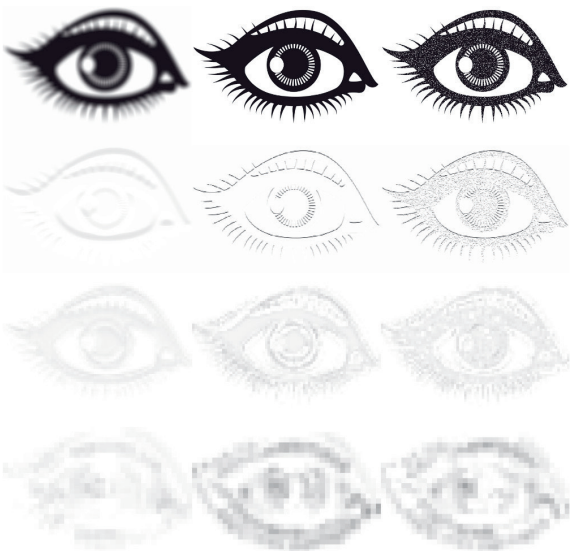
DOCTORAL THESIS

Deep Perceptual Loss and Similarity



Gustav Grund Pihlgren

Machine Learning

LULEÅ UNIVERSITY OF TECHNOLOGY

Deep Perceptual Loss and Similarity

Gustav Grund Pihlgren

Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden

Supervisors:

Marcus Liwicki and Fredrik Sandin

To all the teachers I ever had, at school, at home, and beyond

Abstract

This thesis investigates deep perceptual loss and (deep perceptual) similarity; methods for computing loss and similarity for images as the distance between the deep features extracted from neural networks. The primary contributions of the thesis consist of (i) aggregating much of the existing research on deep perceptual loss and similarity, and (ii) presenting novel research into understanding and improving the methods. This novel research provides insight into how to implement the methods for a given task, their strengths and weaknesses, how to mitigate those weaknesses, and if these methods can handle the inherent ambiguity of similarity.

Society increasingly relies on computer vision technology, from everyday smartphone applications to legacy industries like agriculture and mining. Much of that groundbreaking computer vision technology relies on machine learning methods for their success. In turn, the most successful machine learning methods rely on the ability to compute the similarity of instances.

In computer vision, computation of image similarity often strives to mimic human perception, called perceptual similarity. Deep perceptual similarity has proven effective for this purpose and achieves state-of-the-art performance. Furthermore, this method has been used for loss calculation when training machine learning models with impressive results in various computer vision tasks. However, many open questions exist, including how to best utilize and improve the methods. Since similarity is ambiguous and context-dependent, it is also uncertain whether the methods can handle changing contexts.

This thesis addresses these questions through (i) a systematic study of different implementations of deep perceptual loss and similarity, (ii) a qualitative analysis of the strengths and weaknesses of the methods, (iii) a proof-of-concept investigation of the method's ability to adapt to new contexts, and (iv) cross-referencing the findings with already published works.

Several interesting findings are presented and discussed, including those below. Deep perceptual loss and similarity are shown not to follow existing transfer learning conventions. Flaws of the methods are discovered and mitigated. Deep perceptual similarity is demonstrated to be well-suited for applications in various contexts.

There is much left to explore, and this thesis provides insight into what future research directions are promising. Many improvements to deep perceptual similarity remain to be applied to loss calculation. Studying how related fields have dealt with problems caused by ambiguity and contexts could lead to further improvements. Combining these improvements could lead to metrics that perform close to the optimum on existing datasets, which motivates the development of more challenging datasets.

Contents

Acknowledgments

The research for and work on this thesis was conducted at Luleå University of Technology in the Machine Learning Group of EISLAB at the Department of Computer Science, Electrical and Space Engineering. As such, I would like to thank everyone in and around the University, department, division, and research group who have made my work and time here at LTU possible and enjoyable. I wank to thank everyone, from administrators, custodians, and support personnel making sure that there was a university for me to work at to researchers and teachers providing guidance and assistance along the way.

I also want to give my thanks to the friends and family who have encouraged me in times of need and made my days enjoyable. From the friends and family that have supported me from afar and whom I always looked forward to visiting, to the friends and family I met in Luleå that have made my time here rewarding.

