have the same values, otherwise, $y = 1$. Therefore, the red points with the coordinates $(0,0)$ and $(1,1)$ belong to class 0 and the blue points with the coordinates $(0,1)$ and $(1,0)$ belong to class 1.

Another limitation of the single neuron in the M-P neuron or perceptron model is that they can only do binary classification where the output y only has boolean value 0 or 1 to represent two classes. One way to extend the existing model to fit the multi-class classification problem is to use multiple neurons to produce multiple output numbers, which leads to the establishment of neural networks.

2.1.3 Neural Networks

A Neural Network (NN) is a collection of neurons that are organized into layers (Figure 2.5). It contains one input layer, one output layer, and at least one hidden layer. The hidden layers consist of a set of neurons that individually functions as a perceptron neuron. Neurons between two adjacent layers are fully pairwise connected. Therefore, these layers are also called fully-connected layers.

One of the advantages of Neural Networks over the perceptron model is that a Neural Network can have one or many neurons in its output layer. Single output neuron results in binary classification, but multiple output neurons allow the network to perform multi-class classification.

In order for a network to function correctly, the weights need to be learned such that the difference between the output prediction and the ground-truth is minimized. A method for iteratively updating the weights to obtain the desired output predictions is *backpropagation*. It was popularized to be used in Neural Network in 1986 by Rumelhart, Hinton and

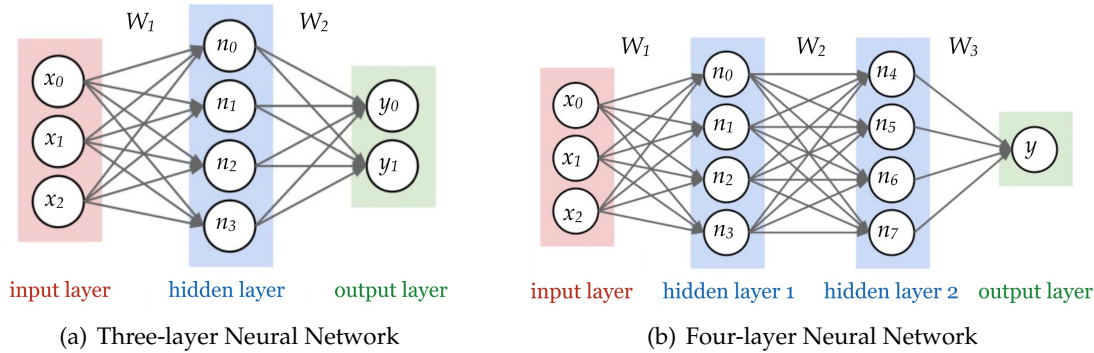


Figure 2.5: Examples of Neural Networks: 2.5(a) a three-layer Neural Network with one input layer, one hidden layer and one output layer. 2.5(b) a four-layer Neural Network with one input layer, two hidden layers and one output layer. The output layers can have one or many neurons for a binary or a multi-class classification, respectively. W_i are weights that pair-wise link neurons between two adjacent layers.

Williams [RHW86], even though the backpropagation itself was first introduced much earlier.

Backpropagation

When a Neural Network is used to perform classification, the input data is fed into the input layer. The information then travels through the hidden layers and arrives at the output layer yielding the output prediction. This process is called *feedforward*. In contrast to the feedforward, the backpropagation travels from the output layer through hidden layers back to the input layer. Backpropagation updates the network's weights using the gradient of the *loss function*. There are many types of loss functions depending on what type of outputs the network has. In principle, a loss function measures the difference between the produced output and the ground-truth. The gradient of the loss function is calculated with respect to the weights and propagated backward through hidden layers. For an input-output pair, the gradient is computed as $\delta\mathcal{L}/\delta w_{jk}^l$ where \mathcal{L} is the loss between the predicted output and the target output, w_{jk}^l is the weight between layer $(l-1)$ and layer l that links the j^{th} neuron in layer $(l-1)$ with the k^{th} neuron in layer l . Backpropagation is efficient, thanks to the chain rule to compute derivatives in backward fashion for one layer at a time.

Different optimizers can be used for updating the weights with backpropagation such as gradient descent, stochastic gradient descent (SGD), Adagrad [DHS11], Adam [KB15], to name a few. Each optimizer uses different strategies to update the weights given the computed gradients. For example, gradient descent updates parameters by using gradients of the whole dataset. In a large dataset, it takes a long time to compute gradients for all the data. SGD, in contrast, performs a parameter update for each training data or on a subset of the training data. Therefore, SGD is faster than gradient descent. However, SGD produces higher fluctuations in the convergence of the objective loss. Adagrad improves over SGD by adapting the learning rate to the parameters. While SGD uses the same learning rate for all the parameters, Adagrad uses different learning rates for every parameter at each time-step.

Similar to Adagrad, Adam uses an adaptive learning rate, but it also incorporates momentum in the optimization. Momentum increases the updated value for a parameter when the gradient has a steeper slope. Hence, it helps the optimizer converge faster.

Activation functions and nonlinearity

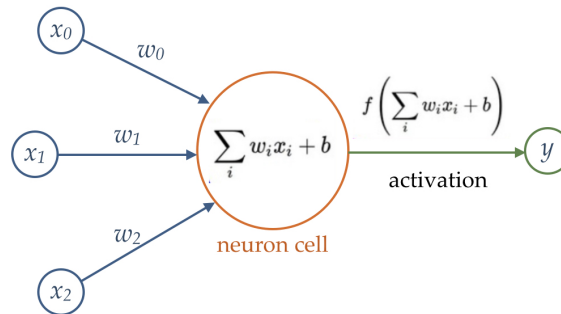


Figure 2.6: An Illustration of a single neuron of a Neural Network. The neuron computes the weighted sum of its inputs and adding a bias value. Based on the value produced by the neuron, the activation function f decides the output value.

The building block of Neural Networks is still a single cell neuron that has a similar mechanism as the perceptron neuron described in Section 2.1.2. However, the perceptron can only separate linear data. Referring back to the activation of a perceptron in Equation 2.5, it is easy to see that this is the step function (Figure 2.7(a)). The step function is linear. The derivative is zero at all points except for the point 0, where the derivative is undefined. Therefore, perceptron neurons cannot classify nonlinear data such as XOR data in Figure 2.4.

The linearity problem with the perceptron neuron is fixed in Neural Networks by using different nonlinear activation functions f (Figure 2.6). Few commonly used activation functions are described as following:

- **Sigmoid:** the Sigmoid function is defined as $\sigma(x) = 1/(1 + e^{-x})$. It is a nonlinear function that maps all real numbers into the range of $(0, 1)$ (Figure 2.7(b)). One drawback of the Sigmoid function is that the neuron's activation is saturated at the tail of 0 and 1. The gradient at these tail regions is close to 0. Hence, almost no signal travels back in the backpropagation process to update the weights.
- **Tanh:** the Tanh function is similar to the Sigmoid function and can be described as $\tanh(x) = 2\sigma(2x) - 1$ where $\sigma()$ is the Sigmoid function (Figure 2.7(c)). The Tanh activation function is nonlinear and maps real numbers to the range of $(-1, 1)$. Similar to Sigmoid, Tanh also suffers from the gradient vanishing problem. Different from Sigmoid, the Tanh function is zero-centered. Depends on the specific requirements of an individual application, one then can choose between Sigmoid and Tanh.
- **Rectified Linear Unit (ReLU):** The Rectified Linear Unit (ReLU) simply threshold the activation at zero $f(x) = \max(0, x)$ (Figure 2.7(d)). Besides the simplicity of the function implementation, it is found to help with the convergence of stochastic gradient

descent compared to the Tanh or Sigmoid function. It does not have a gradient saturation problem like the Tanh and Sigmoid functions when the signal x is a large positive number. However, ReLU also bears the zero gradient problem, which is called "dying ReLU". If a neuron falls into a state that it always produces a negative signal ($\sum_i w_i x_i + b < 0$), which could be due to a large negative bias, the gradient is always 0. Therefore, the neuron's weights and bias can never be updated, and the neuron cannot recover from this dead state.

- **Leaky ReLU:** Leaky ReLU is an upgraded version of ReLU to fix the "dying ReLU" problem by introducing small gradients for negative inputs. It is defined as $f(x) = x$ when $x \geq 0$ and $f(x) = \alpha x$ when $x < 0$. α is a small constant such as 0.01 (Figure 2.7(e)).
- **Exponential Linear Units (ELU):** ELU is another alternative to fix the "dying ReLU" problem. ELU is similar to Leaky ReLU, except the simple linear function for negative inputs is replaced by a nonlinear function. It is defined as: $f(x) = x$ when $x \geq 0$ and $f(x) = \alpha(e^x - 1)$ when $x < 0$, $\alpha \geq 0$ (Figure 2.7(f)). Both Leaky ReLU and ELU have α as an extra hyper-parameter for tuning.

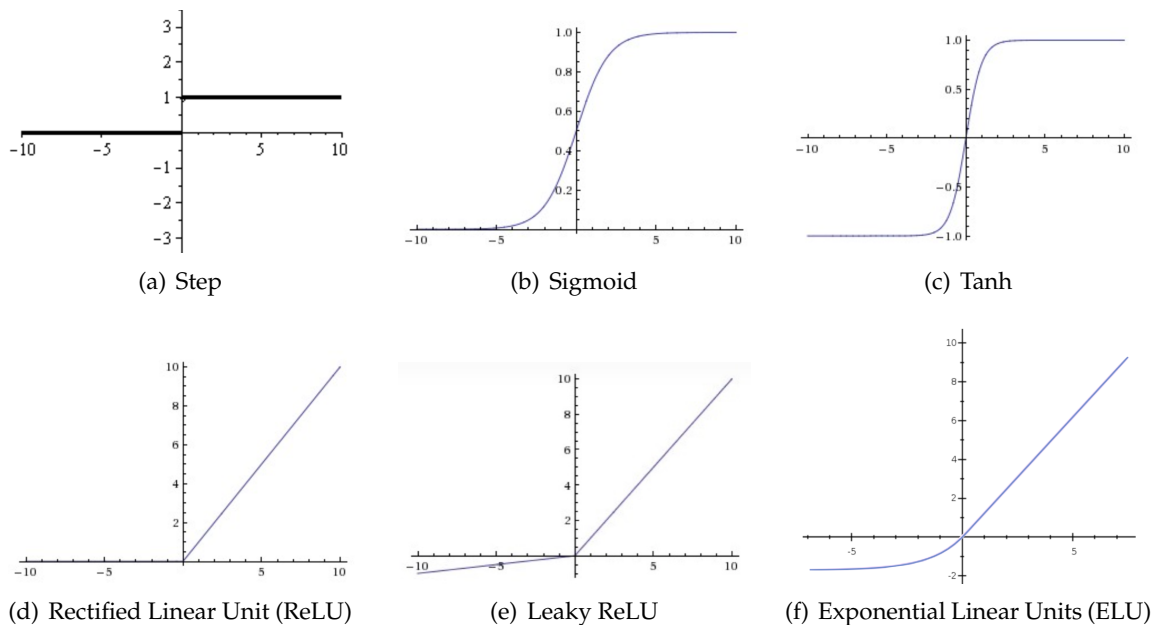


Figure 2.7: Commonly used activation functions in Neural Networks.

Growing in depth

The total number of neurons and the number of hidden layers in a Neural Network define the network's complexity. Increasing the number of hidden layers is increasing the depth of the neural network. It is often the case that a four-layer Neural Network performs better than a three-layer Neural Network. However, increasing the depth of Neural Networks further does not help much with performances. One probable reason is that images contain hierarchical structure, and neighbor pixels and patches are correlated. These correlations

are not captured well in Neural Networks. The search for a better architecture of a neuron inspired network leads to the birth of Convolutional Neural Network.

2.2 Convolutional Neural Networks (CNN)

2.2.1 Fundamentals of CNN

One of the limitations of Neural Networks is the ability to learn feature representations, especially for 2D data such as images and videos. Useful feature representation is a prerequisite for achieving good performances in many Computer Vision problems. An increasing number of neurons or number of hidden layers in a Neuron Network does not help to learn better feature representations. Instead, the number of parameters increases significantly every time another hidden layer is added due to fully pairwise connections between adjacent layers. Fortunately, a new set of architectures that use convolution and pooling, together with existing components from Neural Networks, open a whole new horizon for Machine Learning. They are Convolutional Neural Networks (CNN).

Convolution

A 1D continuous convolution is a function that operates on two input functions and defined as:

$$(f * g)(t) \equiv \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau \quad (2.8)$$

The convolution of the functions f and g is computed by reversing the function $g(\tau)$ to become $g(-\tau)$ and the time offset t is added to make $g(t - \tau)$ shifting along τ axis. The time t spans from $-\infty$ to ∞ . When two functions overlap each other at the time t , the integral of their product is computed and results in the convolution $(f * g)(t)$.

The discrete version of the 1D convolution is defined as following:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m] \quad (2.9)$$

Many Computer Vision applications deal with 2D images and videos. Therefore, popular CNNs use discrete 2D convolutions. A 2D convolution for an image I with a kernel W is defined as:

$$(W * I)(i, j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} W[m, n] \cdot I[i - m][j - n] \quad (2.10)$$

where (i, j) is a 2D pixel coordinate on which the convolution is currently computed. The values of m and n are from 0 to the width and height of kernel W . An example of 2D convolutions is illustrated in Figure 2.8.

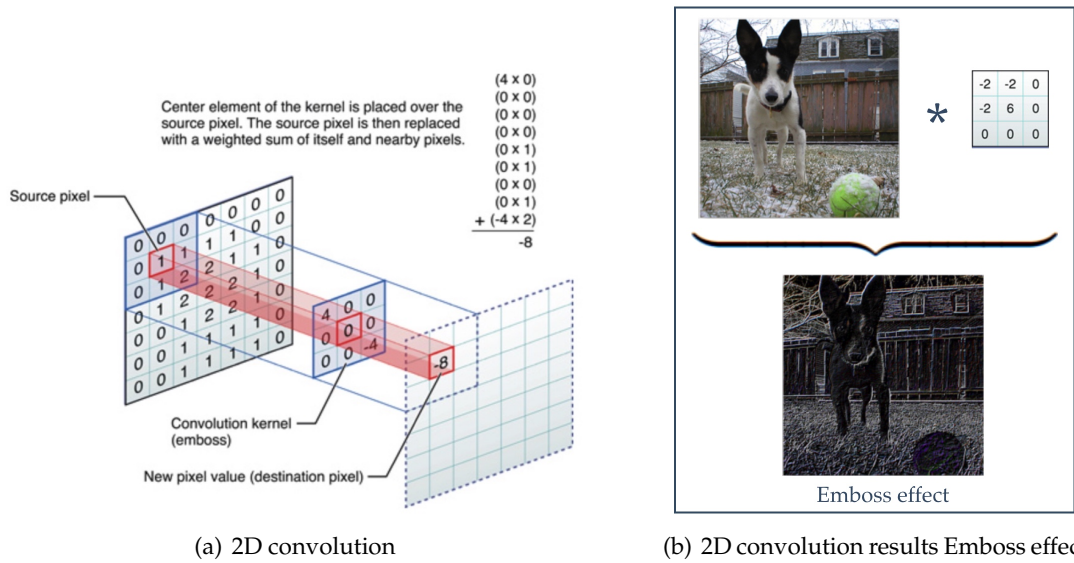


Figure 2.8: Examples of 2D convolutions: 2.8(a) demonstrates a convolution between an image and a 2D convolutional kernel. The input image contains source pixels, and the convolutional kernel (in the middle) carries the weights. The center of the kernel is placed over the source pixel. The operator then computes the weighted sum of the source pixel with its neighbors and outputs to the destination pixel. 2.8(b) shows an example of convolutions between a 2D kernel and an image to produce the Emboss effect. (Source image: <https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>)

Unlike fully connect layers in Neural Network, a convolution kernel contains much fewer parameters. The number of parameters for a kernel W with a size of (m, n) is mn . A kernel W is shifted across all the pixels of the original image to detect a specific pattern of features. Hence, the weights of the kernel are shared commonly for all the pixels of the image. Besides, this mechanism also helps to maintain the spatial relationship of features in the original image (see Figure 2.8(b)).

A representation of an image can be formed from multiple features extracted from the image by applying different filter kernels such as Sobel filters for edge detection, Gabor filters, Gaussian filters, color filters, and more. By applying these filters directly to an RGB image, we can extract the low-level features presented in the image. A set of convolutional kernels applied to the same input form a *convolutional layer* and the output results are called *activation maps* or *feature maps*, interchangeably. As an example, Figure 2.9 displays 96 convolution kernels in the first layer of AlexNet [KSH12] trained on ImageNet [Den+09] dataset.

Bias and Nonlinearity

Bias is a constant added to the result of a convolution, as follows:

$$\mathcal{F} = \sum_i w_i x_i + b \quad (2.11)$$

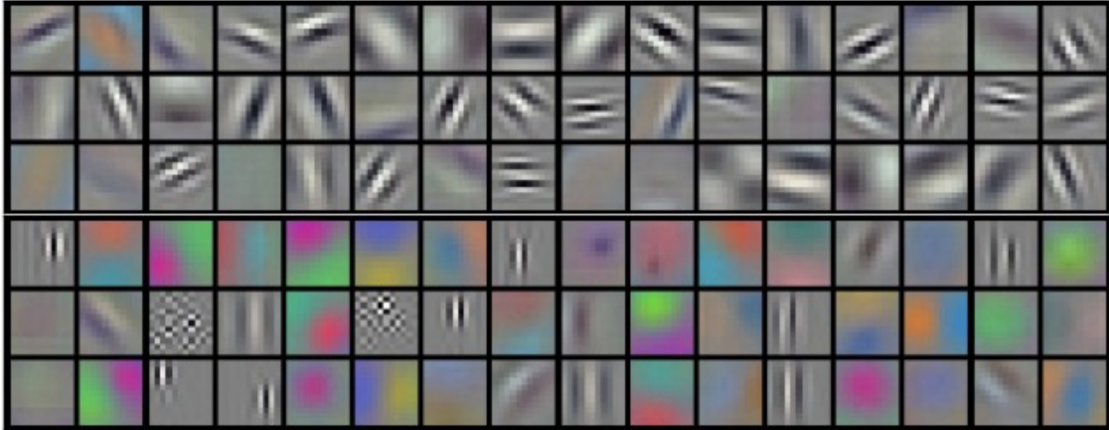


Figure 2.9: A display of 96 convolution kernels in the first layer of AlexNet that is trained on ImageNet.

where \mathcal{F} is a feature map, b is a bias, the weight w_i is an individual value of the convolution kernel W and x_i is a value of a correspondent pixel in an image I . In short, it can be written as $\mathcal{F} = W * I + b$. In the Equation 2.7, the bias b is a threshold for a neuron to produce the output signal. There is an alternative interpretation. The bias helps the model to fit better on a given data. For example, in the task of fitting a straight line $y = ax + b$ to 2D data, without a bias b , the line always passes through the origin $(0,0)$. The bias b allows the line to move out of the origin and therefore, can lead to a better fit to the data (see Figure 2.10). Biases are additional network parameters that a CNN needs to learn.

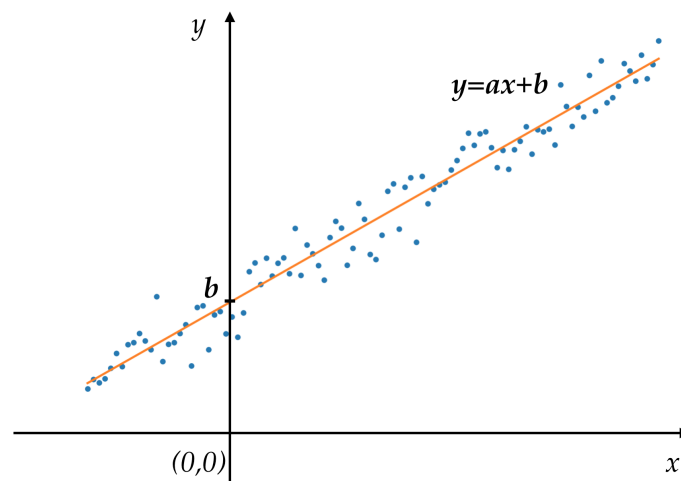


Figure 2.10: An example of a linear regression: fitting a line to a given data. The bias b helps to move the line out of the origin so that the line $y = ax + b$ can fit the data better.

Like in Neural Networks, nonlinearity is achieved by nonlinear activation functions such as Sigmoid, ReLU, etc., as described in Section 2.1.3. When multiple convolution layers are combined as a sequence, the nonlinear activation functions prevent all convolutional layers from collapsing into a single layer of convolution. The sequence of convolutions and activations can be written as: $\sigma_n(W_n * \dots \sigma_2(W_2 * (\sigma_1(W_1 * I))) \dots)$

Pooling

Another important building block of CNNs is pooling. Pooling is applied to reduce the size of feature maps resulted from one convolution layer before applying the next convolution layer. In other words, pooling is an operator to down-sample the feature maps. Two types of pooling commonly used are max pooling and average pooling. The max pooling applies the max operator on each local patch of the feature maps, whereas the average pooling averages each local patch. An illustration of the max pooling is presented in Figure 2.11.

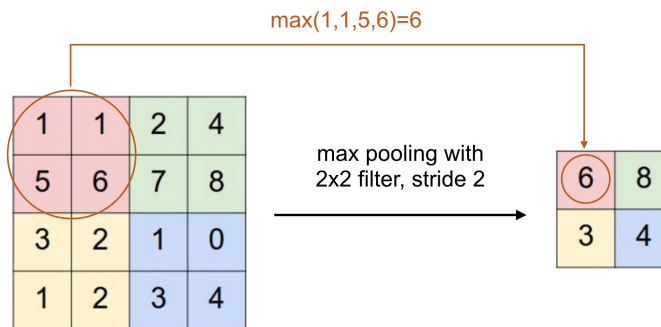


Figure 2.11: An example of a 2D max pooling that has a filter size of (2,2). A stride of 2 means that every time the filter is shifted by 2 pixels horizontally and vertically. The result is the maximum value within the image block that the filter is applied on. (Image adapted from <https://cs231n.github.io/convolutional-networks/>)

Pooling techniques help to reduce the size of the feature maps. As a result, the number of parameters for the subsequent convolution layers are reduced. The progressive down-sampling feature maps also express the hierarchical structure of features from low-level features such as lines and colors into higher features such as eyes and noses. For example, the feature maps are visualized for different convolutional layers from a VGG19 network [SZ15] pre-trained on ImageNet [Den+09] in Figure 2.14. The features in the first few convolutional layers have more local details. In the later layers, the features are more sparse and more abstract that focus on different important parts of objects in the input image.

The hierarchical structure of features is an advantage of a CNN. Different levels of features embed different characteristics that are useful in many Computer Vision applications.

Convolution and Neural Network

The combination of convolutions, nonlinear activation functions, and pooling is a great means to learn different levels of features, from local low-level features to highly abstract features that represent characteristics of images. The next task is to use these features for regression, classification, and other computer vision tasks. Even though there are traditional machine learning techniques for classification such as Support Vector Machine (SVM), Fisher Discriminant Analysis, and so on, Neural Network is the best choice to integrate with convolutional layers (Figure 2.12).

To link the 2D convolutional features to a Neural Network, the 2D features are flattened into a 1D feature vector that can be used as an input to the neural network. The other option is to

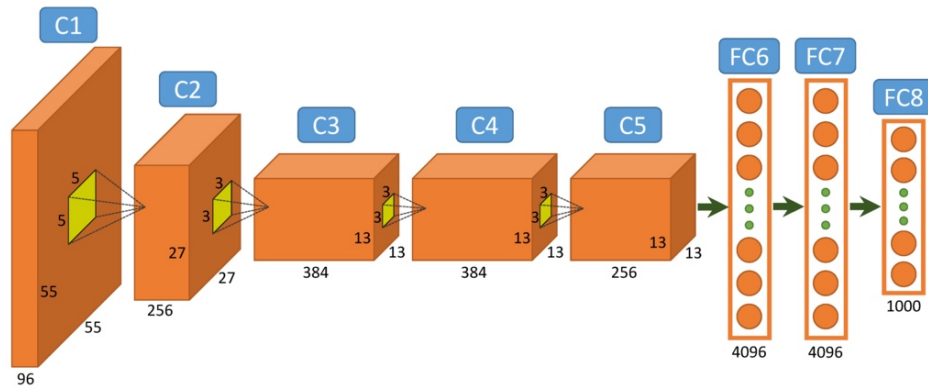


Figure 2.12: AlexNet [KSH12]: the first standard CNN architecture that combines convolutions and neural networks. (Source image: <https://www.saagie.com/blog/object-detection-part1/>)

compute the maximum or the average of individual 2D feature maps into a single number, and then form a 1D vector from these pooling results of all feature maps. Different with pooling for down-sampling, this is called *global pooling* (Figure 2.13).

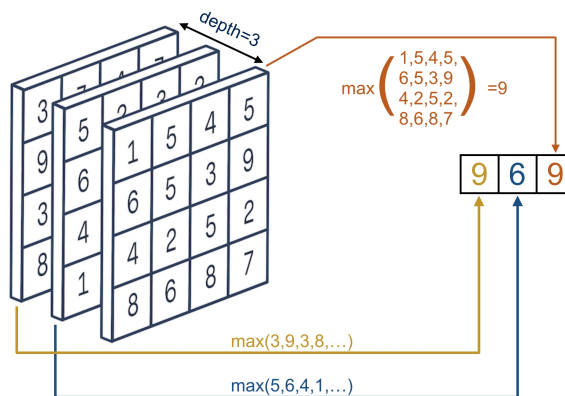


Figure 2.13: An example of a 2D global max pooling on a block that contains three feature maps. The 2D global max pooling computes the max value for each feature map and forms a feature vector. Therefore, the size of the feature vector is the same as the depth of the block of feature maps. (Image adapted from <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-global-average-pooling>)

The size of an input image will decide the size of convolutional feature maps. Therefore, the size of the feature vector produced by flattening the convolutional feature maps will change according to the size of the input image. With global pooling, the feature vector's size is fixed and equal to the number of feature maps. Hence, by using global pooling, the network is invariant to the size of the input images.

For different applications, different activation functions can be applied at the last layer of the network for prediction. For example, to classify 1,000 classes, the last layer of the network is a fully connected layer (Neural Network) with 1,000 neurons (see layer "FC8" of AlexNet [KSH12] in Figure 2.12). For example, with classification, each neuron is expected to return the probability of its corresponding class. The results are in $[0, 1]$, and all the class

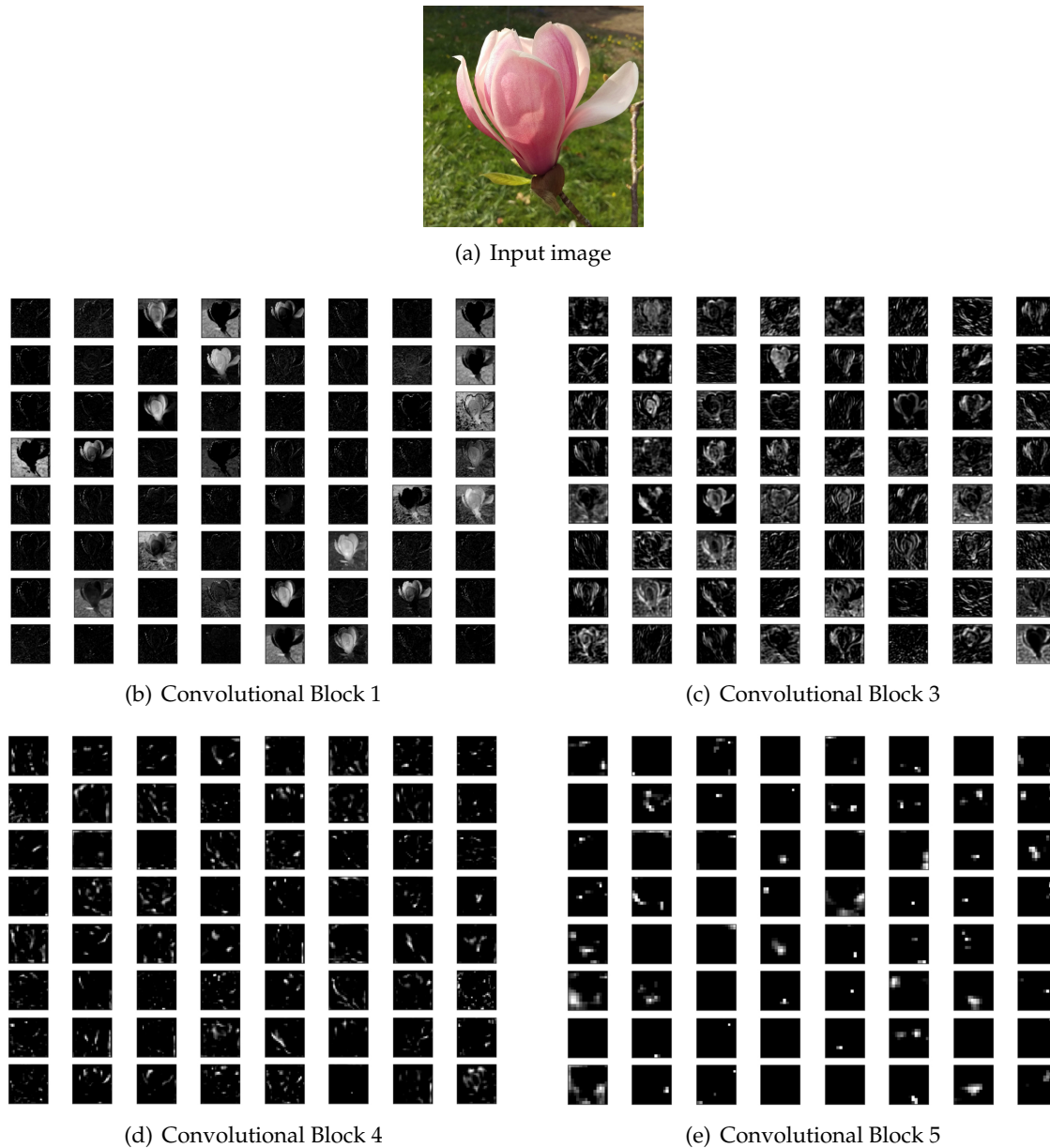


Figure 2.14: Visualizations of the convolutional features extracted from different convolutional layers of the VGG19 [SZ15] network, pre-trained on ImageNet [Den+09]. The first 64 feature maps for each convolutional block are displayed.

probabilities sum to 1. To specify this constraint, a Softmax activation function σ is applied to the feature vector \mathbf{z} from the "FC8" layer. The Softmax function is defined as follows:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}}, \text{ and } \mathbf{z} = (z_1, \dots, z_n) \in \mathbb{R}^n \quad (2.12)$$

The Softmax function has many advantages compared to other functions of which the results satisfy the probabilistic constraints such as argmax or standard normalization. For example, let's $\mathbf{z} = [2, 5, 2, 1]$. Argmax function results $[0, 1, 0, 0]$ which is the actual classification ground-truth. However, argmax is not differentiable whereas Softmax is. When using

the standard normalization, the normalized result is $[0.2, 0.5, 0.2, 0.1]$ and the Softmax result is $[0.0445, 0.8946, 0.0445, 0.0164]$. Comparing the two, the result of Softmax is closer to the one-hot encoded ground-truth than the result of standard normalization.

Training

Combining convolutions and Neural Networks forms end-to-end Convolutional Neural Networks. It makes the CNN feed-forward step simple in the deployment or testing phase. More importantly, all the components of a CNN such as convolutions, activations, and fully connected layers (neural networks) are differentiable. Therefore, it is possible to perform end-to-end backpropagation in training. The techniques and optimizers for training CNNs are the same as for Neural Networks that are described in Section 2.1.3.

2.2.2 CNN Architectures

The standard structure of a CNN is a series of convolutional layers with pooling in between them. Each convolutional layer is usually accompanied by an activation function. The features of the last convolutional layer are flattened or pooled to a single feature vector. This feature vector is connected to a series of fully connected layers (neural network). The last layer of the neural network is the prediction layer. An example of this standard architecture is AlexNet [KSH12] illustrated in Figure 2.12.

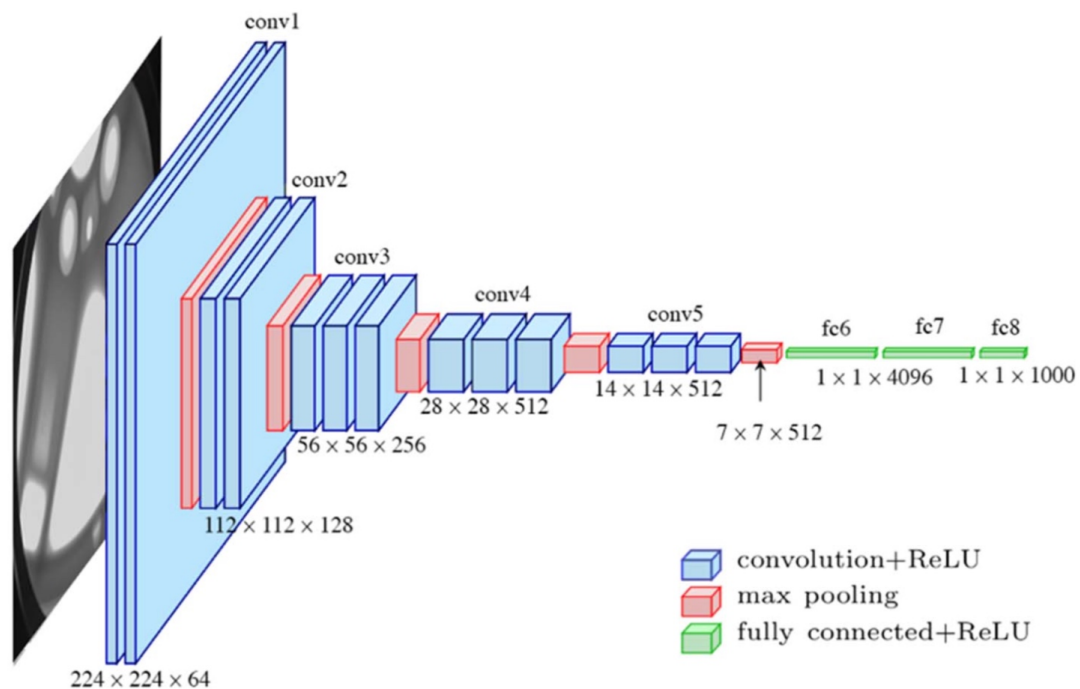


Figure 2.15: An illustration of a VGG16 network architecture to classify 1,000 classes in ImageNet [Den+09] dataset. The final layer "fc8" is activated by a softmax function. A similar network is VGG19 which has one extra convolutional layer in block conv3, conv4 and conv5 [SZ15]. (Source image: [Fer+17])

When adding more layers, a CNN network becomes "deeper" and therefore, the name *Deep Convolutional Neural Networks (DCNN)*. Unlike Neural Networks, these deep networks can

improve the performances further. The field that studies DCNNs or uses DCNNs in research is called *deep learning*. Examples of a deeper CNNs than AlexNet are VGG networks, which are widely known as VGG16 and VGG19 networks (see Figure 2.15).

One potential problem with going deep in CNN is the vanishing gradient problem. This is due to the large numbers of multiplications during backpropagation. This challenge does not stop the deep learning research community. Instead, more complicated or elaborated architectures are developed to support CNNs to move deeper. One solution is to introduce *residual blocks* in ResNet architecture [He+16a]. An illustration of a residual block is in Figure 2.16.

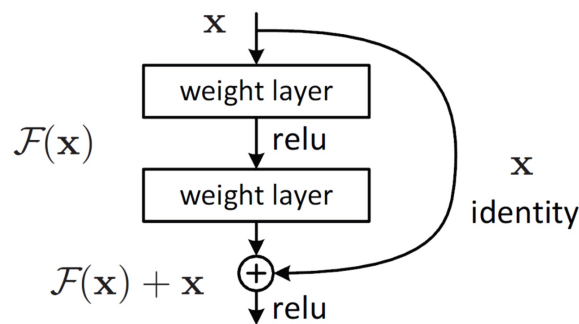


Figure 2.16: A residual block in ResNet architecture [He+16a]. The identity function maps the output to be the same as the input. The connection to add the identity from the previous layer is called *skip connection* or *residual connection*. (Source image: <https://mc.ai/resnet-architecture-explained/>)

The residual block uses a *skip connection* to add data from one layer to a later layer and skip intermediate layers between them. In the backpropagation, the gradient flows through the skip connection helps to recover the diminishing gradient happens through the normal path between adjacent layers. Using residual blocks, ResNet proves to maintain its performance or even improve the performance when the network comprises hundreds or thousands of layers.

Another problem with deep networks is over-fitting when the number of parameters increases with the depth of the network. A solution to this is *inception modules* that, at the core, take advantage of 1×1 convolution kernels. When applied on a block of activation maps, a 1×1 kernel performs a weighted sum of all the activation maps into one single activation map (Figure 2.17). Therefore, the subsequent convolutional kernel will have fewer weights. As a result, the number of network parameters is reduced. Figure 2.18 illustrates how 1×1 convolutions can be used in an inception module for dimension reduction. The popular network using the inception module is GooLeNet [Sze+15a].

Taking the strength of both work, Szegedy *et al.* integrated both the inception module and the residual block to form an Inception-ResNet network. It is proven to accelerate the training process and slightly improve the performance over the Inception network [Sze+17].

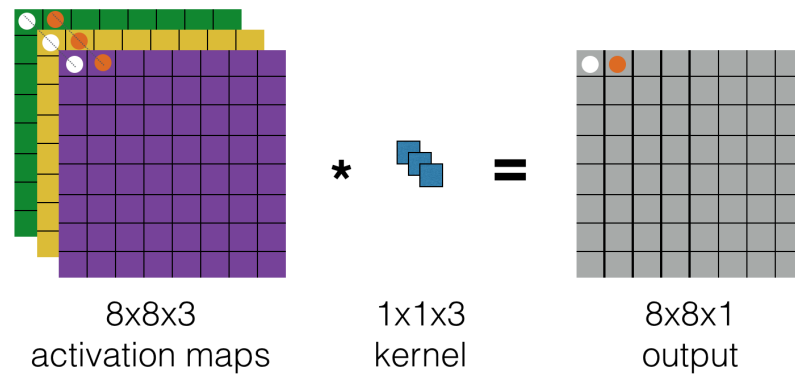


Figure 2.17: An example of a 1×1 convolutional kernel applied on a block of 3 activation maps. It produces an output of which the depth is 1. The result is a linear combination of the activation maps and the weights are stored in the 1×1 convolutional kernel.

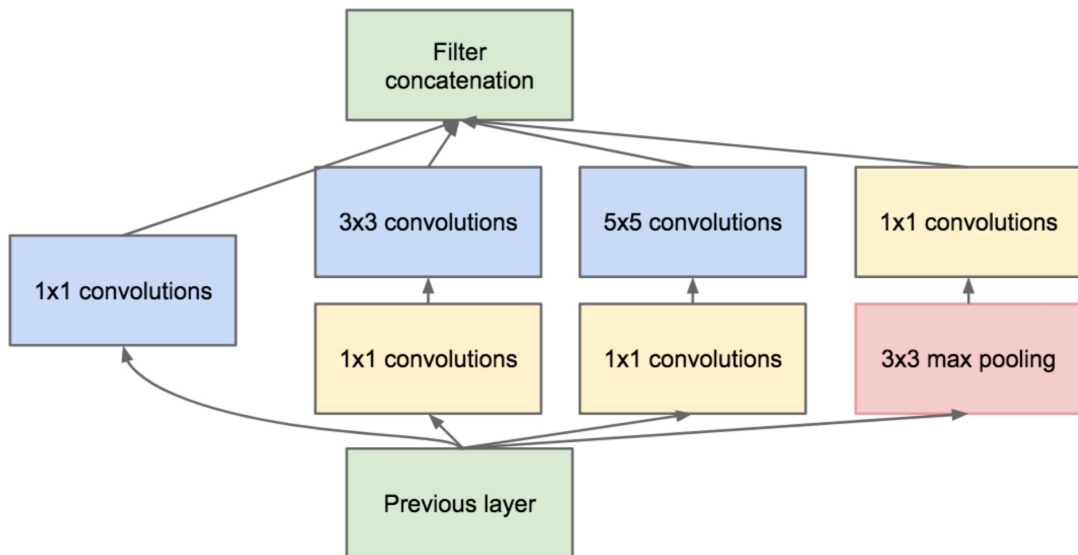


Figure 2.18: An inception module with dimension reduction. A 1×1 convolution is applied to the activation maps before applying 3×3 and 5×5 convolutions. Another 1×1 convolution is also applied after 3×3 max pooling to reduce the depth of the pooling result. The final output is the concatenation of all the results. (Source image: [Sze+15a])

Many other network architectures are successfully applied in various fields of research and applications such as Generative Adversarial Networks (GAN), U-Net, Variational Auto Encoders (VAE), to name a few. The architectures presented in this part are the core and fundamentals of Deep Convolutional Neural Networks (DCNN). They marked significant milestones for the development of deep learning in particular and machine learning in general. The key to move forward and dive deeper into this field is to understand these core concepts and what they can or cannot do. The majority of the work in this thesis stems from the motivation to understand DCNNs, which are often considered as black boxes, study different levels of deep features and apply them in solving different tasks in novel ways.

Part I

Perceived Similarity

"The eye sees only what the mind is prepared to comprehend."

Robertson Davies

Part I of the thesis contains work related to perception. As we all know, perception is in the eye of the beholder. The world that we perceive is not always the same as the reality of the physical world. For example, in Adelson's checker-shadow illusion in Figure 2.19, both square A and square B have the same shade, color, and brightness. However, it appears that square B is brighter than square A in Figure 2.19(a). It is debunked in 2.19(b) when a rectangle of the same color connects squares A and B.

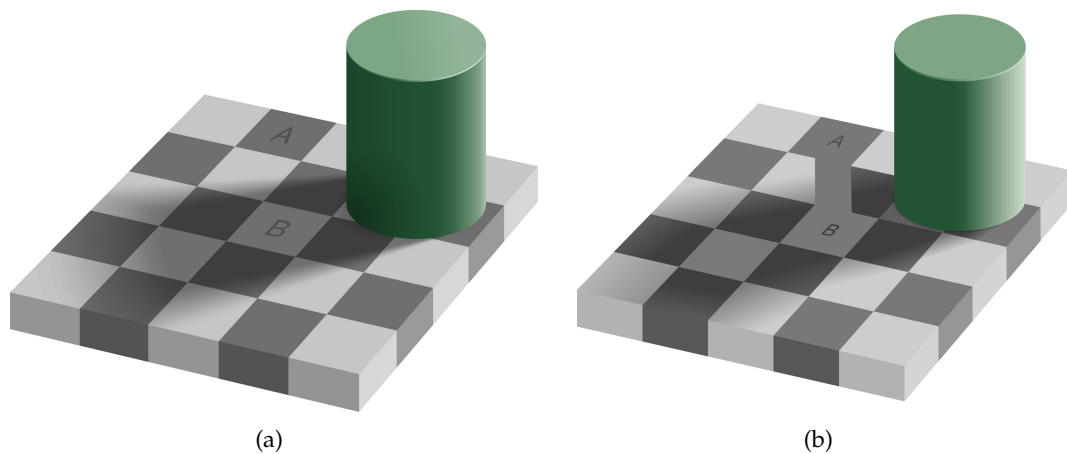


Figure 2.19: Adelson's checker-shadow illusion. In 2.19(a), we perceive that square B has lighter gray color than square A. However, both square A and square B have the same shade of gray (2.19(b)).

First, we perceive that there is a light source from the right side of the checkerboard that casts a shadow on the board in the area containing block B. The first illusion is to see the shadow that is not there because of the shading on the cylinder and the soft edges at the boundaries of shadow versus no shadow regions. The simultaneous contrast of the soft edges is related to the Cornsweet illusion (Figure 2.20). In the Cornsweet illusion, there is a high contrast of gradient in the middle, and the brightness on the two sides is perceived differently, but in fact, they are the same. The simultaneous contrast is an effect of ganglion cells, which receive signals from photoreceptive cone cells and rod cells in the retina and sending the signal to the brain. The ganglion cells are activated when there is a contrast between the center area and its surrounding. The second effect is the contrast between checks. Inside the perceived shadow area, square B is surrounded by darker checks, so it appears to be much brighter. In the light area, square A is surrounded by brighter checks, so it looks much darker. As a whole, square B looks lighter than square A.

On the other hand, we often cannot distinguish two colors if their difference is small. Statistically, in only half of the trials or less, we are able to tell two colors apart if their difference

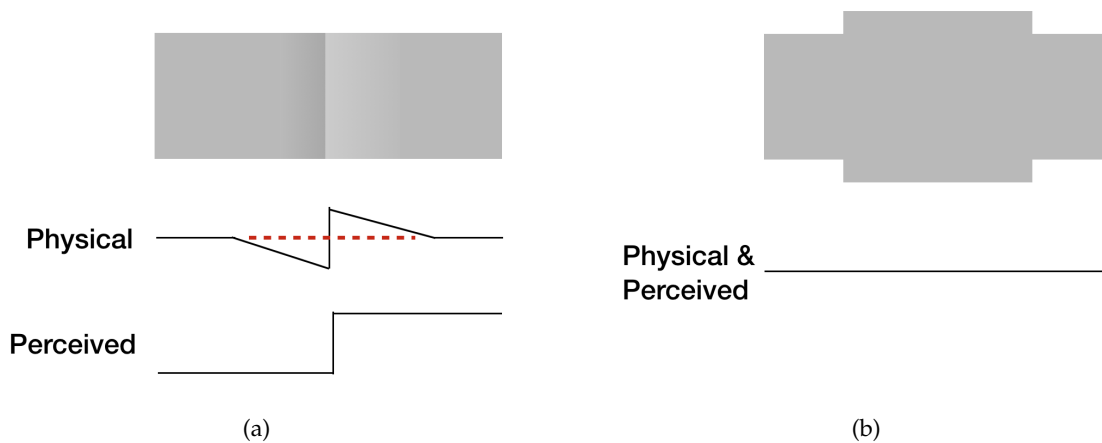


Figure 2.20: Cornsweet illusion. In 2.20(a), we perceive that the right side is brighter than the left side due to the high contrast between them but the truth is they have the same brightness (2.20(b))

is below a just-noticeable difference (JND) threshold. Studies on JND show that there are physically different signals that we cannot distinguish reliably if they are too similar.

Inspired by visual perception, we develop an intrinsic imaging evaluation metric that accounts for the perceptual color difference but also avoids illusions due to strong shadow at the same time. The metric evaluates the errors in reflectance results using color differences between pairs of neighboring pixels. Compared to existing metrics, our proposed perception inspired metric yields fairer judgments for intrinsic imaging methods. The details of the work are presented in Chapter 3.

In Chapter 4, we investigate color composition similarity that goes beyond pixel-based or patch-based. Given a pair of images, the similarity is measured by their global color compositions, including color distributions, colors of foreground objects and backgrounds, and the overall tone. It is challenging due to the complexity of natural images, and evaluating the color similarity on the level of the whole image is very subjective. That is why this type of similarity is classified as perceptual similarity.

We solve the problem by developing an active learning framework for collecting meaningful data for user experiments. It results in a large-scale coherent dataset for learning and evaluating color composition similarity. Using the dataset, we train DCNNs for color composition similarity metrics and extract global color descriptors. Combining color and category similarities, we create a new model for visual similarity that produces better performance in fine-grained image retrieval.

We also apply our deep metric for color composition similarity in color and style transfer. Results of existing neural style transfer techniques [GEB16; Gat+17] are highly perceptual and subjective. However, our neural color transfer method, which uses the correlation between mid-level features extracted from our deep CNN model trained for color composition similarity, produces good results that are faithful to the content of the source image. The method’s details and many example results are shown in Chapter 4.

Declaration for Chapter 3 - Pixel Based Perception Metric for Intrinsic Imaging

This work was done in collaboration with Dr. Shida Beigpour and published at the British Machine Vision Conference (BMVC) in 2016: "Multi-view Multi-illuminant Intrinsic Dataset" [Bei+16]. The research was partially funded by the German Research Foundation (DFG) as part of the research training group GRK 1564 "Imaging New Modalities", supervised by Prof. Dr. Andreas Kolb and Prof. Dr. Volker Blanz.

In this collaboration, Dr. Shida Beigpour worked on creating a new intrinsic dataset. I worked on developing a new perceptual metric for evaluating intrinsic imaging methods. My work in the paper includes developing the algorithms, coding, evaluating, and writing parts of the paper related to the metric and its evaluation. While we collaborated closely on the paper, each of us took full responsibility on our part.

Chapter 3

Pixel-Based Perception-Inspired Metric for Intrinsic Imaging

The discrete colors in a computer are represented by integers from 0-255 in three channels: red, green, and blue (RGB). By increasing or decreasing each of these values by 1, we change the color, and that can be easily identified or recognized by the computer. However, in most cases, humans cannot differentiate two colors if their RGB value difference is small. We can only discriminate between two colors if their color difference is above a threshold called Just Noticeable Difference (JND). In the field of developing perception metrics for Computer Vision applications, it is important that these metrics reflect the perceptual characteristics of human vision. Understanding these components has helped us to develop a useful perceptual metric to evaluate intrinsic imaging methods.

3.1 Intrinsic Imaging

Intrinsic imaging is a process of decomposing an image into its intrinsic components, such as reflectance, shading, and specularities. It is a fundamental research problem in Computer Vision and an important technique for rendering in Computer Graphics.

3.1.1 Intrinsic Imaging Formulation

Intrinsic imaging was introduced by Grosse *et al.* [Gro+09] based on the compact form of the image formation model, which is widely accepted in this field as the following:

$$I(\mathbf{x}) = S(\mathbf{x})R(\mathbf{x}) + C(\mathbf{x}) , \quad (3.1)$$

where S , R , and C are the Shading, Reflectance, and Specularity components of the image I , \mathbf{x} indicates pixel coordinates. In many algorithms and datasets, for simplicity, the Specularity component C is often omitted (Figure 3.1(a)).