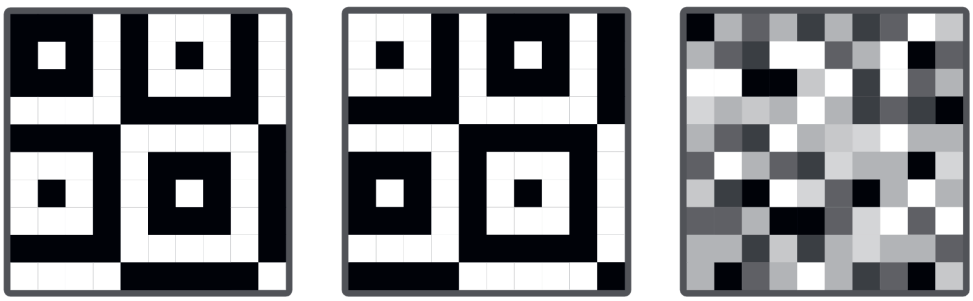
Among these people are some I would like to mention especially. To my supervisors, Marcus Liwicki and Fredrik Sandin, thanks for encouraging me to stake a path of my own while guiding and supporting me above and beyond expectations. To my close colleagues here at EISLAB who have become great collaborators and friends, thanks for the discussions around the meeting room and lunchroom. To my closest family, including mom, dad, brothers, sister, bonus parents, grandparents, partner, and beyond, thanks for your unending support, enthusiasm, and pride. To my friends in "Torsdagsgänget", thanks for keeping me sane and entertained. To my friends in LARPing, thanks for the wonderfully inclusive and welcoming community.

Finally, I want to give thanks and credit to those who made this thesis possible. Thanks to my collaborators: Marcus Liwicki, Fredrik Sandin, Konstantina Nikolaidou, Oskar Sjögren, Nosheen Abid, Rajkumar Saini, and others whose work did not fit in this thesis. Thanks to those who have reviewed this thesis: Marcus Liwicki, Fredrik Sandin, Nosheen Abid, Richa Upadhyay, Karl Löwenmark, and Nicklas Lallo. Thanks to the opponent: Hugo Larochelle; the grading committee: Kary Främling, Atsuto Maki, Athanasios Katsamanis, and Mikael Sjödahl; and the chairperson; Foteini Simistira Liwicki.

Luleå, April 2023
Gustav Grund Pihlgren

Chapter 1

Introduction



(a) (b) (c)

Figure 1.1: The currently most popular image similarity metrics used in machine learning calculate (a) and (b) to be less similar than (b) and (c).

In Fig. 1.1, which of the side images (a) and (c) is most similar to the middle image (b)? Seems simple enough, and most people would likely agree with you. However, the most common ways that image similarity is computed for training machine learning models, pixel-wise metrics, would disagree with you. One can wonder how it is possible for models that are trained to generate images to get anything close to the images they are trained at when the way that the similarity is measured has such blatant flaws. Yet, despite these well-known flaws and many alternatives being available, pixel-wise metrics remain popular.

Even without adopting newer improved metrics, machine learning has been successful [1], and now those successes are being turned to improve similarity metrics further. In recent years, the deep features learned by image processing artificial neural networks have been used to better than ever approximate the human perception of similarity [2, 3, 4].

This breakthrough in using machine learning models for similarity metrics seems to have finally inspired the use of those similarity metrics to improve machine learning. The coming chapters will present how the paradigm of using deep features to approximate image similarity solves the issue presented in Fig. 1.1 and others. This thesis will cover how to best use these similarity metrics, their flaws, how they can be improved upon, and discuss the challenges posed by the very definition of similarity.

1.1 Deep Learning and Loss Calculation

In the last decade, Artificial Intelligence (AI) and machine learning have been rapidly integrated into many parts of society, from systems recommending entertainment to individuals to integral parts of essential industries. During that same time, machine learning and its related fields of research experienced a large shift. This shift in machine learning and its accelerated integration into society are related, and the common denominator is the incredible performance of artificial neural networks (henceforth called neural networks) and deep learning.

Prior to deep learning, the typical machine learning setup consisted of a feature-engineered system that extracts features that human experts have decided on, and then a simple learning system is trained to use those features to generate appropriate outputs. While this approach was successful in some cases, it had three major flaws. First, it required human experts to decide which features would be useful for a given task, and then someone had to design an algorithm for extracting each of those features. This is resource intensive and cannot easily be adapted to new tasks. Second, the features needed to be such that a human could come up with an algorithm to extract them. For example, for a cat detection system, it would be useful to have a "furry"-feature, but designing a system by hand to detect fur from pixels is an incredibly complex problem. Third, some features that would be useful for prediction are difficult for a human to think of and articulate. This means that many potentially useful features are not extracted because the expert could not think of them or explain them to the system designer.