Under different lighting conditions, the Shading S and Specularity C of the object surfaces are changed accordingly. However, the Reflectance R is invariant to illumination changes. It

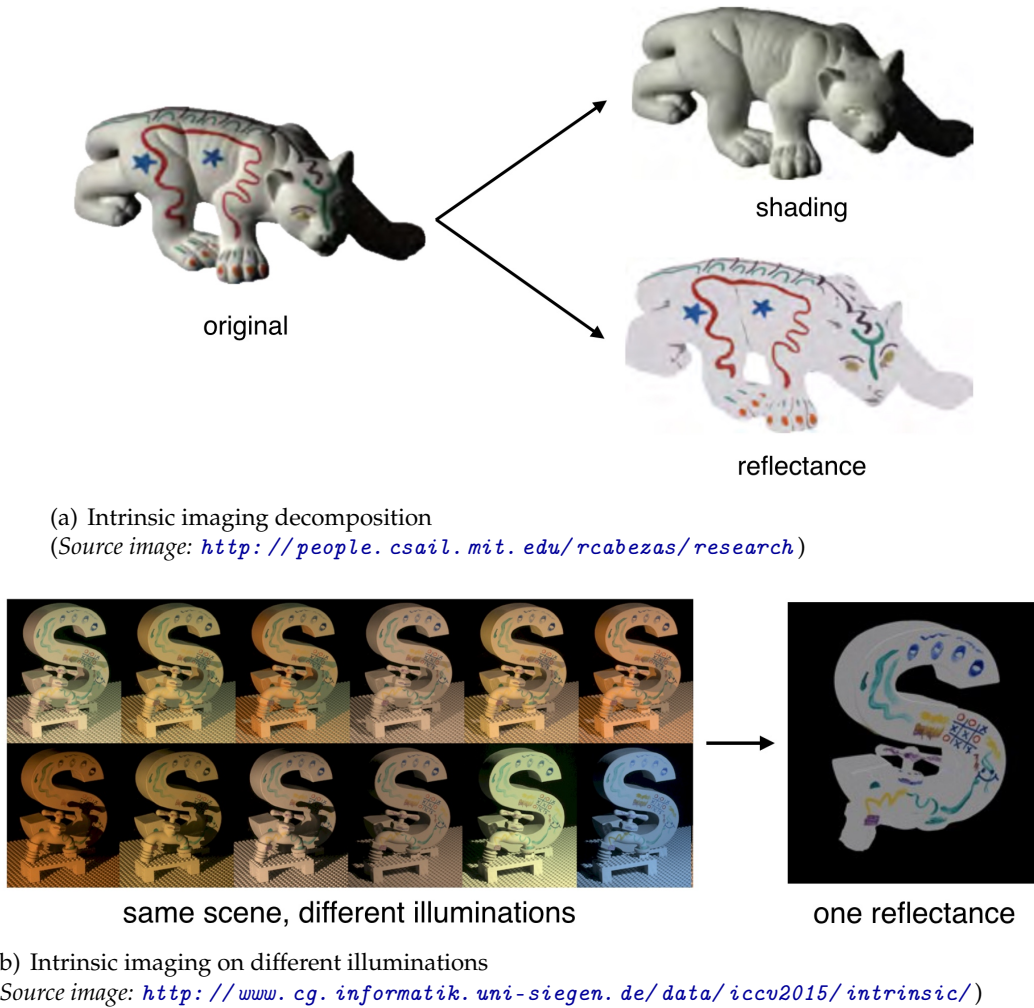


Figure 3.1: An example of intrinsic imaging. In 3.1(a), an RGB image is decomposed into Shading and Reflectance components. The Reflectance represents the color of the object’s surfaces. The Shading is created by illuminations. In 3.1(b), different illuminations will result different Shadings but the Reflectance is unchanged due to the intrinsic color property of the objects’ surfaces.

is important to reproduce sensible Reflectance R as it reflects the surfaces’ actual colors that play a crucial role in realistic scene rendering (Figure 3.1(b)).

3.1.2 Multi-View Multi-Illuminant Intrinsic Dataset

Datasets with ground-truths are needed to evaluate intrinsic imaging methods. They can be created synthetically [BM13a; Bei+13; But+12] or from real images [BKK15; BBS14; Gro+09; TAF06]. One quest in our paper [Bei+16] is to create a novel high-resolution multi-view dataset of complex multi-illuminant scenes with precise pixel-wise reflectance and shading ground-truth. The dataset contains five scenes (Figure 3.2). Each scene was captured by six cameras to create six different views with high resolution (5208×3476) images. The scenes were created with different colored objects and textured surfaces. There were 20 different illumination conditions, including single, multiple, and specular illuminants. The dataset is known to be high quality and more extensive than existing real-world intrinsic datasets. We



Figure 3.2: Multi-View Multi-Illuminant Dataset: the five scenes (#1 to #5 from left to right) captured by one of the six cameras.

will use this dataset and our proposed perception-inspired metric based on the reflectance consistency to evaluate different intrinsic imaging methods in Section 3.3.

3.2 Existing Evaluation Metrics

The reflectance component is the intrinsic property of the objects' surface and does not change due to different illuminations. For scene reconstruction under various lighting conditions tasks, such as scene re-lighting, a high-quality reflectance estimation is an important factor for a realistic scene rendering. Perceptual inconsistencies in the reflectance caused by effects of illuminations such as cast shadows will show up in the rendered image. That leads to the need to have good perceptual metrics to evaluate the quality of estimated reflectances.

Intrinsic image methods often use the Mean Squared Error (MSE) or Local Mean Squared Error (LMSE) metric to evaluate the results. As Grosse *et al.* stated in [Gro+09], MSE is a strict metric. A large error in just one small area can result in a high error for the entire image. The LMSE tries to average out the errors across the whole image by computing the MSE and estimating the scale factor for individual patches [Gro+09]. Hence, the global consistency of the evaluation is not enforced. Bell *et al.* introduced a weighted human disagreement rate (WHDR) metric [BBS14]. The evaluation is based on human judgment on individual pairs of points without ground-truth. Bell *et al.* stated that the WHDR has a margin of error of 7.5%. This could potentially result in the metric being less discriminative for methods with similar performance.

In order to overcome the limitation of existing metrics [BBS14; Gro+09], we propose a new perception-inspired error metric, which is based on *CIE Lab* color space and the standard visual color difference measurement CIE DE2000 [LCR01; SWD05], to evaluate reflectance results generated from intrinsic image methods against the ground-truth. The proposed metric measures the perceptual error of the estimated reflectance without any human subjective input and can be easily and automatically calculated for any new dataset. It can also prevent evaluation error due to human visual illusion, which might occur in the case of shadows and lighting changes [OCS05]. Therefore it is more reliable than the previous human-based intrinsic image metric by Bell *et al.* [BBS14]. Our proposed metric is described in the Section 3.3.

3.3 Point-wise Consistency Metric (PCM)

Our main inspiration in designing a new perceptual metric is rooted in the observation that the quality of an intrinsic image method's result is proportional to its *reflectance consistency* with respect to the ground-truth throughout different illuminations, strong shadows, and specularities. Here, our notion of reflectance consistency conforms to the following principles:

- Firstly, if two points p and q are perceptually similar in the ground-truth reflectance, they should also be similar in the estimated reflectance.
- Secondly, the brightness difference between a pair of points in the estimated reflectance should be similar to the ground-truth for the same pair.

Figure 3.3 shows an example of two different estimated reflectances from two different methods compared to the ground-truth. It is worth noting that the colors from method B's result are more similar to the ground-truth by the overall look than method A's result. However, in intrinsic imaging, it is acceptable that the estimated reflectance is different from the ground-truth reflectance by a constant magnitude. This can be seen from the ill-posed problem in Equation 3.1. Therefore, it is not meaningful to evaluate reflectance results by comparing the absolute values of the results against the ground-truth. Instead, we should evaluate the consistency in the reflectance results with respect to the ground-truth. This consistency can be easily broken due to shadow, shading, and specularities.

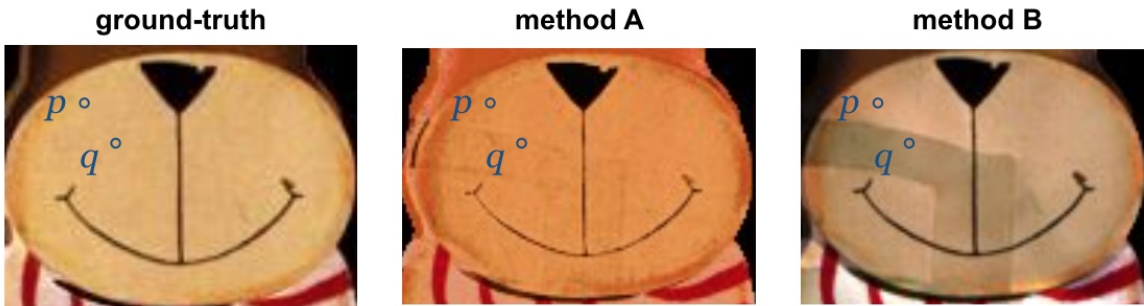


Figure 3.3: An example of comparing the reflectance results of method A and method B to the ground-truth using point-wise consistency principles. Both the results from method A and method B have artifacts caused by the cast shadow from another object. However, the cast shadow from method B is stronger than A. Therefore, visually, the result from method A is better than the result of method B. This evaluation can be formulated by comparing the results at the pair of points (p, q) to the ground-truth. The difference in colors and brightness between point p and point q in method A is closer to the ground-truth than in method B. The final evaluation result is the average error from many pairs of points (p, q) sampled on the entire scene.

Given a selected pair of points (p, q) that are perceptually similar in the ground-truth, the PCM method will evaluate the values of both methods at these two points. Due to cast shadows from the scene, both method A and B have artifacts in their reflectance results. In this example, point q is in the cast shadow region whereas point p is not. The colors and brightness of p and q in method B's result are much more different than in method A's result. Therefore, method B's result is less consistent with the ground-truth compared to method A. The PCM is proposed based on these consistency principles for color and

brightness. A complete PCM algorithm will evaluate the average error from many pairs of points (p, q) across the entire scene. This example also shows the importance of metrics to evaluate the consistency in intrinsic imaging. When we re-render the scene in different lighting conditions, this shadow artifact from method B yields an unrealistic scene with wrong shadows compared to method A.

3.3.1 Overview of the Algorithm

The algorithm for the Point-wise Consistency Metric (PCM) is summarized in the Algorithm 1 below:

Algorithm 1 Algorithm for the Point-wise Consistency Metric (PCM)

Input: a scene image I , a ground-truth reflectance G and an estimated reflectance T

Output: an average PCM error

- 1: Create a mask M for region of interest from the scene image I
 - 2: Sample a set of pairs of points in the region of interest M
 - 3: **for** each pair of points **do**
 - 4: Compute the PCM error using their colors in the ground-truth reflectance G and the estimated reflectance T
 - 5: **end for**
 - 6: Compute the average of PCM errors from all the selected pairs of points.
 - 7: Return the average PCM error.
-

3.3.2 Point-wise Consistency Error

Given the ground-truth reflectance image G and the estimated reflectance image T , our *point-wise consistency metric* works as follows:

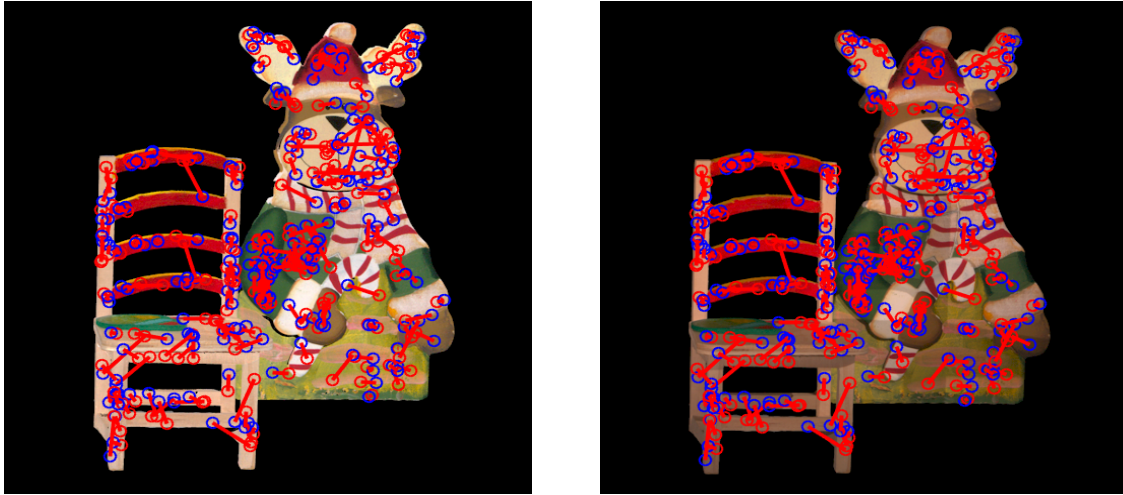
1. Select a set of point pairs $\mathcal{S} = \{(p, q)\}$ in G that are *perceptually similar*.
2. Compute the *point-wise consistency error* $PCE(G, T)$ as follows:

$$PCE(G, T) = \frac{1}{|\mathcal{S}|} \sum_{(p, q) \in \mathcal{S}} f(pce_{G, T}(p, q)), \quad (3.2)$$

$$f(x) = \begin{cases} 1, & x > \sigma \\ \frac{x}{\sigma}, & otherwise \end{cases}, \quad (3.3)$$

$$pce_{G, T}(p, q) = \Delta E_{00}(G(p) - G(q), T(p) - T(q)), \quad (3.4)$$

where ΔE_{00} denotes CIE DE2000 color distance [LCR01; SWD05] in the *Lab* color space. In the *Lab* space, channel L represents the brightness whereas a and b are two color channels. By using the *Lab* color space in the metric, both color and brightness are taken into consideration. $pce_{G, T}(p, q)$ is the difference between the similarity of the points in the ground-truth and the estimated reflectance for a pair of points (p, q) . $G(p)$ and $G(q)$ are the colors in the ground-truth G of point p and q respectively. $T(p)$ and $T(q)$ are the estimated reflectances of point p and q respectively. All the colors are in the *Lab* color space. The differences



(a) Selected pair points on ground-truth reflectance

(b) Selected pair points on estimated reflectance under white illumination

Figure 3.4: An illustration of 200 pairs of points selection with the colour difference threshold $\epsilon = 10$ and the mean distance between point pairs $\mu_d = 20$

$G(p) - G(q)$ and $T(p) - T(q)$ are vector differences. f is a linear function to normalize $pce_{G,T}(p, q)$ to $[0, 1]$ with a cut-off threshold σ . The value of σ is set such that there are about 10% of point pairs (p, q) or less which have $pce_{G,T}(p, q) > \sigma$ for all the evaluated methods. We do not want to have many error values normalized to 1 because it will make the evaluation between different methods less distinctive and less accurate.

3.3.3 Point Selection Strategy

We randomly select pairs of perceptually similar points in the ground-truth image G . For every pair, the points are selected in the region of interest and need to be perceptually similar in colors, taking both chroma and luminance into account. Furthermore, the distance between two points in a pair also follows a normal distribution.

We define the region of interest to be on object surfaces, at properly lit pixels and not on objects' contours or edges. We build the mask M , forming the region of interest as below:

$$M = \text{ero}(M_{\text{scene}} \cap \overline{M_{\text{under-exp}}}) \cap \text{ero}(\overline{M_{\text{edges}}}) , \quad (3.5)$$

where M_{scene} , $M_{\text{under-exp}}$, and M_{edges} are the masks for the scene, the under exposed pixels, and the edges inside the object, respectively. M_{scene} is created manually to mask out the background of the scene. $M_{\text{under-exp}}$ is produced automatically by setting the threshold for under-exposed pixels. M_{edges} is the edge map computed using the Canny edge detection method. The erosions make sure that selected points are not close to the contours or edges.

We select point pairs (p, q) by first randomly select p within the region of interest M . Point q is selected also inside M such that pixel distances in x and y axes from q to p follow a normal distribution with standard deviation σ_d . To do this, a random (x_q, y_q) coordinate for point

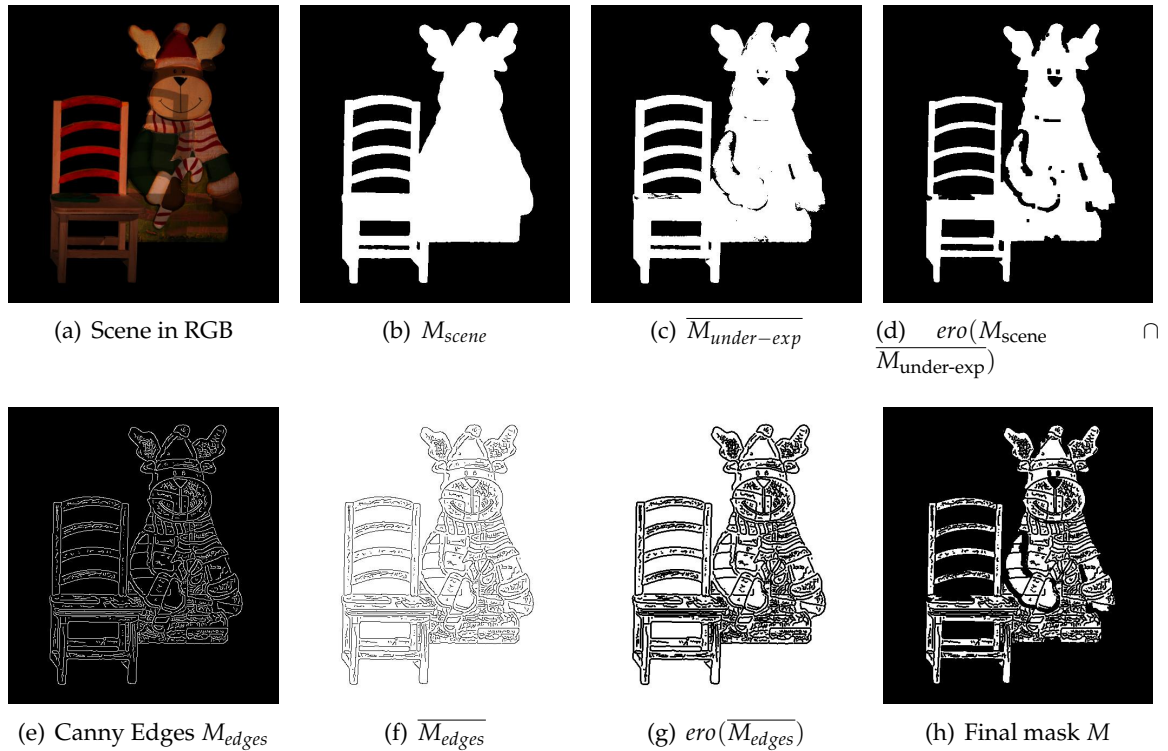


Figure 3.5: An example of masks derived for scene #5. 3.5(a) is the RGB image of scene #5. The mask for objects in the scene is manually created in 3.5(b). 3.5(c) is created to eliminate under-exposed pixels. The under-exposed pixels are those whose sum values of three R, G, B channels are below a threshold of 20. The combination of the object mask in 3.5(b) and the none under-exposed mask in 3.5(c) followed by erosion is shown in 3.5(d). The erosion uses a disk structuring element with size 3. 3.5(e) shows the Canny edge detection result of the scene. The edge map's negation is shown in 3.5(f), followed by the erosion shown in 3.5(g). This erosion uses a disk structure element of size 1. Combining 3.5(d) and 3.5(g) results in the final mask M in 3.5(h).

q is generated follow a normal distribution of which the mean is the location of p and the standard deviation is σ_d . We choose σ_d to be 20 for the experiments. We then check if this point q is inside the map M . Furthermore, we only accept perceptually similar point pairs, i.e.

$$\Delta E_{00}(G(p), G(q)) < \epsilon, \quad (3.6)$$

where ϵ is set to 10 as we observe that two colors are perceptually similar if their ΔE_{00} is less than 10 (Figure 3.4).

Notes on CIE color measures

- We use ΔE_{00} to measure the similarity between the two selected points in Eq. (3.6) because ΔE_{00} offers the perceptual uniformity by correcting problems with blue colors and also improves the performance on gray colors [LCR01]. It also takes into account the notion of a just-noticeable difference. The point selection criteria set the constraint for the first principle of the reflectance consistency to be fulfilled.
- To compute the pair-wise consistency error, we utilize ΔE_{00} creatively in Eq. (3.4).

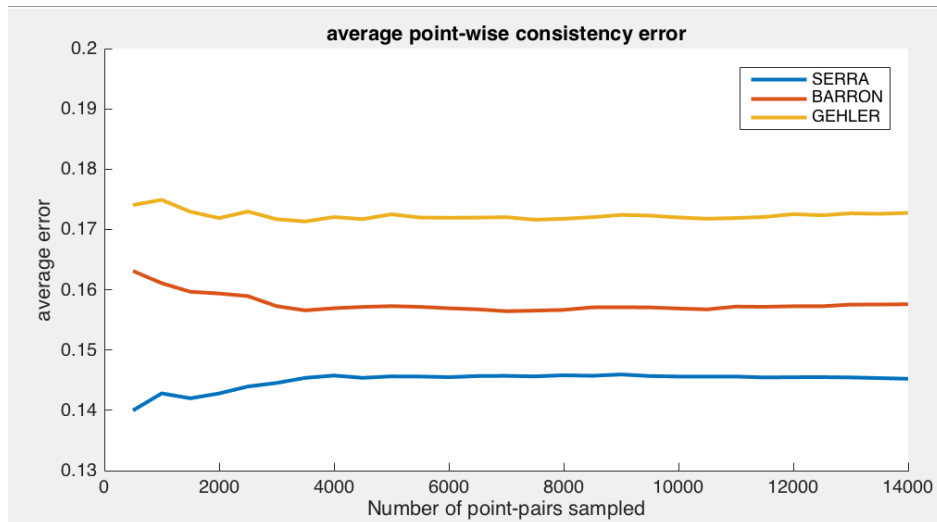
One can think of using $\Delta E(G(p), G(q))$, $\Delta E(T(p), T(q))$, and compute the difference between them. However, $\Delta E(G(p), G(q))$ is used as a criterion to choose the pair of points (p, q) that are perceptually similar. Hence, the values of $\Delta E(G(p), G(q))$ are usually small. It leads to the difference in $\Delta E(G(p), G(q))$ and $\Delta E(T(p), T(q))$ closes to 0.

- We use ΔE_{00} with *Lab* color space which has a separate channel *L* for brightness and two channel *a* and *b* for colors. Therefore, the error function $pce_{G,T}(p, q)$ in Eq. (3.4) accounts for both brightness and color differences. When combining $pce_{G,T}(p, q)$ in Eq. (3.4) with the point selection criteria in Eq. (3.6), our reflectance consistency principles hold.
- ΔE_{00} is not a real distance metric. Even though it is positive and symmetric, it does not always hold the triangular inequality criteria. For example, $x = (50, 2, -4)$, $y = (50, 1, 1)$, and $z = (50, 1, 4)$ are three colors in the *Lab* color space. We have $\Delta E_{00}(x, y) = 4.74$, $\Delta E_{00}(y, z) = 2.73$, and $\Delta E_{00}(x, z) = 7.52$. Hence, $\Delta E_{00}(x, y) + \Delta E_{00}(y, z) = 7.47$. Therefore, $\Delta E_{00}(x, y) + \Delta E_{00}(y, z) < \Delta E_{00}(x, z)$, which breaks the triangular inequality. Since PCM uses ΔE_{00} to compute consistency errors, PCM is not a distance metric but rather just a perceptual error metric.

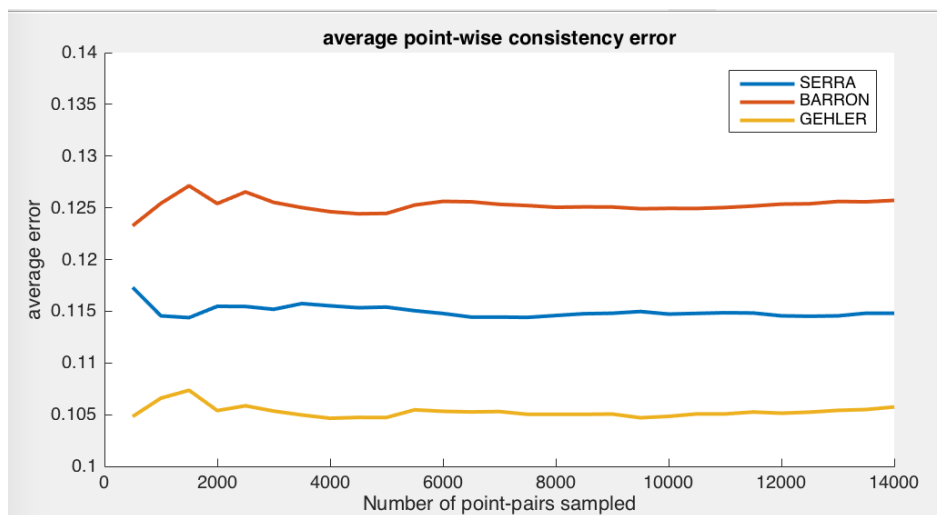
3.3.4 Sampling And Results

In our experiment, we scale the images in the dataset to 1042×696 pixels to reduce the reflectance reconstruction time for different evaluated methods. We first start by sampling 500 pairs of points and increase 500 pairs for each iteration. The maximum number of sampling is 16,000 pairs for an average of 120,000 pixels in the regions of interest *M* in our dataset. For *p* percentile inliers selection, we choose $p = 95$. In other words, we eliminate 5% of the highest errors. When the number of samplings is large, this 5% elimination guarantees there are no outliers due to noises. However, we still have enough number of samplings to evaluate the reflectance between different methods fairly. The results analysis in the Fig. 3.6 shows that the metric is able to distinct well the performance of different methods (Serra *et al.* [Ser+12] vs. Barron and Malik [BM12] vs. Gehler *et al.* [Geh+11]) on different scenes with different illuminations and is stable when the number of pairs increases to 8,000 and beyond.

Our point-wise consistency metric (PCM) evaluates the reflectance reconstruction of different methods based on the reflectance consistency principles. The principles are inspired by human perception but, at the same time, avoid visual illusion caused by strong shadows. The PCM can be extended to evaluate reflectance at pairs of points that are perceptually different. It is also possible to develop the error estimation for shading based on this perceptual and statistical approach.



(a) Scene #5 with green and orange illuminant



(b) Scene #2 with white and yellow illuminant

Figure 3.6: The average point-wise consistency error of scene #5 (a) and scene #2 (b) with the colour difference threshold $\epsilon = 10$ and the mean distance between pair points $\mu_d = 20$. The evaluations are on three different methods: Serra *et al.* [Ser+12], Barron and Malik [BM12] and Gehler *et al.* [Geh+11]. PCM can discriminate the performances of these methods well and stable when the number of sample pairs increases.

3.3.5 Evaluation

Table 3.1 presents an evaluation of three intrinsic image methods, namely: Barron and Malik [BM12], Gehler *et al.* [Geh+11] and Serra *et al.* [Ser+12] using their publicly available codes and default parameters on a subset of our dataset (i.e., images captured by one of the six cameras for all the scenes and illuminations)¹. We further group our illumination conditions into four categories, i.e., easy (whitish), moderately colored, hard (strongly colored), and specular based on their complexity level. Evaluation is performed using PMC and LMSE.

¹Due to high computation time of the evaluated methods, we scale our images to 20% of their original resolution.

As PCM only evaluates reflectance data, we restrict LMSE in the same way in order to get comparable results.

Evaluation	Method	easy	moderate	hard	specular	Total
PCM	Barron <i>et al.</i>	0.149	0.151	0.157	0.157	0.154
	Gehler <i>et al.</i>	0.151	0.154	0.158	0.164	0.157
	Serra <i>et al.</i>	0.123	0.125	0.125	0.126	0.125
LMSE (reflectance)	Barron <i>et al.</i>	0.305	0.383	0.485	0.387	0.403
	Gehler <i>et al.</i>	0.277	0.341	0.441	0.336	0.360
	Serra <i>et al.</i>	0.253	0.300	0.319	0.289	0.296

Table 3.1: Evaluation results of the methods Barron and Malik [BM12], Gehler *et al.* [Geh+11] and Serra *et al.* [Ser+12] using Point-wise Consistent Metric PCM and Local Mean Squared Error (LMSE).

To give further insight into the two measures, we pick a sequence of 20 different illuminations for a fixed camera pose for scene #5 (Fig. 3.2, the right most). Fig. 3.7 compares the evaluation based on PCM and LMSE and shows the results of all three methods Barron and Malik [BM12], Gehler *et al.* [Geh+11], and Serra *et al.* [Ser+12]. PCM delivers consistent results ranking the methods’ performance over all 20 illumination conditions. LMSE, on the other hand, shows large variations across different illuminations. Specifically, with LMSE, Serra performs the best in many lighting conditions but also the worst in some others such as in illumination #4, #5, #6, and #17.

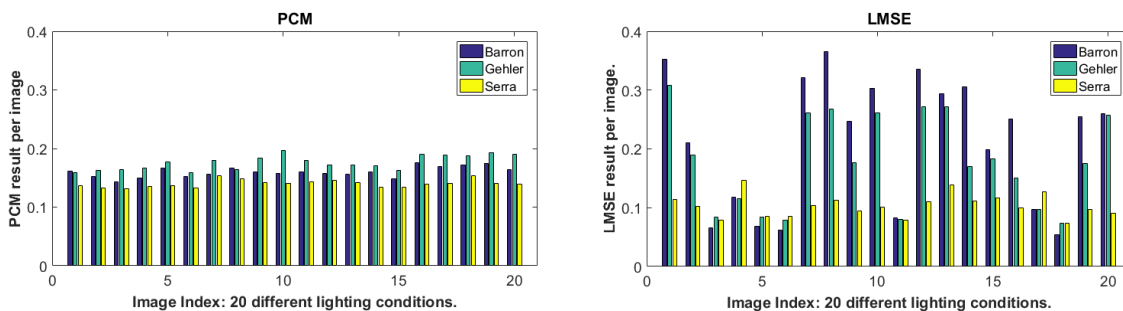


Figure 3.7: PCM and LMSE results on scene #5 under 20 different illuminations.



Figure 3.8: Reflectance results for the three methods on scene #5, 4th illumination.

Consulting LMSE for the 4th illumination, Barron and Gehler are similar, and Serra is worse, while for PCM Serra is better than Barron and both better than Gehler. Visually on Fig. 3.8, all three methods deliver imperfect results for this example, but Gehler’s methods appear to yield stronger color artifacts, e.g., on the bear’s sleeve for which PCM accounts.

The consistent evaluation results from PCM are important because that means PCM provides higher confidence in an intrinsic imaging method's performance. In applications where there are only one or few lighting images for a scene, and we need to pick an intrinsic method that works well in general, knowing that PCM has more consistent evaluation than LMSE, it will be a logical choice to use PCM to select the best intrinsic imaging method for the application.

3.3.6 Chapter Summary

In this chapter, my research contribution is a new perceptually inspired metric to evaluate the intrinsic methods' results based on the reflectance consistency principle. The metric measures the point-wise consistency between local pairs of points, thus the name Point-wise Consistency Metric (PCM). To facilitate the perceptual color difference, we use the CIE DE2000 metric on the *Lab* color space [LCR01; SWD05]. We evaluate three state-of-the-art intrinsic methods Barron and Malik [BM12], Gehler *et al.* [Geh+11] and Serra *et al.* [Ser+12] on our dataset using LMSE and our proposed PCM metrics. Compared to the popularly used LMSE metric, PCM evaluation is more stable across different illumination conditions and more faithful to the visual appearance of intrinsic results. We believe that our dataset and the PCM metric can help to improve the quality of intrinsic imaging methods in complex scenes.

Declaration for Chapter 4 - Perceptual Color Composition Similarity

This work was in collaboration with Dr. Vlad Hosu at the University of Konstanz and published at the Winter Conference on Applications of Computer Vision (WACV) Conference in 2020 "Color Composition Similarity and Its Application in Fine-grained Similarity" [HHB20]. The research was funded by the German Research Foundation (DFG) as part of the research training group GRK 1564 "Imaging New Modalities" and part of Project-ID 251654672 – TRR 161 (Project A05). This work was supervised by Prof. Dr. Volker Blanz.

In this collaboration, Dr. Vlad Hosu helped with technical set up and advised on using a crowd-sourcing platform. We did the user experiments on www.crowdfunder.com (now appen.com). In the fine-grained image retrieval, I worked closely with Dr. Hosu on developing techniques to combine color and object category. Dr. Hosu implemented the joint features and SVM for classification, and wrote the corresponding part of the paper. My contributions to this work are:

- Designing the active learning framework.
- Manually selecting data at the beginning together with M.Sc. Roberto Cespi.
- Designing different crowd-sourcing experiments and test questions.
- Analyzing crowd-sourcing data.
- Training and testing different CNN models for color composition similarity.
- Running experiments with color descriptors and comparisons with the existing color descriptors.
- Developing the entire work on color and style transfer.
- Leading the writing of the paper and the corresponding parts mentioned above.

Chapter 4

Perceptual Color Composition Similarity

Visual similarity is a long-standing research problem that has not been studied thoroughly. Its challenges come from the ambiguity in the problem definition as well as the subjective evaluation due to individual human perception. There are many factors that contribute to the overall visual similarity evaluation, such as object categories, image composition, color layout, image style, and so on (see Figure 4.1 for examples).

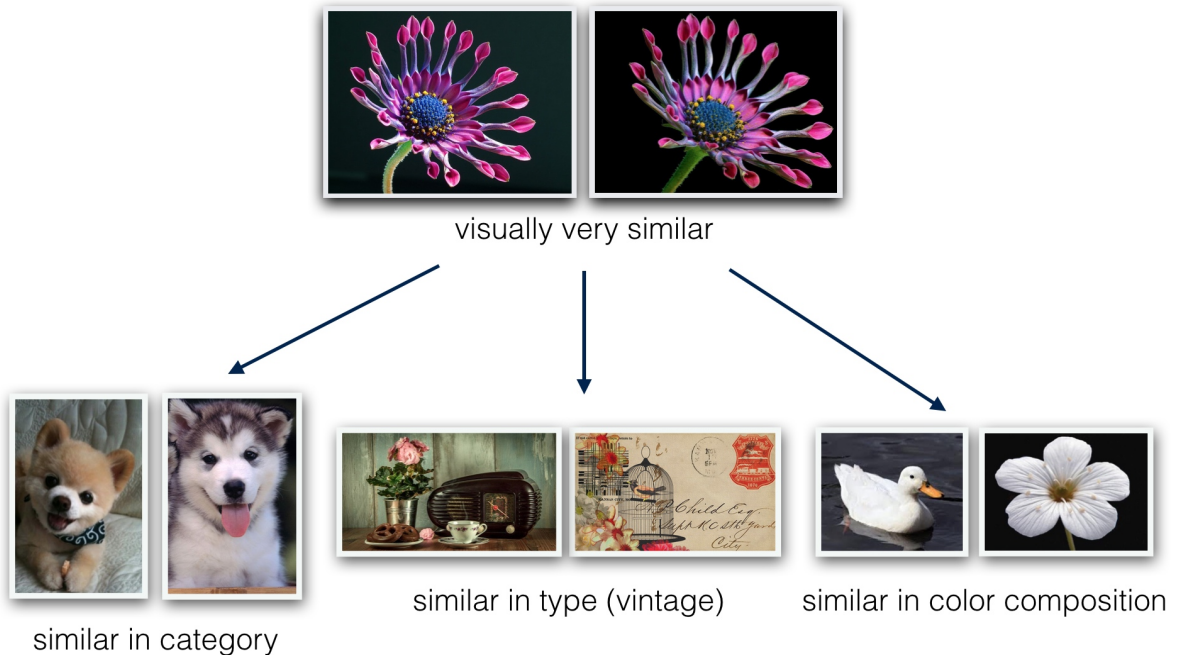


Figure 4.1: Different aspects of visual similarity. Visual similarity is challenging and subjective. It can be seen as visually very similar, or it can be understood as similar in different aspects such as categories, image type, color composition, etc.

This chapter aims to study the fundamental problem of visual similarity and propose novel ways to reduce its ambiguity in order to create better predictive models. We break down visual similarity into sub-problems (category and color similarity), finding a way to collect meaningful training data, and developing metrics and descriptors for color similarity. In

contrast to existing approaches, we study visual color similarity “in-the-wild”, which goes beyond pixel-based or patch-based approaches. For specific colors, we have a good standard to measure the perceptual similarity that is established via the CIE ΔE_{2000} metric [LCR01; SWD05]. The metric indicates how similar two colors are, based on the ability of people to distinguish individual colors. However, multiple colors interact in complex ways in natural images. Existing metrics like CIE ΔE^* and hand-crafted color descriptors [AF06; BZM08; BG09; Geu+01; Kha+13; LS13; Man+01; SGS10; WGB06] are not able to accurately predict color composition similarity.

While recent methods learn visual features for image search and visual similarity [BB15; Che+10; PM15; Wan+14; WKH17], they lack a dataset built directly from human judgments on color similarity for training and validation. These methods are trained on datasets labeled with object categories such as ImageNet [Den+09] or Pascal [Eve+10]. Another set of approaches try to separate aspects of perception by discovering and learning visual attributes for image search and retrieval [DRS11; FZ07; KG13; PG11; Ras+13; SFD11; Yan+16]. The information they rely on involves textual description, attribute labeling, and supervised learning on attribute labels. Attributes simplify the objectives of visual similarity by mapping a full range of perception into a discrete set of textual descriptions. Our approach is to ask participants to visually compare and rate images directly without having to use less accurate means of assessment such as textual descriptions.

As stated in [YG14], fine-grained similarity comparisons (including color) are critical for building perceptually accurate models. However, it is not easy to measure the color similarity for images in-the-wild due to the high complexity of natural images and the subjectivity of perceptual judgments. Therefore, we introduce a new way to define visual color similarity, as color composition, and study it directly via human evaluations. The color composition assessment emphasizes hues and shades, their distributions, and overall layout, independent of the semantic category.



Figure 4.2: Different rating examples for color composition similarity. The ratings are on a 5-point scale where 1 is the least similar, and 5 is the most similar to almost identical.

We create a dataset annotated with 5-point similarity ratings for color composition (Figure 4.2). Our dataset contrasts with other image similarity datasets that often rely on binary labels such as INRIA Holidays [JDS08] or the triplets dataset [Wan+14]. While the dataset

empowers us to embed the image color composition similarity according to human perception, the participants' opinion distributions show the subjectivity of the evaluation and comparison processes.

One of the challenges for building a fine-grained similarity rating large-scale dataset is the cold start problem that arises from the very low probability in obtaining similar image pairs if we were to sample them randomly. We overcome this by using an *active learning approach* and iterating from binary to fine-grained ratings. We also account for many measures to ensure the quality of the dataset. As a result, we contribute a large-scale (31,248 image pairs with at least 20 ratings each), high quality (ICC of 0.69, very high for crowd-sourcing), and novel dataset for color composition similarity. The Intra-class Correlation Coefficient (ICC) measures the degree to which raters resemble each other in the whole experiment and is explained in details in Section 4.2.4. Our dataset is the first annotated dataset of its kind up to date, to the best of our knowledge¹.

Using the dataset, we train a Siamese network to predict the distributions and mean opinion scores. The network serves as a metric and a feature extractor for color composition similarity. We compare performances and create a benchmark for existing color descriptors and our trained color features in the field of color similarity for images in-the-wild. Trained global features using CNNs produce a very good performance (0.913 SROCC, Spearman correlation w.r.t the ground-truth). Even though $L1$ and $L2$ measuring on existing hand-crafted local descriptors with dense samplings yield lower performances, it is promising to train these descriptors to capture global features of color composition, leading to better performances (the best case is 0.862 SROCC with HueSIFT).

Furthermore, we validate our color features and metrics in a fine-grained similarity application (Section 4.5). Color had been previously modeled implicitly together with category. We propose a novel approach to combine category features via pair-wise correlations and color similarity as predicted from our models. These combined features are extracted from pairs of images leading to improvements in accuracy compared to learning on individual content or color features alone. Training an SVM using our proposed features on a small dataset yields better accuracy than the state of the art. Compared to a common baseline, the best existing method DeepRanking [Wan+14] trained on millions of images achieves a relative improvement of 3.5%. Our model, trained on less than 50K images in total, improves by a much higher margin of 12.5%. Despite using three orders of magnitude less training data, the absolute improvement of our method on different validation sets is still better than the state of the art, 86.2% for ours vs. their 85.7%.

The importance of perceptual color similarity is also reflected in its applicability in Computer Vision and Computer Graphics. We successfully apply the color composition similarity models in another domain: neural style transfer (Section 4.6.2). Our models provide a new option to generate “stylized” images for which the detail style and colors are taken

¹dataset download link: <https://github.com/hamailan/Color-Composition-Similarity>

from separate sources. The color transfer results produced by our deep metrics are faithful to the source images.

In summary, our contributions in this chapter are:

- A general framework for modeling sub-aspects of image similarity, that is designed to handle highly subjective measurements via crowd-sourcing and active learning.
- The first large-scale perceptual color composition similarity dataset in-the-wild with 5-point ratings collected directly from human judgments.
- A global color similarity benchmark for color descriptors.
- A new type of brief but highly generalizing features for fine-grained similarity: the concatenation of the correlation of category features and color similarity extracted from pairs of images. Triplet ranking using SVM on these features surpasses the state of the art even when trained on a much smaller dataset.
- A new direction for neural style transfer in which colors are transferred from a third input image, in addition to textures being taken from a reference (second input) image. Thanks to our color similarity metric, we obtain good color transfer results.

4.1 Related Work For Perceptual Color Similarity

There are two groups of measurement methods that compare the colors between two images: the first group uses hand-crafted features, and the second one learns features. These color features are known as color descriptors.

4.1.1 Hand-crafted Features for Color Similarity

From the famous SIFT descriptor [Low04] that describes local features for a set of interesting points in an image by histograms of gradient orientations, different extensions of SIFT are derived for color descriptors [AF06; BZM08; BG09; Geu+01; WGB06]. A common objective of these descriptors is to be robust against changes in lighting, scale, rotation, and so on. A complete evaluation of these SIFT variational color descriptors can be found in [SGS10]. Another set of color descriptors, introduced in the MPEG-7 standard [Man+01], relies on transformations to various color spaces. Color descriptors are often designed to be compact for fast indexing [LS13], or to maintain a level of photometric invariance [Kha+13]. However, all these descriptors operate locally on low-level image features and therefore lack the ability to capture global color information.

4.1.2 Learned Features for Visual Similarity

For complex natural images, it is challenging for hand-crafted descriptors to perform well. Deep Convolutional Neural Networks (DCNN) have been successfully applied to image similarity. One type of methods learns similarity metrics for pairs of images using pairwise

similarity data [BB15; Che+10; YG14; Zha+18]. Another approach uses triplet data where a reference image is paired with a positive and a negative example [Wan+14; WKH17]. In either case, pairwise image similarity is labeled by category, attributes, or binary classification. When labeling by category, colors are ignored. Binary classes often relate to generic visual similarity rather than specifically to colors. With attribute learning, the visual attributes are expressed in terms of textual descriptions [DRS11; FZ07; KG13; PG11; Ras+13; SFD11; Yan+16] and therefore over-simplify the objectives of visual and color similarity. In this work, we aim to develop better metrics for color similarity that can also provide color features that are beneficial for many Computer Vision applications.

4.1.3 Datasets for Perceptual Similarity

In order to train or validate perceptual similarity metrics and descriptors, we need to have datasets that are assessed by people as ground-truth. However, there is no fine-grained rating dataset for perceptual similarity for images in-the-wild. All existing datasets are either classification or coarse in the level of similarity or not completely in-the-wild comparison. INRIA Holidays [JDS08] is a dataset where similar images are grouped together, and dissimilar images are assigned to different groups. Another type is a triplet dataset [Wan+14] that provides a coarse level of similarity. Recently published, the BAPPS dataset [Zha+18] contains natural images and their generated distortions for learning perceptual patch similarity. In this work, we fill in the missing gap by contributing a perceptual color similarity in-the-wild dataset for which the similarity is measured by participants' ratings on color composition using a fine-grained 5-point scale rating. Our dataset is the largest human-annotated dataset of its kind up to date, to the best of our knowledge.

4.2 Process to Define Perceptual Color Composition Similarity

The complexity of color composition on natural images makes it extremely challenging to write down a set of rules or formulae to define the perceptual color composition similarity. If we need to give a verbal definition, the similarity criteria that we are aiming at are the layout of colors, color distribution, dominant colors, and the overall perceptual appearance of colors in the images. Instead of relying on such a description, our approach is to capture this definition directly via human judgments. Given a pair of reference and test images, participants rate the degree to which the pair is similar with respect to colors exclusively. We face two challenges. The first is selecting images for which ratings for color similarity make sense. Statistically, numbers of pairs that are different in color compositions are much higher than similar pairs. The second is to convey an unambiguous definition of color composition similarity to participants so that they can understand and provide useful and reliable ratings. Unlike other types of annotations, it is not easy to describe the degrees of similarity.

Our solution to the first problem is to build the dataset in two stages. In the first stage, we create a small binary dataset in which similarity is clearly defined: either very similar

or completely dissimilar. We start with the least subjective data. It is possible to collect a small set of pairs or groups of similar images by using INRIA Holidays [JDS08] dataset and Pixabay images (<https://pixabay.com/>). From this starting binary dataset, we train a small binary classification network (*binary-net*) to identify similar/dissimilar labels for pairs of images. We then use the *binary-net* to sample more image pairs and have them annotated by participants as similar or dissimilar. The network performance is further improved with the extended set of annotated pairs. We name the improved binary network as *improved-binary-net*. The details of the first stage development are described in Section 4.2.1.

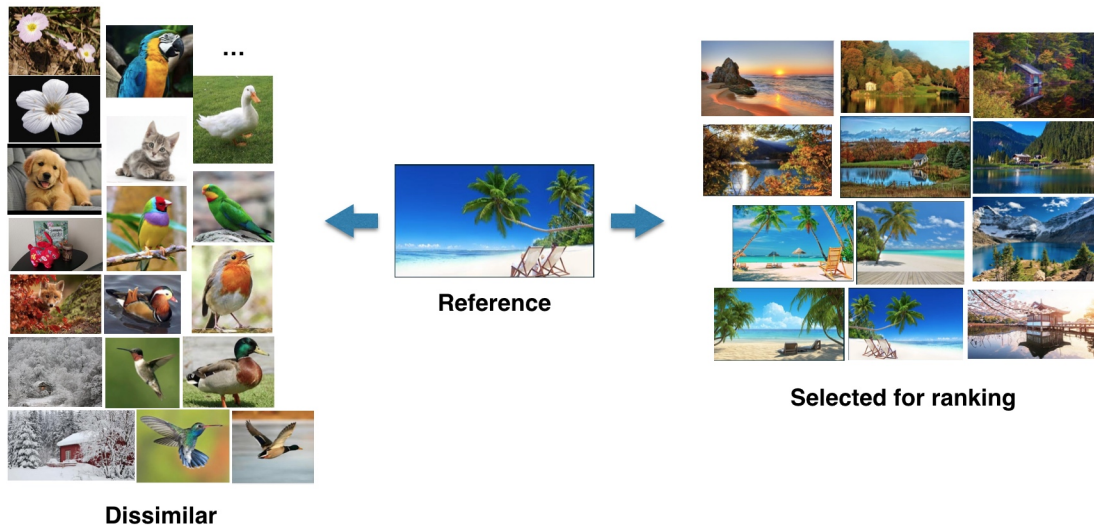


Figure 4.3: Binary classifying images into two groups. One contains dissimilar images. The other contains similar images in general, which could be similar in color or category. The images in the second group can be used for ranking the similarity measurements in terms of color composition.

In the second stage, we use the *improved-binary-net* to select images for the rating dataset. We ask participants to evaluate similarity on a finer-grained 5-point scale where 1 means a pair is completely different, and 5 means the images are very similar or almost identical. It is important to choose evaluated images such that the ratings are present for all five options. We describe the detailed strategy in Section 4.2.2.

To solve the second problem, we ask participants to consider several cues that help them consistently compare pairs of images, such as the presence of dominant colors, distribution of colors, colors of foreground objects and the background, and the overall perceptual appearance of colors in the whole image. We quantify the rating from 1 to 5 as follows: 1 - the pair of images are totally different, 2 - below 50% similar colors, 3 - about 50% similar colors, 4 - above 50% similar colors and 5 - very similar to identical (e.g., Figure 4.4). We present many rating examples (Figure 4.6), conduct an entrance test before participants can start working on the project, and embed hidden test questions seamlessly into work items. We made a pool of 150 test questions in total. Five questions are randomly selected from the pool to make up a quiz for an individual participant. The test pairs and their expected ratings serve as ground truth to assess whether participants' rating criteria are consistent

with the task's requirements. In the test cases, to allow room for subjectivity, for very similar pairs, we set the candidates for correct ratings to $\{4, 5\}$. For pairs that are absolutely different, the correct rating candidates are set to $\{1, 2\}$. For non-extreme similarities, ratings of $\{2, 3, 4\}$ are allowed (Figure 4.7). Participants must pass the entrance test and maintain their accuracy above 70% throughout the study.

We have 1.781 people who participated in the experiment. Among those, 1.470 passed the entrance test (82.54%) to be qualified for the experiment. In the working process, 1.337 participants maintained their accuracy above 70% of the hidden test questions, and hence their ratings were valid. This is 90.95% of the qualified participants that passed the entrance test, showing who passed the entrance test understood the tasks well and were able to perform consistently. To get a variety of participants, we limited each participant to work on a maximum of 100 tasks. Each task contained a group of 8 pair ratings for 1 reference image (for example, Figure 4.7 is 1 task). There were 837 participants maxed out the task (62.6% of participants that have valid results). The statistic is shown in Figure 4.5. Finally, the quality of our rating dataset is evaluated in Section 4.2.4.

Rating Scale

The similarity in color composition of an evaluated image to its reference image can be rated as:

1. **Very different:** neither the image colors, nor their layouts is similar at all.
2. **Substantially different:** less than 50% of the colors are similar, and their layouts are a bit similar.
3. **Fairly similar:** about 50% of the colors are similar and their layouts are similar.
4. **Substantially similar:** more than 50% of the colors are similar and their layouts are similar.
5. **Very similar:** image colors and their layouts are mostly the same.

Examples of Rating Scale



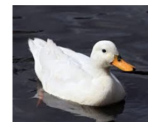
Image A



Reference Image

score of image A = 5

Reason: even though image A and the reference image contain different objects, they are very similar in terms of color composition: white object in the middle of a dark background.



Reference Image



Image B

score of image B = 3

Reason: even though image B and the reference image both contain a white duck, the background colors are very different. Each duck covers about 50% of the image, therefore around 50% of the colors are similar.

Figure 4.4: Instructions and examples for crowd-source participants to rate color composition similarity. The example to emphasize the similarity is measure in *color composition* aspect and not the category aspect that most participants might misunderstand.

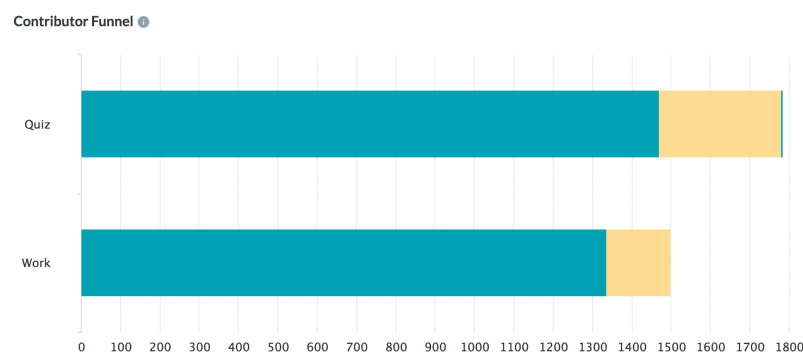


Figure 4.5: Statistic on passing rate of the participants.



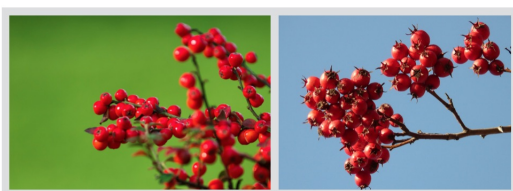
(a) Score 5: very similar to identical



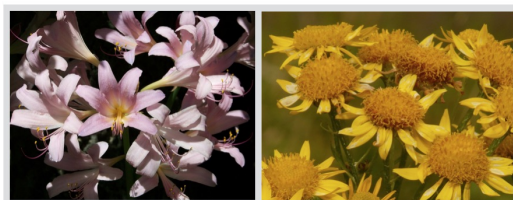
(b) Score 4: above 50% similar colors



(c) Score 3: about 50% similar colors

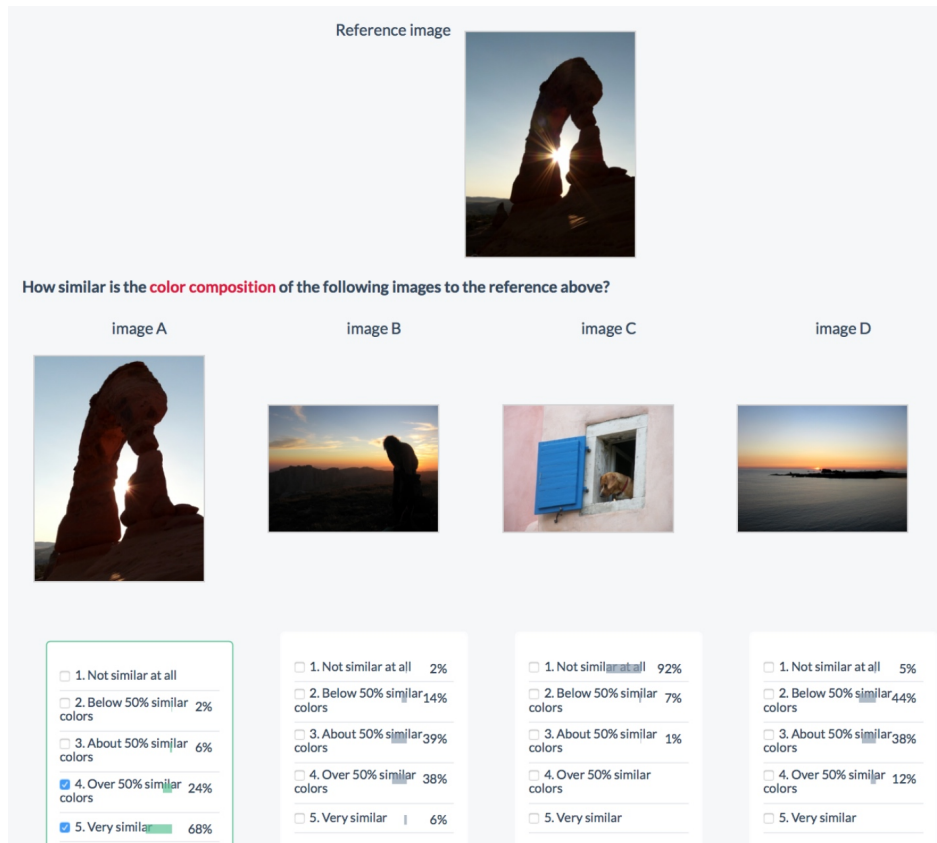


(d) Score 2: below 50% similar colors

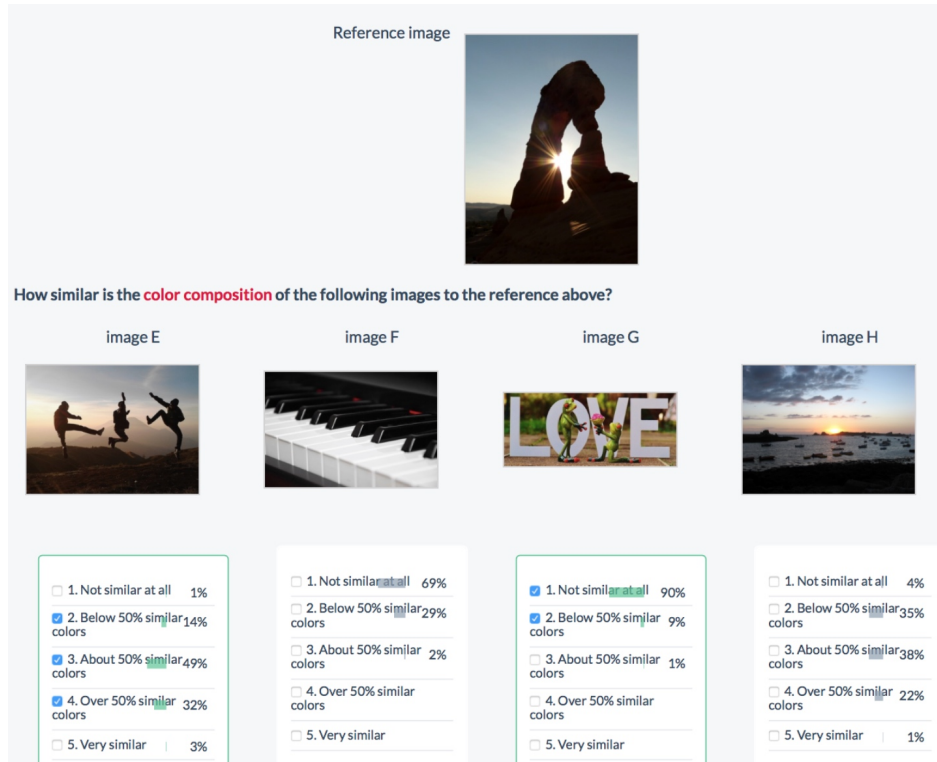


(e) Score 1: the images in each pair are totally different

Figure 4.6: Examples for each similarity score (two pairs), shown in the introduction of the crowd-sourcing experiment.



(a) Scores 4,5 for very similar images



(b) Scores 1,2 for dissimilar images and 2,3,4 for similar images

Figure 4.7: Different settings for test questions. The tick boxes are options set in advance. The percentages are the rating from participants. Those images with ticked boxes are controlled questions. Majority (around 95% on average) of the participants have the correct answers.

4.2.1 Binary Dataset and Network

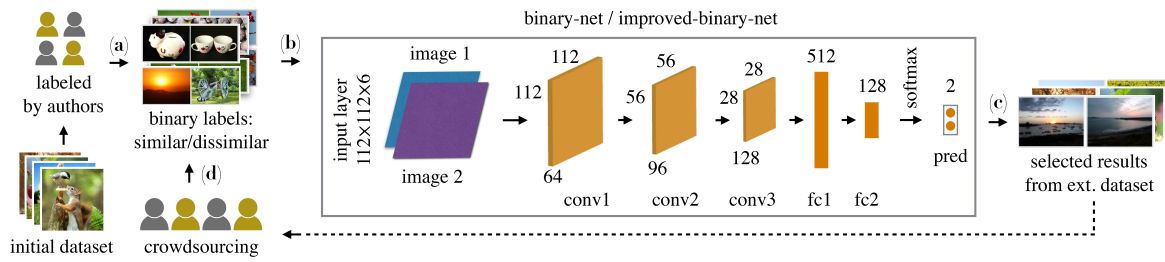


Figure 4.8: Our proposed active learning framework for building color composition similarity binary dataset: it starts with hand-picked similar image pairs (a), on which a classifier is trained (b) to select more similar image pairs (c), which in turn are annotated for similar or dissimilar by crowd-sourcing participants. The process is repeated by using accumulated user annotated data.

We combine the images from INRIA Holidays [JDS08] and Pixabay (<https://pixabay.com/>) datasets to create our own dataset. In the INRIA Holidays dataset, similar images are grouped together. Similar images are those of the same scene with a slight change in view or zoom (Figure 4.9(a)). There are usually two to five images per group. With the Pixabay dataset, similar images can be distributed widely in the dataset, but there are also cases where similar images are clustered together by filenames (Figure 4.9(b)). However, images with similar filenames do not always look similar. Even though the INRIA Holidays dataset is more convenient than Pixabay for choosing similar images, it has only more than one thousand images, whereas Pixabay has millions of images. Therefore, we need to have a good strategy to select similar images.

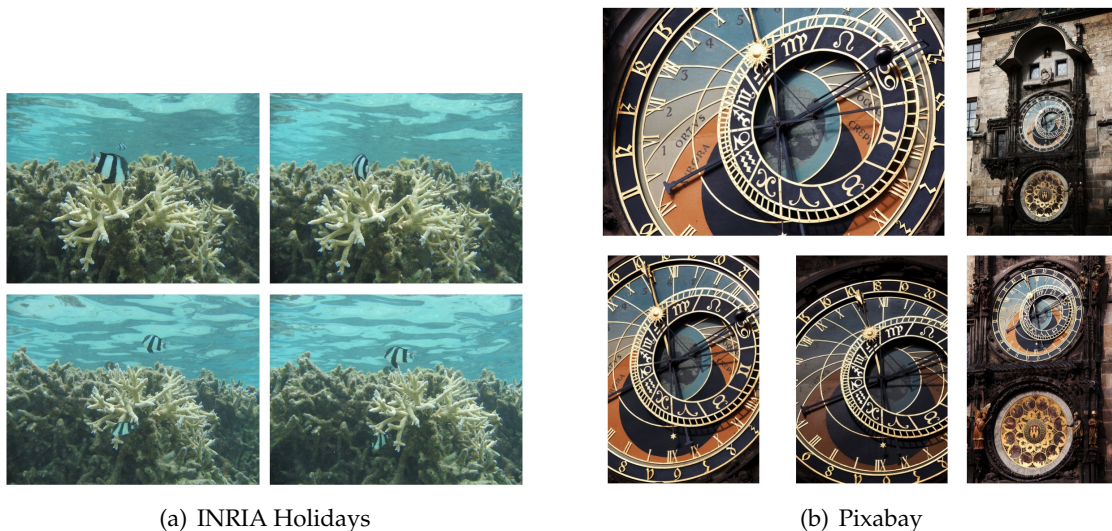


Figure 4.9: Examples of similar images from INRIA Holidays [JDS08] dataset in (a) and Pixabay <https://pixabay.com/> dataset in (b). Similar images in INRIA Holidays are grouped by filename. In contrast, images in Pixabay are not pre-grouped. Therefore, we need to have a good strategy to select similar images.

We use an active learning approach to improve the binary network and expand the dataset (Fig. 4.8). The process starts with an equal number of 3,591 labels each for similar and dissimilar image pairs. This small set of labels are manually annotated by M.Sc. Roberto

Cespi and me (Fig. 4.8(a)) and are used to train the initial binary network named *binary-net* to classify similar or dissimilar images in term of color composition (Fig. 4.8(b)). We chose pairs images such that there is the least subjectivity possible. If two images are considered dissimilar, they should contain almost no color in common. If two images are similar, they contain the same set of colors and layout, and they often contain the same objects as well (Figure 4.10).



Figure 4.10: Similar vs dissimilar pairs of images.

Due to the limited amount of training data, we design a CNN architecture with few parameters. Instead of using a Siamese model, we stack pairs of RGB images into six-channel inputs. We augment the images by horizontal flips, small rotations, and swap the two inputs. By swapping image 1 and image 2, we can double the training data and, therefore, increase the network's accuracy. The output of the network is softmax scores for two classes: similar and dissimilar.

In the next step, we generate data for participants' evaluation on new pairs of images for binary classification using the initial binary network *binary-net* (Fig. 4.8(c)). We select 1,302 reference images that cover a wide variety of objects, textures, and scenes. We use *binary-net* to evaluate the binary similarity between each reference image against a set of 3,000 images from our large pool dataset. The results from the *binary-net* are then sorted from the most similar to the most different based on their similarity scores.

For each reference image, only the first few dozen images are similar, and the majority of images are different. Therefore, we select only 24 evaluated images per reference for participants to evaluate similar or dissimilar (Figure 4.11). These 24 images consist of 1 highly similar image from the initial set of 3,591 labels that are manually selected at the beginning, the first 20 images resulted from the *binary-net* and 3 dissimilar images that are taken randomly at the end of the *binary-net* result list. It yields 31,248 pairs of comparisons in total.

Finally, the participants' evaluations are added to the binary dataset (Fig. 4.8(d)) and fed to re-train the initial *binary-net* to increase its accuracy (Fig. 4.8(b)) up to 98.9%. This re-trained network is called *improved-binary-net*. An example of retrieval results for the *improved-binary-net* is presented in Figure 4.12. While the retrieval results for similar images to the reference image are visually good, the order of the retrieval can be improved and the similarity score to indicate the degree of similarity is missing. Therefore, in the next step, we use *improved-binary-net* to select images for fined ratings in Section 4.2.2.

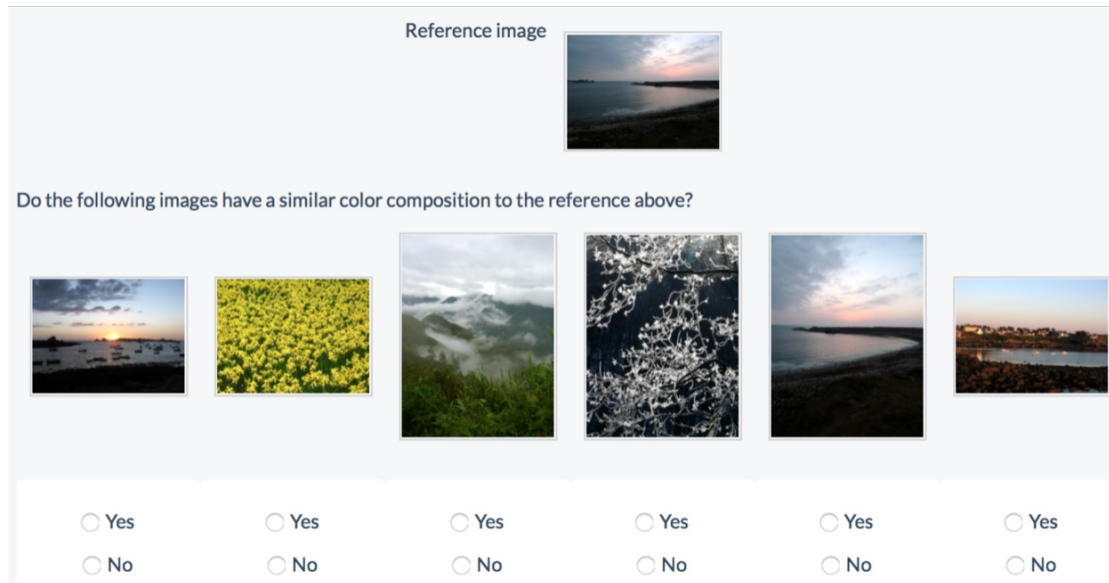


Figure 4.11: Crowdsourcing for binary evaluation: similar versus dissimilar. Users were asked to answer yes/no if an image is similar to the reference image.

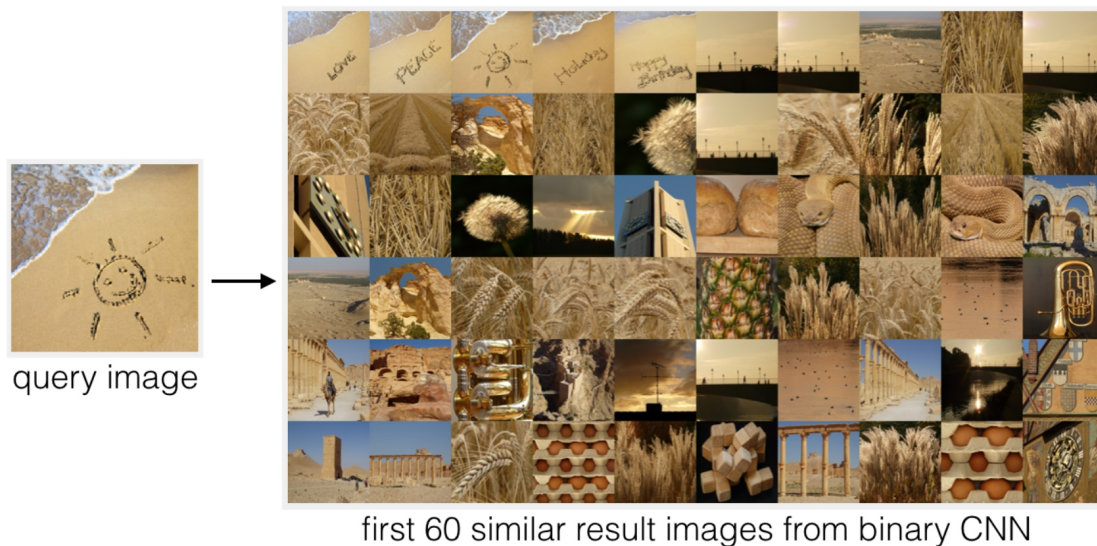


Figure 4.12: Retrieval results of the *improved-binary-net*. Even though the retrieval results for similar images to the reference image are visually good, the order of the retrieval can be improved, and the similarity score to indicate the degree of similarity is missing.


4.2.2 Rating Dataset

In the subsequent crowd-sourcing process, we create a fine-grained rating dataset from the binary set. We ask participants to evaluate the similarity for pairs of images using a 5-point Likert-type scale, ranging from absolutely dissimilar (1) to very similar or identical (5). The rating data comprises 1,302 reference images. There are 24 evaluated images for each reference. It is important to choose the evaluated images such that their ratings span the entire 5-point scale. For very similar to identical (rating 5) pairs of images, we choose pairs from the 3,591 manual labeling data. For pairs of images for which the similarity

ratings potentially range from 2 to 4, we select pairs from the top results of *improved-binary-net* sorted by similarity scores. Dissimilar pairs of images (rating 1) are accurately chosen from the bottom of the sorted *improved-binary-net* result list.

The critical factors that control the quality of the rating dataset are the rating accuracy and consistency among work items of individual participants as well as the consistency among all participants for each work item. To reduce biases and promote the coherence of participants' ratings, for every reference image, we presented to participants a group of evaluated images at a time. We asked the participants to not only rate each pair of images individually but also compare among the group of evaluated images (Figure 4.13). If an evaluated image A is more similar to the reference image R than an evaluated image B to R , then the rating for A should be higher than for B and vice versa. If both images A and B are equally similar to the reference image, then the ratings for both should be the same (Figure 4.14). This strategy provides an additional context for rating, thus helping participants to adjust their ratings to become more consistent.

Reference image



How similar is the **color composition** of the following images to the reference above?





image A	image B	image C	image D
			
<p>Image A vs Reference:</p> <input type="radio"/> 1. Not similar at all <input type="radio"/> 2. Below 50% similar colors <input type="radio"/> 3. About 50% similar colors <input type="radio"/> 4. Over 50% similar colors <input type="radio"/> 5. Very similar	<p>Image B vs Reference:</p> <input type="radio"/> 1. Not similar at all <input type="radio"/> 2. Below 50% similar colors <input type="radio"/> 3. About 50% similar colors <input type="radio"/> 4. Over 50% similar colors <input type="radio"/> 5. Very similar	<p>Image C vs Reference:</p> <input type="radio"/> 1. Not similar at all <input type="radio"/> 2. Below 50% similar colors <input type="radio"/> 3. About 50% similar colors <input type="radio"/> 4. Over 50% similar colors <input type="radio"/> 5. Very similar	<p>Image D vs Reference:</p> <input type="radio"/> 1. Not similar at all <input type="radio"/> 2. Below 50% similar colors <input type="radio"/> 3. About 50% similar colors <input type="radio"/> 4. Over 50% similar colors <input type="radio"/> 5. Very similar

Figure 4.13: Rating for color composition similarity. Images are presented in groups. Participants not only rate the similarity between a rating image to the reference image but also compare the score among the group's images.

4.2.3 Study on Rating Strategies

We did experiments for rating color similarity between pairs of images using different strategies: (a) pair comparison, (b) triplet comparison, and (c) group rating. To study the effect of ratings of paired comparisons without and with context i.e., presenting images in a group, we conducted a small experiment.

In pair comparison without extra context (Fig. 4.15(a)), we presented users with individual pairs of images (randomly selected) and asked them to rate the color composition similarity

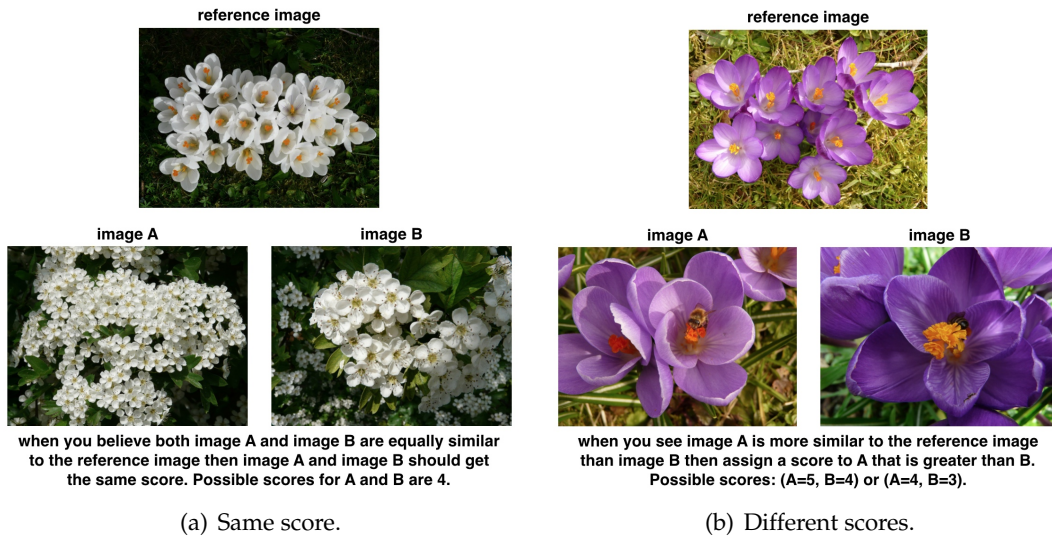
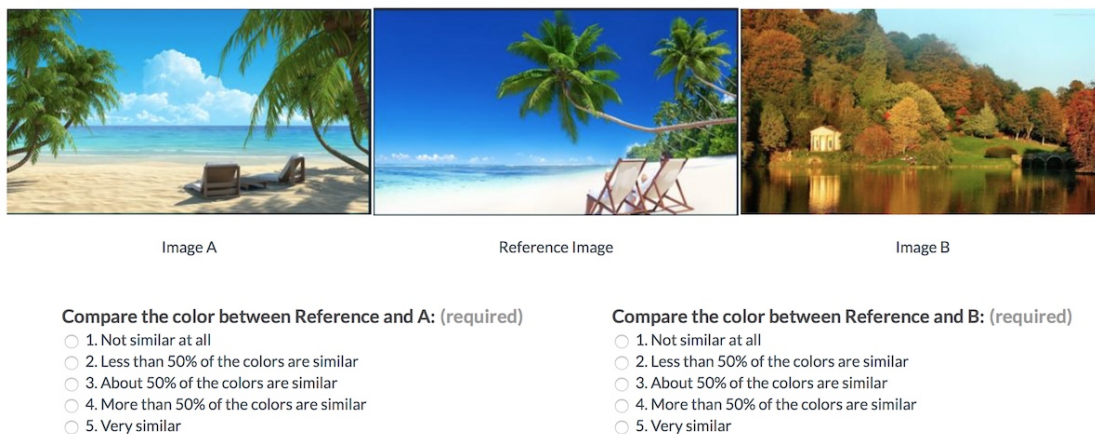


Figure 4.14: Examples of different scenarios for score comparison within a group of image.



(a) User's rating on pair comparison.



(b) User's rating on triplet comparison.

Figure 4.15: Different rating strategies: pairwise rating vs. triplet comparison and rating. In pair rating, participants are asked to rate the similarity between two images. In triplet rating, participants are required to not only rate the similarity between image A to the reference and image B to the reference, but also compare the similarity scores of A and B to the reference.

for the displayed pair. In the with context comparison (Fig. 4.15(b)), we presented users with three images (triplet comparison): a reference image R , an image A , and an image B . We asked users to rate the color composition similarity of image A to the reference R , and of image B to the reference R . Moreover, we also asked users to compare image A and B , making sure the ratings are three-way consistent such that they follow the requirement in Eq. 4.1.

$$\begin{aligned} S(R, A) > S(R, B) &\Leftrightarrow Q(R, A) > Q(R, B) \\ S(R, A) = S(R, B) &\Leftrightarrow Q(R, A) = Q(R, B) \end{aligned} \quad (4.1)$$

where S is the perceptual similarity judgment, R is the reference image, A and B are the evaluated images and Q is the rating.

If an evaluated image A is more similar to the reference image R than an image B is to R , then the rating for A should be higher than for B , and vice versa. If both images A and B are equally similar to the reference R , then their ratings should be the same (Figure 4.14).

We analyze the user agreement for both triplet comparisons and pair comparisons. Given a reference image R and an evaluated image A , the rating $\tilde{Q}(R, A)$ is a rating distribution from 1 to 5 for the color composition similarity between the reference image R and the evaluated image A from all the users. Let $H(\tilde{Q})$ be the normalized histogram of \tilde{Q} (bins sum to 1), and $H^k(\tilde{Q})$ be the normalized histogram of \tilde{Q} at bin k , $k \in [1..5]$, and $p(H)$ is the mean of $H(\tilde{Q})$. $p(H)$ is also known as Mean Opinion Score (MOS). Ideally, all users agree on one rating such that $p(H) \in [1..5]$ and $H^{p(H)}(\tilde{Q}) = 1$. However, users do not always agree with their ratings. The less diverse the user ratings are, the higher the agreement. An illustration of users' rating distribution is in Fig. 4.16.

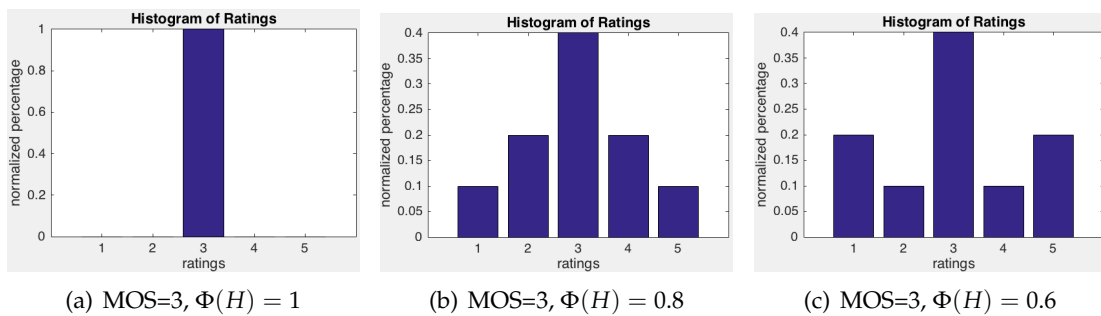


Figure 4.16: Example of different rating distributions. In all three examples, the Mean Opinion Score are the same (MOS = 3). $\Phi(H)$ is computed as $\sum_a^b H(\tilde{Q})$ where $a = 2$ and $b = 4$ which are one scale below and one scale above MOS value. In (a): 100% users agree on the rating 3 and therefore, $\Phi(H) = 1$. In (b): $\Phi(H) = 0.8$, users' ratings spread out such that 80% of users vote around the MOS. The worst case is (c) where $\Phi(H) = 0.6$, only 60% of users rate around MOS.

To measure the users' agreement $\tilde{Q}(R, A)$, we sum up the normalized histogram $H(\tilde{Q})$ around the MOS $p(H)$. The result is $\Phi(H) = \sum_a^b H(\tilde{Q})$ where $p(H) \in [a, b]$, and a and b are the floor and ceiling values of the MOS. When the MOS is an integer, a and b are chosen

to be one rating below and one above the MOS. We then compute the mean and standard deviation of $\Phi(H)$ for all pairs of rating images.

$\Phi(H)$ measures agreement, ranging from the lowest value of 0 to the highest value of 1. In Table 4.1, we show that both the paired and the triplet comparison have a very high average agreement. However, the triplet comparison has a much smaller standard deviation ($\sigma(\Phi(H))$), meaning that the user agreement is roughly the same high value for all pairs. Even the mean ($\mu(\Phi(H))$) is slightly lower for the triplet comparison than for paired comparison. We prefer an experimental methodology that gives more consistent results in the general case i.e., a high minimum agreement for any pair. This type of consistency is suggested by the much lower standard deviation in the case of triplet comparisons. The statistical result confirms that three-way comparative judgments reduce the overall subjectivity of user ratings.

	$\mu(\Phi(H))$	$\sigma(\Phi(H))$
Pairwise rating	0.983	0.204
Triplet rating	0.952	0.083

Table 4.1: Evaluation of users' agreement for pairwise and triplet ratings.

Based on the numerical results, we conclude that pairwise rating with relative comparisons between a group of images increases the users' agreement. Thus, in the rating experiment, we extend the triplet rating to group ratings where each reference image has 24 evaluated images. The 24 evaluated images are divided into six groups of four images. Users are then asked to compare each evaluated image to the reference image, making sure to consider the consistency of relationships within the group (Fig. 4.13).

For this experiment, we choose to use the standard deviation described above to measure the consistency between pair and triplet rating instead of ICC. It is because analyzing users' rating distributions using standard deviation is more intuitive and easy to understand. We use ICC to evaluate the final color composition similarity dataset to compare the quality of the dataset with other subjective study datasets.

4.2.4 Quality of the Rating Dataset

One important aspect of crowd-sourcing experiments is to have a sufficient number of participants working on each question. In highly subjective perceptual comparison tasks, we need a more significant number of user judgments per item compared to less subjective tasks such as object labeling. We conducted a preliminary experiment on a small part of the dataset (559 pairs) using 40 ratings per pair. We studied how well a smaller number of ratings can reproduce the mean of 40 ratings. We found that the mean opinion derived from 20 ratings suffices to obtain a 0.994 Pearson linear correlation with the mean for 40 ratings, with an MAE of 0.033 on a scale of [1,5] (Figure 4.17). Hence, we chose 20 ratings per pair.

We compute the histogram of Mean Opinion Scores (MOS) on the entire color similarity dataset. If we were to randomly sample pairs of images, there would be a very low chance

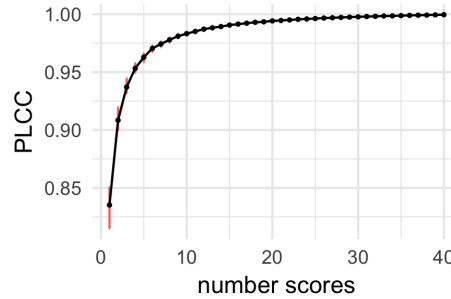


Figure 4.17: Correlation (PLCC) between the MOS from ‘number scores’ and the MOS from all 40 scores. The random repeated sub-sampling leads to confidence intervals that are displayed in red.

that two images were highly similar i.e., a score of 4 and above. In our dataset, we have about 2,500 pairs (6.4% of the dataset) with ratings of 4 and above. We also have a fair amount of pairs with MOS ratings distributed in the middle of the rating scale (See Fig. 4.18). This distribution was possible only due to our pair selection procedure employed during the active learning process.

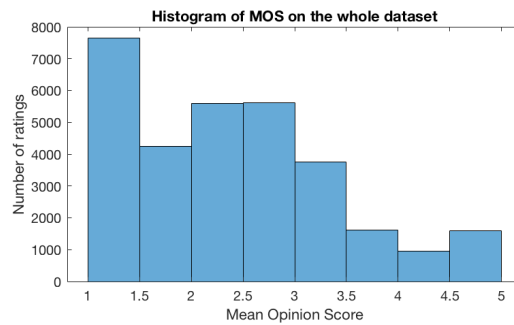


Figure 4.18: Histogram of Mean Opinion Scores for the entire color composition similarity dataset.

To evaluate the quality and reliability of the dataset, we use the Intra-class Correlation Coefficient (ICC). In subjective ratings, ICC is often used to measure inter-rater reliability. A high value of the ICC indicates a high degree of agreement between participants. There are three models for ICC [KL16b]:

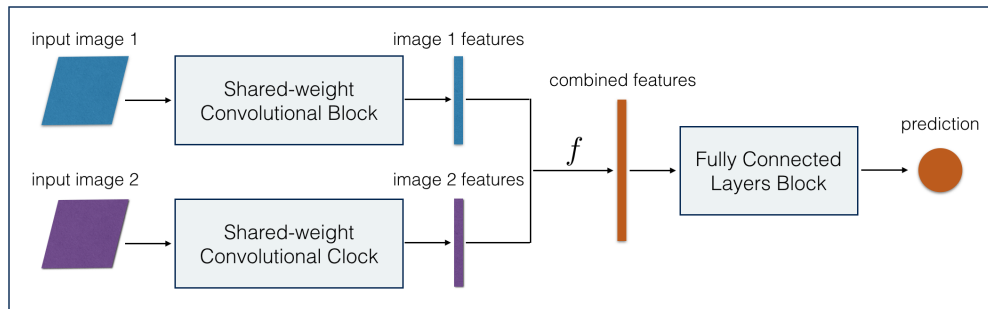
- One-way random-effects: each subject is rated by different sets of participants, which are randomly selected in a population and can be generalized.
- Two-way random-effects: each subject is rated by the same set of participants, which is randomly selected in a population and can be generalized.
- Two-way mixed-effects: only specific participants such as experts are selected, and this cannot be generalized to other participants such as a general population.

In our dataset, due to a large number of comparisons, no participant completed all the questions. Therefore, we use the one-way random-effects ICC. The one-way ICC value on our dataset is 0.69 (computed using `ICCest()` function from package `ICC` in the software R). It

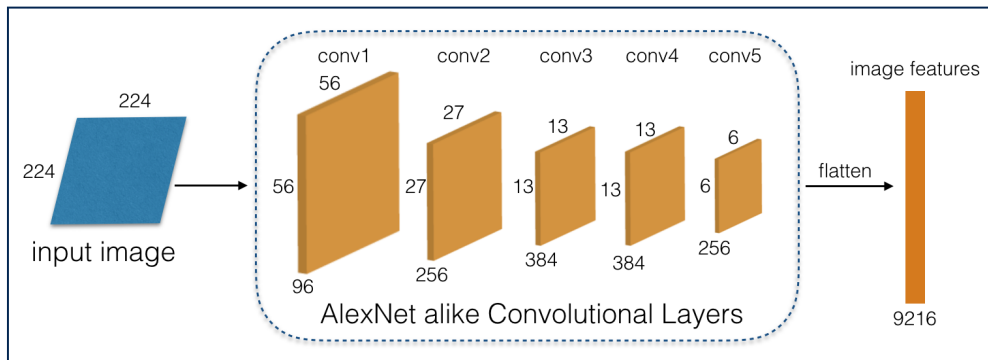
suggests a high agreement in the context of crowd-sourcing rating experiments for subjective studies, where values between 0.22 and 0.56 have been previously reported on different rating datasets [HLS18; SHR16].

4.3 Computational Model of Color Composition Similarity

We train Convolutional Neural Networks (CNN) to evaluate the perceptual color composition similarity using the rating dataset. These networks can be used as similarity metrics and color feature extractors.



(a) Siamese network for predicting ratings.



(b) Shared-weight Convolutional Block.

Figure 4.19: Siamese architecture (a) using Convolutional Neural Network (b) for training our color similarity metrics.

Different from binary networks, rating networks allow us to rank image similarity. We train two types of rating networks: **COLSIM_RATE** to predict the participants' rating distribution and **COLSIM_MOS** to predict the participants' Mean Opinion Score (MOS). Both networks use the same architecture as described in Fig. 4.19(a). The only difference is in the prediction layer, where we have a single output for MOS and five outputs for rating distributions. The overall architecture is a Siamese network that has two input images. Each input is fed into a shared-weight Convolutional Block that contains a series of convolutional layers to extract features. The features from the two input images are combined by a function f defined in Eq. 4.3. Finally, a neural network with a few fully connected layers performs the predictions based on the combined features.

Shared-weight Convolutional Block: we use five convolutional layers that are similar to the Caffe implementation of AlexNet with Batch Normalization on the first two layers. The last convolutional layer's responses are flattened to form a feature vector v (Fig. 4.19(b)). We also train a compact network that contains only three convolutional layers on images of size 112×112 pixels. The compact network is smaller and faster, but there is a slight drop in performance (Section 4.4).

Image Features Combination: to combine features of image 1 (v_1) and features of image 2 (v_2), we use 3 different metrics: the absolute difference d_a , squared difference d_s and Hadamard product d_h as follows:

$$\begin{aligned} d_a(v_1, v_2) &= \|v_1 - v_2\| \\ d_s(v_1, v_2) &= \|v_1 - v_2\|^2 \\ d_h(v_1, v_2) &= \|v_1^j \odot v_2^j\| \end{aligned} \quad (4.2)$$

where $j = 1..n$, $v_1 \in R^n$ and $v_2 \in R^n$. Let C be a concatenating operator, i.e., stacking vectors vertically. Combined features of two images are defined by function f as follows:

$$f(v_1, v_2) = C(d_a(v_1, v_2), d_s(v_1, v_2), d_h(v_1, v_2)) \quad (4.3)$$

The combined features resulting from Eq. 4.3 are used as the input to the Fully Connected Layer (FCL) block (Fig. 4.19(a)).

Fully Connected Layer (FCL) Block: comprises two fully connected layers of size 512 and 128. We use dropout 0.5 for the first FCL and 0.2 for the second FCL. ReLU activation is used throughout the whole network.

Rating Distribution Prediction: participants' ratings are distributed over the 5-point scale. Given a pair of images, we want to predict the participants' rating distribution. We use different metrics for computing the distribution losses, including Mean Absolute Error (MAE), Kullback-Leibler (KL) divergence, and Huber loss. The objective of the training is to minimize these distribution losses. From numerical results, KL divergence consistently performs the best. Thus, we use KL divergence in all of our rating networks. KL divergence loss between the participants' rating distribution (ground-truth) P and predicted rating distribution Q is computed as follows:

$$L_{KL}(P||Q) = \sum_{i=1}^n P(i) \log \frac{P(i)}{Q(i)} \quad (4.4)$$

where $n = 5$ for the 5-point rating scale.

Mean Opinion Score (MOS) Prediction: from the participants' rating distributions, we can compute MOS values that are useful for image ranking. The MOS is computed as $MOS =$

$\sum_{i=1}^n i \cdot P(i)$ where P is the normalized rating distribution and $n = 5$ for a 5-point rating scale. We also train networks that predict MOS (**COLSIM_MOS**) using Mean Squared Error (MSE) loss. Our experiments show that MOS derived from predicted rating distributions has lower errors than the results of networks that are trained directly on MOS data.

Color Features Embedding Visualization: We use t-SNE embedding to visualize the MOS prediction from our **COLSIM_MOS** network. t-SNE is a technique to visualize high dimensional data. One can provide raw data points to the t-SNE embedding function and it will calculate Euclidean distances between all pairs by default. Another option is to provide t-SNE with a pair-wise distance matrix. Here, we use the matrix of which each element is a value between each pair of images (x, y) defined as: $d(x, y) = 1 - \text{MOS}(x, y)$, $d(x, y) \in [0, 1]$. d is positive and symmetric but is not guaranteed to follow the triangle inequality. However, t-SNE is designed to work with non-metric similarities [MH12]. Fig. 4.20 and Fig. 4.21 show the t-SNE embedding of the MOS predictions on 3,000 and 5,000 images respectively. The 3,000 images are mixed selections of Pixabay and INRIA Holiday datasets. The 5,000 images are randomly selected from Pixabay. The images form clusters based on their dominant colors.

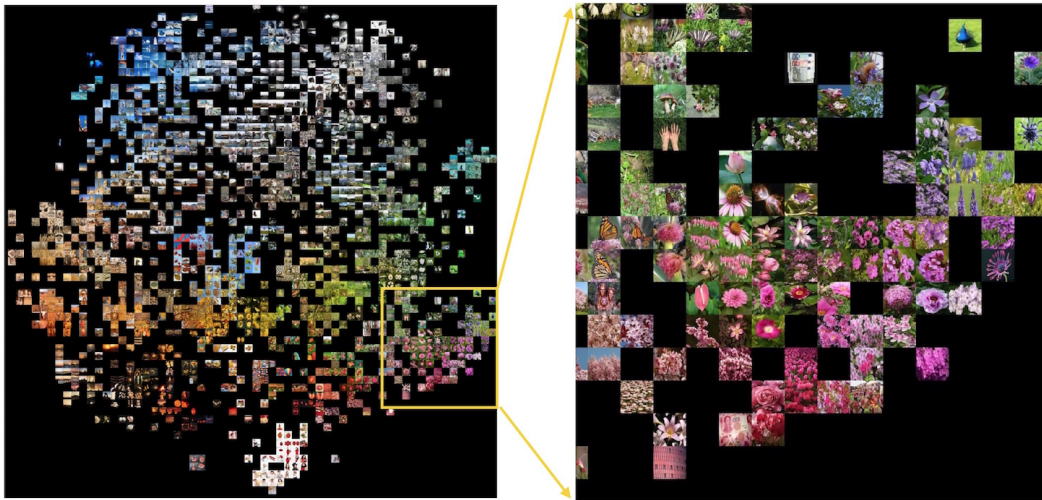


Figure 4.20: t-SNE embedding on MOS prediction from AlexBN color similarity network for 3K images. The 3K images are mixed selections of Pixabay images and INRIA Holiday dataset. The left image is the overall embedding and the right image is a zoom in of a small region from the left one.

To evaluate the color features extracted from the metric network, we visualize the L2 distances between color features from our metric in Fig. 4.22(a). For comparison, we also visualize the L2 distances of fc7 content features from the VGG19 network in Fig. 4.22(b). It shows that with content features, the colors are scattered around, such as red and green images. Whereas with color features, the images are visually grouped correctly.

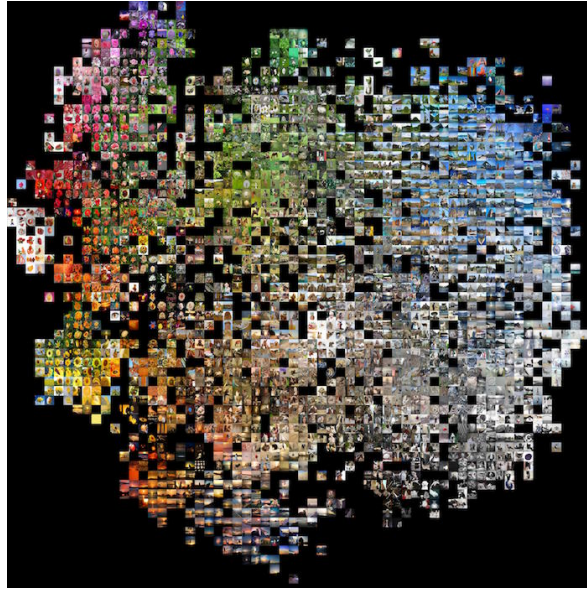


Figure 4.21: t-SNE embedding based on MOS prediction from AlexBN color similarity network for 5K images from Pixabay. The result shows that images are clustered very well based on colors. For a clear visualization, only images that do not overlap with each other are rendered.

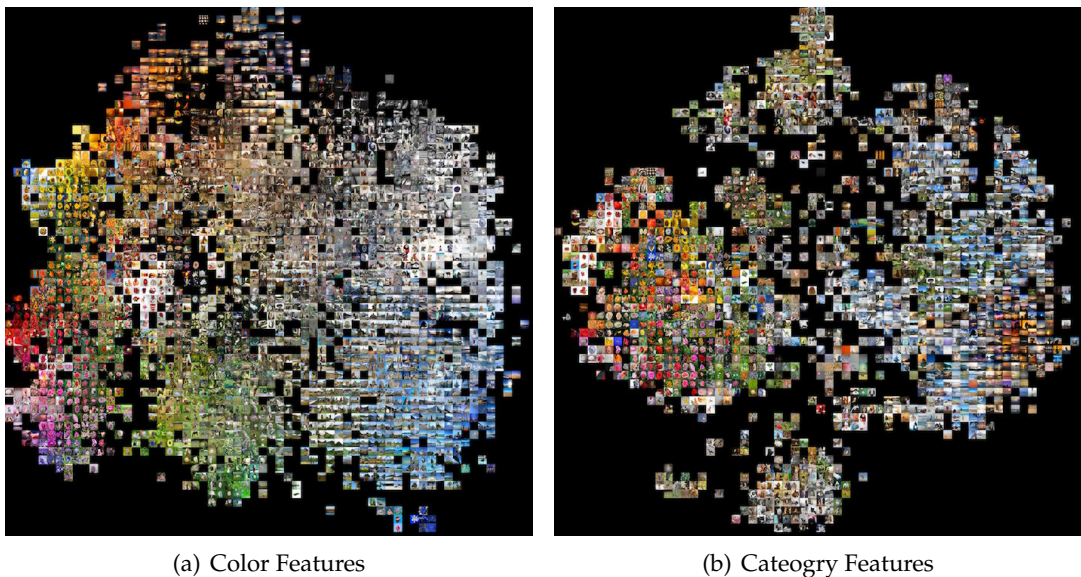


Figure 4.22: Visualization for color features versus category content features. Only images that do not overlap with each other are rendered.

Using the color composition similarity dataset, we develop different color descriptors and apply them in three applications: global color descriptor, fine-grained image retrieval, and neural style transfer with perceptual color similarity. Each application is described in the following sections.

4.4 Application 1: Global Color Descriptor

Color is one of the essential elements of visual content. Color descriptors are important to describe images and are used in many Computer Vision applications such as image retrieval, object recognition, scene understanding, to name a few. In the first application, we train deep convolutional neural networks to learn global color descriptors using our color composition similarity dataset. We compare the trained global color descriptors with existing hand-crafted color descriptors in ranking image similarity based on color composition.

In order to evaluate and compare the performances of different descriptors and networks on perceptual color similarity measurements, we split the dataset into an 80% training set (24,840 pairs) and a 20% test set (6,210 pairs). There are no common reference images in the two sets. All the algorithms are trained and validated on the training set and tested on the test set. The results reported in Table 4.2 are the SROCC on the test set, which measures the Spearman Rank Order Correlation between the predicted results and participants' ratings. We choose SROCC over other metrics such as MAE or MSE because it accounts for the changes in scale and non-linearity of the measurements coming from different descriptors and methods.

Group	Descriptor	correlation ρ			
		L1	L2	Histogram Intersection	Trained MOS/Rating
Histogram	nrgistogram	0.503	0.546	0.503	-
	opponent histogram	0.604	0.498	0.604	-
	hue histogram	0.631	0.535	0.648	-
	lab histogram	-0.260	-0.336	0.670	-
SIFT	rgsift	0.259	0.277	-	-/0.754
	hvsift	0.327	0.277	-	-/0.757
	csift	0.351	0.318	-	-/0.687
	opponentsift	0.604	0.498	-	-/0.636
	huesift	0.631	0.535	-	-/0.862
MPEG7	CLD	0.290	0.562	-	-/0.715
	CSD	0.653	0.692	-	-/0.737
	SCD	0.692	0.646	-	-/0.720
Learning	VGG19 + L2	-	-	-	0.467/-
	VGG19 Transfer	-	-	-	0.780/0.812
	VGG19 Fine-tune	-	-	-	0.832/0.863
	Compact (ours)	-	-	-	0.860/0.869
	COLSIM (ours)	-	-	-	0.902/0.913

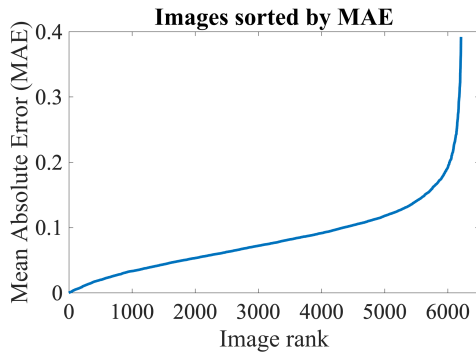
Table 4.2: Evaluation of color descriptors and learning methods on color composition similarity. Predictions are based on L1 and L2 norms, or trained on 'MOS' and distribution of 'Ratings'. The Spearman ρ between the predictions and the MOS computed from user ratings is reported. Performance is highest when training on distributions of ratings using the Kullback Leibler Divergence loss function.

We divide color descriptor methods into three groups: histogram-based, SIFT-based and MPEG7. We use L1 and L2 for all descriptors in these three groups to measure the similarity between pairs of images in the test set, rank them, and compute the SROCC. We also train the descriptors using CNNs, and neural networks for SIFT and MPEG7 descriptors, respectively. To extract SIFT descriptors, we densely sample the images and compute SIFT color features at each sampled point. The resulting data is enough to train a CNN that has a similar architecture to our **COLSIM_RATE** network (in Section 4.3). MPEG7 descriptors, on the other hand, are very compact. Their sizes are 192 for Color Layout Descriptor (CLD), and 256 for Color Structure Descriptor (CSD) and Scalable Color Descriptor (SCD). Thus, we train a small neural network that has two fully connected layers with one prediction layer. The features produced by descriptors for pairs of images are combined using the function f as described in Eq. 4.3.

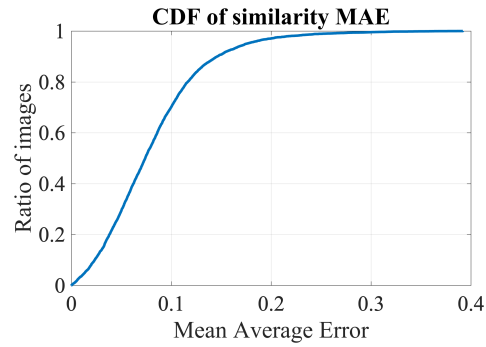
The numerical results show that descriptors, even though designed for color similarity, do not correlate well with human evaluations. The maximum SROCC is 0.692 obtained with CSD and SCD descriptors. Training a DCNN or Neural Network on the descriptors can improve the results up to a maximum of 0.860 SROCC in the case of **huesift**. Nevertheless, it takes an additional step to first compute descriptors before training them to get decent results.