One significant factor in the success of deep learning is that it can be used to create systems that learn not only to make predictions from features but also to learn the features needed to make that prediction. This process of learning predictions directly from raw data is referred to as end-to-end learning [5]. By stacking several simple neural network layers, a Deep Neural Network (DNN) is created where each layer uses the features from the previous layer to themselves output slightly more advanced features for the next layer to use. This process is illustrated in Fig. 1.2, where each layer is trained to create increasingly more complex features by combining previous features. While these deep learning models are larger and therefore need more data and computation power to train, the features learned by one DNN can often be reused in another DNN for a comparable problem. This procedure of reusing parts of a trained model for another task is called transfer learning and can be used to save resources and train machine learning models on tasks where there is too little data to train a very large network.

The most popular way to train DNNs is to use the backpropagation algorithm [5]. The

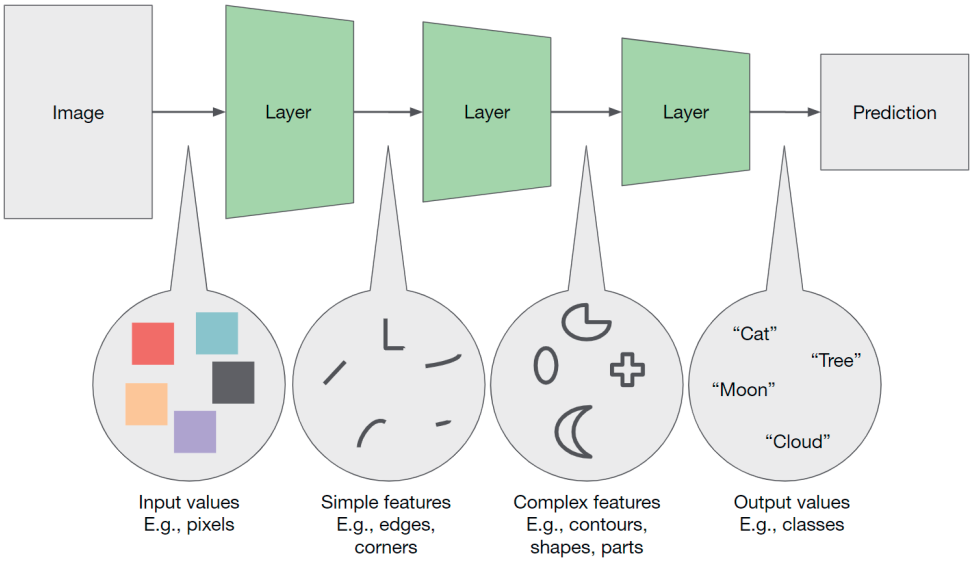


Figure 1.2: A schematic abstraction of an end-to-end deep learning system illustrating how higher-level concepts are learned. Each layer has learned to combine the simpler features of the previous layer to create more complex features and eventually use those to make a prediction.

input to the network is processed through each layer to give an output. That output, and potentially other factors, are used to calculate the loss of the network. The loss is used as a performance indicator for the network. The network is optimized during training to minimize the loss and therefore improving performance. Optimization is performed by sending the loss backward through the network, and at each layer, the influence of each parameter on the loss is computed. This measurement of influence can then be used to change the parameters in a way that would lower the loss. In summary, backpropagation computes how each parameter of the network affected the loss, and those parameters are updated to lower the loss.

Since loss is such a fundamental part of deep learning, how to calculate it is an important subject. An algorithm that calculates a loss is known as a loss function. There are many ways to define a loss function, several of which can be used jointly [1, 6]. Extrinsic rewards defined by the environment could be used directly to punish the network for not reaching some specified goal. Regulatory losses can force the network to adhere to certain desired properties, such as using all neurons equally. However, the most common type of loss calculation is to compare the output of the network directly with some defined desired output [6].

Due to its significance in training many popular machine learning models, studying how to compare models and desired outputs are of interest. For some machine learning

tasks, this can seem straightforward. For example, a model meant to estimate the publishing date of a book based on its writing could directly use the difference between the model output and the true date as a loss. So if the model outputs 1475 and the true date is 1525, then the loss would be 50. However, even in this simple example, there are many things to consider. For example, the loss does not have to be the absolute difference, and the squared difference is often used instead. When predicting multiple output values, should the difference use the sum of absolute differences or the distance in Euclidean space? Despite these complexities, the field of machine learning has established practices for deciding how to calculate the loss of simple cases where single numbers or classifications are being compared. In other cases, however, calculating the difference between two instances are entire research areas.

1.2 Similarity Metrics and Computer Vision

To computationally measure the difference between two instances, or conversely their similarity, falls under the umbrella term of similarity metrics. Such metrics are directly applicable to a host of problems. Content-based image retrieval search is a problem of finding the most similar images to the query in a given dataset [7]. Finding out how individuals or even entire species are related could be done with a similarity measure of their respective genomes [8]. Image Quality Assessment (IQA) is the problem of finding the quality of an image, often by comparing the similarity of that image to an ideal version [9].

However, similarity is not an easily defined concept. If you ask which pair of symbols in Fig. 1.3 is more similar compared to the other combinations, you will get many different answers, and more than a few would likely ask "similar in what way?". This is a fair question and begs the further question "what is the goal of calculating similarity?". If similarity itself has no fixed definition, perhaps it is better to focus on what the purpose of measuring is. In our examples above, the purposes could be to find the origin of the query image, to find out what species are more closely related, and to tell which image compression humans prefer. For a given purpose, a similarity metric is better if it manages to fulfill the purpose better than another metric. For loss functions that calculate the similarity of model outputs and desired outputs, a good metric is one where the model trained with the loss function performs well.

One field where machine learning has been successful and where no clear definition of similarity exists is computer vision. Computer vision deals with the computational processing of visual data, such as images and video. The field includes a wide breadth of applications and has had a large impact on society. These applications include improving the efficacy of medical imaging [10], development of autonomous driving and related technologies [11], and automated fault detection in production [12].

With the prominence of machine learning within the field and its recent successes, looking at how loss, and therefore similarity, is calculated is of interest. Though, not all image processing requires similarity comparisons of images to calculate the loss. Image classification, for example, can be performed by simply comparing the output label

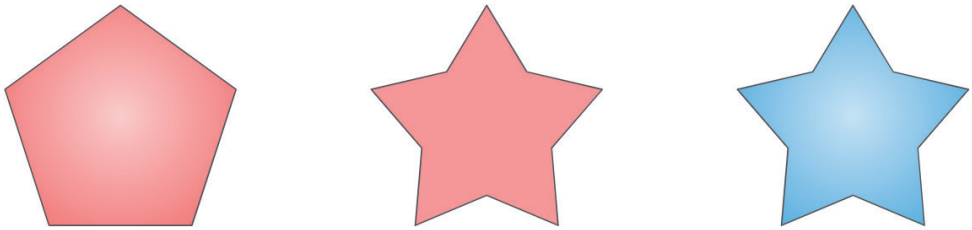


Figure 1.3: Which pair of symbols are most similar to each other can have many different answers.

with the actual label. However, even with those tasks removed, a wide array of models remain that rely on image comparisons for loss calculation. These include every model that outputs an image, such as image generation [13], image autoencoding [14], and super-resolution upscaling [15]. It also includes models with image-like outputs in the form of pixel-lattices, such as image segmentation [16], object detection [17], and depth prediction [18]. Therefore, calculating image similarity is a relevant task to consider.

The difficulty of comparing images stems from the highly complex correlations between the pixels they are composed of. In other domains, directly comparing the values that composite the output can be sensible, but, as is exemplified by Fig. 1.1, this is not the case for images. By only comparing the pixels directly, the correlations to other pixels are ignored, leading to higher-level structures in the image not being considered. Additionally, each pixel is considered equally valuable to the image, while a human would likely think that some regions of an image are more important than others. An example of these complexities could be a portrait. In a portrait, it is more important that the pixels of the eyes have the correct structure than that each pixel individually are as close as possible to the correct color. It is also more important that the region of the face is correct than the background behind it.

The difficulty of calculating the similarity of images has been a well-known problem for decades [19] and has become a significant problem within the field of similarity metrics. While innumerable definitions of image similarity exist, one that has been the primary focus is perceptual similarity. Perceptual similarity is defined as how similar a human perceives the images to be. While it can be difficult for a human to give a numerical value of similarity exactly, it can be approximated through other questions, such as which pair of three images are more similar, as presented in Fig. 1.3. Even perceptual similarity, however, is ambiguous as different people have different opinions, and even a single person might think that two pairs of images are similar in different ways. This ambiguity has not been as large an issue as it first seems to be. There are three main reasons for this. First, the average judgment of a large population of people has been used to approximate what humans generally tend to agree on [20, 21, 22, 2]. Second, the human judgments are gathered on images that are different in similar ways, which to some degree eliminates the ambiguity [20, 21, 22, 2]. Third, until recently, perceptual metrics were not even close to human performance even with the simplifications described, which meant that

the problem of ambiguity could be mostly postponed since the metrics struggled even where people tend to strongly agree [2].