A straightforward approach is to fine-tune a network or train one from scratch on our dataset. We do transfer learning from pre-trained features and then fine-tuning using the VGG19 network [SZ15]. As VGG19 is trained for object categorization, it cannot perform well out of the box on color similarity. The SROCC result for L2 distance on fc7 features of the VGG19 is 0.467. It shows that content and color are not highly correlated. The SROCC result of VGG19 transfer learning is 0.812 and improves to 0.863 with fine-tuning. Even though the results are satisfying, we observe that the features in VGG19 favor classification and hence still affect the performance of color similarity measurement. Thus, we train a rating network **COLSIM_RATE** described in Section 4.3 from scratch. The SROCC of **COLSIM_RATE** is 0.913, the best of all methods. We also train a Compact network that contains 3 convolutional layers, 2 fully connected layers and 1 prediction layer on images of size 112×112 pixels. Even though the performance is lower at 0.869 SROCC, the network has fewer parameters while having comparable performance to the fine-tuned VGG19. The MOS prediction network **COLSIM_MOS** has an SROCC of 0.902, which is slightly lower than **COLSIM_RATE**.

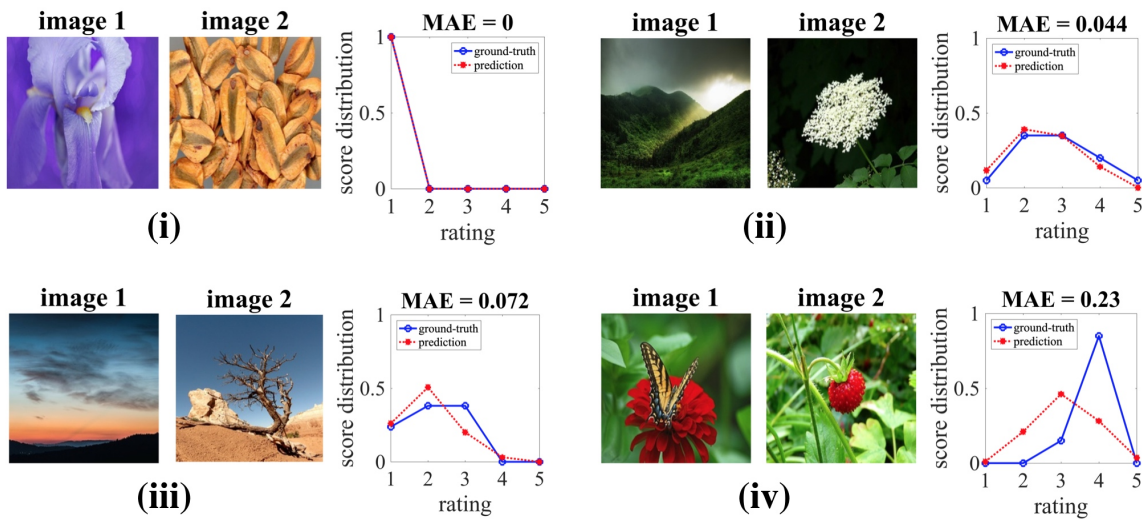
Regarding errors, we sorted the images by their MAE and plot the ranked images in Figure 4.23(a). We plot the cumulative distribution function (CDF) of the MAE between the participants' distribution of ratings and our **COLSIM_RATE** network's predictions in Figure 4.23(b). The MAE is below 0.1 for 70% of the test data and only increases substantially in the last 5%.



(a) Sorted prediction errors for all images.



(b) CDF of similarity ratings MAE.



(c) Examples of different error levels (MAE). Blue graph: ground-truth, red graph: prediction.

Figure 4.23: The MAE between participants' rating distributions and the `COLSIM_RATE` network's predictions on the test set. For most images the MAE is small, e.g., (i) and (ii) whereas only 3% have an MAE > 0.2, e.g., (iv).

4.5 Application 2: Fine-Grained Image Retrieval

Fine-grained image similarity measures not only the content difference among image classes but also the visual difference within a class. Image retrieval by class or categorical features does not consider colors as a part of the ranking procedure. For instance, when searching for an image of a black poodle, retrieval prioritizes semantic information and returns poodles with various colors. This is not always desirable. We show that by using our visual color similarity metric, the relevance of the ranking results is improved.

4.5.1 Related Work

Existing methods relate visual similarity to fine-grained classification or visual attribute similarity. These two main approaches are only beginning to tackle the complex nature of perceptual comparisons as part of visual search. Visual similarity is contextual because of the subjective judgments and its use-case. For instance, a query for an image depicting a leopard

pup at the zoo could be intended to retrieve images of leopards (pure class), young leopards (object attribute and class), or yellow animals (color and class).

The first type of methods learn features for general visual similarity [BB15; Che+10; PM15; Wan+14; WKH17], starting from category labels, textual descriptions, or triplet data. The second type of approaches separate aspects of visual similarity, by learning from human-nameable visual attributes or discovering new ones for image retrieval [DRS11; FZ07; KG13; PG11; Ras+13; SFD11; WKH17; Yan+16]. Attribute learning complements category-level recognition by learning the degree to which one or more attributes are present in an image. Attributes are very specific and combining them is challenging [SFD11] due to their interactions and relative importance. For instance, an animal can be yellow or furry (absolute scale), taller or shorter (relative attributes).

We propose to separate visual similarity into multiple factors that can be individually studied. In this work, we focus on the color composition factor. This is not a per-image attribute as we cannot quantify the amount of color composition in an image, nor can we say that an image has more or less color composition than another. However, it allows us to better specify the context in visual search. We use the correlation between pairs of content features and color similarity to improve fine-grained visual similarity prediction. We show how the colors factor can improve retrieval rankings, by controlling it separately from content semantics and category (Section 4.5.2). Subsequently, we use the correlation of color and category features to improve fine-grained visual similarity prediction (Section 4.5.3).

4.5.2 Content versus Color Retrieval

We perform retrieval using content features versus color features on a subset of ImageNet [Den+09] dataset. In the context of this chapter, the term "content" can also be used interchangeably with "category". We measure content similarity using L2 for fc7 from VGG19 [SZ15] network, and color similarity is from our color-sim network. The similarity between the query q and a retrieved image r is $\mathcal{H}(q, r)$. For a query q , we ranked the results by content and color similarity. The visualized results show all retrieved images r such that $\mathcal{H}(q, r) > 0.5$.

In the case of retrieving the sunflower image in Figure 4.24(a), all the yellow flowers are pushed up in their rank in the color similarity compared to the content similarity. Moreover, the images that contain sunflowers with a blue sky background are also higher in rank for color similarity. For the sunset image retrieval example in Figure 4.24(b), the red-orange tone images are ranked visually more correct in color similarity compared to the content similarity. Similarly, the same consistent results are noticeable for the mushroom retrieval example in Figure 4.24(c).

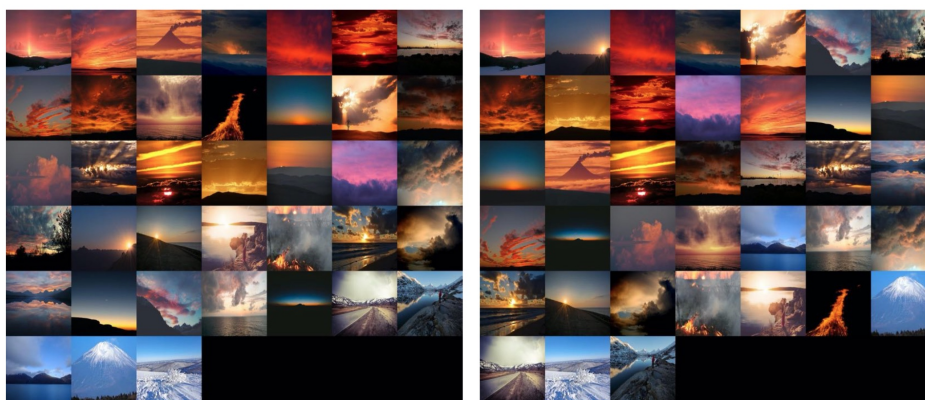
Retrieval by combining category similarity and color similarity: We are also combining color and category criteria for retrieval in a simple way. First we retrieve images by category and from the first 100 retrieved images, we sort them by color similarity to the query



Color similarity

Content similarity

(a) Sunflower retrieval



Color similarity

Content similarity

(b) Sunset retrieval



Color similarity

Content similarity

(c) Mushroom retrieval

Figure 4.24: Retrieval results, independently ranked by content and colors. The first image in each figure is the query image.

image. The examples are shown in Figure 4.25 and Figure 4.26. We also display the results of retrieval by category and color for reference. It shows that retrieval by each criteria works



(a) Retrieval by category



(b) Retrieval by color



(c) Retrieval by category followed by color

Figure 4.25: Different retrieval results: retrieval by category only, by color only and by combination of category and color. In the combination, the images are retrieved by category first, followed by color similarity. The first image in each figure is the query image.



(a) Retrieval by category



(b) Retrieval by color



(c) Retrieval by category followed by color

Figure 4.26: Different retrieval results: retrieval by category only, by color only and by combination of category and color. In the combination, the images are retrieved by category first, followed by color similarity. The first image in each figure is the query image.

well based on their own measurements. The retrieval by combining both criteria results more visual similarity towards the fine-grained image retrieval. In the Section 4.5.3, we introduce a novel features that combined color similarity features and category features that can be trained fast by SVM and be able to produce the state of the art results in fine-grained image retrieval.

4.5.3 Features and Training Model for Fine-Grained Retrieval

From visual ranking, we go further in evaluating the potential of the COLSIM network to be used for fine-grained similarity with more quantitative results and analysis.

Model	Validation 14k (not available)	Subset 5k (\subset 14k)
L2 on ConvNet AlexNet fc8	82.8% (baseline)	73.7% (baseline)
Single-scale Ranking	84.6%	-
OASIS on Single-scale Ranking	82.5%	-
Single-Scale & Visual Feature	84.1%	-
DeepRanking	85.7% (+3.5%)	-
L2 on *sift descriptors	-	62.9% - 65.4%
L2 on MPEG descriptors	-	62.3% - 65.1%
L2 on COLSIM features	-	69.1%
L2 on ResNet GAP	-	79.1%
SVM on COLSIM correlation	-	73.7%
SVM on ResNet50 GAP correlation	-	84.3%
SVM on ResNet50 + COLSIM	-	86.2% (+12.5%)

Table 4.3: Evaluation on the DeepRanking triplet dataset. Results for the 'Validation 14k' column are reproduced from [Wan+14]. We evaluate our methods on the 'Subset 5k' triplets. "ConvNet AlexNet fc8" and "ResNet" use L2 distance on fc8 and ResNet50 Global Average Pooling (GAP) content features respectively. "COLSIM features" uses the result scores from our **COLSIM** network. The three SVM approaches rely on COLSIM features, ResNet50 GAP content features, and the combination between ResNet50 GAP and COLSIM features. The relative improvement of "SVM content + **COLSIM**" method to the baseline is significant, in comparison to other methods.

Color Descriptor	Combined features	Color features
csift	84.5%	50.7%
rgsift	84.6%	61.3%
oppsift	84.8%	64.5%
hvsift	85.1%	62.7%
huesift	85.3%	65.4%
CLD	84.4%	61.8%
CSD	85.5%	68.9%
SCD	85.5%	62.8%
COLSIM (ours)	86.2%	73.7%

Table 4.4: Evaluation on the DeepRanking triplet dataset. We evaluate and compare our color similarity descriptors with existing color descriptors. We show the SVM results on the 5k subset when training with (combined) and without content features. Except COLSIM, we use L2 for the rest of color descriptors to compute SVM features.

Our hypothesis for improving fine-grained similarity is that the combination of category and color features helps to better predict the similarity of image pairs compared to the individual features alone. The similarity in the categorical feature space is computed as the correlation between two feature vectors of pairs of images. The color similarity features are extracted from our color composition similarity metric or L2 distance for existing hand-crafted color descriptors. The detailed formulations for the content correlation and color similarity are

explained below. Our hypothesis is verified by numerical results in Table 4.3 and Table 4.4.

Wang *et al.* [Wan+14] have introduced a fine-grained similarity database which contains 5,033 ranked triplets. A triplet comprises a query Q , and two compared images A and B . If the visual similarity $sim(Q, A) > sim(Q, B)$ which means A is more similar to Q than B , then the correct ordering of the triplet is (Q, A, B) .

Using this dataset, we study different similarity measures on category and color features individually and in combination. We use the L2 distance to measure the visual similarity between pairs of images. For content features, we evaluate L2 on the fc8 layer of AlexNet and the Global Average Pooling (GAP) layer of ResNet50. For color features, we evaluate L2 for all color SIFT descriptors, MPEG7 descriptors and COLSIM features extracted from our model. The L2 distance on individual types of features does not yield good results (Table 4.4). Therefore, we train a binary classifier (SVM, RBF kernel) on the triplet data using combinations of features. In general, the input features to the SVM are a pair of similarities $(sim(Q, A), sim(Q, B))$ for a correct triplet (Q, A, B) . Wrongly ranked triplets are created from the correct ones, by reversing the relationships $(sim(Q, B), sim(Q, A))$.

The features that are used when training the SVM are: the direct color similarity produced by the **COLSIM** network $S_{COLSIM}(X, Y)$, and the Pearson Linear Correlation Coefficient (PLCC) between GAP content features ² F^{GAP} extracted from a pre-trained **ResNet50** network. They are defined as following:

$$S_{GAP}(X, Y) = PLCC(F^{GAP}(X), F^{GAP}(Y)) \quad (4.5)$$

$$PLCC(x, y) = \frac{1}{n-1} \sum_{i=1}^n \left(\frac{x_i - \bar{x}}{\sigma_x} \right) \left(\frac{y_i - \bar{y}}{\sigma_y} \right) \quad (4.6)$$

where n is the number of dimensions of the features x, y . Therefore, the input features for ‘‘SVM ResNet GAP correlation’’ is $[S_{GAP}(Q, A), S_{GAP}(Q, B)]$ which contains only content features for a triplet (Q, A, B) . The input features for ‘‘SVM ResNet + COLSIM’’ are the combination of content similarity and color similarity, and defined for a triplet (Q, A, B) as:

$$g(COLSIM, GAP) = [S_{COLSIM}(Q, A), S_{COLSIM}(Q, B), S_{GAP}(Q, A), S_{GAP}(Q, B)] \quad (4.7)$$

4.5.4 Results Analysis and Discussion

The DCNN methods in [Wan+14] have been evaluated on a validation dataset of 14,000 triplets. However, the authors [Wan+14] make available only a subset of 5,033 triplets. We evaluate our models on this subset by using 20 repetitions of random 80%/20% train/validation splits. The optimal hyper-parameters for each split are estimated by 5-fold cross-validation.

²the terms category and content features are used interchangeably

Our proposed model using the combined category and color feature similarities performs best. We do not have access to the other methods to directly compare their performances on the “Subset 5k” database. Thus, we use a shared baseline model for comparison: the L2 distance between fc8 features from AlexNet [KSH12], named “ConvNet AlexNet fc8”. This common baseline performs much better on the 14K dataset than on the subset 5K (Table 4.3). Therefore, we expect equivalent methods will perform better when tested on the 14K compared to the 5K subset.

State-of-the-art performance: The accuracy of our method is 86.2% compared to 85.7% for the best DeepRanking [Wan+14] approach. However, our method shows a substantially higher improvement of 12.5% relative to the shared baseline, compared to the improvement of 3.5% for DeepRanking. As the performance of the baseline method on ‘Subset 5K’ (73.7%) is much lower than on ‘Validation 14K’ (82.8%), the relative % improvement suggests a much better overall performance for our method.

Feature combination vs individual features: Even though the SVM training on ResNet GAP correlation and COLORSIM scores achieves the best results, we also test the model on different hand-crafted descriptors. The results, on the right of Table 4.4, show that: (i) COLSIM outperforms hand-crafted descriptors; (ii) the combination of content feature correlation and color similarity yields better accuracy compared to using L2 on descriptors or ResNet GAP alone.

Feature correlation vs L2 distance: Using content or color descriptors alone, we find that training an SVM on the PLCC of the features results in a better accuracy than L2 distance on the respective features. For instance, the accuracy for SVM on ResNet GAP correlation is 84.3% compared to 79.1% for L2 on ResNet GAP features (Table 4.3).

Features vs end-to-end training: While DeepRanking [Wan+14] used 14 million Google search images during training, and a large set of triplets ($\approx 50k$), our method relies on a much smaller set of 5,033 triplets and our own database of 30k image pairs. The improved performance of our approach, using combined category and color features, shows that embedding domain knowledge in our model achieves both excellent performance and efficient training. Training on the proposed low-dimensional pairwise features is much faster than an alternative end-to-end triplet network.

In this section, we present experiments with image retrieval from different aspects: color and category and their combination. Retrieval based on combination of color and category features produces better results toward visual similarity image retrieval. We create an improved model for visual ranking similarity, by introducing a novel way to combine non-homogeneous representations such as color similarity and category features. These multi-aspect, low-dimensional features have proven to be extremely effective in training visual ranking models, surpassing the existing state of the art ‘DeepRank’ that was trained on substantially more data. Overall, the results prove that our proposed approach better predicts visual similarity.

4.5.5 Analogy Image Retrieval - An Inspiration

The classic popular image retrieval that we are familiar with is retrieval using keywords such as Google search. However, keywords are usually short and not descriptive enough to specify in detail what image we want to retrieve. For example, the Google search for images with keyword "duck" returns all sorts of images including natural and toy ducks (Figure 4.27).

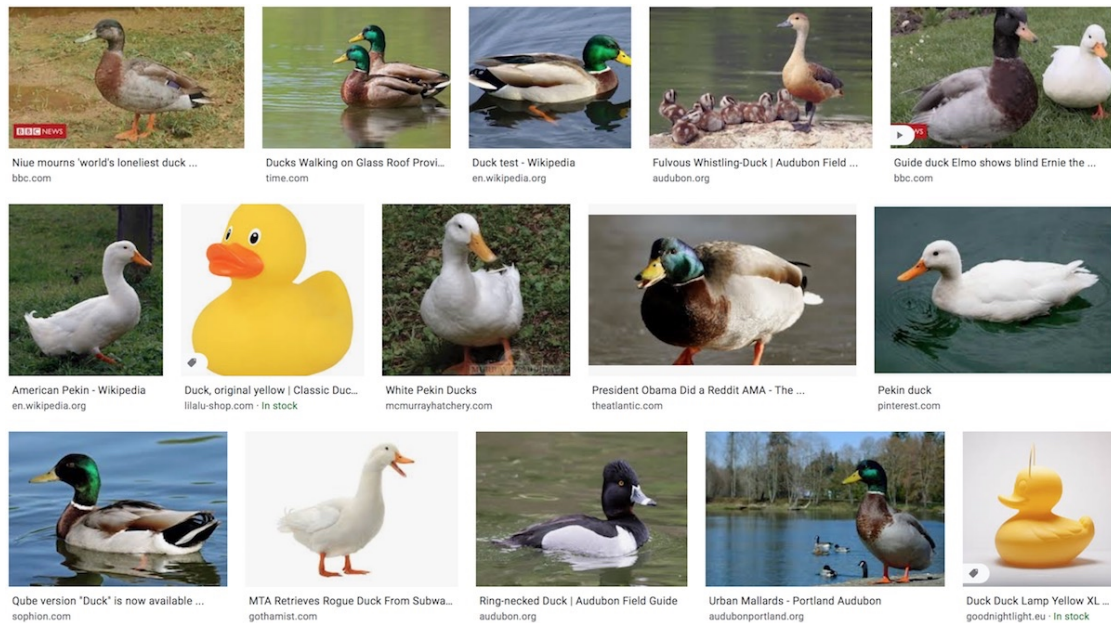


Figure 4.27: Results of Google search on keyword "duck". They include many types of ducks: natural images and graphical generated, real ducks and toy ducks.

To better specify the users' requirements for image search, we have example based image retrieval. In this approach, a user inputs an image of which she wants to retrieve similar images. There are not many commercial platforms that currently support this type of search. Pinterest is one of the platform that users can retrieve images based on one input image (Figure 4.28). Once the user inputs an example image, the system will return a set of similar images. It is unclear what criteria are used in the search engine. The resulting images can be seen as similar to the input image in terms of image type, category or texture. Some photo-stock search engines try to combine visual and keywords search to propose similar images such <https://www.shutterstock.com> or <https://pixabay.com>. Google image search also has an option for the input to be an image and return similar images. However, for many cases, Google infers the input image's label and use it as one of the criteria for image search. Besides keywords, users can also specify colors as another search criterion. The color criterion is a single solid color rather than the color composition of the input image.

Studying multiple aspects of visual similarity leads us towards future research on analogy image retrieval where the inputs are multiple images. The system will identify which aspects the input images have in common and retrieve accordingly, assuming that the common features in the input images reflect the user's intention. In Figure 4.29, we show examples of analogy retrieval. Given a pair of input images, we expect the retrieval results contain

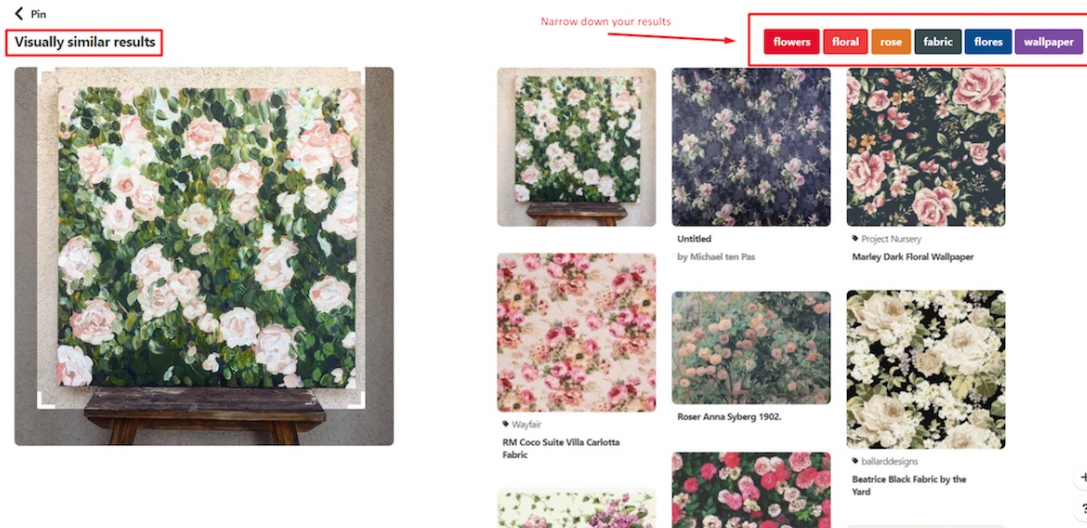


Figure 4.28: Results of Pinterest search on example images. The results show similar types of images. The retrieved images can be seen as similar in terms of image type or category content or texture. (Source image: <https://www.searchenginejournal.com/best-image-search-engines/299963/#close>)

common features with the input images. In Figure 4.29(a), the common features are color composition similarity: images that have white objects in the center of the image with black backgrounds. In Figure 4.29(b), the input images are two types of ducks and therefore, we expect the results will be natural images of ducks that look like each of the input image. Not only with color and category, images can also be evaluated in different aspects such as image type, structure or texture. If the two input images are identical, the similarity search criteria are high for all the aspects. Therefore, we will expect the system to return highly visually similar images to the degree of almost identical. In the next Section 4.6, we will explore a metric that is used to measure the texture similarity of pairs of images using deep features generated by Convolutional Neural Networks.

In this part of the work, my broader motivation is to answer the questions "Are two images similar?" If they are, then "Why are they similar? (or "What aspects of similarity do they represent?"). It is the first step towards reasoning and image understanding. The results in this chapter show that studying each individual aspect of similarity is possible and visual similarity is better formalized from combining these individual aspects. It is the proof of concept to the answers of my problem formulation questions "Why".

Even though we succeed in separating some factors of visual similarity and produce good results, there are still challenges in image understanding in Computer Vision research. Understand the contexts in the images leads to a higher level of similarity such as pose, illusion, etc. For example, in Figure 4.30, how can an algorithm abstracts the feminine lying pose between the leopard and the woman? Or how would a system interpret a cat with antlers? Reaching this abstract level of understanding also means to abide with subtleties and refinement cognitive systems, as one said "the devil is in the detail".

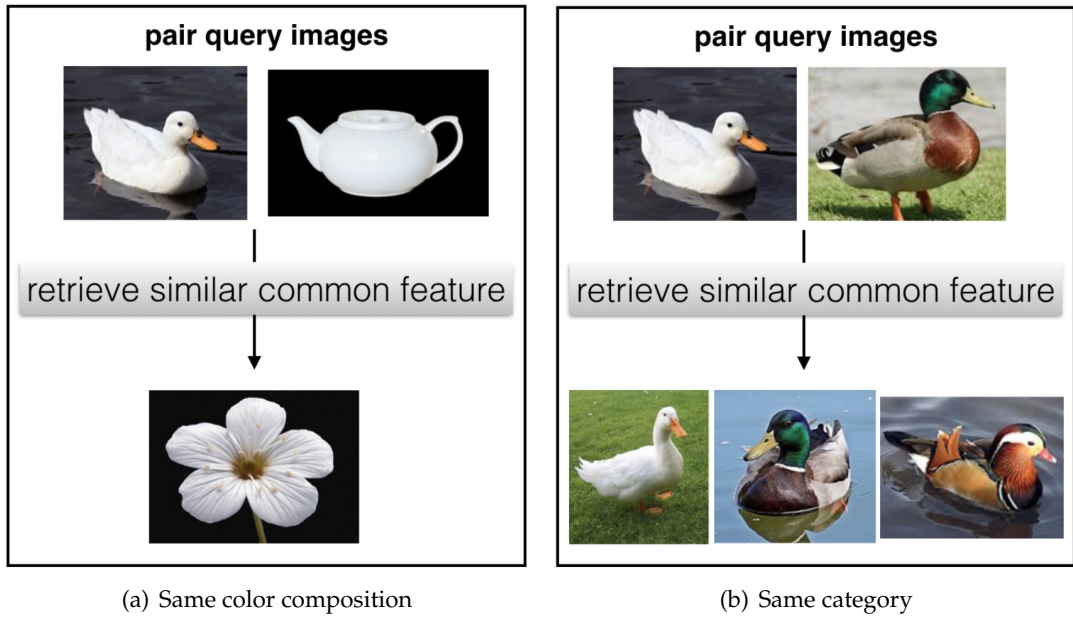


Figure 4.29: Analogy retrieval: color versus category. The requirement is to understand different similarity aspects and retrieve similar images accordingly.

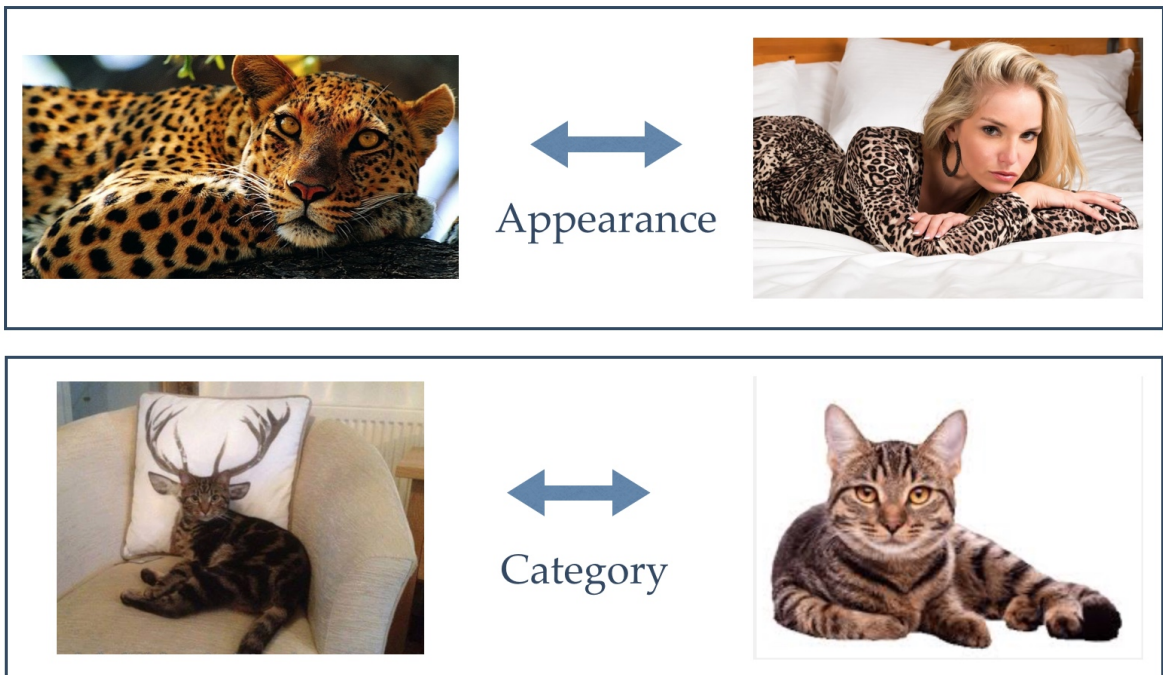


Figure 4.30: The challenge of Appearance versus Category.

4.6 Application 3: Neural Style Transfer with Perceptual Color Similarity

In this section, we show the benefit of using a perceptual color similarity network in neural style transfer. Neural style transfer is a set of techniques to generate a new image from a content and a style image using Convolutional Neural Networks (CNNs). By combining one content image with different style images and paintings, it opens unlimited possibilities in creating artworks for everyone even without having painting skills.



Figure 4.31: An example of style versus color transfer. Both methods use the same source and target images. The style includes both texture and color, which represents the style of the painting in the target image. The color transfer result has the target image's color palette while maintaining clear content from the source image. The style transfer result is created using Gatys *et al.* method [GEB16], and the color transfer result is created by using our perceptual color similarity network.

4.6.1 Related Work

One of the famous and pioneering techniques in the field is proposed by Gatys *et al.* [GEB16]. The method computes content and style representations of input images using the filter responses from various layers of VGG19 [SZ15]. The image content representation is simply the filter responses from the DCNN. The image style representation is computed using a Gram matrix $G^l \in R^{N_l \times N_l}$ which measures the correlations of image features in layer l :

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \quad (4.8)$$

where F_i^l and F_j^l are features maps i and j in layer l .

An example of how a Gram matrix is being used in texture synthesis [GEB15] is shown in Figure 4.32. The Gram matrix is used to measure the difference between convolutional response layers of the source texture and synthesized texture. It is also the main component for the loss that the DCNN needs to optimize to generate target texture. The synthesized results (Figure 4.33) show that the Gram matrix captures textures of images very well. Therefore, it is used as the core technique for style transfer.

The style transferred image is generated by backward propagation through a CNN. The optimization required is to minimize the total distance of content representation and style

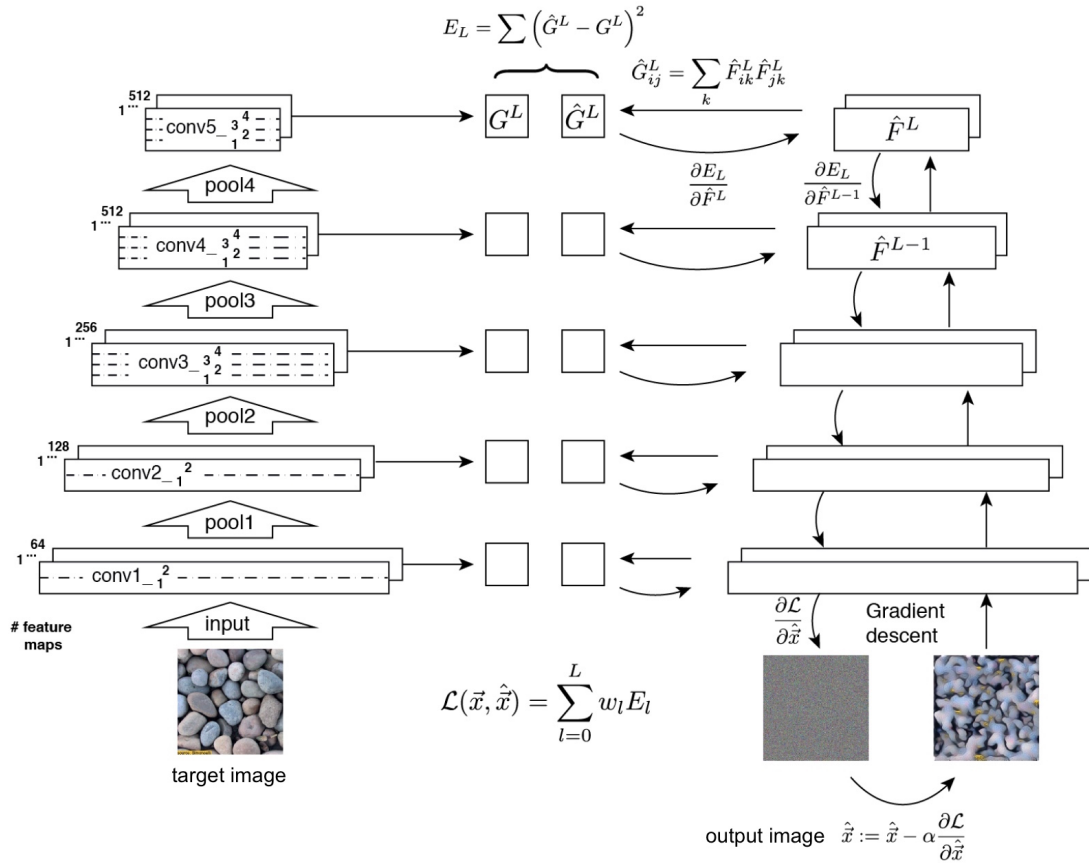


Figure 4.32: Gram matrix is calculated based on mid-level deep features of a Convolutional Neural Network. For each convolutional response layer shown in the figure, the Gram matrix calculates its feature correlation. At every layer, the difference between feature correlation of the target image and the output image is computed. The objective loss is the weighted sum of all the differences between two images for all the layers. (Source: [GEB15])

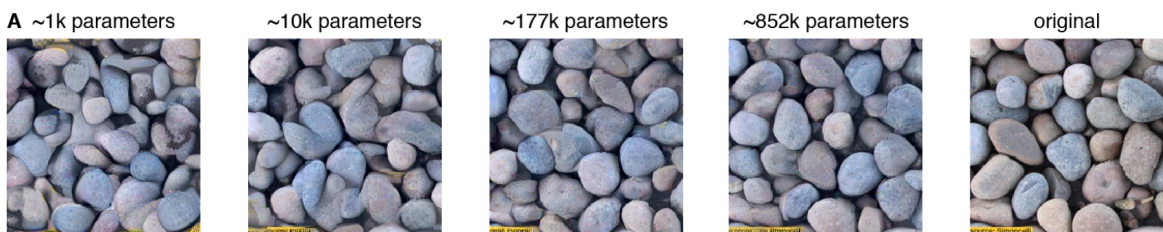


Figure 4.33: Results of texture synthesis using the Gram matrix. The number of parameters increases from left to right. The larger the number of parameters is, the more response layers in the network are selected to compute texture features. The synthesized results are more realistic when the number of parameters increases. With 852K parameters, all the layers from the first layer up to and including 'pool4' are selected. When reducing the number of layers such that only one layer for each scale in the network is selected, the number of parameters is about 177K. However, the results from 177K parameters and 852K parameters are very close in terms of quality. Based on this study, we use one convolutional response layer for each scale in both style transfer and color transfer methods. (Source: [GEB15])

representation with a linear weighted combination. Gatys *et al.* [GEB16] proposed a model using Gram matrix to compute the style similarity between the input and the target images (Figure 4.34). Having a similar goal, Li and Wand [LW16] replace the Gram matrix

by Markov Random Fields (MRF) on local patches of CNN responses. Thus, local spatial constraints are enforced in the synthesized image. The other approach to synthesize style transferred images is to use generative networks [JAF16; Zhu+17]. Johnson *et al.* [JAF16] use content and style representations that are similar to Gatys *et al.* [GEB16] to compute loss functions with an additional total variation regularization for smoothness. Johnson *et al.* train a feed-forward network for every style and therefore, the style transferred image is generated through a single forward pass. Different with other neural style transfer methods that learn the feature differences between a content and a style image, Zhu *et al.* [Zhu+17] use Generative Adversarial Networks (GAN) to generate images by learning the mappings between two image collections. Ying *et al.* [Jin+17] present a review on different neural style transfer methods and comparisons in more details.

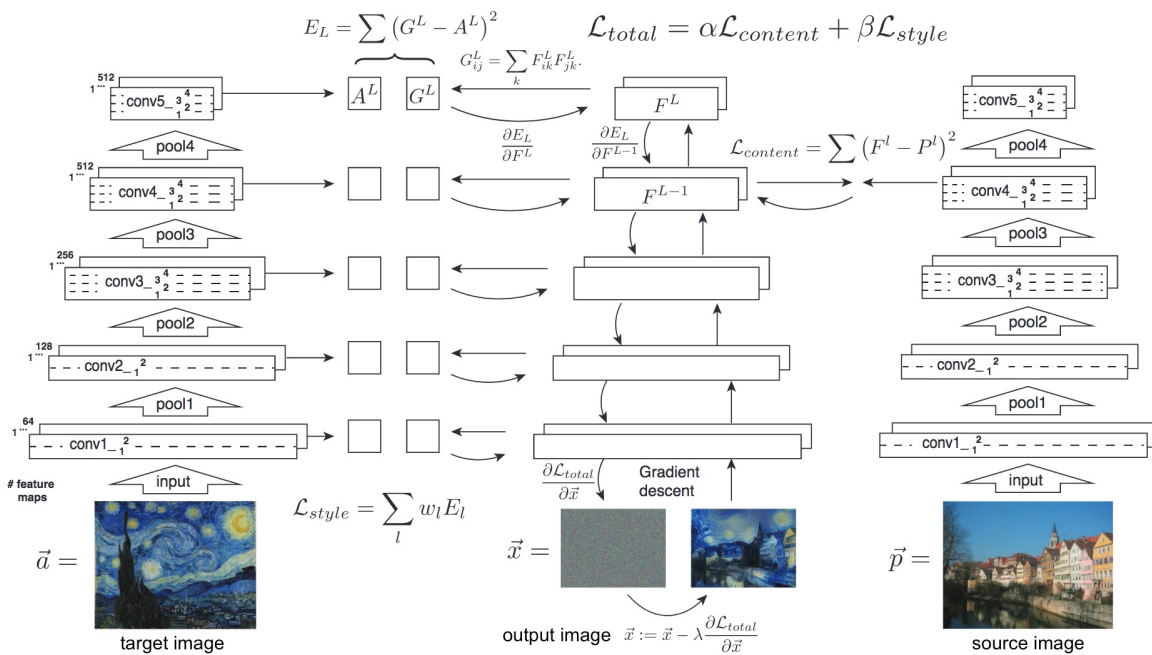


Figure 4.34: A style transfer model using Gram matrix [GEB16]. The output image \vec{x} is generated by back propagation such that its content is close to content input \vec{p} and its style is similar to the style input \vec{a} . Gram matrix is used to calculate feature correlation at every convolutional response layer for all three images $\vec{x}, \vec{p}, \vec{a}$. The content loss is computed by the sum difference between layers of the output image \vec{x} and the content input image \vec{p} . The style loss is calculated by the sum difference between layers of the output image \vec{x} and the target style input image \vec{a} . The objective loss is the weighted combination of the content loss and the style loss. (Source: [GEB16])

The common property of all the neural style transfer methods is to repaint the content image using textures and colors in the style image. Textures and colors are coupled together in representing styles. In the most recent work, Gatys *et al.* [Gat+17] propose a method to preserve the color of the content image. Therefore, it provides an option to synthesize style transferred image with the color from either content or style image. Inspired by the remarkable results from Gatys *et al.* [GEB16; Gat+17], we propose a method to generate a style transferred image whose colors come from a third input image using our perceptual color similarity network.

4.6.2 Perceptual Color Transfer

We train a new network from scratch on our color similarity dataset. The new network, called **VGG19-COLSIM**, has a similar architecture to VGG19 [SZ15] and predicts color similarity rating distributions. Therefore, we believe the network captures features for color representations. We do not use **COLSIM** networks because the Batch Normalization layers normalize filter responses. Thus, the back propagation cannot reconstruct the original features from an input image.

Given a content image I and a color image C , a perceptual color transferred output O_C^P is produced by altering the content I during the back propagation on **VGG19-COLSIM** such that the perceptual color of C and O_C^P are minimized. Therefore, we use I as the initialization for the back propagation process. The color loss is calculated as follows:

$$L_{color}(C, O_C^P) = \sum_{l=0}^L \left(\frac{1}{N_l^2} \sum_{i,j} \left(G_{ij}^l(C) - G_{ij}^l(O_C^P) \right)^2 \right) \quad (4.9)$$

where $G_{ij}^l(I)$ and $G_{ij}^l(O_C^P)$ are the color representation for content image I and generated image O_C^P respectively, calculated using the Gram matrix (Eq. 4.8), N_l is the size of layer l . We use 5 layers {"conv1_1", "conv2_1", "conv3_1", "conv4_1", "conv5_1"} in the **VGG19-COLSIM** for perceptual color transfer. The structure of our color transfer model is similar to the structure of Gatys *et al.* in Figure 4.34, except that our network is trained on color. In the style transfer model from Gatys *et al.* in Figure 4.34, the total loss is defined as $L_{total} = \alpha L_{content} + \beta L_{style}$. For our color transfer model, the L_{style} is replaced by our L_{color} in Equation 4.9 such that $L_{total} = \alpha L_{content} + \beta L_{color}$. Hence, the transfer factor is color composition and not image style.

4.6.3 Combining Style Transfer and Perceptual Color Transfer

In the new approach of neural style transfer with perceptual color similarity, we need three input images: content I , color C and style S . We use Gatys' method [GEB16] to transfer the style of image S to the content image I and produce the output image $O_S = Y(I, S)$. Our perceptual color transfer between content I and color C is O_C^P described in Section 4.6.2. The perceptual color and style transfer result is the combination of the luminance channel of O_S and the UV color channels of O_C^P in the YUV color space. The results are shown in Section 4.6.4.

4.6.4 Results and Analysis

To achieve the style and color transfer results without our perceptual color similarity metric, we explore two options. The first one is to use Gatys *et al.* method [GEB16] to transfer color from the color image C to the content image I and the output is O_C^G (Fig. 4.35(c)). We then combine the luminance channel of O_S for style transfer and the color channels of O_C^G in YUV color space to produce the output in Fig. 4.35(g). The Gram matrix of VGG19 filter responses captures both texture and color features. Thus, the result does not only contain










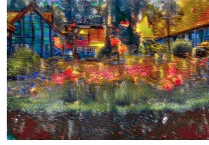
Input	Method	Reinhard	Gatys	Perception
color transfer  content I color C		 (b) O_C^R	 (c) O_C^G	 (d) O_C^P
style transfer  content I style S			 (e) O_S	
color + style transfer  content I color C style S		 (f) O_S	 (g) $L(O_S) + C(O_C^G)$	 (h) $L(O_S) + C(O_C^P)$

Figure 4.35: Perceptual style and color transfer results. (a) input images: input content I , color C , style S . (e) O_S : result of Gatys *et al.* [GEB16] transferred from style S to content I . (b) O_C^R : result of color transfer from color C to content I using Reinhard *et al.* method. (f) O_S^R : results of Gatys *et al.* [Gat+17] transferred from style image S to O_C^R as content image with color preservation from O_C^R . (c) O_C^G : result of Gatys *et al.* [GEB16] transferred from color C to content I . (g) combining style in (e) and color in (c). (d) O_C^P : result of color transfer from color C to content I using our perceptual color similarity network. (h) combining style in (e) and color in (d).

colors, but the style of color image C as well. On the other hand, our proposed perceptual color similarity network **VGG19-COLSIM** can transfer only the colors and keep the content faithful to the input image I (Fig. 4.35(d)). The results of the final color style transfer using perceptual similarity network are visually more pleasant (Fig. 4.35(h) vs Fig. 4.35(g)).

In the other approach, we use the color transfer technique from Reinhard *et al.* [Rei+01] to transfer the color from the color image C to the content image I and apply color preserving style transfer using the Gatys *et al.* [Gat+17] method. We choose to transfer the color to the content image I instead of the style image S to compare visually, even though both produce similar results. The color transfer result of Reinhard *et al.* is O_C^R in Fig. 4.35(b). We then use Gatys *et al.* [Gat+17] to transfer style in style image S to re-color image O_C^R and result O_S^R (Fig. 4.35(f)). Due to the inaccurate color transfer from the Reinhard *et al.* method, the colors of O_S^R are not faithful to the color image C compared to our method.

To evaluate more on the color transfer, we compare our perceptual color transfer with color transfer method from Reinhard *et al.* [Rei+01]. Color transfer is a big field of research on its own. Our aim is not to perform content aware color transfer. However, we observe that many color transfer methods work well if source and target images are similar in content or belong to the same image categories [FK10; Laf+14; Shi+14; Zhu+17]. The advantage of our method is that it is versatile, working with any pairs of source and target images. We compare ours with the established color transfer technique from Reinhard *et al.* [Rei+01].

It is shown that in a majority of cases, the color transfer results from our perceptual metric are more faithful to the source image than Reinhard *et al.*'s results [Rei+01] (Figure 4.36). Noting that even though we re-train VGG19-COLSIM from scratch on our perceptual color similarity dataset which is much smaller than ImageNet, the color transfer using VGG19-COLSIM performs very well.

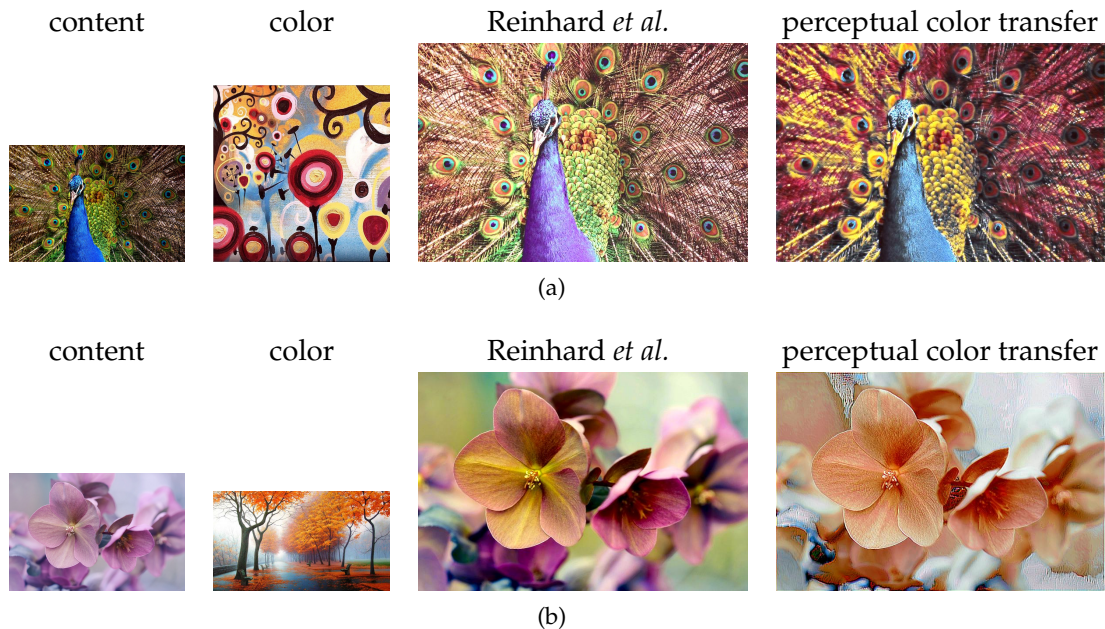


Figure 4.36: Comparison of color transfer results. In Fig. 4.36(a), the result of Reinhard *et al.* method [Rei+01] retains the green color of the peacock from the content image and also produces purple, even though there is no green or purple color in the target color image. Our result, on the other hand, produces high contrast and correct colors (blue, yellow, red and dark brown), which are all present in the target color image. In Fig. 4.36(b), Reinhard *et al.*'s method [Rei+01] generates purple and green colors in the result, which are not present in the target color image.

Our color transfer model also has its limitation. Due to the back propagation process, the edges in the target image could be blurred or lose their smoothness. Even though at normal sizes of images, the defects are not obvious, they show in high resolution images. Therefore, in our future work, we will investigate to add the edge preserving constraints and smoothness prior in our loss function.

In this section, we presented a new option for neural style transfer that allows us to have colors transferred from a different image than just from the style or content image. In general, there are many ways to combine color and style transfer such as using different color transfer methods or using our perceptual metric to transfer color to the content and/or style images before applying style transfer techniques. We experiment with different methods and examine the results. We recommend using our perceptual color metric to combine with Gatys *et al.* [GEB16] style transfer technique described in section 4.6.3 to achieve the best visual results that contain clear contents from the content images, well-defined styles from the style images and faithful colors from the color images.

4.6.5 More Style-Color Transfer Results

We compare the style transfer results that use our color similarity metric (sub-figure (f)) with those using the original method of Gatys *et al.* [GEB16] (sub-figure (e)). When applying the latter, colors from different objects become intermixed, whereas in our results each object's colors are coherently transformed. Good examples are the peacock's body in Fig. 4.37 or Lena in Fig. 4.39. Our method preserves better the separation between the foreground and background, for example, the flower in Fig. 4.38 and the horse in Fig. 4.39.