Over the decades, several metrics have been proposed to calculate the similarity of images. From simple pixel-wise metrics, which average the differences between the corresponding pixels of the two images, to the more complex Structural Similarity Index Measure (SSIM) [9], which uses a combination of the means, variances, covariances, and more of several patches of the images. Until recently, these metrics were mathematical functions handcrafted by experts using their intuition of what it means for images to be similar. However, these handcrafted metrics were far from achieving human performance and had known cases where they diverged from human judgements [13, 2]. Recently, however, deep learning and feature learning have been successfully turned to the task. The deep features learned by image processing networks such as Convolutional Neural Networks (CNNs) while trained on tasks like image classification can also be used to measure the similarity of the images. Rather than directly comparing two images with one another, like in pixel-wise metrics, the two images are used as inputs to a neural network. The features that the deep layers extract from each image can then be compared directly to each other. Due to their use of deep features, these metrics have been dubbed deep perceptual similarity metrics. The reasoning behind the metrics is that the neural networks learn complex features representing higher-level concepts of the images and that two images with similar features likely contain similar concepts. This reasoning is backed by the excellent performance of deep perceptual similarity metrics, which comes close to competing with humans [2].

The general setup for deep perceptual similarity is visualized in Fig 1.4. A metric consists of a loss network and a comparison method. The loss network is an image-processing neural network from which deep features from the input images are extracted. The comparison method is the function by which the difference between the deep features of the two images is calculated. Implementations of deep perceptual similarity metrics can therefore vary by changing loss networks and comparison methods. For the loss networks, there are many different neural network architectures that could be used, the parameters of that architectures can be set through many different procedures (typically through pretraining), and there are many different ways to choose the features to extract. Loss network implementation is further analyzed in Chapter 3. Just like the similarity of outputs can be calculated in many different ways, so too are there many methods for comparing deep features. Comparison methods are covered in greater detail in Chapter 4.

Despite their known flaws, loss calculation for images in the past decades mostly used pixel-wise metrics [13, 23]. Even as better perceptual similarity metrics have been developed, pixel-wise losses have remained the most popular. Only in the last few years, with the advent of deep perceptual similarity, have other similarity metrics started being tested for loss calculation. Older perceptual similarity metrics, like SSIM, have recently been investigated as loss functions [24, 25]. Though, deep perceptual similarity metrics are the ones that have seen the widest adoption and most success [13, 15, 23, 14]. With deep perceptual similarity used in state-of-the-art perceptual similarity metrics and its growing use in loss functions, it is an interesting subject of study.

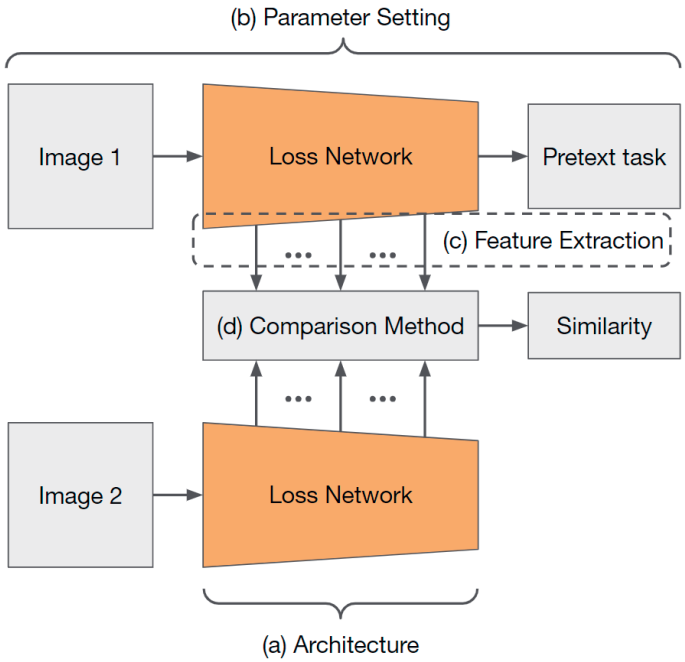


Figure 1.4: An overview of the implementation of a deep perceptual similarity metric. The metric calculates a similarity score of two images by comparing the deep features extracted by a neural network called a loss network. The implementation of metrics can vary by using loss networks with different (a) architectures, (b) parameters of that architecture, and (c) feature extraction. Besides the loss network, the (d) comparison method used to get the difference between the deep features can also be changed.

The focus of this thesis is the study of deep perceptual similarity and how it is being used to calculate the so-called deep perceptual loss. While it is a promising method within computer vision, much remains to be explored. How to define a deep perceptual similarity metric that best suits a given task? What neural network should be used to extract deep features? From which layers should those features come? How should the network be trained? Is the metric that performs best on perceptual similarity necessarily the one that gives the best training when used in a loss function? How should the extracted features be compared to each other? Do the metrics have flaws similar to those that came before them? Now that metrics are coming close to human performance is the inherent ambiguity of similarity an issue? These questions, among others, are the ones that will be addressed in this thesis.