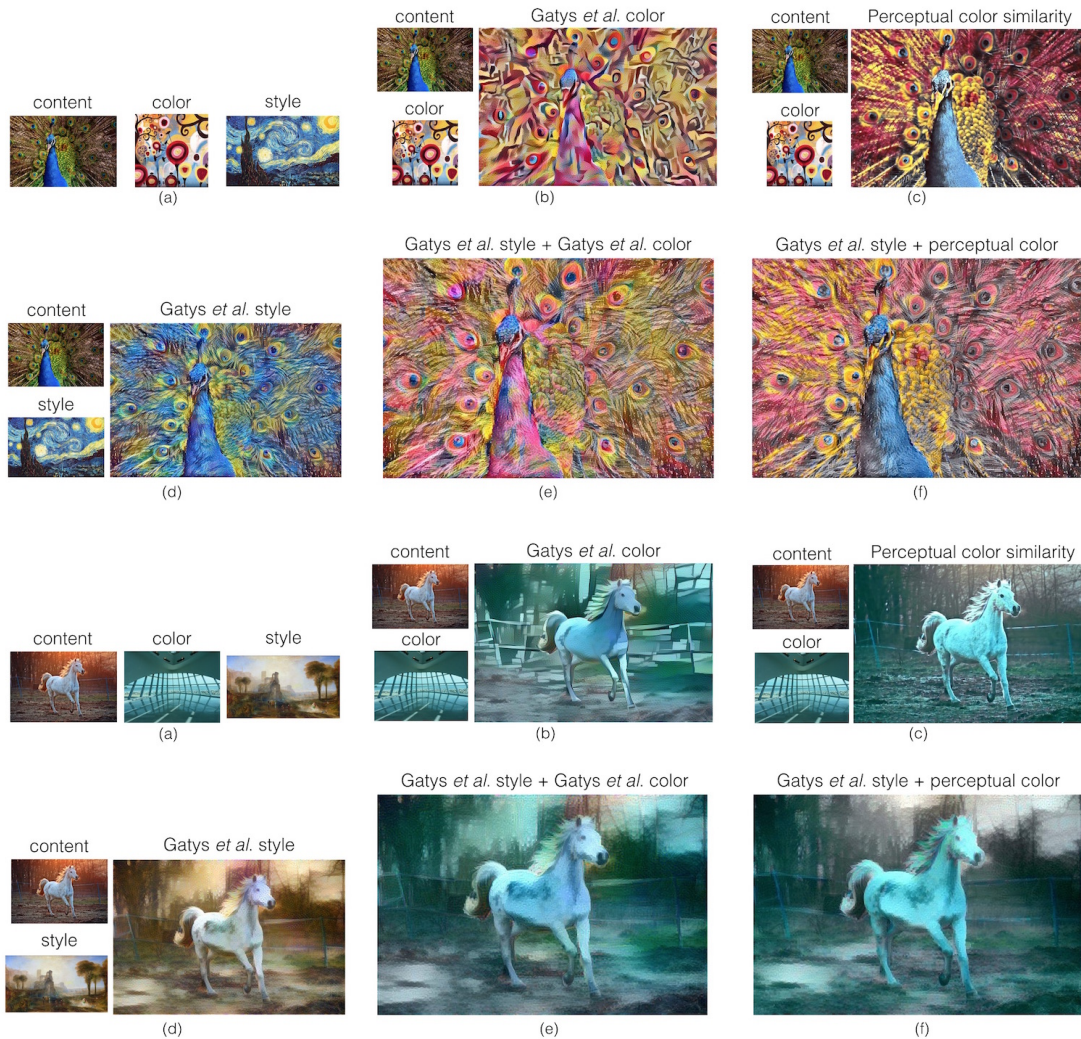


Figure 4.37: (a): input images. (d): result of Gatys *et al.* style transfer [GEB16] between content and style image. (b): result of Gatys *et al.* style transfer [GEB16] between content and color image. (c): result of color transfer from color image to content image using our color similarity network. (e): combination of color in (b) and style in (d). (f): combination of color in (c) and style in (d).

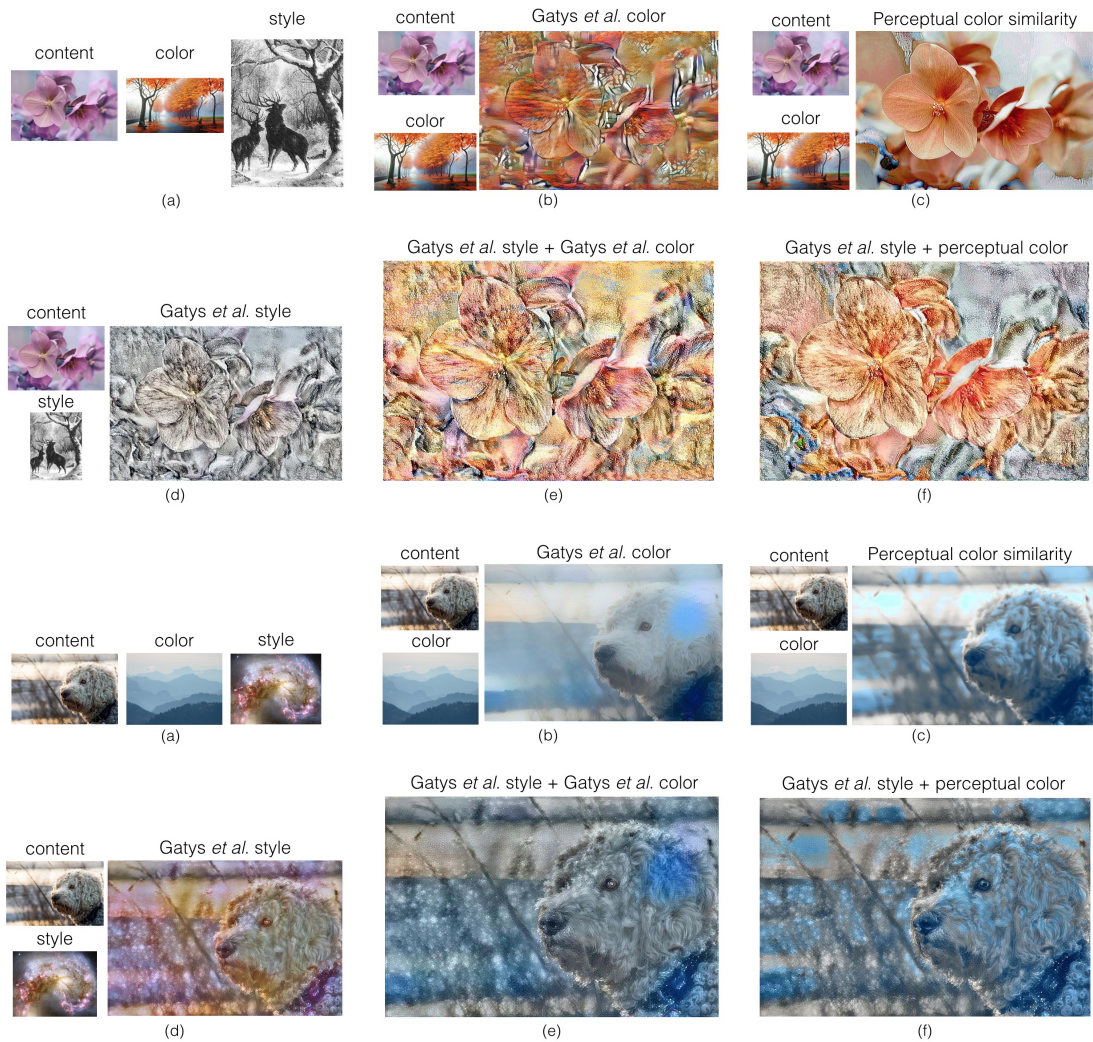


Figure 4.38: (a): input images. (d): result of Gatys *et al.* style transfer [GEB16] between content and style image. (b): result of Gatys *et al.* style transfer [GEB16] between content and color image. (c): result of color transfer from color image to content image using our perceptual color similarity network. (e): combination of color in (b) and style in (d). (f): combination of color in (c) and style in (d).

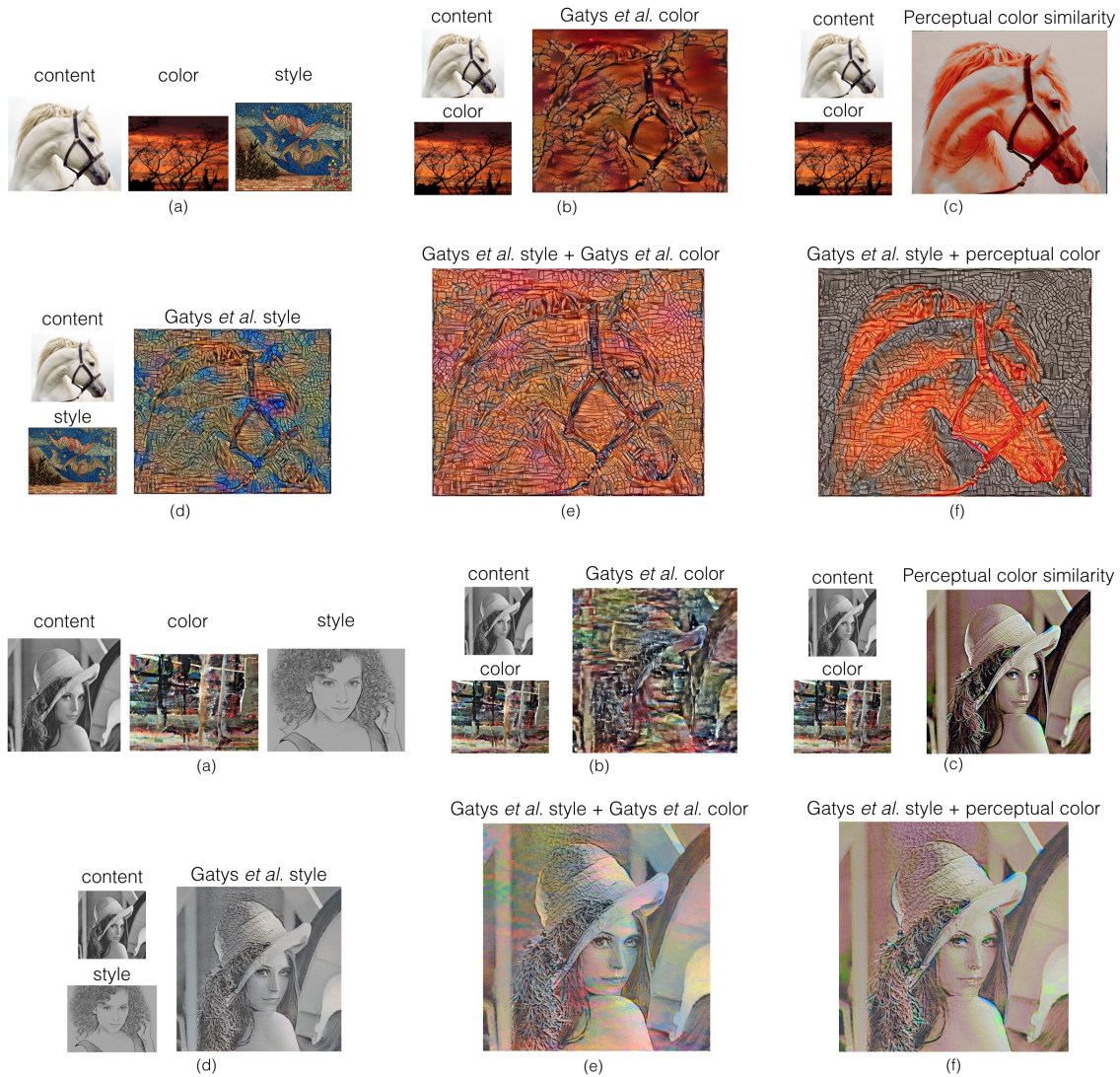


Figure 4.39: (a): input images. (d): result of Gatys *et al.* style transfer [GEB16] between content and style image. (b): result of Gatys *et al.* style transfer [GEB16] between content and color image. (c): result of color transfer from color image to content image using our perceptual color similarity network. (e): combination of color in (b) and style in (d). (f): combination of color in (c) and style in (d).

4.6.6 More color transfer results

We compare the results of perceptual color transfer using our color similarity metric with Reinhard *et al.* color transfer method [Rei+01]. The comparison shows that our metric produces results that are more faithful to the color image (Fig. 4.36 and Fig. 4.40).

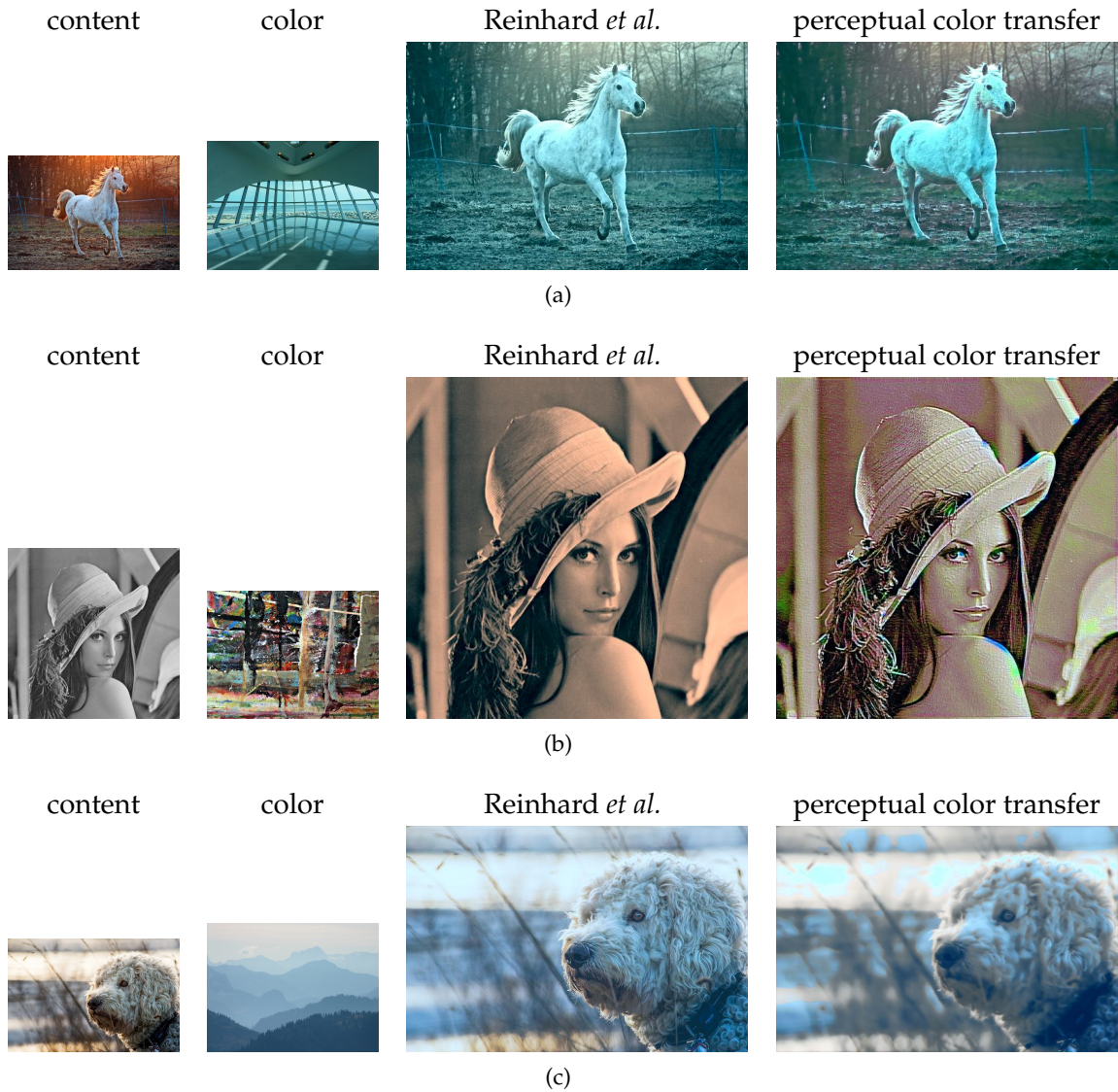


Figure 4.40: Comparison of color transfer results. In Fig. 4.40(a), our result has higher contrast between the horse and the background than the result from Reinhard *et al.* method [Rei+01]. We can see that the input images also have high contrast between the foreground and the background. In Fig. 4.40(b), the colors in our result contain a larger diversity of hues from the color image whereas Reinhard *et al.* method [Rei+01] produces fewer hues. Therefore, their result is less visually similar to the color image than ours. In Fig. 4.40(c), both results are equally good.

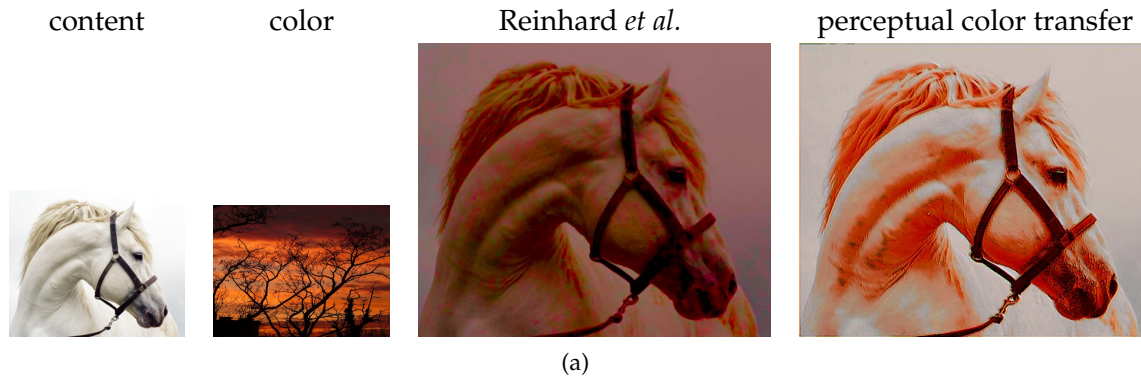


Figure 4.41: Comparison of color transfer results. Our result contains off-white background where there is no such color in the color input image. Artistically, our result looks better with higher contrast and sharper than the result from Reinhard *et al.* method [Rei+01]. However, the color palette from Reinhard's result is closer to the color input image's palette compared ours.

Occasionally, our color transfer results are less than ideal. For example, in Figure 4.41, the color input image has mainly black and orange colors. However, in our result, the background color is light beige, which is influenced by the the content image's background. It could be due to the features of the horse in the content network being significantly larger in magnitude compared to the features of the color image in the color network. In other words, the content loss is much higher than the color loss. Therefore, the color transfer method's optimizer reduces the loss to fit the content image more than the color image. This can happen because all the examples for color transfer in this chapter use the same set of parameters for the weights of the content loss and the color loss. Changing the weights to optimize more for the color loss can improve the color palette of the final result but might reduce the sharpness that comes from the content image.

4.7 Chapter Summary

Assessing image similarity is one of the core abilities of human vision, and yet it is not easy to design a computational model for visual similarity. It is technically relevant in Computer Graphics and many topics of Computer Vision. However, images can be similar in one aspect but different in others. The notion of image similarity is, therefore, ambiguous and subjective. In this work, we unravel the perceptual similarity by studying one particular aspect, which is color composition, and by combining it with content and style similarity. Our model considers the color distributions and layouts in images. Thus, our color similarity metrics go beyond pixel-based, patch-based, or histogram-based methods. We take an active learning approach to build a large color similarity rating dataset via crowd-sourcing. Our dataset is the first of its kind and sufficiently large and diverse for learning with Deep Convolutional Neural Networks (DCNN). We create perceptual metrics by training deep networks using the dataset. Our metrics perform better than all existing descriptors and methods on color comparisons. The importance of perceptual color similarity is also reflected in its applicability in Computer Vision and Computer Graphics. We successfully

apply the trained models in two domains: neural style transfer and fine-grained visual similarity. Our models provide a new option to generate “stylized” images for which the detail style and colors are taken from separate sources. The color transfer results produced by our deep metrics are faithful to the source images. In fine-grained similarity, it is demonstrated that color features extracted by our metrics improve the ranking accuracy in comparison to content features alone.

In the future, our perceptual similarity metrics can be used to select meaningful images to train for classification that uses an active learning approach. Learning through similarities helps to boost the generalization ability. It is also a smarter way to learn with a small amount but more meaningful data, making the learning process more efficient.

Part II

Deep Features

“Wonder is the beginning of wisdom.”

Socrates

One of the significant motivation of my work is to understand the potential and features of Deep Convolutional Neural Networks (DCNN). In the last few years, DCNNs offer great performances in different research fields in Computer Science. Besides searching for better architectures, optimizers, and other ways to boost performance, a branch of research is setting out to understand the knowledge encapsulated in a DCNN and is working towards explainable DCNNs. In the previous chapter, we explored the mid-level features inside a DCNN trained for classification tasks or color similarity regression tasks. We showed that these features encapsulate the textures and colors of input images very well such that one can have a successful color and style transfer using these features. The high-level features (right before the classification layer or regression layer) are discriminative and representative enough to have good visual similarity image retrieval.

In this part of the thesis, we will explore the characteristics of the abstract features from DCNNs and what they can potentially achieve in different research areas without the need for re-training DCNNs using a large amount of data. We start by examining the classification features in visual retrieval, with objects and textures, through a small application of multiple example-based image retrieval. Inspired by a work to display Class Activation Maps, which presents regions a DCNN finds features used for prediction in an input image, we study if these features could embed extra information such as shape and, therefore, can potentially be applied to weakly supervised segmentation. It is weakly supervised learning because we do not need to train a network on a segmentation dataset. However, we use existing DCNNs trained on classification and infer the shape and segment objects based on the deep features provided by the most abstract layers of the DCNNs.

Lastly, we study the discriminative property of classifying features of DCNNs. We propose a Neural Discriminant Analysis that increases the inter-class variance and reduces the intra-class variance. The results show that there is room for improvement in the clustering or discriminative optimization for deep features. We prove this in the general classification and especially in the fine-grained classification tasks.

Declaration for Chapter 5 - Shape Extraction and Semantic Segmentation

This work was in collaboration with Dr. Gianni Franchi and Prof. Dr. Michael Möller, and published at the Winter Conference on Applications of Computer Vision (WACV) Conference in 2018 "Segmentation and Shape Extraction from Convolutional Neural Networks" [Ha+18]. The research was partially funded by the German Research Foundation (DFG) as part of the research training group GRK 1564 "Imaging New Modalities", supervised by Prof. Dr. Andreas Kolb and Prof. Dr. Volker Blanz.

In this work, Dr. Gianni Franchi and I worked together in almost all the parts of the paper. It was the first deep learning project for both of us. Therefore, we shared and collaborated closely on the implementation, including setting up the Caffe framework in different systems, coding, debugging, testing, and discussing algorithms. However, in this Chapter, the study on multiple input example-based retrievals was done solely by me, before the collaboration. I brought in the Conditional Random Field (CRF) while Dr. Franchi implemented the variogram technique. Both techniques contributed to the improved performance results.

Chapter 5

Shape Extraction and Semantic Segmentation

5.1 Image Feature Representation

Even though Computer Vision has many different tasks, the fundamentals of Convolutional Neural Network architectures are usually evaluated on image classification. Often, the performance of a newly proposed deep network architecture is evaluated on the ImageNet dataset [Den+09] with 1,000 classes. In the image classification domain, ImageNet is considered the largest dataset with ground-truth for deep learning with a high number of image classes. However, 1,000 classes are still just a fraction of the number of objects in real life that can be captured by cameras. We want to examine if a Deep Convolutional Neural Network (DCNN) trained on the ImageNet dataset can present features for object classes that it has never seen or trained on before without domain adaptation or fine-tuning using a new set of data. This will give us an answer if deep features from a pre-trained DCNN can generalize to unseen classes.

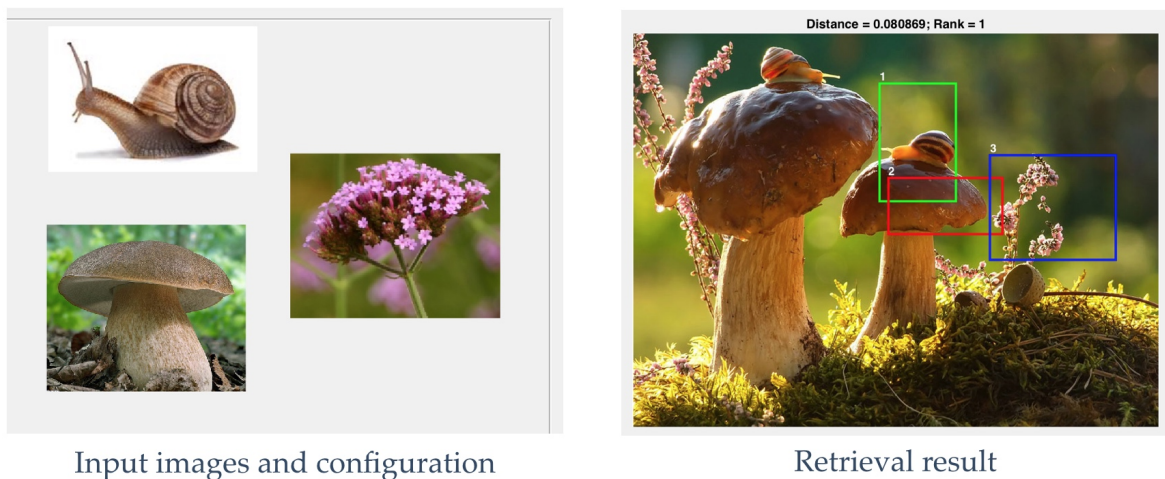


Figure 5.1: An example of image retrieval from multiple input images.

To study deep features, I implemented a small application of image retrieval in Matlab with a graphical interface where input images can be those that are not in the same set of classes

in the ImageNet dataset. The special part of this application is that users can input multiple images or snapshots of objects and place them in a canvas where they think the retrieval images should have a similar configuration (Figure 5.1). While the application is instructive for exploration, it is also an exciting idea that, so far, there is no commercial application yet.

5.1.1 Deep Feature Extraction

One characteristic of images in the ImageNet dataset [Den+09] is that they usually contain only one object because one image has only one label to describe its content. On the other hand, images in the wild are much more complicated where multiple objects can interact and are presented in a single frame of an image. Therefore, we need to select individual objects in every image and extract deep features for each of them. The general framework to extract deep features for multiple regions of images in a dataset is described as in Figure 5.2.

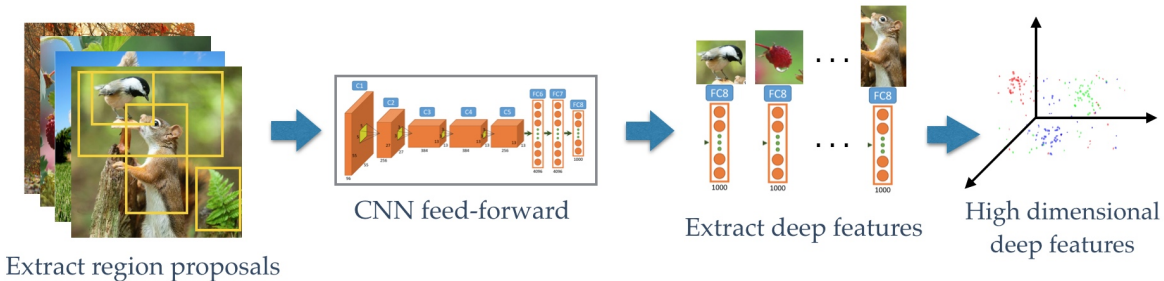


Figure 5.2: Process of extract deep features for different region proposals from images. The proposed regions in an image are first selected using algorithms in [ZD14]. Each region is fed to a CNN *BAIR Reference CaffeNet* (bvlc_reference_caffenet) model from Caffe Model Zoo (https://caffe.berkeleyvision.org/model_zoo.html). The deep features from the last layer before the classification layer are extracted to form a high dimensional feature space.

The region proposals of images are selected using algorithms in [ZD14]. It is a method to generate bounding box proposals using edges. The goal is that the bounding boxes do not cut edges. In other words, the object contours should entirely reside inside its bounding box(es). Each region proposal is then passed through a CNN to extract the deep features. In this application, a Caffe model *BAIR Reference CaffeNet* (https://caffe.berkeleyvision.org/model_zoo.html) is used and the deep features are those extracted from the last fully connected layer FC8, before the classification layer. We build a KD-tree to index the deep features for all region proposals due to the high number of images and multiple region proposals in each image. KD-tree helps to speed up the retrieval time.

5.1.2 Retrieval with Spatial Constraint

The deep features for each input image are extracted using the same Caffe model (*BAIR Reference CaffeNet*). The system uses these deep features to perform k -nn searches in the KD-tree that contains features of all the region proposals in the whole dataset to retrieve the best k retrieval candidates for each input. Each retrieval candidate contains at least one object instance of one query image (Figure 5.3).

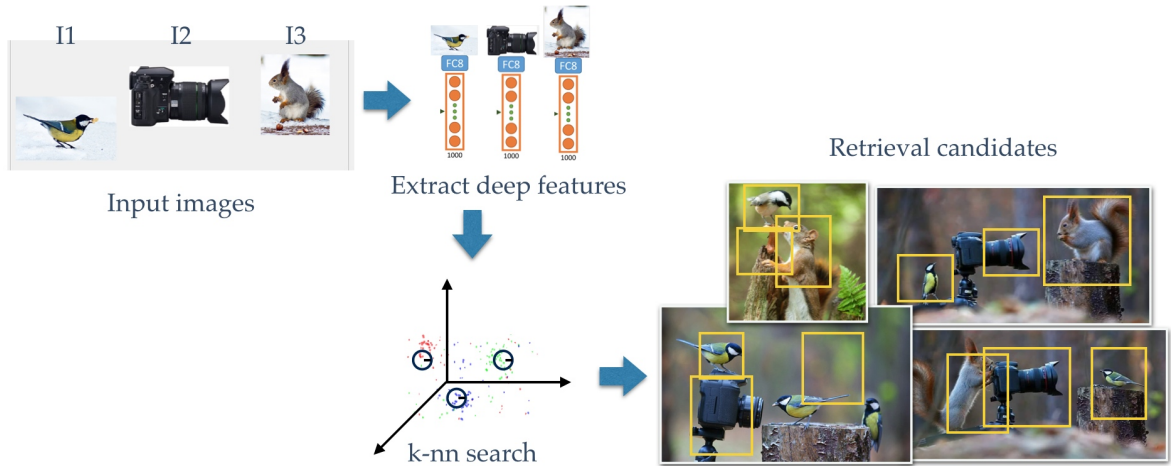
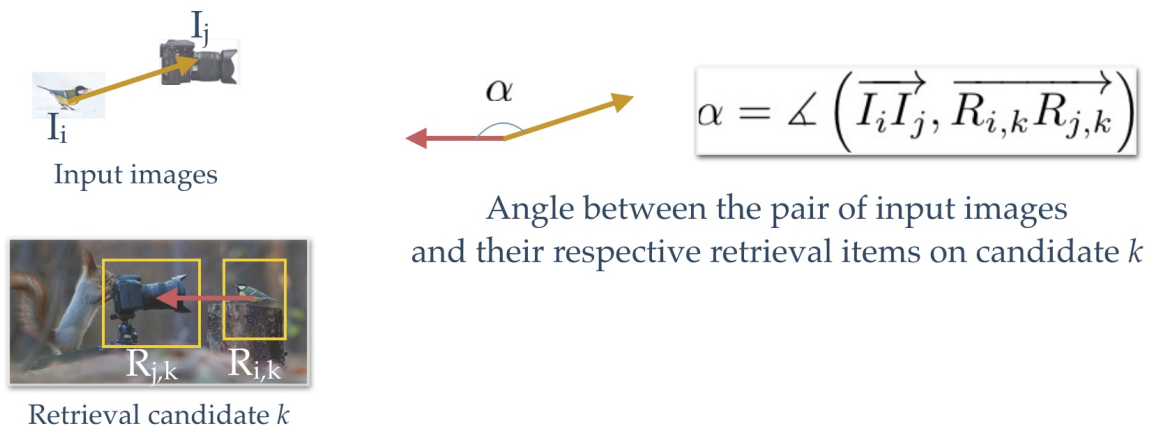
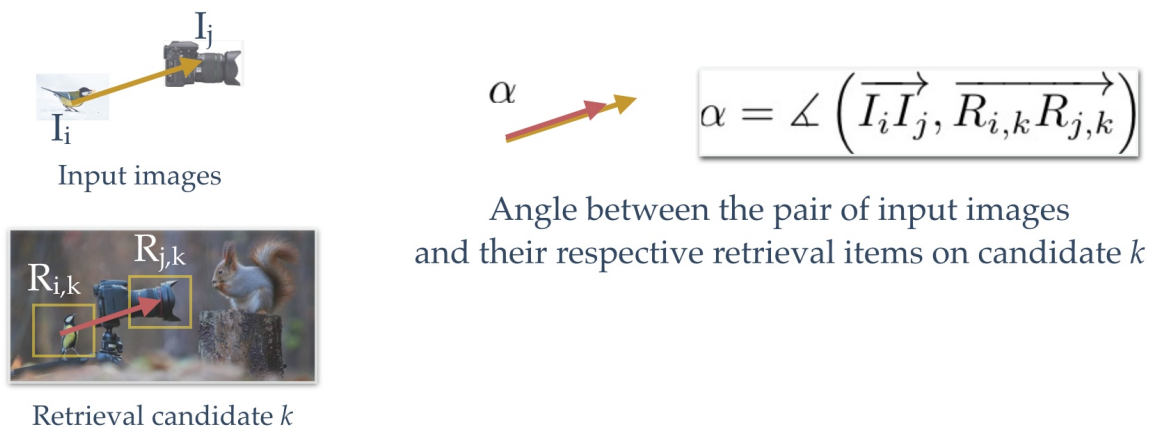


Figure 5.3: Retrieving images based on region proposals.



(a) Large angle between pair input and pair proposals.



(b) Small angle between pair input and pair proposals.

Figure 5.4: Demonstration of spatial constraint on input images and region proposals.

Finally, the constraint on the configuration (or relative positions) of the input images is used to filter out the retrieval candidates. The system returns only images that contain all the

instances from the input images with similar spatial configurations. A pair-wise angle between each pair of input images is calculated to measure the spatial constraint (Figure 5.4). For each retrieval candidate, a spatial distance function is computed as:

$$D_k = \sum_{i,j} \angle \left(\overrightarrow{I_i I_j}, \overrightarrow{R_{i,k} R_{j,k}} \right) \quad (5.1)$$

where I_i, I_j are a pair of input images. $R_{i,k}, R_{j,k}$ are two bounding boxes in the retrieval candidate image k that contain objects represented in image I_i and I_j respectively. The angle between two images or two bounding boxes is formed by a vector between two center points. The best retrieval result is the retrieval candidate k such that $k = \operatorname{argmin} D_k$.

5.1.3 Results

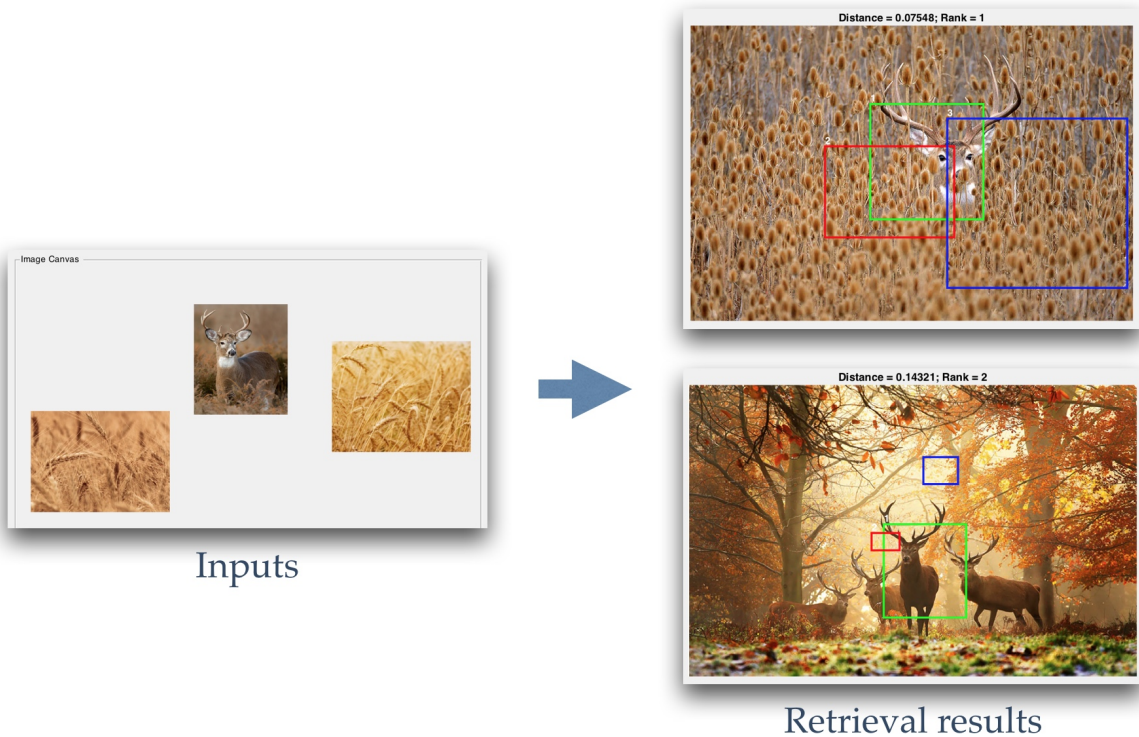
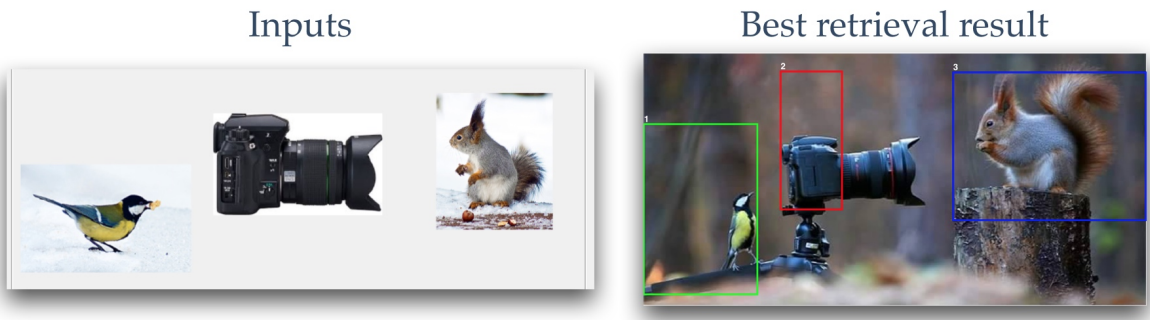
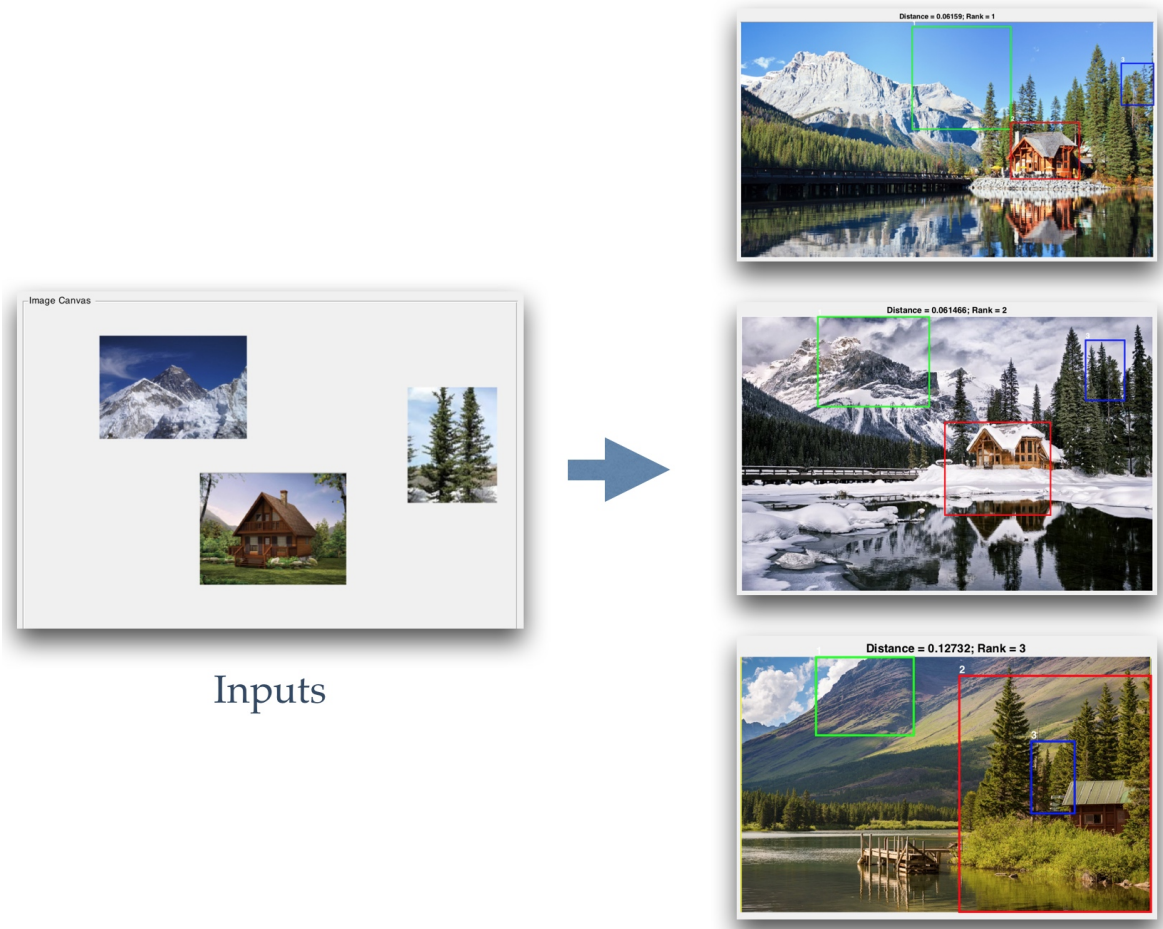


Figure 5.5: The best retrieval results from 3 input images with a spatial constraint on texture images.

We tested the results using images of which object classes are in 1,000 classes of ImageNet [Den+09] such as bird, camera, squirrel, and classes that are not in ImageNet such as mountain, tree, deer and especially wheat field textures. We get surprisingly good retrieval results, especially for the case in Figure 5.5 and Figure 5.6(b). It shows that even with unseen classes in training, the CNN model is able to represent distinctive features of new images well.



(a) The best retrieval results from 3 input images with spatial constraint.



(b) The best 3 ranked results from 3 input images with spatial constraint (ranking: top down).

Figure 5.6: Retrieval results for multiple query images with the spatial constraint.

Encouraged by this finding, we proceed to study more in-depth what features embedded in the high-level features of a CNN that are not so easily perceived. That leads us to the discovery of object shapes hidden in the high dimensional deep feature space from networks trained for image classification. We then extract the shape and use it for weakly supervised semantic segmentation. The work is described in detail in Section 5.2.

5.2 Shape Extraction And Weakly Supervised Segmentation

The era of Deep Convolutional Neural Networks (DCNNs) has led to impressive advances in the problem of *image classification*. The improvements in the network architectures, for example in AlexNet [KSH12], VGG [SZ15], or GoogLeNet [Sze+15b], as well as the training of deeper models were made possible by the availability of extremely large-scale datasets such as ImageNet [Den+09] in which images are annotated with labels.

On the contrary, it is challenging to create big datasets for learning-based approaches to image segmentation. Such datasets require pixel-accurate labeling of thousands of images by human observers. This is the reason why researchers have turned their attention to *weakly supervised segmentation methods* such as [Baz+16; BV16; CVS17; Li+16a; Oqu+15; Zho+16] that take advantage of training on labeled images without any pixel-wise ground-truth information (Figure 5.7). The goal is still to provide an accurate segmentation without relying on the availability of large-scale segmentation datasets.

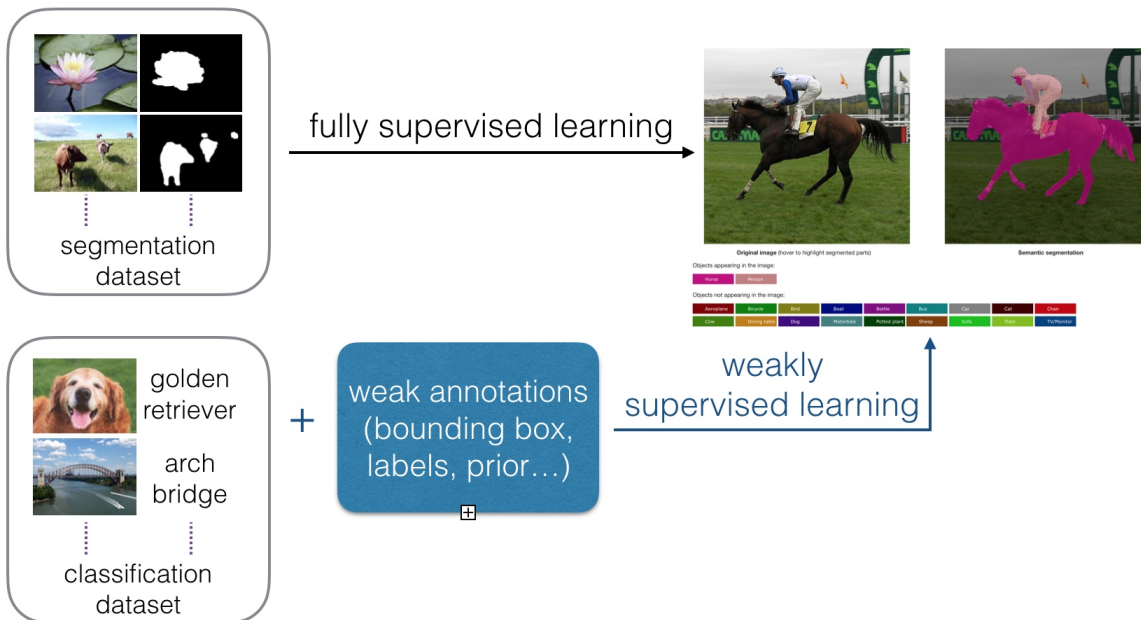


Figure 5.7: An illustration of fully supervised learning versus weakly supervised learning on the image semantic segmentation task.

Zhou *et al.* showed in [Zho+16] that some localization information about the main object, i.e., the object with the highest classification score, can be extracted from a DCNN that had only been trained on image classification. Their technique is based on computing a class activation map (CAM), which identifies those regions in an image that leads the classification network to make a specific prediction about the image label.

Our work goes a step further by providing a high-resolution CAM that not only localizes all the instances of the main object in the image and but also provides shape information

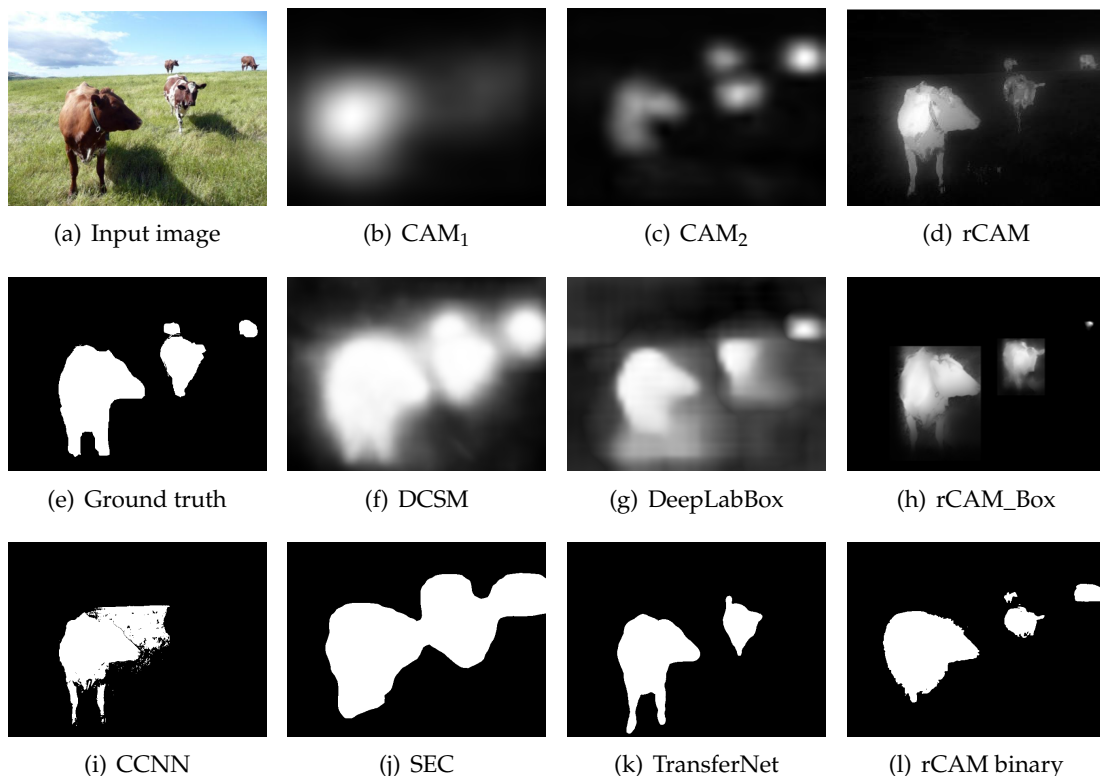


Figure 5.8: Illustrating the behavior of the proposed rCAM based shape extraction: To detect the main objects in 5.8(a), the classical CAM method [Zho+16] provides the rough location of the foreground cow in 5.8(b). Smaller cows are located in 5.8(c) when the input image is upsampled. However, neither 5.8(b) or 5.8(c) is accurate enough to provide objects' shape. Our extensions in 5.8(d) and 5.8(h) provide segmentations that are at least as detailed as the competing methods [SY16] and [Kho+17] shown in 5.8(f) and 5.8(g), respectively, while generalizing well over a wider variety of different datasets. Our rCAM binarized version in 5.8(l) also shows better result than those methods that produce binary segmentation maps such as [PKD15], [KL16a] and [Hon+16] in 5.8(i), 5.8(j) and 5.8(k) respectively.

which is accurate enough to be used for image segmentation without requiring any additional training (see Figure 5.8). Opposed to the original CAM method [Zho+16], the proposed method is able to locate the whole object body rather than only discriminative regions, which often cover only parts of the objects. For example, animals' heads are the most discriminative parts and can be effectively used to classify different animals. However, we aim to discover the whole animal body rather than just its head.

Note that our method still differs from *semantic image segmentation*, where every pixel in an image is classified, and from *object segmentation*, where all the objects in an image are segmented. The proposed method segments all instances of the main object in an image only. To bridge the gap between our method and the segmentation methods, we apply our high-resolution CAM algorithm on region proposals produced by Faster-RCNN [Ren+15]. In either case, our method is comparable to the state-of-the-art weakly supervised segmentation methods, which are intensively evaluated in Section 5.2.3. Although our method does not contain any fine-tune training stage of the classification network, it performs favorably

in comparison to previous CAM methods and state-of-the-art, weakly supervised segmentation methods, particularly with respect to the ability to generalize across different datasets.

Our proposed method can be summarized in four steps: (i) Firstly, we create two CAMs at different scales from two different resolutions of the input image using the GoogLeNet-GAP network [Zho+16]. (ii) We extract the shape information from GoogLeNet-GAP using a principal component analysis (PCA) on a particular set of response maps. (iii) The two CAMs are upsampled by the guided filter [HST13] that uses the extracted shape information. (iv) The upsampled CAMs are merged to create a high-resolution class activation map. Finally, we use the Conditional Random Field in [Zhe+15] to improve the accuracy of the shape prediction.

5.2.1 Related Work

The difficulty of creating large-scale image segmentation datasets for training deep neural networks on the one hand and the urgent need to extract localization and shape information from images, on the other hand, has sparked two lines of research, namely localization and weakly supervised segmentation. CAM methods, which are a subset of localization methods, try to localize objects by identifying pixels that activate the class of interest. Alternatively, weakly supervised segmentation techniques use different constraints and information that is less than segmentation ground truth to train or fine-tune DCNNs to perform segmentation tasks. Our work falls in between these two types of approaches.

Understanding DCNNs and Class Activation Maps

In order to have a better understanding of the image classification process, various works identify the most important pixels used by a DCNN to classify an image. Bazzani *et al.* [Baz+16] apply masks at different locations on an image and classify each result. They study the link between the positions of the masks and the classification scores to localize objects. Simonyan *et al.* [SVZ13] predict a heat map by altering the input image. Oquab *et al.* [Oqu+15] use a particular DCNN composed of a fully convolutional network which outputs K images, where K is the number of classes, followed by a global max pooling (GMP) and then a fully connected layer. Thanks to the K images before the fully connected layer, Oquab *et al.* localize the pixels that activate the class. Similarly, Zhou *et al.* [Zho+16] proposed a DCNN architecture, illustrated in Figure 5.12(a), that is able to classify an image. While their architecture is similar to GoogLeNet [Sze+15b], a global average pooling (GAP) followed by a fully connected layer is used after the fully convolutional network. According to [Zho+16], GAP provides better localization results than GMP. Selvaraju *et al.* [Sel+17] propose a technique to extract the discriminative pixels based on the gradient of a DCNN. Based on CAMs produced by Zhou *et al.* [Zho+16], Wei *et al.* proposed an adversarial erasing method to iteratively expand the discriminative object regions [Wei+17]. Their mined regions are then used to train semantic segmentation. All the above techniques aim to localize the most important pixels used by a DCNN to classify an image. However, they can only provide very crude estimations of the objects' shape.

Weakly Supervised Object Segmentation

Recent works [Hon+16; Kho+17; KL16a; Oh+17; PKD15; SY16] have explored weakly supervised object segmentation. While weakly supervised learning algorithms do not have access to the complete (semantic) segmentation of the training images, they vary strongly based on the amount of training images and the types of annotations. [KL16a; Oh+17; PKD15; SY16] use image class labels only, which provide information about labels of objects that are present in each image, but do not contain any localization information. More information can be exploited via bounding boxes as for instance, in [Kho+17]. [Hon+16; Oh+17; SY16] learn shape information from other databases to improve semantic segmentation results. Other techniques like [KL16a; PKD15] add some constraints on the shape of the objects. These constraints are used as priors in order to improve the segmentation results.

Class Activation Map (CAM)

When Deep Convolutional Neural Networks (DCNNs) started to take off, they were considered as black boxes that somehow can produce impressive results, especially in classification tasks. To understand DCNNs, what are inside the DCNNs, and how they can perform tasks well, researchers have looked inside different parts of DCNNs and tried to visualize their information. In classification tasks, the big question is how a DCNN makes a prediction. For example, where the DCNN looks at to get discriminative features to decide an image contains a dog. Zhou *et al.* [Zho+16] proposed a method to find discriminative regions that a DCNN uses to predict classes. These regions are displayed by a map called Class Activation Map (CAM). The process of creating CAM is described in Figure 5.9.

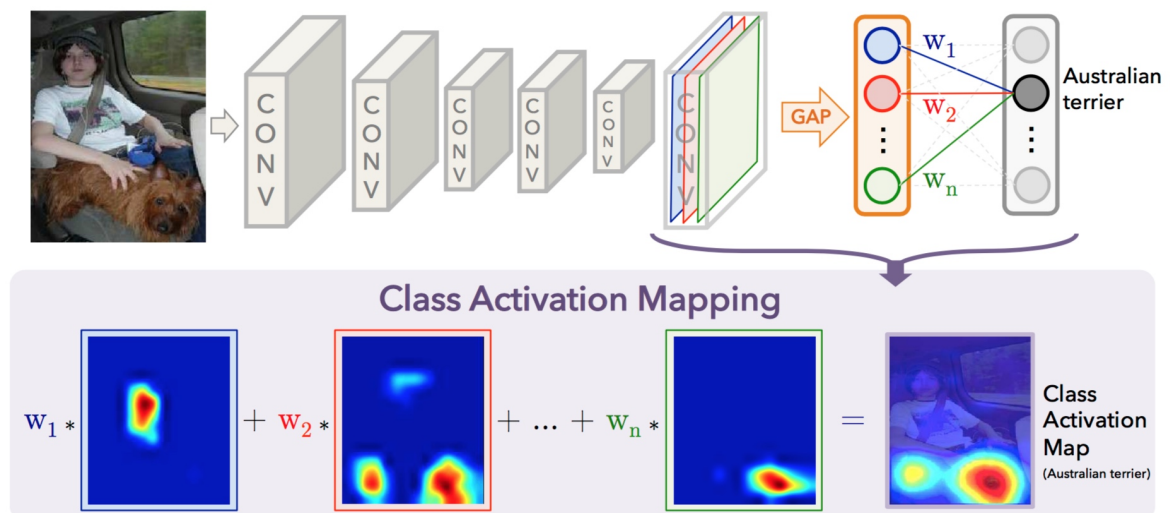


Figure 5.9: A Process of an existing Class Activation Map (CAM) extraction method proposed by Zhou *et al.* [Zho+16]. (Source image: [Zho+16])

The process starts with GoogLeNet [Sze+15b] or similar networks such that they consist of a series of convolutional layers, followed by a Global Average Pooling (GAP) that connected to the final prediction layer. In the last convolutional layer, the network produces n

response/feature maps. The GAP layer will compute the average for each map and, therefore, return a feature vector of size n . The weighted sum on the values of this feature vector yields the class probability results that are usually normalized with the Softmax function (See the upper part of Figure 5.9).

The weight w_k^c tells how important a k^{th} component of the feature vector's contribution to the prediction for class c is. In other words, it tells how important the k^{th} response/feature map in the last convolutional layer contribute to the prediction for class c . Zhou *et al.* proposed to use these weights w_k^c on the response/feature maps of the last convolutional layer to construct the class activation map (the lower part of Figure 5.9). To extract a CAM for a particular class c , we only need to use all the weights w_k^c of class c (Figure 5.10). Some examples of CAM results for action recognition are shown in Figure 5.11.

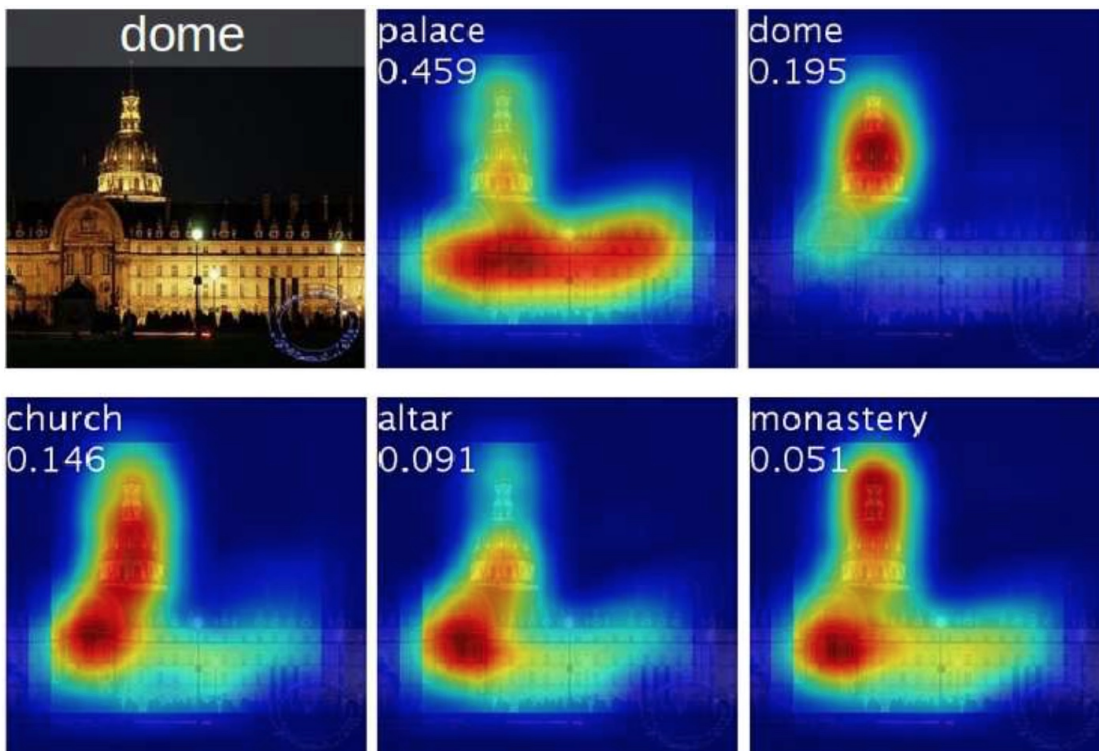


Figure 5.10: Results of CAMs for different prediction classes of the same input image. (Source image: [Zho+16])

It is impressive to be able to visualize the discriminative regions that a DCNN uses for its predictions using a simple and elegant solution. However, the method only can produce low-resolution CAM. This CAM method does not work well if there are different instances of the same objects or objects with different resolutions or sizes. In the next part, we will propose our method based on CAM to extract shape and construct high-resolution CAM. Our method can also successfully perform weakly supervised semantic segmentation.



Figure 5.11: Results of CAMs for predicting actions. For the teeth brushing action, the CAMs show regions that contain part of a hand holding a whole toothbrush. For the cutting tree action, the CAMs show regions contain a face of a human and the chainsaw. (Source image: [Zho+16])

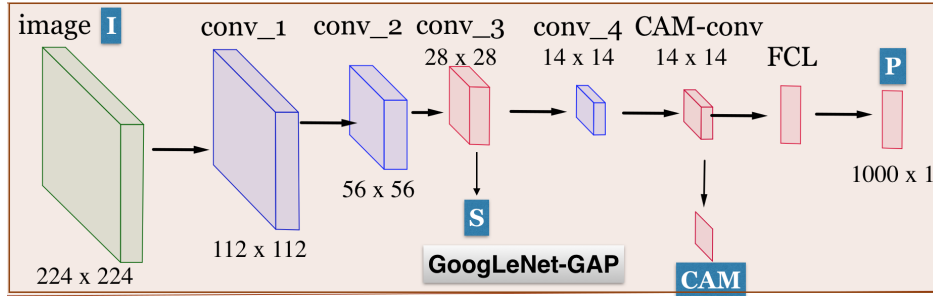
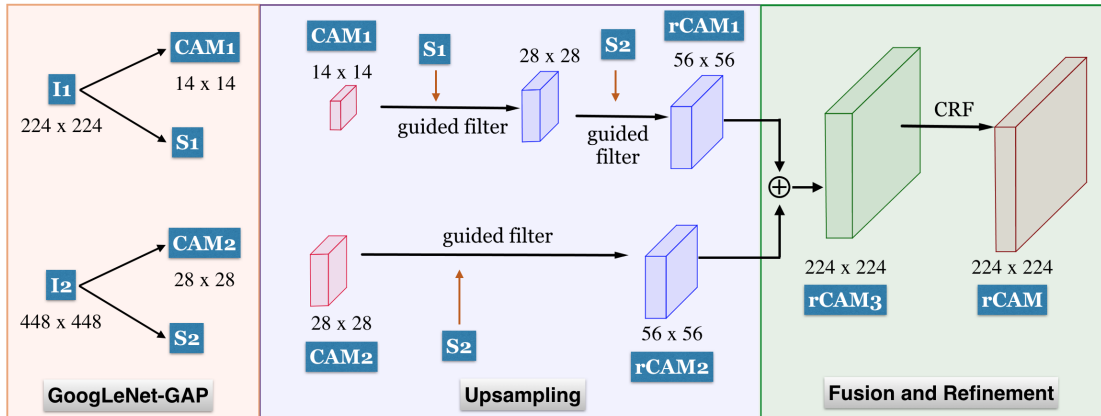
5.2.2 High Resolution Class Activation Map (rCAM) and Shape Extraction

In this section, we present our method for producing high-resolution class activation maps (rCAMs) that not only localize the main object in an image but also predict its shape accurately. The proposed method is based on extracting shapes from the GoogLeNet-GAP network [Zho+16] and using such information together with multi-scale CAMs to increase their resolution. The processes and overall structure of the framework are illustrated in Figure 5.12. It consists of extracting CAMs and shapes at two different scales, using the shape information for an upsampling of the activation maps, and finally fusing and refining the latter to obtain the rCAM result.

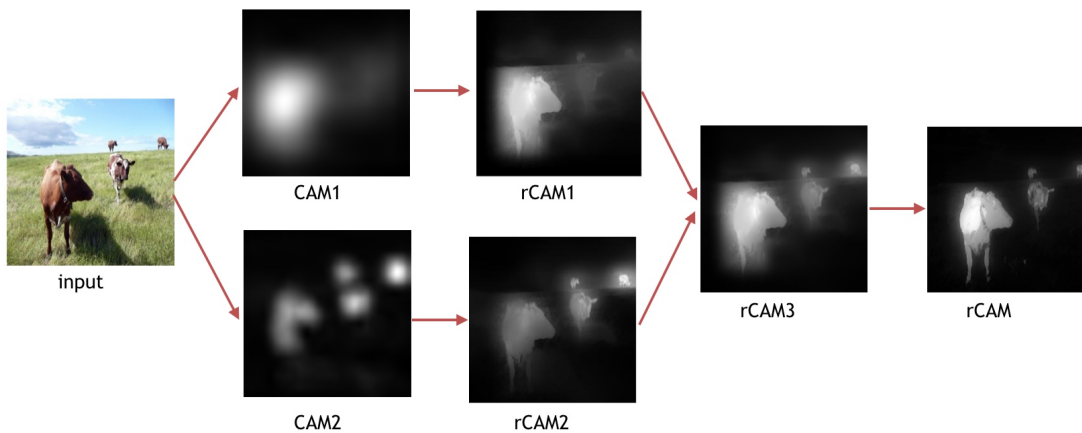
Multi-scale CAMs Extraction

We use the GoogLeNet-GAP network to create CAMs [Zho+16] as the basic components for constructing rCAM. The GoogLeNet-GAP mainly consists of convolutional layers. After the last convolutional layer, a Global Average Pooling (GAP) is performed, and the GAP results are fed into a fully connected layer for the final classification producing a 1000-dimensional vector denoted P , which holds the class probabilities for the classification result. Let us denote C_{CAM} the set of response maps of the CAM layer and w_{ij} the fully connected weight connecting the response map i (denoted C_{CAM}^i) and the coordinate j of P . The CAM of the class j at the position x is defined in [Zho+16] as: $\text{CAM}(x)^j = \sum_{i=1}^N w_{ij} C_{\text{CAM}}^i(x)$, where N is the number of response maps of the CAM layer. For an input image I of size 224×224 , GoogLeNet-GAP produces CAM of size 14×14 that localizes the first object with the highest classification probability (Figure 5.12(a)).

Instead of using a single scale we resize every input image to images I_1 of size 224×224 and

(a) GoogLeNet-GAP with an input image size 224×224 

(b) Upsampling, Fusion and Refinement process



(c) An example of results at every stage of the whole process

Figure 5.12: Overview of the proposed rCAM method. The input image is fed into a GoogLeNet-GAP network, [Zho+16], operating on two different scales 224×224 and 448×448 . They produce the class activation maps CAM_1 and CAM_2 , the shape information maps S_1 and S_2 , and the class probability maps P_1 and P_2 , respectively. In the upsampling process (middle part of (b)), S_1 and S_2 are used as guidance images for a guided filter [HST13] that upsamples CAM_1 and CAM_2 to $rCAM_1$ and $rCAM_2$. In the fusion and refinement process (right part of (b)), $rCAM_1$ and $rCAM_2$ are combined to create $rCAM_3$ and finally, the $rCAM$ is produced by applying a dense Conditional Random Field (CRF) [Zhe+15] to $rCAM_3$. The example results show that $rCAM$ s resolutions are higher than CAM such that shapes can be perceived. With input size 448×448 , we are able to discover multiple instances that have small sizes in the input images. By combining $rCAM$ s at different resolutions and refining the result by using CRF, we obtain a clear shape and high resolution $rCAM$.

I_2 of size 448×448 by bilinear interpolation. The images I_1 and I_2 are feed-forwarded to GoogLeNet-GAP to generate CAM_1 of size 14×14 and CAM_2 of size 28×28 , respectively (Figure 5.12(a)). We discover that while CAM_1 provides the coarse discriminative regions for the main object, CAM_2 gives us finer discriminative regions that are sometimes overlooked by CAM_1 (see Figure 5.16 for an example).

The usage of the image I_2 of size 448×448 creates a zoom-out effect. The dominance of the discriminative regions discovered in the image I_1 of size 224×224 is reduced, and the finer discriminative regions have an opportunity to be discovered in the image I_2 . According to our experiments, CAM_2 is especially useful when there are multiple instances of the main object, for example, many cows in the image in Figure 5.8. On the other hand, CAM_1 is very important for the classification and localization of the main object due to the suppression of small objects. Therefore, CAM_1 and CAM_2 do not compete but complement each other.

Shape Extraction from GoogLeNet-GAP

Traditionally, object recognition or shape estimation uses hand-crafted features such as SIFT [Low99], or descriptors like the color, texture, or gradient of an image. The robustness of a method is based on the invariance of such features to factors such as scale, illumination, or rotation. However, in DCNNs, one does not need to define features. Instead, the features are learnt and embedded inside DCNNs for us to discover [Goo+09; Zho+15].

A DCNN can be divided into two parts. The first part involves a set of layers that form a Fully Convolutional Network (FCN). Each layer in FCN contains a series of convolutional operations followed by non-linear operators such as activation and pooling. The second part consists of Fully Connected Layers (FCL), which lead to the classification results. We focus on the FCN of GoogLeNet-GAP. The output of each convolution kernel in the FCN is a response map. Our goal is to find a set of response maps that contain shape information and extract the shape.

The FCN of the GoogLeNet-GAP architecture is a concatenation of convolution and pooling layers: for an input image I_1 of size 224×224 , it produces response maps of sizes 112×112 , 56×56 , 28×28 and 14×14 . By gathering all these response maps into four groups according to their sizes, we have four sets of response maps C^l with $l \in \{112, 56, 28, 14\}$. Each C^l is a cubic tensor such that $C^l \in \mathbb{R}^{l \times l \times D_l}$, where D_l is the number of response maps of size $l \times l$. Therefore, C^l can be decomposed into l^2 vectors v_k where $v_k \in \mathbb{R}^{D_l}$ and $k \in [1, l^2]$.

To condense the information of the feature maps C^l , we apply a Principal Component Analysis (PCA) [Jol02] to reduce the dimension of v_k from D_l to 3 by extracting the first three components, mapping $C^l = \{v_k\}_{k=1}^{l^2} \subset \mathbb{R}^{D_l} \mapsto \tilde{C}^l = \{\tilde{v}_k\}_{k=1}^{l^2} \subset \mathbb{R}^3$. The resulting principal components represent the response maps C^l by more compact sets \tilde{C}^l and yield a better understanding of the information contained in each of the feature maps, see Figure 5.13.

In order to find response maps that contain shape information, we build a small database that is composed of 200 binary shape images, as illustrated in Figure 5.14. We perform

color and texture transformations on these shape images and study how the response maps change when the color and texture information varies (Figure 5.15).

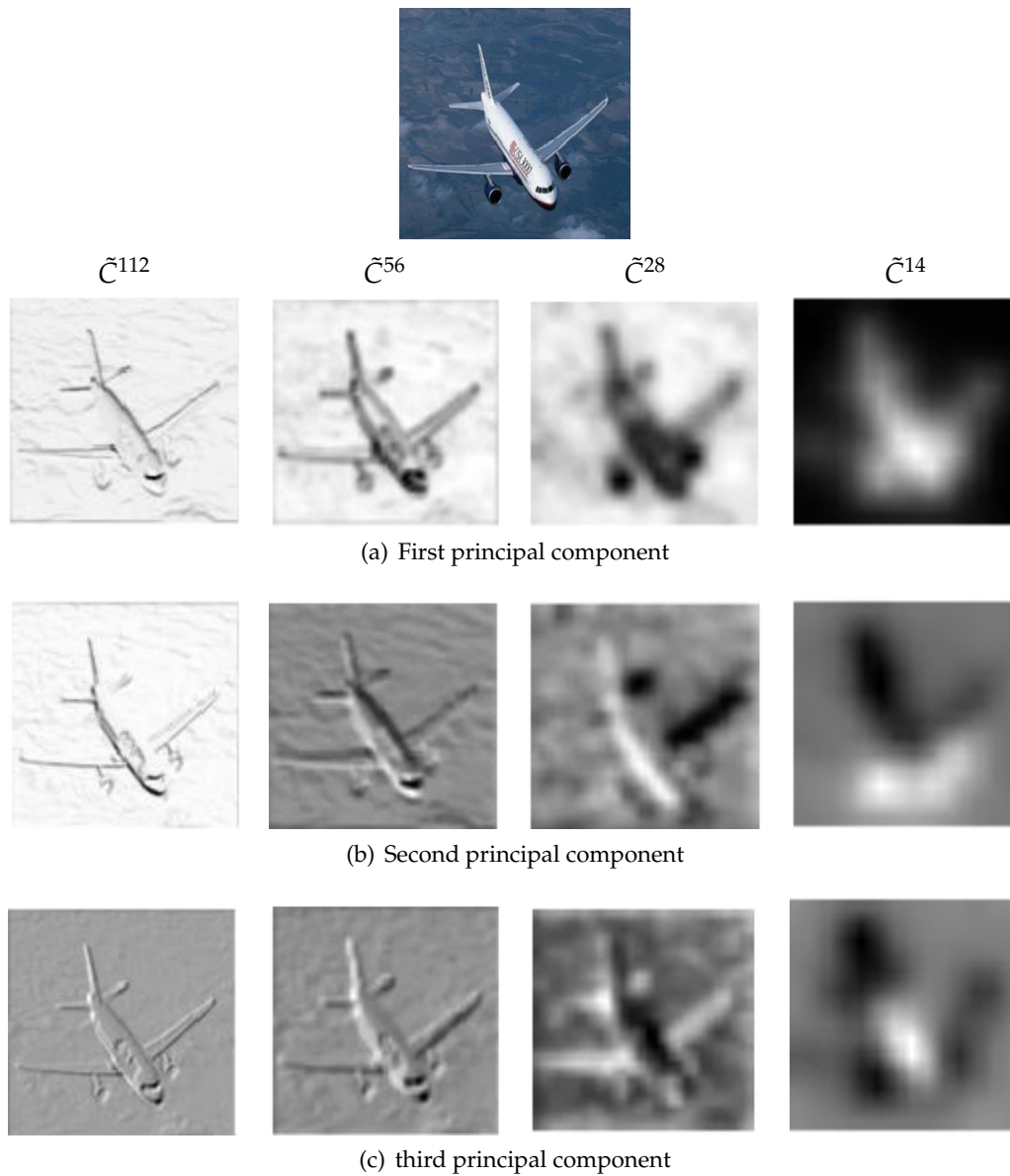


Figure 5.13: Illustration of the first three principal components of layers C^{112} , C^{56} , C^{28} , and C^{14} . While C^{112} and C^{56} yield gradient information, C^{28} and C^{14} contain mostly shape information.

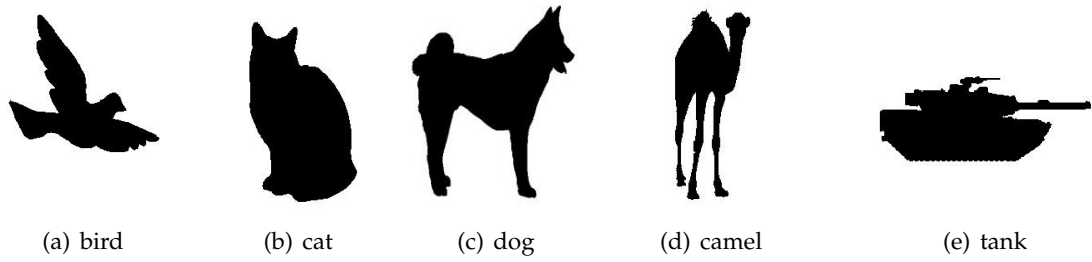


Figure 5.14: An example of different shape images using white background since ImageNet contains some images that have the objects with white background.

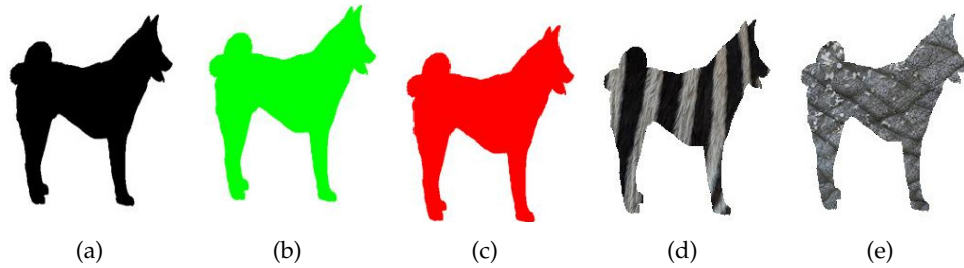


Figure 5.15: An example of a shape image with different colors and textures. (a) Initial shape with white background. (b) and (c): shape images with different colors. (d) and (e): shape images with different textures.

Let F_i be a family of color transformations, where $i \in [1, n]$ and n is the number of transformations. For all n color transformations, we compute the variance V of each response map j as follows:

$$V(j) = \frac{1}{200 \cdot |\Omega_j|} \sum_{x \in \Omega_j} \sum_{k=1}^{200} V_k(x), \quad (5.2)$$

$$V_k(x) = \frac{1}{n} \sum_{i=1}^n \left(\phi_j(F_i(I_k)) - \frac{1}{n} \sum_{i=1}^n \phi_j(F_i(I_k)) \right)^2 (x), \quad (5.3)$$

where $F_i(I_k)$ is the result of color transformation i applied on image I_k . $\phi_j(F_i(I_k))$ is the response map resulting from convoluting kernel j on $F_i(I_k)$. x is the pixel position of the response map ϕ_j . Ω_j is the spatial domain of ϕ_j and $|\Omega_j|$ is the number of pixels in ϕ_j . We compute the variance for texture transformations in the same way. We want to find a layer that has the smallest variance.

According to our numerical experiments \tilde{C}^{112} and \tilde{C}^{56} contain mainly gradient information, \tilde{C}^{28} provides shape structures, and \tilde{C}^{14} yields a heat map revealing the location of the main object. This leads us to define $S_1 := \tilde{C}^{28}$ to be a shape representation of the input image I_1 .

By feeding an input image I_2 of size 448×448 into the network and performing a PCA of the feature maps, one again obtains four compact response maps whose resolution is

four times larger than the resolution of the corresponding feature maps of I_1 . Again, the shape information S_2 is defined as the compact response map of the third layer, such that $S_2 \in \mathbb{R}^{56 \times 56 \times 3}$.

We use the shape information S_1 and S_2 to guide the upsampling process in Section 5.2.2. Interestingly, our numerical experiments indicate that the main shape information can be found in the first principal component on about 70% of the images. On the other 30%, shapes can be found in the second or third principal component. Sometimes, the shapes can also be contrast inverted, as shown in Figure 5.13. In the next section, we will show how the shapes S_1 and S_2 can be used to increase the resolution of CAMs in order to provide localization and shape information in one high-resolution image.

Upsampling Using Guided Filters

Localization results from CAMs are expressed in the form of blobs of discriminative regions (see CAM_1 results in Figure 5.16). They may contain only parts of the objects, for example, heads of the animals, rather than the whole objects' bodies. Besides that, the blob regions cannot depict the shapes well. To solve these two problems, we use the shape information recovered from GoogLeNet-GAP to guide the process of increasing the CAMs' resolution. The results we achieve are rCAMs that localize the main objects as a whole and make the objects' shapes perceivable. The process to increase resolution is illustrated in Figure 5.12(c).

The guided filter proposed in [HST13] is an image processing operator that smoothens images while preserving sharp edges using a guidance image G . It relies on the assumption that inside a local window w_k that is centered at pixel x_k , there is a linear model between the guidance image G and the output image O as defined in [HST13]. Hence, the guided filter preserves edges from the guidance image while being independent of its exact intensity values. This is an important property because the shape information that we extract from GoogLeNet-GAP can be contrast inverted.

However, similar to non-parametric kernel regression [Alt92], the size of the window w_k is very important. If the window size is too big, a large number of observations will be considered during the regression process, and it leads to an over-smoothed estimation of the output O . If the window size is too small, the output O will depend on too few observations, which leads to a high variance solution. To find the optimal values for the window sizes, we estimate them on the shapes S_1 and S_2 using the variogram proposed in [Cre85].

We assume that S_1 and S_2 follow a random process that is homogeneous and has second-order stationary properties. That implies that two observations of the random process are independent of their locations and only depend on their spatial distance. To measure the spatial dependence of the data, we use the empirical variogram defined as follows:

$$\hat{\gamma}(\mathbf{h}) = \frac{1}{2|N(\mathbf{h})|} \sum_{i,j \in N(\mathbf{h})} (S_1(x_i) - S_1(x_j))^2 \quad (5.4)$$

where $N(\mathbf{h})$ is the set of observations pairs (i, j) such that $\|x_i - x_j\| = h$, which is the spatial distance between two observations, and $|N(\mathbf{h})|$ is the cardinality of this set.

This empirical variogram $\hat{\gamma}$ is approximated by a model function:

$$\gamma(h) = c_1 \cdot \left(\exp \left(-\frac{\|h\|^2}{2\sigma^2} \right) \right) + c_2 \quad (5.5)$$

The model function increases the generalization power of the empirical estimator. Three parameters c_1, c_2, σ are estimated such that the variogram function fits the empirical one. The σ parameter provides information about the average size of objects. So we use σ as the size of the filter. As a result, the size of our guided filter is adapted to each image.

In order to double the resolution of a CAM using a shape prior S , we first double the size of the CAM by bilinear interpolation. Then we apply a guided filter on the upsampled CAM using S as the guidance image – a process which we denote by $G^f(U^2(CAM), S)$ where U^2 is the upscaling bilinear interpolation with a factor of 2 and G^f is the guided filter process.

We increase the resolution of CAM_1 of size 14×14 using guided filters as follows:

$$C\tilde{A}M_1^{28 \times 28} = G^f(U^2(CAM_1), S_1), \quad (5.6)$$

$$rCAM_1^{56 \times 56} = G^f(U^2(C\tilde{A}M_1^{28 \times 28}), S_2), \quad (5.7)$$

where S_1 and S_2 are shapes extracted from GoogLeNet-GAP and used as guidance images. The CAM_2 extracted from the higher resolution input image is of size 28×28 already and is further upsampled via

$$rCAM_2^{56 \times 56} = G^f(U^2(CAM_2), S_2). \quad (5.8)$$

As the result, we increase the resolution of both, CAM_1 and CAM_2 , to $rCAM_1^{56 \times 56}$ and $rCAM_2^{56 \times 56}$ both of which are of size 56×56 .

As explained in Section 5.2.2, CAM_1 and CAM_2 complement each other in providing coarse and fine discriminative regions – a property that is preserved during the proposed upsampling, see Figure 5.16. Therefore, it is beneficial to combine two of them in order to take advantage of both.

Fusion and Refinement

Our goal is not only to provide high-resolution in localization and shape but also to discover all the main object's instances. To achieve the latter, we combine $rCAM_1$ and $rCAM_2$, which provide localization and shape information at different scales.

To do so, the $\text{rCAM}_1^{56 \times 56}$ and $\text{rCAM}_2^{56 \times 56}$ images described in the previous section are up-sampled to a resolution of 224×224 pixels using bilinear interpolation.

We fuse the resulting maps rCAM_1 and rCAM_2 via:

$$\text{rCAM}_3 = \text{rCAM}_1 \cdot P_1(\text{idx}_1) + \text{rCAM}_2 \cdot P_2(\text{idx}_1), \quad (5.9)$$

where idx_1 is the index of the highest classification score of the image I_1 , and P_1 and P_2 are vectors of classification probability for image I_1 of size 224×224 and image I_2 of size 448×448 , respectively. Therefore, $P_1(\text{idx}_1)$ is the classification probability of the highest predicted class for image I_1 and $P_2(\text{idx}_1)$ is the classification probability of the highest predicted class from image I_1 for image I_2 . The output is rCAM_3 that combines the advantages of both rCAM_1 and rCAM_2 .

To refine the accuracy of the shape prediction, we use the dense Conditional Random Field (CRF) implemented in [Zhe+15] on rCAM_3 . CRF is a classical tool often used to refine the segmentation from a coarse prediction to one that has well-defined boundaries. CRF minimizes an energy function that consists of two terms: unary and pairwise energy components as follows:

$$E(x) = \sum_i \psi_u(x_i) + \sum_{i < j} \psi_p(x_i, x_j) \quad (5.10)$$

where $\psi_u(x_i)$ is the unary cost of assigning a label x_i to a pixel i and $\psi_p(x_i, x_j)$ is the pairwise cost of assigning labels x_i, x_j to pixels i, j respectively. The pairwise cost creates a smoothing condition such that similar neighboring pixels will be assigned similar labels.

We first normalize rCAM_3 to $[0, 1]$ to create the probability map that indicates the main object's presence. We use rCAM_3 and $(1 - \text{rCAM}_3)$, which represent the foreground and background probability, respectively, as the inputs to the CRF algorithm. The inference output from the dense CRF is our final high-resolution rCAM.

5.2.3 Evaluations

Evaluation Datasets

Our proposed method delivers results in two aspects: main objects' location and shape. While many weakly supervised learning methods output bounding boxes for objects' locations, CAM and rCAM produce probability maps (heatmaps). Therefore, instead of evaluating CAM and rCAM methods on bounding box datasets, we use three datasets: Pascal-S [Li+14], FT [Ach+09] and ImgSal [Li+13]. These datasets provide locations and shapes of salient objects and are commonly used to evaluate salient object detection. Each dataset has its own characteristics. While the FT dataset mainly provides a single object in each image, Pascal-S includes multiple-object images. Pascal-S is also a fair choice for the evaluation because many weakly supervised segmentation methods are trained on Pascal VOC

2012 dataset [Eve+]. For more diversity, ImgSal contains not only single-object and multiple-object images, but also a fair amount of natural landscapes. ImgSal also contains objects that do not have the same labels as in Pascal nor ImageNet. It is the most challenging dataset for weakly supervised segmentation methods in our evaluation.

Evaluation Metrics

We use different F-measures [Arb+11] and Mean Absolute Error (MAE) [Per+12] to analyze the performance of various CAM methods as well as weakly supervised segmentation methods. For F-measures, we use the Optimal Image Scale (OIS) and Optimal Dataset Scale (ODS) [Arb+11]. OIS is computed using the best threshold for the individual image, while in ODS, an optimal threshold is selected on the whole dataset. Despite the fact that OIS and ODS use different approaches in selecting optimal thresholds, both F-measures are calculated using the same formula in Eq. (5.11).

$$F_{\beta} = \frac{(1 + \beta^2) \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}, \quad (5.11)$$

where $\beta^2 = 0.3$ as suggested in [Ach+09].

While F-measure metrics use the binarized heat map with optimal thresholds, the Mean Absolute Error (MAE) proposed in [Per+12] measures the error of the original heat map without thresholding to the binary ground truth. The results are then averaged for all the images.

It is important to note that higher numbers of F-measures indicate improved results, whereas, with MAE measurement, the smaller value is better.

Numerical results for various CAM methods

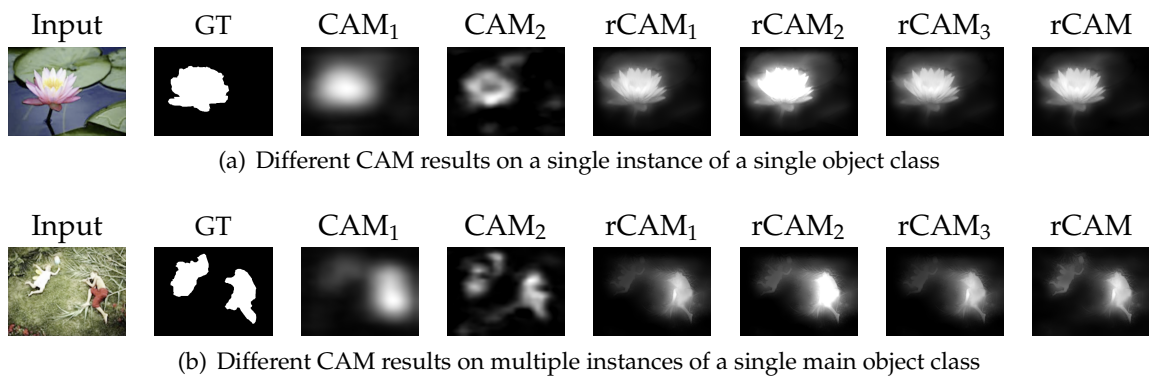


Figure 5.16: Localization and shape extraction results from various CAMs. The results are shown for single object and multi-object images.

We analyze the performance of different CAM methods on the Pascal-S, FT, and ImgSal datasets. The results in Table 5.1 show that the CAM method with input resolution 448×448 does not produce better results than CAM with input resolution 224×224 . It can be viewed

Pascal-S								
Metric	G-Weak	CAM ₁	CAM ₂	rCAM ₁	rCAM ₂	rCAM ₃	rCAM	% increase
OIS	0.398	0.682	0.684	0.725	0.733	0.736	0.773	13.34%
ODS	0.339	0.566	0.566	0.617	0.613	0.625	0.665	17.49%
MAE	0.395	0.298	0.338	0.290	0.314	0.291	0.276	-
FT								
Metric	G-Weak	CAM ₁	CAM ₂	rCAM ₁	rCAM ₂	rCAM ₃	rCAM	% increase
OIS	0.506	0.710	0.660	0.789	0.751	0.792	0.878	23.66%
ODS	0.448	0.643	0.568	0.714	0.660	0.714	0.803	24.88%
MAE	0.367	0.223	0.280	0.206	0.250	0.215	0.160	-
ImgSal								
Metric	G-Weak	CAM ₁	CAM ₂	rCAM ₁	rCAM ₂	rCAM ₃	rCAM	% increase
OIS	0.388	0.509	0.502	0.577	0.574	0.597	0.623	22.40%
ODS	0.273	0.419	0.417	0.491	0.478	0.505	0.533	27.21%
MAE	0.330	0.247	0.250	0.231	0.247	0.237	0.188	-

Table 5.1: Results of various CAM methods on Pascal-S, FT and ImgSal datasets. G-Weak [Oqu+15]. CAM₁: CAM method [Zho+16] with the input size of 224×224 , CAM₂: CAM method [Zho+16] with the input size of 448×448 , rCAM₁: high resolution of CAM₁, rCAM₂: high resolution CAM₂, rCAM₃: combination of rCAM₁ and rCAM₂, rCAM: the result of applying CRF on rCAM₃. The best value for OIS and ODS measurements are 1. The ideal value for MAE is 0. The last column shows the relative improvement of rCAM in comparison to CAM₁ for the OIS, and ODS metrics.

as two CAMs at two different scales complement each other, rather than compete with each other. At the resolution of 224×224 , CAM₁ and rCAM₁ localize the main object at the largest size. At the resolution of 448×448 , CAM₂ and rCAM₂ can discover other smaller instances of the main object class or secondary feature locations of the main object class, if there is only one instance (Figures 5.8 and 5.16).

However, there are significant improvements between the existing CAM methods and the high-resolution CAMs. In more details, rCAM₁ (high resolution of CAM₁) is better than CAM₁ and rCAM₂ (high resolution of CAM₂) is better than CAM₂. By combining rCAM₁ and rCAM₂, the result (rCAM₃) is better than any of the individual rCAM₁ or rCAM₂. Finally, the evaluation results are topped by applying CRF on rCAM₃ to create our final high-resolution CAM (rCAM). On the other hand, G-Weak [Oqu+15] is the method that uses Global Max Pooling (GMP) [Oqu+15]. The results indicate that G-Weak yields a weaker performance than the CAM method which uses Global Average Pooling (GAP), and also a weaker performance than our method.

Weakly Supervised Segmentation Comparison

We divide the weakly supervised segmentation methods into two groups: the first group provides a binary segmentation for each class. The second group provides continuous values that represent the likelihood of the foreground (similar to a probability map after normalization to the range (0,1)). We call the first one Binary Map methods and the latter one Continuous Map methods. To evaluate the Binary Map methods, we set all the foreground classes to 1 and the background to 0. As a result, OIS and ODS measurements are the same for the Binary Map methods (Table 5.2).

Dataset		Pascal-S			FT			ImgSal		
Metric		OIS	ODS	MAE	OIS	ODS	MAE	OIS	ODS	MAE
G1	CCNN	0.530	0.530	0.231	0.276	0.276	0.176	0.169	0.169	0.099
	SEC	0.638	0.638	0.208	0.553	0.553	0.150	0.399	0.399	0.123
	TransferNet	0.735	0.735	0.156	0.714	0.714	0.120	0.442	0.442	0.119
G2	DCSM	0.708	0.607	0.293	0.234	0.207	0.245	0.341	0.308	0.220
	DeepLab_Box	0.781	0.716	0.318	0.805	0.747	0.329	0.564	0.503	0.356
	rCAM	0.773	0.665	0.276	0.878	0.803	0.160	0.623	0.533	0.188
	rCAM_Box	0.765	0.696	0.254	0.807	0.716	0.184	0.663	0.527	0.164

Table 5.2: Comparison results for different weakly supervised segmentation methods: CCNN [PKD15], SEC [KL16a], TransferNet [Hon+16], DCSM [SY16], DeepLab_Box [Kho+17] and our rCAM methods. The methods are divided into 2 groups: Binary Map (G1) and Continuous Map (G2) based on their CAM outputs.

The rCAM method that we describe in this chapter localizes and extracts the shapes of instances of the main object at different scales. To compare with weakly supervised segmentation methods, we use Faster-RCNN [Ren+15] to retrieve bounding boxes for all detected objects. We then apply rCAM algorithm on these bounding boxes. The evaluation for this approach is called rCAM_Box.

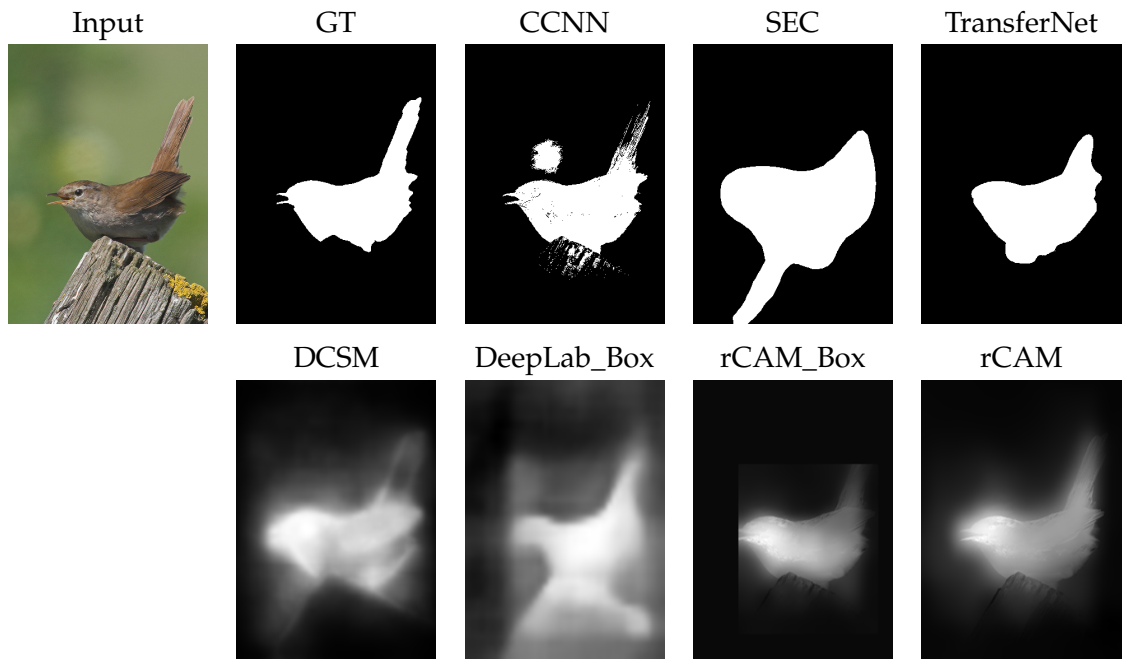
From the numerical results in Table 5.2, rCAM and rCAM_Box perform better than all the competing weakly supervised segmentation methods in terms of F-measures on FT and ImgSal datasets, on which none of the methods were trained. On the Pascal-S dataset, rCAM and rCAM_Box also outperform majority of the methods except DeepLab_Box [Kho+17] and TransferNet [Hon+16]. Similar to rCAM_Box, DeepLab_Box [Kho+17] method also segments object instances inside bounding boxes proposed by the Faster-RCNN network [Ren+15]. The performance of rCAM is inferior to DeepLab_Box [Kho+17] on the Pascal dataset by 2-3%. It is also shown that for methods that are trained only on image labels such as CCNN [PKD15], DCSM [SY16], and SEC [KL16a], the accuracies are consistently lower on all three datasets than the accuracies of methods that are trained using both image labels and segmentation ground-truth such as TransferNet [Hon+16] and DeepLab_Box [Kho+17]. We also observe a significant drop in performance from Pascal-S dataset to FT and more to ImgSal, especially for CCNN [PKD15], SEC [KL16a] and DCSM [SY16]. This reflects the limitation of these methods to generalize beyond the datasets on which they have been trained.

They are prone to fail for classes they have not seen during training. The proposed method is able to maintain a much higher accuracy across different datasets without the need for any weakly supervised training or fine-tuning. It is, therefore, much better suited for datasets, where the training data does not need to be highly representative of the test data.

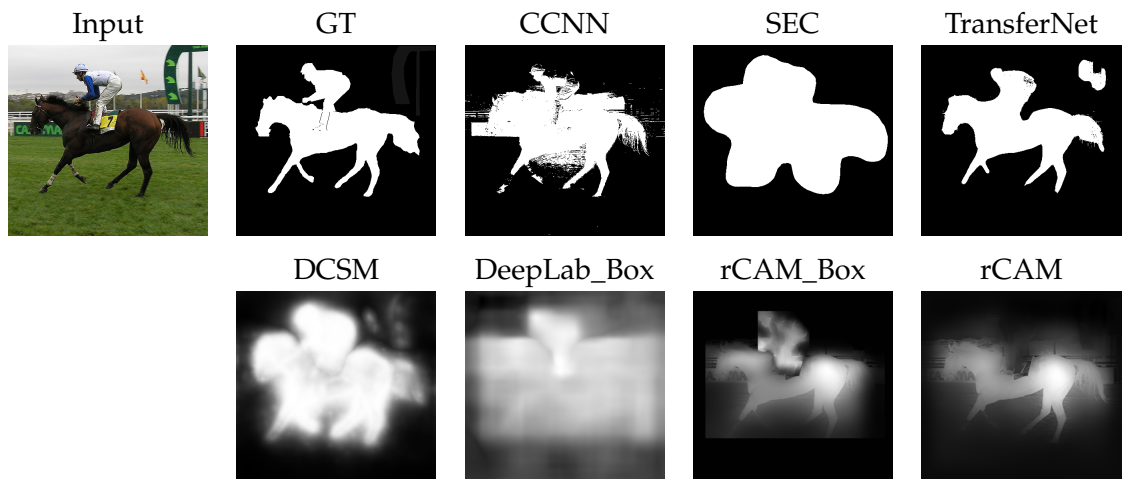
With the MAE metric, Binary Map methods such as CCNN [PKD15], SEC [KL16a] and TransferNet [Hon+16] have low error values. In the Continuous Map group, rCAM or rCAM_Box has the lowest errors. However, we observe that the Binary Map methods miss out more often all the segmented objects. As they cannot detect any object in an image, the output results contain only the background.

Different difficulty levels of segmentation are illustrated in Figure 5.17 and Figure 5.18. In the cases that objects' labels are in the Pascal dataset, all the methods perform relatively well even though some of the results lack some details in the shape information, e.g. CCNN [PKD15], SEC [KL16a] and DeepLab_Box [Kho+17] in Figure 5.17(b), or CCNN [PKD15] and SEC [KL16a] in Figure 5.17(c). If the object label is in ImageNet [Den+09] but not in the Pascal VOC 2012 [Eve+] dataset, an accurate segmentation becomes significantly more challenging: In Figure 5.18(a), DCSM [SY16] is unable to detect any object, and SEC [KL16a] as well as TransferNet [Hon+16] show degraded shape results. In the most difficult case where the object's label is neither in the Pascal VOC 2012 [Eve+] nor in the ImageNet [Den+09] datasets, none of the methods are able to produce reasonable results except our proposed rCAM and rCAM_Box methods (Figure 5.18(b)).

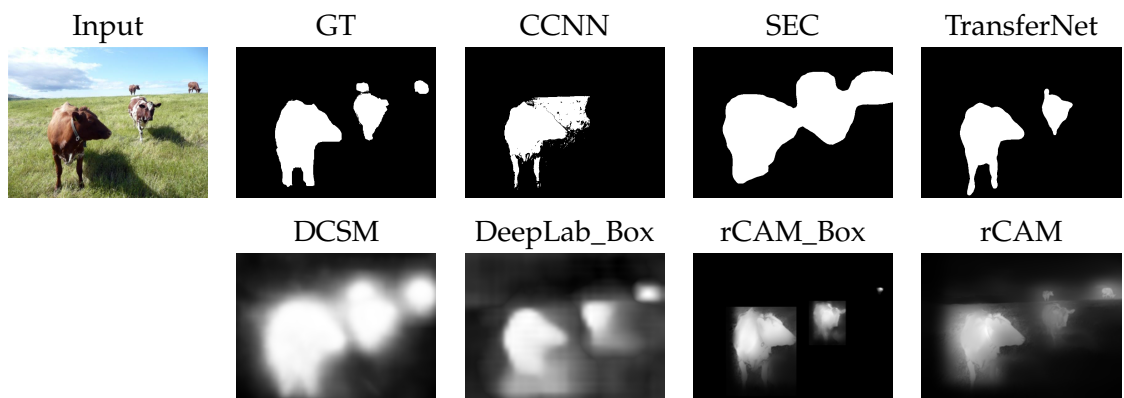
To understand the above results, it is important to note that all the weakly supervised segmentation methods that we use in our comparison are trained on the Pascal dataset. They all do well on Pascal, but their performance drops significantly when they are evaluated on different datasets such as FT and ImgSal. Although rCAM does not need training or fine-tuning on any dataset, its performance is already comparable to, if not better than, most of the competing methods on Pascal. Furthermore, rCAM is able to maintain the top performance on both FT and ImgSal, which demonstrates its robustness, as well as the ability to generalize to a wide variety of different types of data.