1.3 Scope of the Thesis

Deep perceptual loss and (deep perceptual) similarity has grown in popularity recently. However, using features and outputs of neural networks is an older practice that covers many techniques beyond deep perceptual loss and similarity [26, 27, 28]. The existence of closely related techniques with comparable use cases makes defining a strict border around deep perceptual loss and similarity difficult.

In this thesis, a *perceptual similarity metric* is a metric that takes two images as input and is supposed to approximate human perception of similarity. *Deep perceptual similarity* refers to perceptual similarity metrics where the similarity is computed as the difference of the deep features of a neural network when the images are used as input. Following these definitions, *perceptual loss* and *deep perceptual loss* are loss functions that include, respectively, a perceptual and deep perceptual similarity metric. Finally, the network which performs the feature extraction used for comparison in deep perceptual loss and similarity is referred to as a *loss network*.

The research encompassing deep perceptual loss and similarity is wide, and while this thesis will touch on most of that field, it cannot go into depth on all subjects. Instead, this thesis focuses on understanding and improving deep perceptual similarity and loss with pretrained loss networks for image and image-like applications. The thesis covers the history and applications of the methods (Chapter 2), the effect of using different pretraining, architectures, and extraction layers (Chapter 3), what known flaws exist and their possible mitigation techniques are (Chapter 4), how the methods handle the inherent ambiguity in similarity (Chapter 5), and finally a summary, ethical considerations, and a discussion on future work (Chapter 6).

1.4 Knowledge Gap

The research on deep perceptual loss and similarity has tended towards its applications rather than the methods themselves. Particularly, deep perceptual loss research has rarely explored how the method is best utilized in favor of using the same implementations as previous works or not justifying the chosen implementation at all. In the field of deep perceptual similarity, more work has been conducted regarding how to make better metrics, but there is still much left to explore. This has left a knowledge gap in the understanding of why the methods work so well, how best to implement them for a given task, and whether significant drawbacks exist. Additionally, the studies that do exist do not tend to build on each other, leaving gaps in places where they could overlap. Some of these knowledge gaps have been formalized in the research questions that comprise the backbone of this thesis.

1.5 Research Questions

This work follows three research questions, each of which is further subdivided into three sub-questions. The questions each deal with understanding deep perceptual loss and similarity and are described in detail along with their sub-questions in the subsections below.

Q1 How does the loss network implementation affect downstream performance?

Q2 What significant flaws do deep perceptual similarity metrics have, and how can they be mitigated?

Q3 To what extent can deep perceptual similarity handle the inherent ambiguity of similarity?

1.5.1 Research Question 1

How does the loss network implementation affect downstream performance? Many previous perceptual metrics have only a few variants with a handful of parameters to decide on. For these metrics choosing the version best suited to a given task can be as simple as trying all the alternatives or using established best practices. However, deep perceptual similarity is not so simple. At the core of deep perceptual similarity metrics are DNNs to use for feature extraction, the loss networks. There are thousands of neural network architectures to choose from, thousands of ways to pretrain them, and thousands of ways to extract the deep features. All of these factors are part of the loss networks, and testing all combinations for every application is impossible. Instead, a study of general strategies for implementing loss networks is a path forward. Such strategies could then be used to implement the appropriate loss network for a given task. With this in mind, the research question can be split into three sub-questions to consider.

Q1.1 How does the DNN architecture of a loss network affect downstream performance?

Q1.2 How does the pretraining of a loss network affect downstream performance?

Q1.3 How does the feature extraction of a loss network affect downstream performance?

1.5.2 Research Question 2

What significant flaws do deep perceptual similarity metrics have, and how can they be mitigated?Older metrics of perceptual similarity have well-documented flaws. Examples include preferring different image pairs than humans [2] and, in some cases considering random noise to be more similar than simple transformations of the original image [29]. Comparatively, deep perceptual similarity does not have as many known flaws. Due to being relatively new, deep perceptual similarity has yet to face serious scrutiny, and the complexity of implementation means that it can be difficult to intuit or discover such

flaws. For some perceptual metrics, it is easy to imagine a scenario where the calculation will go against human perception, but to do so for deep perceptual similarity is more difficult. Investigation of deep perceptual similarity is both needed and complicated. However, many methods for investigating the behavior of deep learning methods exist that can be used for this purpose. With this in mind, the research question can be subdivided into the following sub-questions.