(a) Results on a single instance of a single object that is labeled in Pascal dataset

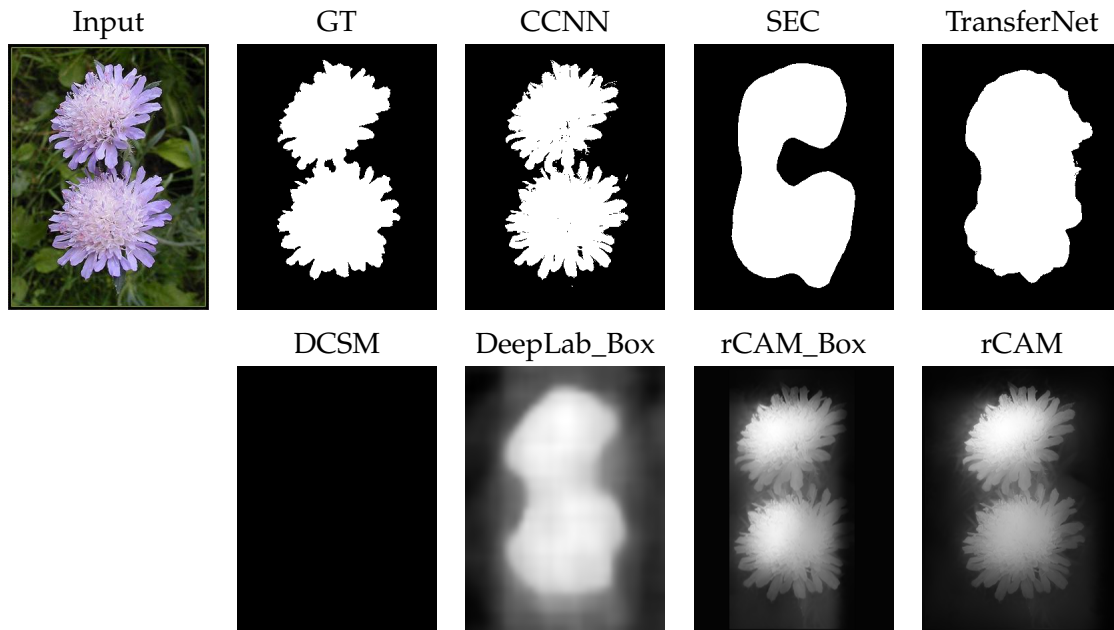


(b) Results on multiple objects that are labeled in Pascal dataset

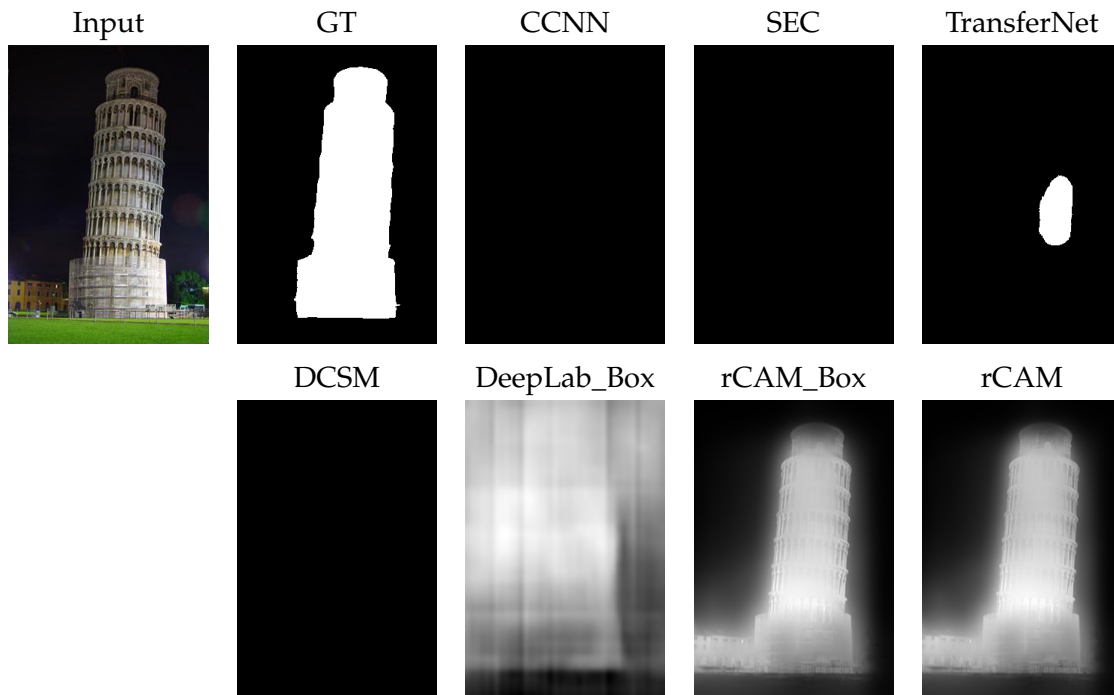


(c) Results on multiple instances of one main object that are labeled in Pascal dataset

Figure 5.17: Weakly supervised segmentation results from comparison methods for various scenarios for trained objects in Pascal dataset.



(a) Results on multiple instances of one main object that is not labeled in Pascal dataset



(b) Results on single object that the label is neither in Pascal or ImageNet

Figure 5.18: Weakly supervised segmentation results from comparison methods for various scenarios for untrained objects.

5.2.4 Chapter Summary

In this chapter, we study high-level deep features. We show that we can use features produced by a classification DCNN trained on 1,000 ImageNet classes to retrieve images of which labels are not part of training dataset. Furthermore, we proposed a method for extracting the location and shape information of all instances of the main object class in an image. To do so, we recover the primitive shape information from inside the GoogLeNet-GAP network. This shape information is used as guidance for the guided filter in our upsampling process to create high resolution class activation maps (rCAMs). We ascertain the benefits of using multi-scale rCAMs in our method, which does not require any extra training or fine-tuning. Our evaluation shows that, regardless of the simplicity, our proposed method outperforms existing CAM methods. Moreover, it performs on-par with competing state-of-the-art weakly supervised segmentation methods, while being far more robust to image data that is not well-represented by the training domain of the respective networks.

Declaration for Chapter 6 - Neural Discriminant Analysis

This work presented in this chapter is an independent work without any collaboration. It is entirely done by me, under the supervision of Prof. Dr. Volker Blanz. A part of the work is published at the International Conference on Image Processing (ICIP) in 2020: "Neural Discriminant Analysis for Fine-Grained Classification" [HB20].

I am currently collaborating with Dr. Gianni Franchi to extend the use of NDA to other research fields such as Semi-Supervised Learning (SSL) and Out Of Distribution (OOD) prediction. We achieve good results indicating that NDA helps in improving the performances in those tasks.

Chapter 6

Neural Discriminant Analysis

Feature learning is an important process in image and object classification, even more so for fine-grained classification where objects have a similar global structure and subtle differences between discriminative parts, such as between different breeds of dogs or different sub-types of birds. In this chapter, we study methods to improve the discriminant of deep features after training a Deep Convolutional Neural Network on a classification task. Inspired by Linear Discriminant Analysis (LDA), we propose a two-phase optimization to transform deep features from their original space to a lower-dimensional space using Neural Networks with two primary goals for inter- and intra-class variances. The first one is to minimize variances within each class. The second goal is to maximize pairwise distances between features coming from different classes. The approach produces more discriminative features that lead to improvements in classification performance.

The effectiveness of our method is shown through the evaluations of five popular fine-grained classification datasets. We mainly focus on fine-grained classification as our proposed optimization directly addresses its intrinsic challenges. At the end of the chapter, we present a way to combine two-phase optimization into a single-phase and evaluate it on CIFAR-10, a general classification dataset.

6.1 Introduction To Discriminant Analysis For Fine-Grained Visual Classification

The task of Fine-Grained Visual Classification (FGVC) is to classify subordinate classes under one common super-class. Examples are recognizing different breeds of dogs and cats [Kho+11; Par+12], sub-species of birds [Van+15; Wah+11], different models and manufactures of cars and airplanes [Kra+13; Maj+13; Yan+15], sub-types of flowers [NZ08] or natural species [Hor+18a] and so on. On the one hand, it is challenging because the subordinate classes share the same visual structures and appearance; the differences are very subtle. In many cases, it requires domain experts to distinguish and label these classes by recognizing their discriminative features for specific parts of the objects. Therefore, it is also a great challenge to obtain large-scale datasets for FGVC. On the other hand, the intra-class variance can

be visually higher than the inter-class variance. Such cases can be seen in different colors and poses of objects in the same class (Figure 6.1).



(a) All three are miniature poodles



(b) Siberian Husky vs. Malamute vs. Eskimo

Figure 6.1: Examples of a high intra-class variance in (a) and a low inter-class variance in (b). In Figure (a), the miniature poodles have different colors and poses. In Figure (b), three different breeds of dogs (Siberian Husky, Malamute and Eskimo) that are difficult to distinguish. Example images are selected from Stanford-Dogs dataset [Kho+11].

Research on FGVC has achieved remarkable results in the past few years due to advances in Deep Convolutional Neural Networks (DCNNs). The key to their success is in the ability to form and extract high-level features that are discriminative in large-scale image databases. High-level features are derived from various low-level features through convolutions and poolings. When adapting DCNNs models to FGVC, many works focus on transfer learning [Cui+18; ZTJ18], pooling techniques [Gao+16; LRM15; LM17], part based CNNs [Bra+14; Hua+16; Kho+11; Lin+15; Liu+12; Zha+16a; Zha+14a], etc. The common goal is to improve the discriminability of features that can address the subtle differences between classes and the high variance within classes, which is specifically challenging for FGVC.

One recent approach that produces state-of-the-art results is to extract a subset of the ImageNet dataset [Den+09] that is visually similar and relevant to fine-grained classification classes and combine it with other datasets to do transfer learning and fine-tuning [Cui+18; ZTJ18]. These methods are data-driven and take advantage of DCNNs. While the large-scale data empowers this domain adaptation and transfer learning approach, the task of learning discriminative features for FGVC is left to the DCNNs themselves.

Inspired by Linear Discriminant Analysis (LDA), we take on the method's objectives and combine it with the power of Neural Networks to improve the discriminative of features for fine-grained classification. Our goal is to optimize the learned features produced by transfer

learning or domain adaptation so that they become more discriminative and lead to better classification results. This optimization focuses on solving these three criteria:

- Reducing intra-class variance by minimizing the total distance between features of objects in the same class to their means.
- Increasing inter-class variance by transforming the feature space such that the distances between two classes in the target space are pushed further apart.
- Improving a classification DCNN's accuracy when integrating the optimizer to the DCNN.

To create an optimizer that satisfies those three criteria, we implement a small Neural Network with a two-phase optimization approach called Neural Discriminant Analysis (NDA). The first phase of the NDA optimization is to pull feature points within each class to the class center (class mean). The class mean optimization is the condition for the first criterion. In the second phase, we construct a Siamese network that uses the NDA network as a shared base. The Siamese optimization will pull pairs of features of the same class together and push pairs of features that belong to different classes apart. The Siamese includes both inter- and intra-class variance optimizations using pairwise data. The pairs of features in Siamese optimization are moved relative to each other. Thus, the intra-class constraint provided by the Siamese is weaker than the explicit constraint in the first phase, where all features of the same class are forced to move closer to their class mean. Therefore, the Siamese optimization is a strong condition for the second criterion, even though it also helps to hold the first criterion. NDA optimizes both phases alternately using the explicit intra-class variance constraint in the first phase and the Siamese optimization in the second phase, yielding better results than using only the Siamese optimization. Finally, we train a classification layer on the features produced by the NDA. This last step in re-train the classification layer is to achieve the last criterion.

We evaluate our NDA optimization on five different FGVC datasets. The main goal of NDA is to increase the performance of the original features, which are evaluated as the baseline. With that objective, NDA performs very well. Our results top the best transfer learning method up to 7.5% on four datasets. We also note that NDA surpasses state-of-the-art on two datasets, matches the state-of-the-art on one dataset, and are worse in the other two. The improvements are up to 5.4% over the state-of-the-art. We also compute standard deviations of the classification accuracy with and without NDA optimization. The statistics show that NDA optimization results in a more stable accuracy across different rounds of training with random initializations.

The advantages of our method are not only about the performance but also the ease of deploying NDA as a component to an existing network. We can connect NDA to the last feature layer of an existing DCNN and make it end-to-end. In this work, we introduce NDA optimization as an independent component that works on feature domains. We also develop another version of NDA in which the losses are weighted combined and integrate with the feature DCNN. We then train the DCNN end-to-end for classification.

Contributions: We propose an effective two-phase optimization for a new type of feature transformation, the Neural Discriminant Analysis (NDA). It optimizes for the objectives of Linear Discriminant Analysis (LDA), implemented as a fully-connected Neural Network. The NDA optimization helps to transform the original features such that they are more discriminative for FGVC by improving the clustering for each fine-grained class. Our two-phase NDA method significantly improves performances on several fine-grained classification benchmark datasets. As a small independent component, NDA can be easily added to the last feature layer of an existing DCNN to form an end-to-end network. We also combine two-phase into a single-phase NDA optimization that integrates with the feature DCNN for end-to-end training. The classification performances are improved significantly over the base-line in CIFAR-10 dataset for the general classification.

6.2 Related Work

6.2.1 Fine-Grained Visual Classification (FGVC)

In the field of FGVC, the inter-class differences are often subtle. Experts distinguish subordinate classes based on specific parts of the objects. Therefore, a straight-forward approach is to learn features of object parts [Bra+14; Che+19; Far+11; Hua+16; Kho+11; Kra+14; Lin+15; Liu+12; Par+11; Zha+16a; Zha+14a; Zha+14b; Zha+15a]. This approach often requires heavy part annotations from domain experts and therefore it is difficult to extend to larger scale datasets, or relies on object detection and localization [GLY19]. Some other works rely on attribute annotations and text descriptors [HP17; Ree+16; Ved+14; Xu+18a]. Stepping away from those types of annotations, another set of works focus on learning and using visual attention on discriminative regions [FZM17; Liu+16; PHZ18; Sun+18; Xia+15; Yan+18; Zha+17; Zhe+17; Zhe+19]. Analyzing the filter responses from DCNNs has also led to good part descriptors and localization [Wan+15; Zha+16b]. Utilizing the internal responses from DCNNs, different pooling techniques have also been developed such as bilinear pooling to study the interactions of sets of local features [CZZ17; Cui+17; Gao+16; LRM15; LM17; Wei+18; Yu+18].

Besides the high intra-class and low inter-class variance challenge, FGVC also faces a problem from small datasets. A small size dataset with high intra-class variation does not have enough training images to cover a variety of visual appearance such as dogs with different poses. Thus, it can easily lead to over-fitting in training with DCNNs [Dub+18]. In order to address this issue, researchers work on different strategies to collect more relevant images to enrich the datasets [GHF17; Kra+16; Xie+15; Xu+18b; ZTJ18] or employ human in the loop and human interaction to bootstrap datasets [Bra+10; Cui+16; Den+16].

Instead of collecting images from the web to augment the FGVC datasets, Cui *et al.* [Cui+18] and Zhang *et al.* [ZTJ18] use subsets of ImageNet [Den+09] that are visually similar to FGVC classes, in combination with iNat [Hor+18b] or L-Bird [Kra+16] dataset to do transfer learning and fine-tuning. These methods hold state-of-the-art results in several benchmark

datasets. This approach confirms the importance and effectiveness of discriminative feature learning from large-scale datasets.

From high level features produced by pre-trained DCNNs for classification such as Inception [Sze+16], ResNet [He+16a; He+16b] and Inception-ResNet [Sze+17], to name a few, as well as FGVC transfer learning [Cui+18; ZTJ18], we take a step further by transforming deep features in order to push instances between classes apart and bring instances from the same class closer together. Our approach is inspired by the Linear Discriminant Analysis (LDA). However, it is implemented in a neural network architecture, thus the term Neural Discriminant Analysis (NDA).

6.2.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a popular linear data transformation technique in Machine Learning. The goal of LDA is to find a linear projection of data that helps to separate classes well. The objectives are to maximize the variance between classes and minimize the variance within each class. Therefore, the computation of LDA involves between-class and within-class scatter matrices.

Let x is a set of original data that has C classes, μ_i is the mean of each class C_i that has N_i sample data for the class i^{th} and μ is the global mean. The within-class S_W scatter matrix and between-class scatter matrix S_B are computed as follows:

$$S_W = \sum_{i=1}^C \sum_{j=1}^{N_i} (x_{i,j} - \mu_i)(x_{i,j} - \mu_i)^T \quad (6.1)$$

$$S_B = \sum_{i=1}^C N_i (\mu_i - \mu)(\mu_i - \mu)^T \quad (6.2)$$

The goal is to find a projection matrix P to project the original data to a new feature space that maximizes the between-class variance and minimize within-class variance. The matrix P needs to satisfy Fisher's condition:

$$P_{LDA} = \arg \max_P \frac{|P^T S_B P|}{|P^T S_W P|} \quad (6.3)$$

The solution is that column vectors v in P_{LDA} are the generalized eigenvectors corresponding to the largest eigenvalues λ for the equation $S_B v = \lambda S_W v$. It can be re-written as $A v = \lambda v$ where $A = S_W^{-1} S_B$.

Similar to Principal Component Analysis (PCA), LDA is commonly used for dimension reduction but in supervised learning. After projecting the original features into LDA space,

the classification algorithms can be applied to these transformed features. As the LDA optimization and classification are different processes, it is difficult to have the LDA optimization cooperated in a CNN. LDA needs to compute statistics such as mean and variance on the whole dataset, whereas CNN training usually happens in batches due to the limits of hardware resources.

Therefore, we develop an optimization for CNNs using the objectives of LDA, which maximizes the inter-class variance and minimizes the intra-class variance. Even though CNNs can produce high classification performances on their own, there are no explicit constraints on the feature spaces for which they should be optimized. Our optimization is served as a mean to bridge the CNN features and the discriminant analysis objectives.

6.3 Two-phase Neural Discriminant Analysis (NDA)

6.3.1 Pre-optimized Features Extracted from Pre-trained DCNNs

Features for image classifications can be obtained from various types of network models. A straightforward approach is to extract features from models that are pre-trained on the ImageNet dataset [Den+09] with 1,000 classes such as Inception, [Sze+16], ResNet [He+16a; He+16b], etc. However, in the fine-grained classification, many FGVC datasets [BGV14; Kho+11; Maj+13; NZ08; Par+12; Van+15; Wah+11; Yan+15] contain classes that are not in or have little overlap with ImageNet [Den+09]. The features are not representative enough for a majority of fine-grained classes that are present in FGVC datasets. Therefore, transfer learning and fine-tuning are necessary. Yin Cui *et al.* [Cui+18] combines selected images from Imagenet [Den+09] and iNat [Hor+18b] datasets to create a training dataset that contains almost 2 million images to do transfer learning and fine-tuning. As transfer learning from pre-trained architectures to FGVC datasets is a standard approach, that does not modify the base classification network architecture, nor use special techniques or optimizations, we opt to use pre-existing transfer learning network models from [Cui+18] to extract classification features and use them as pre-optimized data for our method.

6.3.2 Feature Discriminant Analysis

Let f be the pre-optimized features extracted from a pre-trained classification network and \mathcal{F} be the transformed features of f .

$$\mathcal{F} = \mathcal{N}(f), \quad (6.4)$$

where \mathcal{N} is a general function for feature transformation and optional dimensionality reduction. The optimization objectives for \mathcal{N} are the following:

- **Maximizing the fine-grained classification results:** Let \mathcal{N}^1 be the classification function to classify feature \mathcal{F} . Let the classification results be $q(f) = \mathcal{N}^1(\mathcal{F})$. Maximizing classification results is equivalent to minimizing the categorical cross entropy loss:

$$\mathcal{H}(p, q) = - \sum_{\forall f} p(f) \log(q(f)), \quad (6.5)$$

where $p(f)$ is the one-hot encoded classification ground-truth for the feature f .

- **Minimizing intra-class variance:** This is to minimize the total distances for all the feature points \mathcal{F}_j^i of class i to their mean features, calculated as below for n classes:

$$\mathcal{L}_{Mean} = \sum_{i=1}^n \sum_j \left(\mathcal{F}_j^i - \bar{\mathcal{F}}^i \right)^2 \quad (6.6)$$

where $\bar{\mathcal{F}}^i$ is the mean feature of class i .

- **Maximizing inter-class variance:** With this optimization, we propose to use pairs of images. If a pair of images belong to the same class, we want to reduce the distance between the image features; otherwise, we want to increase the distance between them. The effect is to push features from different classes apart from each other while keeping features within the same class close to each other. Let $y = 0$ if two features \mathcal{F}_k and \mathcal{F}_l are from the same class and $y = 1$ if they are from different classes. The optimization for inter-class variance minimizes the following function:

$$\mathcal{L}_{Siamese} = (1 - y)(1 - e^{-d}) + y * e^{-d} \quad (6.7)$$

$$d = l_2(\mathcal{F}_k, \mathcal{F}_l) \quad (6.8)$$

d is the Euclidean distance between two features \mathcal{F}_k and \mathcal{F}_l .

When $y = 0$, Equation 6.7, which is the Siamese loss, is optimized for intra-class distance, but this is a weak constraint compared to Equation 6.6. In the Siamese loss, the optimization is done pairwise and batch-based, so that feature points are moved relative to each other. The Equation 6.6 is optimized such that all the feature points move to the pre-calculated class mean. To emphasize the importance of both intra-class variance optimization in Equation 6.6 and inter-class variance optimization in Equation 6.7, we conduct an experiment that uses only the Siamese loss and compare it with our proposed two-phase NDA optimization where both the Equation 6.6 and Equation 6.7 are optimized alternately. The results show that two-phase NDA optimization produces higher classification accuracy than an optimization that uses only Siamese loss alone (see Table 6.2).

6.3.3 Discriminant Analysis Optimization with Neural Networks

We choose to integrate the objectives of discriminant analysis in Section 6.3.2 into Neural Networks and form Neural Discriminant Analysis (NDA) networks due to the following

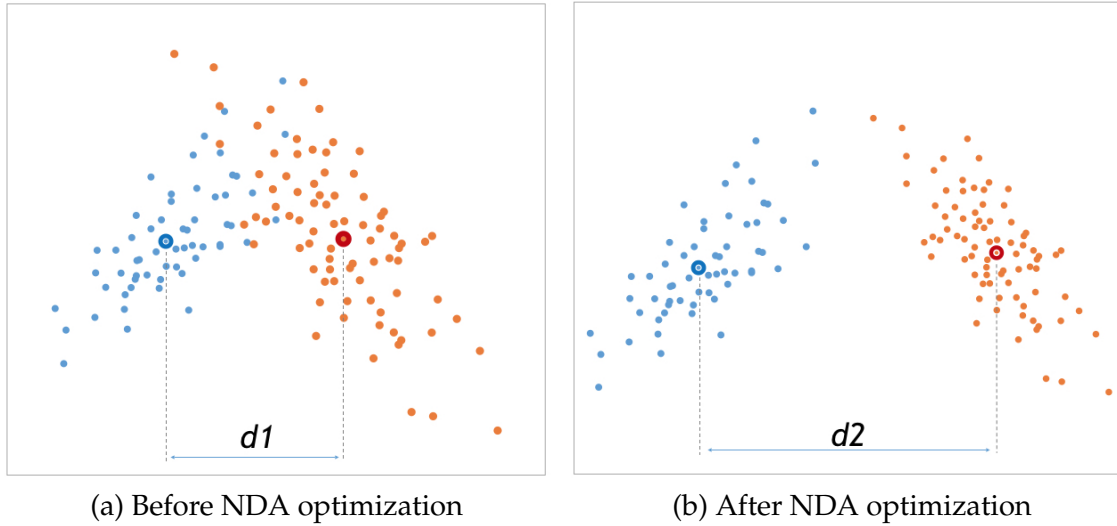


Figure 6.2: A demonstration of NDA optimization. In Figure (a), the extracted features from a pre-trained network or transfer learning network are distributed with some overlap and the distance between two classes is $d1$. In Figure (b), the features after NDA optimization are expected to cluster closer to their mean and the distance between two classes is $d2$. $d2$ is expected to be larger than $d1$ so that the features from different classes are better separated.

reasons. We can easily add an NDA component to an existing classification model in between the final feature layer and the prediction layer. This is an advantage over using Linear Discriminant Analysis (LDA) because the transformed features need to be trained using different tools for classification such as Support Vector Machine (SVM) or single-/multi-layer neural networks. The NDA approach is flexible in design. It can be used to implement a dimensionality reduction or a feature transformation without changing the features' dimensions. However, in practice, it is better to reduce the feature dimensions with a neural network when dealing with small size datasets. Dimensionality reduction helps to reduce the number of training parameters while maintaining accuracy and, therefore, reducing the chance of over-fitting. The NDA model (Figure 6.3) is designed as follows:

The feature transformation function \mathcal{N} in Equation 6.4 is a neural network that step-by-step reduces the dimension of the pre-optimized features, which is our NDA network. In order to satisfy the first objective in Equation 6.5, which is maximizing the fine-grained classification results, we first append a single-layer neural network \mathcal{N}^1 at the end of NDA network \mathcal{N} to train for classification using categorical cross-entropy loss. The result is a classification neural network $\mathcal{N}^2 = \mathcal{N}^1 \circ \mathcal{N}$. This first step of training is to make sure the dimension reduction of the NDA network \mathcal{N} can maintain the classification ability from the original features.

The optimizations for inter-class and intra-class variance should be done jointly. To implement this, we develop a two-phase optimization for each epoch of training. In the first phase, we optimize for the Equation 6.6. All the features f_j^i are fed forward to the NDA network, forming features \mathcal{F}_j^i and the mean features $\bar{\mathcal{F}}^i$ are computed for every class i . We then use the Mean Squared Error loss to minimize the distances between all features \mathcal{F}_j^i of the same class i to their mean feature $\bar{\mathcal{F}}^i$.

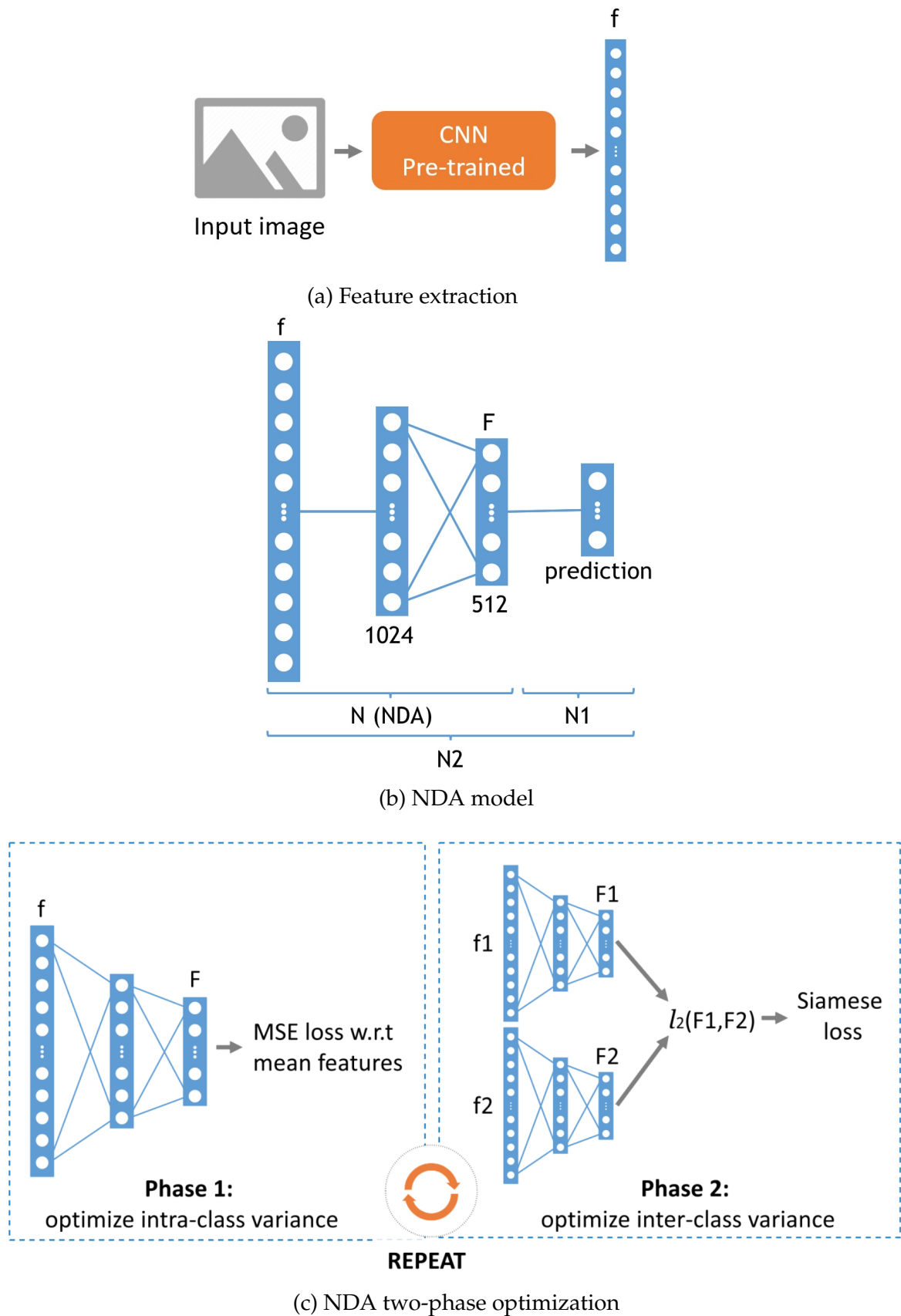


Figure 6.3: An overview of the Neural Discriminant Analysis model and its two-phase optimization.

In the second phase, we optimize for the inter-class variance. We use the NDA network \mathcal{N} as a base to form a Siamese network \mathcal{S} . The Siamese network \mathcal{S} has two input features f_k and f_l and outputs the Euclidean distance between \mathcal{F}_k and \mathcal{F}_l . If f_k and f_l are in the same class, we want the output distance to be reduced, otherwise, increased. The loss function for the Siamese network \mathcal{S} is implemented as in Equation 6.7. In every epoch, after each phase is optimized, the NDA network's weights are updated. Finally, we retrain the 1-layer prediction network \mathcal{N}^1 on the transformed features \mathcal{F} produced by the NDA network \mathcal{N} for fine-grained classification. This will improve the results due to the NDA optimized features. The optimization is summarized in Algorithm 2.

Algorithm 2 Algorithm for optimizing Neural Discriminant Analysis (NDA) network

Input: a set of pre-optimized features $f_j, j = 1..m$

Output: an entire network \mathcal{N}^2

- 1: Train a classification neural network $\mathcal{N}^2 = \mathcal{N}^1 \circ \mathcal{N}$ on the whole set of pre-optimized features $f_j, j = 1..m$ using a categorical cross-entropy loss
 - 2: **for** each training epoch up to NUM_EPOCH **do**
 - 3: (i) compute transformed feature $\mathcal{F}_j = \mathcal{N}(f_j)$
 - 4: (ii) compute mean features $\bar{\mathcal{F}}^i$ for every class i
 - 5: (iii) train the NDA network \mathcal{N} with the Mean-Squared-Error between its outputs and the mean $\bar{\mathcal{F}}^i$
 - 6: (iv) compile a Siamese network using the NDA network \mathcal{N} in each branch with shared weights. The Siamese model outputs the Euclidean distance d between a pair of transformed features \mathcal{F}_k and \mathcal{F}_l .
 - 7: (v) optimize the Siamese loss: $\mathcal{L}_{Siamese} = (1 - y)(1 - e^{-d}) + y * e^{-d}$
 - 8: **end for**
 - 9: Retrain the classification \mathcal{N}^1 layer on the output features of the NDA network \mathcal{N} using a categorical cross-entropy loss for final fine-grained classification results
 - 10: Return network \mathcal{N}^2
-

An important note is that our two-phase NDA optimization alternates between Equation 6.6 (minimize mean variance) and Equation 6.7 (optimize Siamese loss) in each epoch. The two losses are not combined into one single optimization. Therefore, no loss weights are required.

6.4 Experiments with Two-phase NDA

6.4.1 Datasets and Feature Extraction

We evaluate our proposed method on the following FGVC datasets: Stanford-Dogs [Kho+11] that contains 120 breeds of dogs, CUB-200-2011 [Wah+11] that has 200 types of birds, Flower-102 [NZ08] with 102 types of flowers, Stanford-Cars [Yan+15] that has 196 types of cars and NABirds [Van+15] with 555 classes of birds.

We use transfer learning Inception-ResNet-V2, Inception-ResNet-V2-SE and Inception-V3-iNat models from Cui *et al.* [Cui+18] as base networks to compare across all datasets.

6.4.2 Implementation

In this experiment, the NDA network \mathcal{N} is a two-layer neural network with an input feature size of 2,048 or 1,536, depending on the networks that are used to extract the features. The first hidden layer has 1,024 nodes followed by a ReLU activation, and the last layer has 512 nodes, also with ReLU activation. 0.35 and 0.25 dropout is added for each layer, respectively. These parameters are chosen empirically. The classification network \mathcal{N}^1 contains only a single prediction layer with a softmax activation. By concatenating \mathcal{N}^1 after \mathcal{N} , we have a neural network $\mathcal{N}^2 = \mathcal{N}^1 \circ \mathcal{N}$. In the initial training, we train \mathcal{N}^2 to reach the reported accuracy in [Cui+18]. The number of training epochs can range from 100 to 1,000, depending on the datasets and the types of pre-optimized features.

The Neural Discriminant Analysis (NDA) is optimized using 20 to 30 epochs for all the experiments. For each epoch, we need to re-compute the mean and re-generate input data pairs for the optimizations. Therefore, the process takes much longer compared to fine-tuning a 2-layer neural network for transfer learning.

Data sampling: After each epoch, we re-sample pairs of data for the Siamese loss optimization in Equation 6.7. The strategy is for each image in a class, we randomly select one image from the same class and k images from different classes. The intra / inter ratio is therefore $1 : k$, where $k \geq 1$. We chose $k = 2$, which leads to good results and surpasses the state-of-the-art. Increasing k will significantly increase the training time. In training, the data is shuffled randomly. Even though we use a Siamese architecture to optimize for inter-class variance, the number of training samples for each epoch is $(k + 1)n$ where n is the total number of images in the dataset. Therefore, the complexity is $O(n)$, and not $O(n^2)$ that would correspond to sampling all pairs.

The dual objective optimization is not without perils, the two steps can be unbalanced. In some cases, a model collapse can happen, meaning that the entire transformed feature space collapses to one point. To address this issue, we set the learning rate for minimizing variance to a class mean ten times smaller than the learning rate for Siamese optimization (10^{-5} with SGD optimizer vs. 10^{-4} with RMSprop optimizer). Minimizing the variance of features from a class can be interpreted as a pulling force that moves them towards the class centroid. In contrast, the Siamese optimization pushes features of different classes further apart. These two forces should be balanced for good optimization. If the pulling force is too strong, it is easier for the NDA model to collapse after a few training epochs.

The convergence: We experiment with the learning rate and different optimizers (Adam, RMSprop, SGD) to test the balance and solve the model collapse issue. It is shown that the learning rate is a more important factor in reducing the probability of the model collapse. The optimizers have mainly an effect on the convergence when training for the transfer learning network \mathcal{N}^2 . The convergence of our two-phase optimization NDA and model collapse scenario are demonstrated in Figure 6.4. While the Siamese Loss can be still further reduced, the Mean Loss is approaching zero. In the model collapse situation, the Mean Loss

is zero. Therefore, we stop the training before the Mean Loss becomes smaller than an ϵ value, such as 0.003.

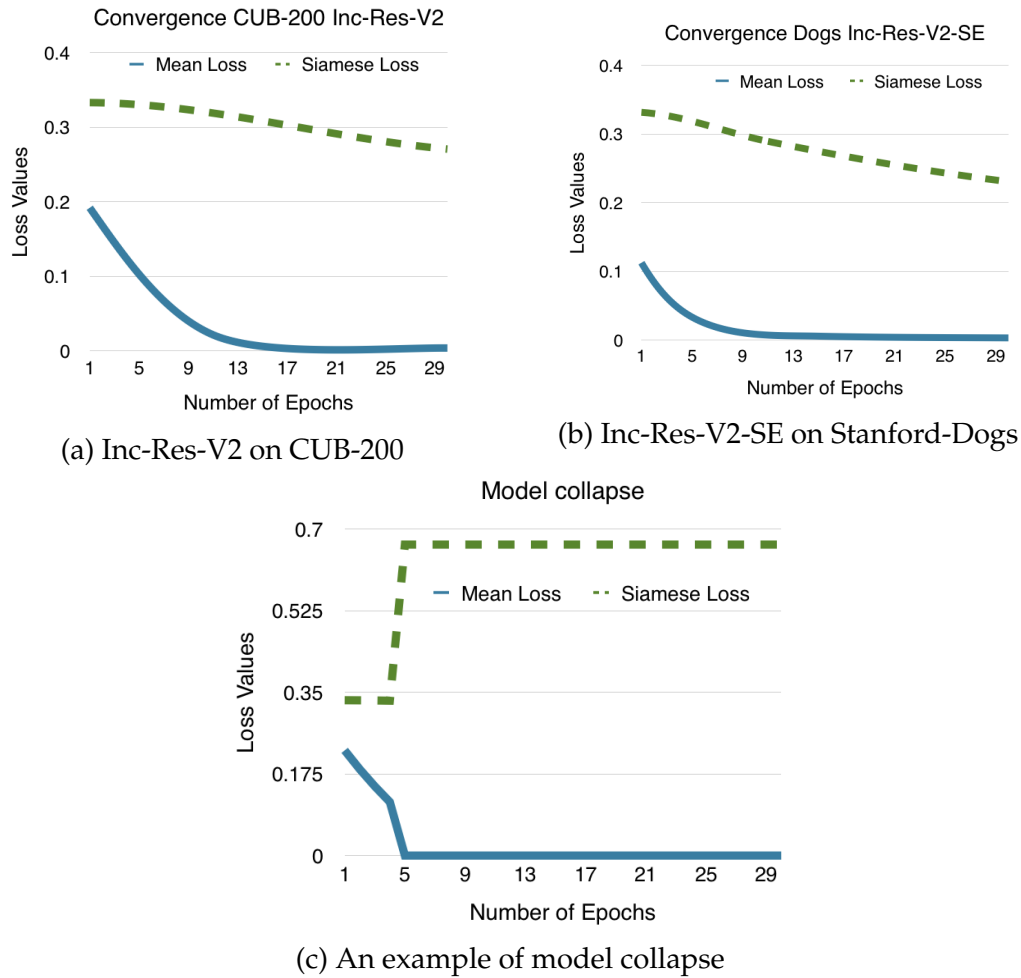


Figure 6.4: (a) and (b): examples of convergences for two-phase NDA optimization (Mean Loss and Siamese Loss) for two different networks (Inception Resnet V2 and Inception Resnet V2 SE) on two different datasets (CUB-200 and Stanford-Dogs) respectively. (c): an example of model collapse after the 5th epoch. When the trained model is collapsed, all the data point come to a single point. Therefore the mean loss $\mathcal{L}_{Mean} = 0$. The Siamese loss $\mathcal{L}_{Siamese} = y * e^{-d} = 1$ where $d = 0$ and $y = 1$. So the $\mathcal{L}_{Siamese} = 1$ for pairs of points from different classes and $\mathcal{L}_{Siamese} = 0$ for pairs of points of the same class. For this experiment, the intra / inter ratio of sample pairs is 1 : 2. If we have n pairs of the same class, then we have $2n$ pairs from different classes. Hence, the sum of the Siamese loss for all pairs of data is $2n$. As we have totally $3n$ pairs, the average $\mathcal{L}_{Siamese}$ when the model collapse is $2/3 \approx 0.67$.

Finally, we re-train the single layer prediction network \mathcal{N}^1 . It usually takes half the number of epochs of the transfer learning for the training to converge.

6.4.3 Results

We compare our results with the state-of-the-art in Table 6.1. We use features extracted from Inception-ResNet-V2, Inception-ResNet-V2-SE and Inception-V3-iNat transfer learning from the method in [Cui+18]. It is worth to note that the best performances of transfer learning networks in [Cui+18] are from Inception-V3 and Inception-ResNet-V2-SE. Even

though the features from Inception-ResNet-V2 transfer learning do not produce the best results on their own, we are still able to top the state-of-the-art in a majority of the evaluated datasets. Authors in [Cui+18] trained Inception-V3 by using different data-sampling strategies, but there is no strategy that is able to consistently produce the best transfer learning result on all the datasets. It is unclear which of the strategies has been applied in training the publicly available network that we use. Thus, for Inception-V3 transfer learning results from [Cui+18], we report the range of accuracy from all the data sampling strategies. The training is repeated ten times, with different random initializations for each network on each dataset. The reported results are the average performance over ten runs.

Method	CUB-200	Stanford -Dogs	Stanford -Cars	Flower -102	NABirds
Krause <i>et al.</i> [Kra+16]	82.0	-	92.6	-	-
Bilinear-Pooling [LRM15]	84.1	-	91.3	-	-
DLA [YWD18]	85.1	-	94.1	-	-
Recurrent-Attention [FZM17]	85.3	87.3	92.5	-	-
Cai <i>et al.</i> [CZZ17]	85.3	-	91.7	-	-
Object-part-Attention [PHZ18]	85.8	-	92.2	97.1	-
Grassmann-Pooling [Wei+18]	85.8	-	92.8	-	-
Improved-Bi-Pooling [LM17]	85.8	-	92.0	-	-
Kernel-Pooling [Cui+17]	86.2	-	92.4	-	-
MA-CNN [Zhe+17]	86.5	-	92.8	-	-
Pairwise-Confusion [Dub+18]	86.9	83.8	92.9	91.4	82.8
Multi-Attention [Sun+18]	86.5	84.8	93.0	-	-
HBP [Yu+18]	87.1	-	93.7	-	-
MetaFGNet [ZTJ18]	87.6	96.7	-	-	-
DCL [Che+19]	87.8	-	94.5	-	-
TASN [Zhe+19]	87.9	-	93.8	-	-
Ge <i>et al.</i> [GLY19]	90.4	97.1	-	-	-
Inception-V3 from [Cui+18]	82.8 - 89.3	78.5 - 85.2	88.3 - 91.4	96.3 - 97.7	82.0 - 87.9
Best of Transfer Learning [Cui+18]	89.6	88.0	93.5	97.7	87.9
NDA (Inception-V3-iNat) (ours)	87.4	89.1	99.9	95.5	83.9
NDA (Inc-Res-V2) (ours)	90.1	95.3	97.4	97.7	88.4
NDA (Inc-Res-V2-SE) (ours)	89.7	95.5	99.9	97.7	89.5

Table 6.1: Comparison of results with the state-of-the-art. Input images of dimensions 448×448 pixels are used for all the experiments. The features used in our methods are from transfer learning networks in [Cui+18]. The best results are reported for Transfer Learning [Cui+18] which are from their Inception-V3 or Inception-ResNet-v2-SE transfer learning networks. Inception-V3 transfer learning results from [Cui+18] are reported as the range of accuracy from all the data sampling strategies as described in [Cui+18]. However, there is no strategy that is able to consistently produce the best transfer learning result on all the datasets. We evaluate our NDA using the Inception-V3-iNat published by the authors of [Cui+18], that is unclear to us which sampling strategy was used. All of our results are averaged over 10 training runs. "Inc-Res" is short for Inception-ResNet.

Method	CUB-200	Stanford -Dogs	Stanford -Cars	Flower -102	NABirds
Siamese (Inc-Res-V2)	89.7	94.9	96.5	97.5	88.2
NDA (Inc-Res-V2) (ours)	90.1	95.3	97.4	97.7	88.4

Table 6.2: Comparison of results for NDA optimization and Siamese network (without Mean Loss). All the parameters are kept the same. All of our results are averaged over 10 training runs. The comparison in Table 6.1 shows that all the methods in this field compete at 0.1%, so the improvement of NDA over Siamese is substantial.

Accuracy: MetaFGNet [ZTJ18] is a method that trains a FGVC network using extra sampled images. The method produces better result on Stanford-Dogs [Kho+11] ([ZTJ18] 96.7% vs. ours 95.5%). However, the performance on the CUB-200-2011 dataset [Wah+11] are substantially lower ([ZTJ18] 87.6% vs. ours 90.1%). On the other hand, we make a 7.5% improvement on Stanford-Dogs [Kho+11] compared to the baseline method [Cui+18]. Ge *et al.* [GLY19] proposed a complex system that combines LSTM trained on different region proposals of a single object with multi-loss and selective joint fine-tuning (SJFT) with images from ImageNet [Den+09] using GoogLeNet [Sze+15b]. It establishes the state-of-the-art results on the CUB-200-2011 [Wah+11] and Stanford-Dogs [Kho+11] datasets. For the CUB-200-2011 [Wah+11], their method exceeds our result by 0.3%. We believe that NDA optimization is able to improve the results of MetaFGNet [ZTJ18] and Ge *et al.* 's method [GLY19] further. Our method matches the state-of-the-art in the dataset Flower-102 [NZ08], and sets new records for the Stanford-Cars [Yan+15] and NABirds [Van+15] datasets.

With the Stanford-Dogs dataset [Kho+11], we also study how good the features produced by the original networks are if they were pre-trained on 1,000 classes from ImageNet [Den+09]. We achieve 93.9% with the original Inception-ResNet-V2 [Sze+17], but below 80% on other pre-trained networks such as ResNet50, InceptionV3, Xception and VGG19. As a data-driven approach, FGVC using deep learning depends on the type of network architectures as well as the amount of training data. It is also widely affected by how relevant the data from the source domain is to the target domain in transfer learning [Cui+18; ZTJ18]. It is observed and confirmed by Guérin and Boots [GB18] that even though different networks can produce competitive classification results, the features provided by those networks have different distributions and clusters.

With NDA optimization on the Stanford-Cars dataset using features extracted from Inception-ResNet-V2-SE and Inception-V3-iNat, we raise the accuracy to 99.9% consistently throughout all 10 runs (the standard deviation is therefore 0). This is due to the very good features provided by Inception-ResNet-V2-SE and Inception-V3-iNat transfer learning networks from [Cui+18] on this dataset. Without the NDA optimization, the average accuracy of Inception-ResNet-V2-SE is 97.4%, and the results fluctuate from 89.1% to 99.9% (standard deviation is 3.88). This shows the consistency of the NDA optimization.

Reliability: We compute and report standard deviations of the accuracy across 10 runs per dataset per network in Table 6.3 and visualize them in Figure 6.5. The standard deviations

are consistently lower in all NDA optimization results compared to transfer learning. It shows that the NDA optimization transforms the features such that the classification becomes more stable and reliable. High average with small standard deviation results are much more desired, compared to lower average and high standard deviation.

Model	Method	CUB -200	Stanford -Dogs	Stanford -Cars	Flower -102	NA- Birds
Inception-ResNet-V2	Transfer Learning	0.74	0.96	2.53	0.22	0.73
	NDA Optimization	0.28	0.81	1.80	0.10	0.08
Inception-ResNet-V2-SE	Transfer Learning	0.69	0.37	3.88	0.13	0.06
	NDA Optimization	0.17	0.11	0.00	0.13	0.06
Inception-V3-iNat	Transfer Learning	3.96	0.86	8.32	0.50	0.09
	NDA Optimization	0.27	0.40	0.00	0.23	0.10

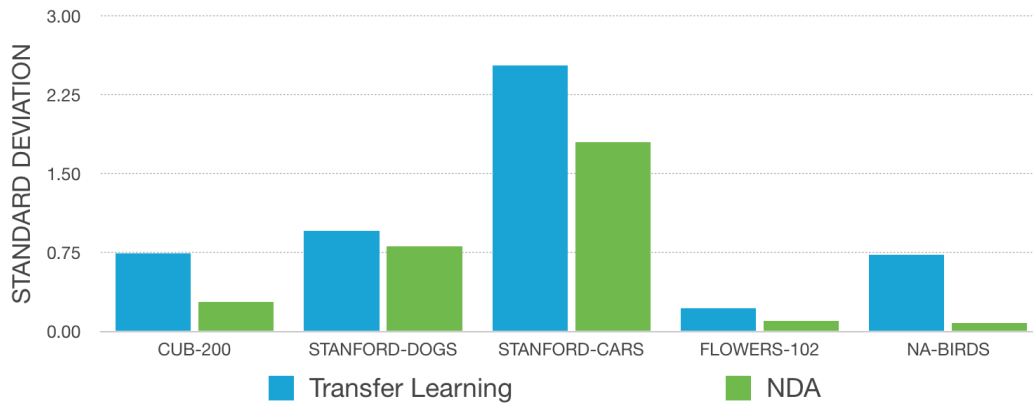
Table 6.3: Training stability: standard deviations of accuracy over 10 trainings. The standard deviations for classification using NDA optimization are consistently smaller than those of classification without NDA optimization. It shows that the NDA optimization produces more stable results across different trainings.

6.4.4 Discussion

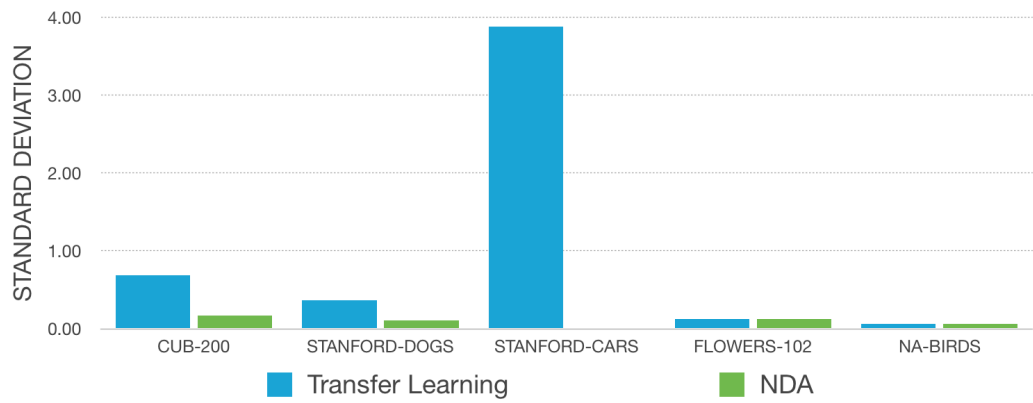
Data augmentation: In this experiment, we do not use data augmentation because we emphasize the importance of features that can represent the objects faithfully. Data augmentation often leads to unwanted distortions, which reduces the validity of the training data. They can be useful in some cases, but they do not enrich the characteristics of the dataset that reflects the variety of images in-the-wild.

Data pre-processing: When we learn discriminative features for fine-grained classification, it is very important that the input images contain the appropriate appearance of the object itself. In some datasets such as Stanford-Dogs [Kho+11], there are images that contain different objects or humans that cover significant parts of the images. Therefore, we decided to crop parts of images that contain only relevant objects using bounding box annotations. When cropping around bounding-boxes, we extend one of the two dimensions (width and height) such that the aspect ratio after cropping is as close to 1 as possible. This is to avoid over distortion when resizing the crops to the size of 448×448 pixels. For Flower-102 [NZ08] and Stanford-Cars [Yan+15], we do not perform any cropping, and use the original images for training. When testing, we use the original images for all the datasets without any cropping or special pre-processing except those required by the deep networks used for feature extraction.

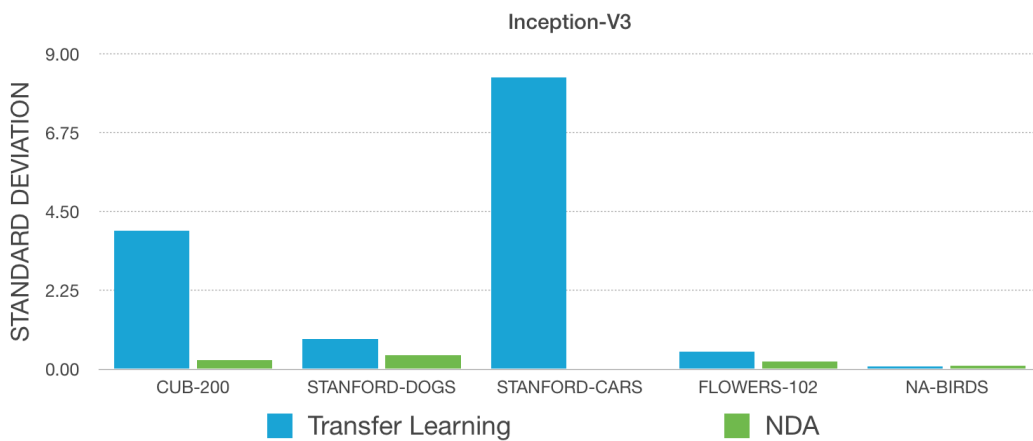
Performance consistency: The low standard deviations during re-training show that our method has stable and consistent performance. Looking closer, our NDA does not always increase the accuracy above the transfer learning network baselines. If we count how often the NDA optimization *reduces* the performance of transfer learning, we find that it happens in 14% of all the training passes.



(a) Inception-ResNet-V2



(b) Inception-ResNet-V2-SE



(c) Inception-V3

Figure 6.5: Comparison of standard deviations between transfer learning and NDA optimization results on 10 training runs.

Two-phase NDA optimization: Last but not least, the two-phase NDA optimization proves its effectiveness throughout various datasets (Table 6.1). If we use only one phase of the NDA optimization, such as Siamese Loss alone, the performance drops (Table 6.2). Without the Mean Loss, it lacks a strong and explicit constraint for intra-class optimization. Without Siamese Loss, there is no inter-class optimization. After the feature-based optimization is completed, we can add the NDA neural network after the last feature layer of the base network that was used for feature extraction to make it a complete end-to-end model for prediction or testing.

6.5 Combined Optimization NDA

Two-phase NDA produces very good results for fine-grained classification. However, it could be a bit complicated to implement the optimization in two phases. To simplify it, we also present combining all the losses to optimize in one phase (Figure 6.6).

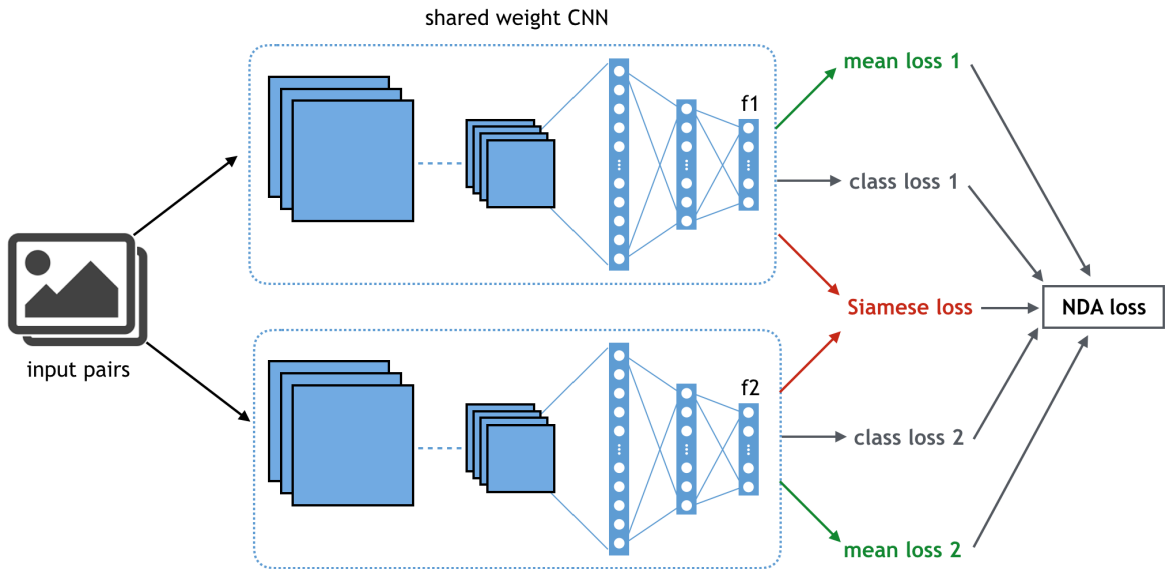


Figure 6.6: A demonstration of NDA optimization with combined loss. The NDA loss is the weighted sum of two classification losses, two mean losses, and a Siamese loss produced by a Siamese network.

The core structure of combining losses for NDA optimization is a Siamese architecture. A pair of input images are passed through a shared weight Siamese network. The network produces each feature for each input image. We have five losses for the network: two Classification losses, two Mean losses (one Classification loss and one Mean loss for each image), and a Siamese loss. All the losses are defined the same as with two-phase NDA. The Classification loss is the categorical cross entropy loss defined in Equation 6.5. The Mean losses are defined as in Equation 6.6 and the Siamese loss is defined as in Equation 6.7. The combination loss is defined as the weighted sum of all the losses as follows:

$$\mathcal{L}_{NDA} = \alpha(\mathcal{L}_{Class1} + \mathcal{L}_{Class2}) + \beta(\mathcal{L}_{Mean1} + \mathcal{L}_{Mean2} + \gamma * \mathcal{L}_{Siamese}) \quad (6.9)$$

where \mathcal{L}_{Class1} and \mathcal{L}_{Mean1} are the Classification loss and Mean loss for the first input image. \mathcal{L}_{Class2} and \mathcal{L}_{Mean2} are the Classification loss and Mean loss for the second input image. The coefficients α , β , and γ are fine-tuned hyper-parameters. For all the experiments in this section with the CIFAR-10 dataset, we use $\alpha = 1$, $\beta = 1e - 3$, and $\gamma = 1$. The selection of the values for these parameters takes into account the difference in magnitude for different types of losses. We want to normalize the losses to the same level of magnitude. CIFAR-10 is a general classification dataset that has 60,000 images in total with ten classes. It is considered as a mini version of ImageNet and is widely used as a benchmark dataset for general image classification.

We train networks end-to-end for classification on the CIFAR-10 dataset. We use the official (train, test) split of the dataset. However, we further split the training set to 80% for training data and 20% for validation. We only validate our training on the validation set and test on the test set. We use two different network architectures for the Siamese: AlexNet [KSH12] with Kaggle implementation and ResNet50 [He+16b]. The results for AlexNet Kaggle are shown in Table 6.4.

	Baseline	Optimization	Improvement Over Baseline
CDA [Zho+18]	60.15%	62.50%	2.35%
NDA (ours)	64.71%	70.92%	6.21%

Table 6.4: Comparison between NDA and CDA [Zho+18] on CIFAR-10 dataset using AlexNet [KSH12] Kaggle implementation. NDA has significant better result than CDA. The NDA result is averaged over five runs.

We compare with a competing method "Convolutional Discriminant Analysis" (CDA) proposed by Zhong *et al.* [Zho+18] using AlexNet Kaggle base network. Different from our Siamese architecture, CDA uses a single branch CNN with a single input image. With the Siamese network, we can specify the Mean loss and the Siamese loss explicitly. On the contrary, CDA's objective is that an input image trained to be close to its class center and further away from other classes' centers. NDA achieves 70.92% accuracy, whereas CDA only reaches to 62.5%. Compared to the baseline, CDA has 2.35% increase, and NDA has 6.21% improvement. We re-implement AlexNet Kaggle using Keras-Tensorflow. Due to different framework supports, our baseline has higher accuracy than the CDA's. The numerical results conclude that our NDA achieves significantly better results than CDA and much higher improvement from the baseline.

We also test our combined loss NDA with ResNet50 [He+16a] using Keras-Tensorflow. We use the code for the model provided by Keras ¹, as well as the pre-defined learning schedule and 200 epochs for each training. It is reported that the accuracy of ResNet50 on CIFAR-10 is 93%. However, it is important to take note that this accuracy is reported for a training that is validated on the test set. On the contrary, we split the default training data into 80% for training and 20% for validation to avoid over-fitting on the test data. It also results in

¹https://keras.io/examples/cifar10_resnet/

having less data for training. We achieve the baseline accuracy 91.77%. NDA improves the accuracy to 93%. The NDA result is averaged over three runs.

We run the experiments with different values for the hyper-parameters α , β , and γ . While we cannot extensively search for many configurations of the hyper-parameters due to limited resources, we find that this configuration $\alpha = 1$, $\beta = 1e - 3$, and $\gamma = 1$ gives slightly better results. However, in this case, the weight β for the Mean losses are quite small compared to the other two; the contribution of the Mean Loss to the total loss is not so high. We believe the results in this scenario are close to the two-phase NDA without the Mean loss.

6.6 Chapter Summary

Inspired by the objectives of Linear Discriminant Analysis (LDA) and making use of the power of deep learning and neural networks, we propose a Neural Discriminant Analysis (NDA) optimization that is useful for Fine-Grained Visual Classification (FGVC). It is a two-phase optimization that minimizes the intra-class variance and maximizes the inter-class variance in the deep feature domain. In order to reach these objectives, we address many technical issues related to deep learning, such as avoiding model collapse, data pre-processing and augmentation, pair data sampling, and most of all, the two-phase optimization architecture that can be fully implemented with neural networks. We exceed the state-of-the-art accuracy on several popular FGVC datasets by a large margin. The analysis also shows that our optimization produces more stable and reliable results. Furthermore, the NDA model can be used on its own in the deep feature domain or as a plug-in component to existing DCNNs. We also propose a combined loss version of the NDA for convenient end-to-end training. We achieve significant improvements from the baseline on CIFAR-10 dataset and much better results than the competing methods.

Chapter 7

Conclusion

In this thesis, we study two aspects of image understanding: perception similarity and deep feature representation. We have two main projects in each category. In the perception similarity topic, the two projects are a perceptual intrinsic imaging metric and color composition similarity. For deep feature representation, we study shape extraction used in semantic segmentation and deep feature discriminant analysis.

We develop a perceptually inspired metric to evaluate reflectance consistency for intrinsic imaging that yields a stable evaluation across different illuminations. The metric takes into account the human ability to distinguish color differences and prevents visual illusions caused by harsh shadows. Intrinsic imaging is often used in Computer Graphics to reconstruct realistic scenes under different lighting conditions. Therefore, it is important to produce correct reflectances with respect to the objects' surface properties and human perception. Perceptually inspired evaluation metrics like ours will help to push the quality of intrinsic imaging methods in complex scenes that leads to more realistic rendering. For future work, our metric can be extended to evaluate the shading components of the intrinsic results and is an inspiration to create more perceptual evaluation metrics in other research fields in Computer Vision and Computer Graphics.

The second project on perception similarity is color composition similarity. We derive a new global color similarity for images in-the-wild directly from human judgments. We carefully design the crowd-sourced experiments and propose an active learning framework to collect meaningful data for human evaluation and ensure the consistency of subjective judgments. We contribute to the research community an active learning framework to solve subjective studies. Its effectiveness is proven by the results of a large-scale, high-quality dataset for color composition similarity. Thanks to the quality of the dataset, we succeed in developing high-performance algorithms in several Computer Vision applications. Using the dataset, we train global descriptors for color similarity that are applied in color similarity prediction and image retrieval. Our global color descriptor surpasses all the existing hand-crafted color descriptors' performances. We also propose a novel type of features for fine-grained image retrieval by combining correlations of color features and category features between pairs of images. Our new visual similarity feature improves the state-of-the-art result by a large margin, using three orders of magnitude less training data than the competing method. We

also successfully use different feature levels of our trained network on color composition similarity in color transfer and achieve good results.

We use crowd-sourcing to conduct user studies. On the one hand, we provide explanations for the correct answers of all the test questions so that the participants understand the tasks better. On the other hand, participants also can give feedback on particular test questions that they find their opinions are different from ours. This interactive process is still weak or missing in many research areas. Computer Vision algorithms often provide the final answers to problems, but they are unable to give reasons and justifications, even much more in perception related topics and subjective study.

In classical Computer Vision problems, we usually seek answers for the questions "*What*" such as "What is the label of an image?", "What similar images can a system retrieve?". Our proposed approach to solve visual similarity by disentangling different factors of similarity is a mean to answer the questions "*How or Why are these images similar?*". The answers are, for example, two images can be similar because they both contain dogs, or they are both pictures of Van Gogh's paintings, or they both have beautiful autumn colors. It motivates us towards creating a machine learning system that is able to give explanations and reasons on abstract and subjective tasks, similar to visual similarity.

For future work, we will study other aspects of visual similarity, such as texture and image structure. By combining different aspects of image similarity, we will develop an explainable Artificial Intelligence (AI) system. Furthermore, there is a potential for using visual similarity in helping other learning algorithms, such as image classification. Active learning in image classification proves to be efficient. It starts with a small initial set of training data for a network to learn classification. Incrementally, new data is selected to add to the training based on the estimated uncertainty of the predictions from the network. There are different measures for choosing new data where the goal is for the network to learn faster with the least amount of training data. We believe visual similarity can be used as a criterion in the data selection process. For example, images that are visually similar or contain objects that look similar but belong to different classes are good candidates for learning discriminative features in classification. Therefore, we will apply visual similarity in active learning for training image classification tasks.

In the second part of the thesis, we focus on learning deep feature representations. In the first work, our goal is to find out the information contained in high-level features from a DCNN trained for classification. It leads to two findings: shape and Class Activation Map (CAM). A CAM is a map of regions in an input image that the DCNN uses for a particular class prediction. Existing methods can only produce low-resolution CAMs. By combining shapes and low-resolution CAMs from two different scales of an input image, we produce a high-resolution CAM (rCAM). The resolution of our rCAM is high enough that we can perform semantic segmentation without any extra training; at the same time, we achieve the state-of-the-art results in weakly supervised segmentation.

It comes as a surprise to many of us that deep features from a classification network can embed many types of attributes (such as shape), that when discovered, can be used efficiently in other tasks (in our case, semantic segmentation). This work can also be applied in creating saliency maps if we combine shape information and color features. As we all know, making a large-scale dataset with pixel-wise ground-truth for segmentation is very expensive. Our work shows that we can use and reuse information smartly in a cheaper way to obtain good results. Furthermore, it inspires us to wonder and discover many features hidden in other types of existing network architectures that are useful to solve various tasks.

Lastly, we study deep feature discriminant analysis. As deep learning is a data-driven approach, we raise the question if the features learned by a DCNN are the most discriminative only by training on large-scale data. The answer is no; we can further improve the discriminative potential of deep features by enforcing the objectives from Linear Discriminant Analysis in the network optimization. The objectives are to reduce the intra-class variance and increase the inter-class variance. We combine these objectives in a DCNN via our Neural Discriminant Analysis (NDA) optimizer. Our NDA improves the baseline results on various datasets with many types of network architectures for both general classification and fine-grained classification. NDA also provides reliable results across different training runs. The variances of the classification results from networks that use NDA are consistently smaller than those of the same networks without the NDA optimization. Every time we train a DCNN for classification, the initial network parameters, and the orders of training data are different due to the randomized initialization and data shuffling processes. As a result, the DCNN can converge to different local optima that lead to different prediction results. In other words, the results from different training runs vary. It is important for a system not only to have high accuracy but also consistent results with a small variance. Our proposed NDA optimization improves both the performance and reliability of classification networks.

We find that NDA also has potential in other learning techniques. We extend the applicability of NDA into Semi-Supervised Learning (SSL) and Out Of Distribution (OOD) detection in an ongoing project. The results show that NDA further improves the performances of those fields. We are pleased to see that traditional techniques and optimizations before the deep learning era are still useful and can be incorporated into DCNNs, leading to improvements in deep learning performances. For future work, we will improve algorithms which apply NDA in SSL and OOD for general classification and fine-grained classification tasks to increase their performances further.

The work in this thesis covers both technical developments in deep learning and perception-related measures. Perception is a subjective and challenging topic. However, we believe it should be more involved in the process of designing machine learning algorithms. There is still not much research in the direction of perception in deep learning. Our work is an effort to bring the two fields together. By focusing on understanding DCNNs, the state-of-the-art techniques in Computer Vision, and studying perceptual visual similarity, we hope to inspire further research into optimizing visual computing towards human perception.

Publications

- [Bei+16] S. Beigpour, M. L. Ha, S. Kunz, A. Kolb, and V. Blanz. “Multi-view multi-illuminant intrinsic dataset”. In: *Proceedings British Machine Vision Conference (BMVC)*. 2016, pp. 10.1–10.13. DOI: [10.5244/C.30.10](https://doi.org/10.5244/C.30.10).
- [Ha+18] M. L. Ha, G. Franchi, M. Möller, A. Kolb, and V. Blanz. “Segmentation and shape extraction from convolutional neural networks”. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 1509–1518. DOI: [10.1109/WACV.2018.00169](https://doi.org/10.1109/WACV.2018.00169).
- [HB20] M. L. Ha and V. Blanz. “Neural discriminant analysis for fine-grained classification”. In: *Proceedings IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 1656–1660. DOI: [10.1109/ICIP40778.2020.9190954](https://doi.org/10.1109/ICIP40778.2020.9190954).
- [HHB20] M. L. Ha, V. Hosu, and V. Blanz. “Color composition similarity and its application in fine-grained similarity”. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 2559–2568. DOI: [10.1109/WACV45572.2020.9093522](https://doi.org/10.1109/WACV45572.2020.9093522).

External Bibliography

- [AF06] A. E. Abdel-Hakim and A. A. Farag. “Csift: a sift descriptor with color invariant characteristics”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2006, pp. 1978–1983.
- [Ach+09] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. “Frequency-tuned salient region detection”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 1597–1604.
- [Alt92] N. Altman. “An introduction to kernel and nearest-neighbor nonparametric regression”. In: *The American Statistician* 46.3 (1992), pp. 175–185.
- [Arb+11] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. “Contour detection and hierarchical image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 33.5 (2011), pp. 898–916.
- [BM12] J. T. Barron and J. Malik. “Color constancy, intrinsic images, and shape estimation”. In: *Proceedings European Conference on Computer Vision (ECCV)*. 2012, pp. 55–70.
- [BM13a] J. T. Barron and J. Malik. “Intrinsic scene properties from a single RGB-D image”. In: 2013, pp. 17–24.
- [Baz+16] L. Bazzani, A. Bergamo, D. Anguelov, and L. Torresani. “Self-taught object localization with deep networks”. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2016, pp. 1–9.
- [BKK15] S. Beigpour, A. Kolb, and S. Kunz. “A comprehensive multi-illuminant dataset for benchmarking of the intrinsic image algorithms”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 172–180.
- [Bei+13] S. Beigpour, M. Serra, J. van de Weijer, R. Benavente, M. Vanrell, O. Penacchio, and D. Samaras. “Intrinsic image evaluation on synthetic complex scenes.” In: *Proceedings IEEE International Conference on Image Processing (ICIP)*. 2013, pp. 285–289.
- [BB15] S. Bell and K. Bala. “Learning visual similarity for product design with convolutional neural networks”. In: *ACM Transactions on Graphics* 34.4 (2015), 98:1–98:10.
- [BBS14] S. Bell, K. Bala, and N. Snavely. “Intrinsic images in the wild”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), 159:1–159:12. DOI: [10 . 1145 / 2601097 . 2601206](https://doi.org/10.1145/2601097.2601206).

- [BV16] H. Bilen and A. Vedaldi. “Weakly supervised deep detection networks”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2846–2854.
- [BSI13] A. Borji, D. N. Sihite, and L. Itti. “Quantitative analysis of human-model agreement in visual saliency modeling: a comparative study”. In: *IEEE Transactions on Image Processing (TIP)* 22.1 (2013), pp. 55–69.
- [BZM08] A. Bosch, A. Zisserman, and X. Muñoz. “Scene classification using a hybrid generative/discriminative approach”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 30.4 (2008), pp. 712–727.
- [BGV14] L. Bossard, M. Guillaumin, and L. Van Gool. “Food-101 – mining discriminative components with random forests”. In: *Proceedings European Conference on Computer Vision (ECCV)*. 2014, pp. 446–461.
- [Bra+14] S. Branson, G. Van Horn, S. Belongie, and P. Perona. “Bird species categorization using pose normalized deep convolutional nets”. In: *Proceedings British Machine Vision Conference (BMVC)*. 2014.
- [Bra+10] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. “Visual recognition with humans in the loop”. In: *Proceedings European Conference on Computer Vision (ECCV)*. 2010, pp. 438–451.
- [BM13b] J. Bruna and S. Mallat. “Invariant scattering convolution networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 35.8 (2013), pp. 1872–1886.
- [BG09] G. J. Burghouts and J.-M. Geusebroek. “Performance evaluation of local colour invariants”. In: *Computer Vision and Image Understanding* 113.1 (2009), pp. 48–62.
- [But+12] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. “A naturalistic open source movie for optical flow evaluation”. In: *Proceedings European Conference on Computer Vision (ECCV)*. 2012, pp. 611–625.
- [CM98] C. C. Tomasi and R. Manduchi. “Bilateral filtering for gray and color images”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 1998, pp. 839–846.
- [CZZ17] S. Cai, W. Zuo, and L. Zhang. “Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 511–520. DOI: [10.1109/ICCV.2017.63](https://doi.org/10.1109/ICCV.2017.63).
- [CC14] I. Cantador and P. Cremonesi. “Tutorial on cross-domain recommender systems”. In: *ACM Conference on Recommender Systems*. 2014, pp. 401–402.
- [Cer+08] M. Cerf, J. Harel, W. Einhäuser, and C. Koch. “Predicting human gaze using low-level saliency combined with face detection”. In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2008, pp. 241–248.
- [Che+10] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. “Large scale online learning of image similarity through ranking”. In: *The Journal of Machine Learning Research* 11 (2010), pp. 1109–1135.

- [Che+19] Y. Chen, Y. Bai, W. Zhang, and T. Mei. "Destruction and construction learning for fine-grained image recognition". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5152–5161.
- [Che+15] M. Cheng, G. Zhang, N. Mitra, X. Huang, and S. Hu. "Global contrast based salient region detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 37.3 (2015), pp. 569–582.
- [CHL05] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005, pp. 539–546.
- [CVS17] R. G. Cinbis, J. Verbeek, and C. Schmid. "Weakly Supervised Object Localization with Multi-fold Multiple Instance Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 39.1 (2017), pp. 189–203.
- [Cre85] N. Cressie. "Fitting variogram models by weighted least squares". In: *Journal of the International Association for Mathematical Geology* 17.5 (1985), pp. 563–586.
- [Cui+17] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie. "Kernel pooling for convolutional neural networks". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3049–3058. DOI: [10.1109/CVPR.2017.325](https://doi.org/10.1109/CVPR.2017.325).
- [Cui+18] Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie. "Large scale fine-grained categorization and domain-specific transfer learning". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4109–4118. DOI: [10.1109/CVPR.2018.00432](https://doi.org/10.1109/CVPR.2018.00432).
- [Cui+16] Y. Cui, F. Zhou, Y. Lin, and S. Belongie. "Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1153–1162.
- [Den+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 248–255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- [Den+16] J. Deng, J. Krause, M. Stark, and L. Fei-Fei. "Leveraging the wisdom of the crowd for fine-grained recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 38.4 (2016), pp. 666–676. DOI: [10.1109/TPAMI.2015.2439285](https://doi.org/10.1109/TPAMI.2015.2439285).
- [DRS11] M. Douze, A. Ramisa, and C. Schmid. "Combining attributes and fisher vectors for efficient image retrieval". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 745–752.
- [Dub+18] A. Dubey, O. Gupta, P. Guo, R. Raskar, R. Farrell, and N. Naik. "Pairwise confusion for fine-grained visual classification". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2018, pp. 71–88.

- [DHS11] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159.
- [ERR16] M. Elahi, F. Ricci, and N. Rubens. “A survey of active learning in collaborative filtering recommender systems”. In: *Computer Science Review* 20 (2016), pp. 29–50.
- [Eve+10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. “The pascal visual object classes (voc) challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338.
- [Eve+] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [Far+11] R. Farrell, O. Oza, Ning Zhang, V. I. Morariu, T. Darrell, and L. S. Davis. “Birdlets: subordinate categorization using volumetric primitives and pose-normalized appearance”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2011, pp. 161–168. DOI: [10.1109/ICCV.2011.6126238](https://doi.org/10.1109/ICCV.2011.6126238).
- [Fer+17] M. Ferguson, R. Ak, Y. T. Lee, and K. H. Law. “Automatic localization of casting defects with convolutional neural networks”. In: *IEEE International Conference on Big Data*. 2017, pp. 1726–1735.
- [FZ07] V. Ferrari and A. Zisserman. “Learning visual attributes”. In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2007, pp. 433–440.
- [FK10] D. Freedman and P. Kisilev. “Object-to-object color transfer: optimal flows and smp transformations”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 287–294.
- [FZM17] J. Fu, H. Zheng, and T. Mei. “Look closer to see better: recurrent attention convolutional neural network for fine-grained image recognition”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4476–4484. DOI: [10.1109/CVPR.2017.476](https://doi.org/10.1109/CVPR.2017.476).
- [Gao+16] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. “Compact bilinear pooling”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 317–326. DOI: [10.1109/CVPR.2016.41](https://doi.org/10.1109/CVPR.2016.41).
- [GEB15] L. A. Gatys, A. S. Ecker, and M. Bethge. “Texture synthesis using convolutional neural networks”. In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 262–270.
- [GEB16] L. A. Gatys, A. S. Ecker, and M. Bethge. “Image style transfer using convolutional neural networks”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423.
- [Gat+17] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. “Controlling perceptual factors in neural style transfer”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3730–3738.

- [GLY19] W. Ge, X. Lin, and Y. Yu. “Weakly supervised complementary parts models for fine-grained image classification from the bottom up”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3029–3038.
- [GHF17] T. Gebu, J. Hoffman, and L. Fei-Fei. “Fine-grained recognition in the wild: a multi-task domain adaptation approach”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1358–1367.
- [Geh+11] P. Gehler, C. Rother, M. Kiefel, L. Zhang, and B. Schölkopf. “Recovering intrinsic images with a global sparsity prior on reflectance”. In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2011, pp. 765–773.
- [Geu+01] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders, and H. Geerts. “Color invariance”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 23.12 (2001), pp. 1338–1350.
- [Goo+09] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng. “Measuring invariances in deep networks”. In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2009, pp. 646–654.
- [Gro+09] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. “Ground truth dataset and baseline evaluations for intrinsic image algorithms”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2009, pp. 2335–2342. DOI: [10.1109/ICCV.2009.5459428](https://doi.org/10.1109/ICCV.2009.5459428).
- [GB18] J. Guérin and B. Boots. “Improving image clustering with multiple pre-trained CNN feature extractors”. In: *Proceedings British Machine Vision Conference (BMVC)*. 2018.
- [HKP06] J. Harel, C. Koch, and P. Perona. “Graph-based visual saliency”. In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2006, pp. 545–552.
- [HST13] K. He, J. Sun, and X. Tang. “Guided image filtering”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 35.6 (2013), pp. 1397–1409.
- [He+16a] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.
- [He+16b] K. He, X. Zhang, S. Ren, and J. Sun. “Identity mappings in deep residual networks”. In: *Proceedings European Conference on Computer Vision (ECCV)*. 2016, pp. 630–645.
- [HP17] X. He and Y. Peng. “Fine-grained image classification via combining vision and language”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 7332–7340.
- [Hon+16] S. Hong, J. Oh, H. Lee, and B. Han. “Learning transferrable knowledge for semantic segmentation with deep convolutional neural network”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3204–3212.
- [Hor+18a] G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. “The inaturalist species classification and detection

- dataset". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8769–8778. DOI: [10.1109/CVPR.2018.00914](https://doi.org/10.1109/CVPR.2018.00914).
- [Hor+18b] G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. "The iNaturalist species classification and detection dataset". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8769–8778. DOI: [10.1109/CVPR.2018.00914](https://doi.org/10.1109/CVPR.2018.00914).
- [HLS18] V. Hosu, H. Lin, and D. Saupe. "Expertise screening in crowdsourcing image quality". In: *International Conference on Quality of Multimedia Experience (QoMEX)*. 2018. DOI: [10.1109/QoMEX.2018.8463427](https://doi.org/10.1109/QoMEX.2018.8463427).
- [Hua+16] S. Huang, Z. Xu, D. Tao, and Y. Zhang. "Part-stacked cnn for fine-grained visual categorization". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1173–1182.
- [Hua+15] X. Huang, C. Shen, X. Boix, and Q. Zhao. "SALICON: reducing the semantic gap in saliency prediction by adapting deep neural networks". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 262–270.
- [IKN98] L. Itti, C. Koch, and E. Niebur. "A model of saliency-based visual attention for rapid scene analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 20.11 (1998), pp. 1254–1259.
- [JDS08] H. Jegou, M. Douze, and C. Schmid. "Hamming embedding and weak geometric consistency for large scale image search". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2008, pp. 304–317.
- [Jin+17] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song. "Neural style transfer: A review". In: (2017). URL: <http://arxiv.org/abs/1705.04058>.
- [JAF16] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2016, pp. 694–711.
- [Jol02] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.
- [Jud+09] T. Judd, K. Ehinger, F. Durand, and A. Torralba. "Learning to predict where humans look". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2009, pp. 2106–2113.
- [Kar+14] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. "Recognizing image style". In: *Proceedings British Machine Vision Conference (BMVC)*. 2014. DOI: [10.5244/C.28.122](https://doi.org/10.5244/C.28.122).
- [Kha+13] R. Khan, J. van de Weijer, F. S. Khan, D. Muselet, C. Ducottet, and C. Barat. "Discriminative color descriptors". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 2866–2873.
- [Kho+17] A. Khoreva, R. Benenson, J. Hosang, and M. Hein. "Simple does it: weakly supervised instance and semantic segmentation". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1665–1674.

- [Kho+11] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. "Novel dataset for fine-grained image categorization". In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011.
- [KB15] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *Proceedings International Conference on Learning Representations (ICLR)*. 2015, pp. 1–11.
- [KL16a] A. Kolesnikov and C. H. Lampert. "Seed, expand and constrain: three principles for weakly-supervised image segmentation". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2016, pp. 695–711.
- [KL16b] T. K. Koo and M. Y. Li. "A guideline of selecting and reporting intraclass correlation coefficients for reliability research". In: *Journal of Chiropractic Medicine* 15.2 (2016), pp. 155–163. DOI: [10.1016/j.jcm.2016.02.012](https://doi.org/10.1016/j.jcm.2016.02.012).
- [KG13] A. Kovashka and K. Grauman. "Attribute adaptation for personalized image search". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 3432–3439.
- [KK11] P. Krähenbühl and V. Koltun. "Efficient inference in fully connected CRFs with gaussian edge potentials". In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2011, pp. 109–117.
- [Kra+14] J. Krause, T. Gebru, J. Deng, L. Li, and L. Fei-Fei. "Learning features and parts for fine-grained recognition". In: 2014, pp. 26–33. DOI: [10.1109/ICPR.2014.15](https://doi.org/10.1109/ICPR.2014.15).
- [Kra+16] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei. "The unreasonable effectiveness of noisy data for fine-grained recognition". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2016, pp. 301–320.
- [Kra+13] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. "3d object representations for fine-grained categorization". In: *International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. 2013.
- [KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2012, pp. 1097–1105.
- [KTB15] M. Kümmerer, L. Theis, and M. Bethge. "Deep gaze i: boosting saliency prediction with feature maps trained on imagenet". In: *ICLR Workshop*. 2015.
- [Laf+14] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays. "Transient attributes for high-level understanding and editing of outdoor scenes". In: *ACM Transactions on Graphics (SIGGRAPH)* 33.4 (2014). DOI: [10.1145/2601097.2601101](https://doi.org/10.1145/2601097.2601101).
- [LW16] C. Li and M. Wand. "Combining markov random fields and convolutional neural networks for image synthesis". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2479–2486.
- [Li+16a] D. Li, J.-B. Huang, Y. Li, S. Wang, and M.-H. Yang. "Weakly supervised object localization with progressive domain adaptation". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3512–3520.

- [LY16] G. Li and Y. Yu. "Deep contrast learning for salient object detection". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 478–487.
- [Li+13] J. Li, M. Levine, X. An, X. Xu, and H. He. "Visual saliency based on scale-space analysis in the frequency domain". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 35.4 (2013), pp. 996–1010.
- [Li+16b] X. Li, L. Zhao, L. Wei, M. H. Yang, F. Wu, Y. Zhuang, H. Ling, and J. Wang. "Deepsaliency: multi-task deep neural network model for salient object detection". In: *IEEE Transactions on Image Processing* 25.8 (2016), pp. 3919–3930.
- [Li+14] Y. Li, X. Hou, C. Koch, J. Rehg, and A. Yuille. "The secrets of salient object segmentation". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 280–287.
- [Lin+15] D. Lin, X. Shen, C. Lu, and J. Jia. "Deep LAC: deep localization, alignment and classification for fine-grained recognition". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1666–1674. DOI: [10.1109/CVPR.2015.7298775](https://doi.org/10.1109/CVPR.2015.7298775).
- [LCY13] M. Lin, Q. Chen, and S. Yan. "Network in network". In: *arXiv preprint arXiv:1312.4400* (2013).
- [LRM15] T. Lin, A. RoyChowdhury, and S. Maji. "Bilinear CNN models for fine-grained visual recognition". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1449–1457. DOI: [10.1109/ICCV.2015.170](https://doi.org/10.1109/ICCV.2015.170).
- [LM17] T.-Y. Lin and S. Maji. "Improved bilinear pooling with CNNs". In: *Proceedings British Machine Vision Conference (BMVC)*. 2017, pp. 117.1–117.12. DOI: [10.5244/C.31.117](https://doi.org/10.5244/C.31.117).
- [Liu+12] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur. "Dog breed classification using part localization". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2012, pp. 172–185. DOI: [10.1007/978-3-642-33718-5_13](https://doi.org/10.1007/978-3-642-33718-5_13).
- [LH16] N. Liu and J. Han. "DHSNet: deep hierarchical saliency network for salient object detection". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 678–686.
- [Liu+16] X. Liu, T. Xia, J. Wang, and Y. Lin. "Fully convolutional attention localization networks: efficient attention localization for fine-grained recognition". In: *CoRR* abs/1603.06765 (2016).
- [LS13] V. Ljubovic and H. Supic. "A compact color descriptor for image retrieval". In: *International Conference on Information, Communication and Automation Technologies*. 2013, pp. 1–5.
- [Low99] D. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 1999, pp. 1150–1157.
- [Low04] D. G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 0920-5691.

- [LKS15] Z. Lun, E. Kalogerakis, and A. Sheffer. "Elements of style: learning perceptual shape style similarity". In: *ACM Transactions on Graphics* 34.4 (2015), 84:1–84:14. DOI: [10.1145/2766929](https://doi.org/10.1145/2766929).
- [LCR01] M. R. Luo, G. Cui, and B. Rigg. "The development of the CIE 2000 colour-difference formula: CIEDE2000". In: *Color Research & Application* 26.5 (2001), pp. 340–350. DOI: [10.1002/col.1049](https://doi.org/10.1002/col.1049).
- [MH12] L. Maaten and G. Hinton. "Visualizing non-metric similarities in multiple maps". In: *Mach. Learn.* 87.1 (Apr. 2012), pp. 33–55. ISSN: 0885-6125. DOI: [10.1007/s10994-011-5273-4](https://doi.org/10.1007/s10994-011-5273-4).
- [Maj+13] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. *Fine-Grained Visual Classification of Aircraft*. Tech. rep. 2013. arXiv: [1306.5151](https://arxiv.org/abs/1306.5151).
- [Man+01] B. S. Manjunath, J. R. Ohm, V. V. Vasudevan, and A. Yamada. "Color and texture descriptors". In: *IEEE Transactions on Circuits and Systems for Video Technology* 11.6 (2001), pp. 703–715.
- [MTZ13] R. Margolin, A. Tal, and L. Zelnik-Manor. "What makes a patch distinct?" In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 1139–1146.
- [MP43] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133. DOI: [10.1007/BF02478259](https://doi.org/10.1007/BF02478259).
- [MP69] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969. ISBN: 978-0-262-63022-1.
- [NZ08] M. Nilsback and A. Zisserman. "Automated flower classification over a large number of classes". In: *Indian Conference on Computer Vision, Graphics Image Processing*. 2008, pp. 722–729. DOI: [10.1109/ICVGIP.2008.47](https://doi.org/10.1109/ICVGIP.2008.47).
- [Oh+17] S. J. Oh, R. Benenson, A. Khoreva, Z. Akata, M. Fritz, and B. Schiele. "Exploiting saliency for object segmentation from image level labels". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5038–5047.
- [Oqu+15] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. "Is object localization for free? - weakly-supervised learning with convolutional neural networks". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 685–694.
- [OCS05] Y. Ostrovsky, P. Cavanagh, and P. Sinha. "Perceiving illumination inconsistencies in scenes". In: *Perception* 34.11 (2005), pp. 1301–1314. DOI: [10.1068/p5418](https://doi.org/10.1068/p5418).
- [Pap+13] T. N. Pappas, D. L. Neuhoff, H. de Ridder, and J. Zujovic. "Image analysis: focus on texture similarity". In: *Proceedings of the IEEE* 101.9 (2013), pp. 2044–2057. DOI: [10.1109/JPROC.2013.2262912](https://doi.org/10.1109/JPROC.2013.2262912).
- [PG11] D. Parikh and K. Grauman. "Relative attributes". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2011, pp. 503–510.

- [Par+11] O. M. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. "The truth about cats and dogs". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2011, pp. 1427–1434. DOI: [10.1109/ICCV.2011.6126398](https://doi.org/10.1109/ICCV.2011.6126398).
- [Par+12] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. "Cats and dogs". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3498–3505. DOI: [10.1109/CVPR.2012.6248092](https://doi.org/10.1109/CVPR.2012.6248092).
- [PKD15] D. Pathak, P. Krähenbühl, and T. Darrell. "Constrained convolutional neural networks for weakly supervised segmentation". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1796–1804.
- [PHZ18] Y. Peng, X. He, and J. Zhao. "Object-part attention model for fine-grained image classification". In: *IEEE Transactions on Image Processing (TIP)* 27.3 (2018), pp. 1487–1500.
- [Per+12] F. Perazzi, P. Krahenbuhl, Y. Pritch, and A. Hornung. "Saliency filters: contrast based filtering for salient region detection". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 733–740.
- [PM15] N. Pourian and B. S. Manjunath. "Pixnet: a localized feature representation for classification and visual search". In: *IEEE Trans. on Multimedia* 17.5 (2015), pp. 616–625.
- [Ras+13] M. Rastegari, A. Diba, D. Parikh, and A. Farhadi. "Multi-attribute queries: to merge or not to merge?" In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 3310–3317.
- [Ree+16] S. E. Reed, Z. Akata, H. Lee, and B. Schiele. "Learning deep representations of fine-grained visual descriptions". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 49–58.
- [Rei+01] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. "Color transfer between images". In: *IEEE Computer Graphics and Applications* 21.5 (2001), pp. 34–41.
- [Ren+15] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: towards real-time object detection with region proposal networks". In: *Proceedings Advances in Neural Information Processing Systems (NeurIPS)*. 2015, pp. 91–99.
- [Ros58] F. Rosenblatt. "A probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: [10.1037/h0042519](https://doi.org/10.1037/h0042519).
- [RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0).
- [SS03] M. H. Safar and C. Shahabi. "Image similarity measures". In: *Shape Analysis and Retrieval of Multimedia Objects*. 2003, pp. 9–11. DOI: [10.1007/978-1-4615-0349-1_2](https://doi.org/10.1007/978-1-4615-0349-1_2).
- [SGS10] K. van de Sande, T. Gevers, and C. Snoek. "Evaluating color descriptors for object and scene recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 32.9 (2010), pp. 1582–1596.

- [Sel+17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. "Grad-cam: visual explanations from deep networks via gradient-based localization". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626.
- [Ser+12] M. Serra, O. Penacchio, R. Benavente, and M. Vanrell. "Names and shades of color for intrinsic image estimation". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 278–285.
- [SWD05] G. Sharma, W. Wu, and E. N. Dalal. "The CIEDE2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations". In: *Color Research & Application* 30.1 (2005), pp. 21–30. DOI: [10.1002/col.20070](https://doi.org/10.1002/col.20070).
- [Shi+14] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. "Style transfer for headshot portraits". In: *ACM Transactions on Graphics* 33.4 (2014), 148:1–148:14.
- [SY16] W. Shimoda and K. Yanai. "Distinct class-specific saliency maps for weakly supervised semantic segmentation". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2016, pp. 218–234.
- [SHR16] E. Siahaan, A. Hanjalic, and J. Redi. "A reliable methodology to collect ground truth data of image aesthetic appeal". In: *IEEE Transactions on Multimedia* 18.7 (2016), pp. 1338–1350. DOI: [10.1109/TMM.2016.2559942](https://doi.org/10.1109/TMM.2016.2559942).
- [SFD11] B. Siddiquie, R. S. Feris, and L. S. Davis. "Image ranking and retrieval based on multi-attribute queries". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 801–808.
- [SVZ13] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep inside convolutional networks: visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).
- [SZ15] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *Proceedings International Conference on Learning Representations (ICLR)*. 2015.
- [Sun+18] M. Sun, Y. Yuan, F. Zhou, and E. Ding. "Multi-attention multi-class constraint for fine-grained image recognition". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2018, pp. 834–850.
- [Sze+15a] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.
- [Sze+17] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Proceedings AAAI Conference on Artificial Intelligence*. 2017, pp. 4278–4284.
- [Sze+15b] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.

- [Sze+16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. “Rethinking the inception architecture for computer vision”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826.
- [TAF06] M. F. Tappen, E. H. Adelson, and W. T. Freeman. “Estimating intrinsic component images using non-linear regression”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2006, pp. 1992–1999.
- [Van+15] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. “Building a bird recognition app and large scale dataset with citizen scientists: the fine print in fine-grained dataset collection”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 595–604. DOI: [10.1109/CVPR.2015.7298658](https://doi.org/10.1109/CVPR.2015.7298658).
- [Ved+14] A. Vedaldi, S. Mahendran, S. Tsogkas, S. Maji, R. B. Girshick, J. Kannala, E. Rahtu, I. Kokkinos, M. B. Blaschko, D. J. Weiss, B. Taskar, K. Simonyan, N. Saphra, and S. Mohamed. “Understanding objects in detail with fine-grained attributes”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 3622–3629.
- [Vel01] R. C. Veltkamp. “Shape matching: similarity measures and algorithms”. In: *International Conference on Shape Modeling and Applications*. 2001, pp. 188–197. DOI: [10.1109/SMA.2001.923389](https://doi.org/10.1109/SMA.2001.923389).
- [Wah+11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011.
- [Wan+15] D. Wang, Z. Shen, J. Shao, W. Zhang, X. Xue, and Z. Zhang. “Multiple granularity descriptors for fine-grained categorization”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2399–2406. DOI: [10.1109/ICCV.2015.276](https://doi.org/10.1109/ICCV.2015.276).
- [Wan+14] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. “Learning fine-grained image similarity with deep ranking”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1386–1393.
- [WKH17] X. Wang, K. M. Kitani, and M. Hebert. “Contextual visual similarity”. In: *arXiv:1612.02534* (2017).
- [Wei+18] X. Wei, Y. Zhang, Y. Gong, J. Zhang, and N. Zheng. “Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification”. In: *Proceedings European Conference on Computer Vision (ECCV)*. 2018, pp. 365–380.
- [Wei+17] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan. “Object region mining with adversarial erasing: a simple classification to semantic segmentation approach”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6488–6496.

- [WGB06] J. van de Weijer, T. Gevers, and A. D. Bagdanov. "Boosting color saliency in image feature detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 28.1 (2006), pp. 150–156.
- [WSJ17] S. Workman, R. Souvenir, and N. Jacobs. "Understanding and mapping natural beauty". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5589–5598.
- [Xia+15] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. "The application of two-level attention models in deep convolutional neural network for fine-grained image classification". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 842–850. DOI: [10.1109/CVPR.2015.7298685](https://doi.org/10.1109/CVPR.2015.7298685).
- [Xie+15] S. Xie, T. Yang, Xiaoyu Wang, and Yuanqing Lin. "Hyper-class augmented and regularized deep learning for fine-grained image classification". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2645–2654. DOI: [10.1109/CVPR.2015.7298880](https://doi.org/10.1109/CVPR.2015.7298880).
- [Xu+18a] H. Xu, G. Qi, J. Li, M. Wang, K. Xu, and H. Gao. "Fine-grained image classification by visual-semantic embedding". In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2018, pp. 1043–1049. DOI: [10.24963/ijcai.2018/145](https://doi.org/10.24963/ijcai.2018/145).
- [Xu+18b] Z. Xu, S. Huang, Y. Zhang, and D. Tao. "Webly-supervised fine-grained visual categorization via deep domain adaptation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 40.5 (2018), pp. 1100–1113. DOI: [10.1109/TPAMI.2016.2637331](https://doi.org/10.1109/TPAMI.2016.2637331).
- [Yan+15] L. Yang, P. Luo, C. C. Loy, and X. Tang. "A large-scale car dataset for fine-grained categorization and verification". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3973–3981. DOI: [10.1109/CVPR.2015.7299023](https://doi.org/10.1109/CVPR.2015.7299023).
- [Yan+16] X. Yang, T. Zhang, C. Xu, S. Yan, M. S. Hossain, and A. Ghoneim. "Deep relative attributes". In: *IEEE Transactions on Multimedia* 18.9 (2016), pp. 1832–1842.
- [Yan+18] Z. Yang, T. Luo, D. Wang, Z. Hu, J. Gao, and L. Wang. "Learning to navigate for fine-grained classification". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2018, pp. 438–454.
- [YG14] A. Yu and K. Grauman. "Fine-grained visual comparisons with local learning". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 192–199.
- [Yu+18] C. Yu, X. Zhao, Q. Zheng, P. Zhang, and X. You. "Hierarchical bilinear pooling for fine-grained visual recognition". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2018, pp. 595–610.
- [YWD18] F. Yu, D. Wang, and T. Darrell. "Deep layer aggregation". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2403–2412.

- [ZF14] M. Zeiler and R. Fergus. "Visualizing and understanding convolutional networks". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2014, pp. 818–833.
- [Zha+16a] H. Zhang, T. Xu, M. Elhoseiny, X. Huang, S. Zhang, A. Elgammal, and D. Metaxas. "SPDA-CNN: unifying semantic part detection and abstraction for fine-grained recognition". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1143–1152. DOI: [10.1109/CVPR.2016.129](https://doi.org/10.1109/CVPR.2016.129).
- [Zha+14a] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. "Part-based r-cnn for fine-grained category detection". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2014, pp. 834–849.
- [Zha+14b] N. Zhang, M. Paluri, M. Rantazo, T. Darrell, and L. Bourdev. "Panda: pose aligned networks for deep attribute modeling". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1637–1644. DOI: [10.1109/CVPR.2014.212](https://doi.org/10.1109/CVPR.2014.212).
- [Zha+15a] N. Zhang, E. Shelhamer, Y. Gao, and T. Darrell. "Fine-grained pose prediction, normalization, and recognition". In: *CoRR abs/1511.07063* (2015).
- [Zha+18] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 586–595.
- [Zha+16b] X. Zhang, H. Xiong, W. Zhou, W. Lin, and Q. Tian. "Picking deep filter responses for fine-grained image recognition". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1134–1142. DOI: [10.1109/CVPR.2016.128](https://doi.org/10.1109/CVPR.2016.128).
- [ZTJ18] Y. Zhang, H. Tang, and K. Jia. "Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data". In: *Proceedings European Conference on Computer Vision (ECCV)*. 2018, pp. 241–256.
- [Zha+17] B. Zhao, X. Wu, J. Feng, Q. Peng, and S. Yan. "Diversified visual attention networks for fine-grained object classification". In: *IEEE Transactions on Multimedia* 19.6 (2017), pp. 1245–1256. DOI: [10.1109/TMM.2017.2648498](https://doi.org/10.1109/TMM.2017.2648498).
- [Zha+15b] R. Zhao, W. Ouyang, H. Li, and X. Wang. "Saliency detection by multi-context deep learning". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1265–1274.
- [Zhe+17] H. Zheng, J. Fu, T. Mei, and J. Luo. "Learning multi-attention convolutional neural network for fine-grained image recognition". In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5219–5227. DOI: [10.1109/ICCV.2017.557](https://doi.org/10.1109/ICCV.2017.557).
- [Zhe+19] H. Zheng, J. Fu, Z.-J. Zha, and J. Luo. "Looking for the devil in the details: learning trilinear attention sampling network for fine-grained image recognition". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5007–5016.

- [Zhe+15] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. “Conditional random fields as recurrent neural networks”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1529–1537.
- [Zho+18] G. Zhong, Y. Zheng, X. Zhang, H. Wei, and X. Ling. “Convolutional discriminant analysis”. In: *International Conference on Pattern Recognition (ICPR)*. 2018, pp. 1456–1461.
- [Zho+16] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. “Learning deep features for discriminative localization”. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2921–2929.
- [Zho+15] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. “Object detectors emerge in deep scene cnns”. In: *Proceedings International Conference on Learning Representations (ICLR)*. 2015.
- [Zhu+17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2223–2232.
- [ZD14] C. L. Zitnick and P. Dollár. “Edge boxes: locating object proposals from edges”. In: *Proceedings European Conference on Computer Vision (ECCV)*. 2014, pp. 391–405.
- [ZPN13] J. Zujovic, T. N. Pappas, and D. L. Neuhoff. “Structural texture similarity metrics for image analysis and retrieval”. In: *IEEE Trans. on Image Processing* 22.7 (2013), pp. 2545–2558. DOI: [10.1109/TIP.2013.2251645](https://doi.org/10.1109/TIP.2013.2251645).