Q2.1 Why is deep perceptual similarity able to solve perceptual similarity tasks where pixel-wise metrics fail?

Q2.2 How can counterexamples be generated where current deep perceptual similarity metrics fail to assess perceptual similarity correctly?

Q2.3 How can deep perceptual similarity be improved to mitigate existing flaws?

1.5.3 Research Question 3

To what extent can deep perceptual similarity handle the inherent ambiguity of similarity? Similarity is inherently ambiguous, with many conflicting definitions. Even when limited to perceptual similarity, which approximates the human perception of similarity, ambiguity is an issue. Currently, ambiguity is not a prevalent issue in the development of perceptual similarity metrics. As metrics come closer to human performance on existing datasets, improved performance might require metrics that are specialized to given contexts. This poses the question of whether deep perceptual similarity would be able to adapt to varying contexts and what such adaption would entail. Furthermore, one must consider whether there is any need for adaptability or if current implementations of deep perceptual similarity are enough for the challenges facing computer vision and machine learning. These thoughts have been formalized into the following sub-questions.

Q3.1 How does ambiguity in similarity pose a problem for perceptual similarity metrics?

Q3.2 How can deep perceptual similarity metrics be adapted to different definitions of similarity?

Q3.3 What are the benefits of having deep perceptual similarity metrics able to adapt to different definitions of similarity?

1.6 Contributions

This thesis aims to give insight into the use of deep perceptual similarity and address the research questions posed above. The questions are addressed based on the research conducted as part of the work for this thesis as well as scientific works produced by other authors.

Many of the contributions have already been made public through the publication of research papers or preprints of unpublished manuscripts. Those works are listed in

Subsection 1.6.1. The specific contributions made by this thesis and the listed works are enumerated in Subsection 1.6.2 Additionally, some scientific works that do not fit this thesis have been produced throughout the research. Those works are listed in Subsection 1.6.3. Along with each work is a description of this author's personal contribution to that work.

1.6.1 Scientific Works Produced for This Thesis

Most of the findings made throughout the creation of this thesis have been formalized into research papers (and yet unpublished manuscripts). These works, along with the author's contribution to them, are listed below.

[14] Gustav Grund Pihlgren, Fredrik Sandin and Marcus Liwicki. Improving image autoencoder embeddings with perceptual loss. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.
Personal Contribution: Conceptualization, methodology, experimentation, first draft, and writing.

[30] Gustav Grund Pihlgren, Fredrik Sandin and Marcus Liwicki. Pretraining image encoders without reconstruction via feature prediction loss. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4105–4111, 2021.
Personal Contribution: Conceptualization, methodology, experimentation, first draft, and writing.

[31] Gustav Grund Pihlgren. *Deep perceptual loss for improved downstream prediction.* Licentiate thesis, Luleå tekniska universitet, 2021.
Personal Contribution: Everything in Part 1 and as specified above for papers [14] and [30] which make up Part 2.

[29] Oskar Sjögren, Gustav Grund Pihlgren, Fredrik Sandin and Marcus Liwicki. Identifying and mitigating flaws of deep perceptual similarity metrics. In *Proceedings of the Northern Lights Deep Learning Workshop 2023*, 2023.
Personal Contribution: Conceptualization and supervision. Methodology, experimentation, and writing jointly with Oskar Sjögren.

[32] Gustav Grund Pihlgren, Konstantina Nikolaidou, Prakash Chandra Chhipa, Nosheen Abid, Rajkumar Saini, Fredrik Sandin, and Marcus Liwicki. A systematic performance analysis of deep perceptual loss networks breaks transfer learning conventions. arXiv preprint, 2023.
Personal Contribution: Conceptualization, organization and planning, methodology, half of the experimentation, result compilation, analysis, and half of the writing.

[33] Gustav Grund Pihlgren, Fredrik Sanding, and Marcus Liwicki. Deep perceptual similarity is adaptable to ambiguous contexts. arXiv preprint

Personal Contribution: Conceptualization, methodology, experimentation, result compilation, analysis, and drafting the manuscript.

1.6.2 List of Scientific Contributions

This thesis and the works produced as part of the research for it have provided several scientific contributions to the field. Summaries of the primary contributions are listed below, ordered by which chapters and works each contribution is primarily related to.

Entire Thesis:

1. Compiling much of the research on understanding and improving deep perceptual loss and similarity in this thesis.

Subsection 2.6.3 and in [14, 31]:

2. Demonstrating that deep perceptual loss can be effectively used for dimensionality reduction for downstream prediction.

Subsection 2.6.3 and in [30, 31]:

3. Demonstrating that knowledge distillation through feature prediction can be used effectively for dimensionality reduction for downstream prediction, yielding similar results to training with deep perceptual loss but with less computational demand.

Chapter 3 and in [32]:

4. Systematically evaluating and analyzing how different architectures and extraction layers affect the performance of loss networks for deep perceptual loss and similarity.
5. Providing guidance for implementing loss networks using deep perceptual loss and similarity to improve downstream performance.
6. Discussing the implications of the findings regarding loss network implementation, including suggesting future studies to cover the knowledge gaps in this area.
7. Suggesting that deep perceptual loss does not adhere to common transfer learning conventions and that those conventions may not be as universal as previously thought.

Chapter 4 and in [29]:

8. Discovering through qualitative analysis several flaws in the most common variant of deep perceptual similarity and demonstrating that other variants do not have those flaws.
9. Proposing a new variant of deep perceptual similarity where the deep features are compared after sorting them by value and demonstrating qualitatively that it does not have the same significant flaws as the most common variant.

10. Demonstrating through qualitative analysis that the proposed variant of deep perceptual similarity along with another variant has other flaws and proposing future research to potentially overcome those.

11. Demonstrating quantitatively that variants of deep perceptual similarity that can handle certain flaws of the most common variant perform better on an actual perceptual similarity dataset.

Chapter 5 and in [33]:

12. Performing initial proof-of-concept analysis into the ability of deep perceptual similarity metric to be adapted to various contexts.

13. Showing that deep perceptual similarity is adaptable in a limited environment.

14. Proposing changes to current training regimes for perceptual similarity to improve performance, generalization, and adaptability further.

15. Proposing future directions of research and use cases for adaptable deep perceptual similarity metrics.

In [14, 30, 29, 32, 33]:

16. Making the technical implementations and experiments that have been conducted as part of the research for this thesis freely available as online repositories online[1][2][3].

1.6.3 Other Scientific Publications

Some research contributions have been made throughout the research, which do not fit in this thesis. Those contributions have been published in the works listed below.

[34] Kumar Shridhar, Ayushman Dash, Amit Sahu, Gustav Grund Pihlgren, Pedro Alonso, Vinaychandran Pondenkandath, György Kovács, Foteini Simistira, and Marcus Liwicki. Subword Semantic Hashing for Intent Classification on Small Datasets. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2020.
Personal Contribution: Assisting in initial experimentation, verifying results, collecting additional data, and assisting in writing.

[35] Johan Edstedt, Amanda Berg, Michael Felsberg, Johan Karlsson, Francisca Benavente, Anette Novak, and Gustav Grund Pihlgren. VidHarm: A Clip Based Dataset for Harmful Content Detection. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1543–1549, 2022.
Personal Contribution: Analysis and evaluation of the gender bias in the trained

[1] `https://github.com/guspih/Perceptual-Autoencoders`
[2] `https://github.com/guspih/deep_perceptual_similarity_analysis`
[3] `https://github.com/LTU-Machine-Learning/Analysis-of-Deep-Perceptual-Loss-Networks`

models. Writing about the same analysis and evaluation as well as assistance in general writing.

[36] Prakash Chandra Chhipa, Richa Upadhyay, Gustav Grund Pihlgren, Rajkumar Saini, Seiichi Uchida, and Marcus Liwicki. Magnification prior: A self-supervised method for learning representations on breast cancer histopathological images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2717–2727, January 2023.
Personal Contribution: Assisting in literature review and writing.

1.7 Thesis Outline

The remainder of the thesis is split into five chapters: *Background*, *Implementing Suitable Loss Networks*, *Investigating Deep Perceptual Similarity*, *Overcoming Ambiguity in Similarity*, and *Conclusion and Future Work*.

Chapter 2 provides the context for what the thesis builds upon. It details the development history of deep perceptual loss and similarity along with closely related technologies. It contains technical details of how deep perceptual loss and similarity are commonly implemented and lists applications for which the methods have been used.

Chapter 3, 4, and 5 address research questions 1, 2, and 3 respectively. Chapter 3 compiles studies on how various loss network implementations affect the performance of the method and makes suggestions on how to implement suitable loss networks for a given task. Chapter 4 inspects flaws in common implementations of deep perceptual similarity and provides options to mitigate those flaws. Chapter 5 questions the ability of deep perceptual similarity to handle the ambiguity of perceptual similarity and investigates how the method can be adapted to different definitions of similarity.

Chapter 6 wraps up the thesis. It summarizes the thesis and how it addresses the research questions. It covers the ethical considerations of the research. Finally, it highlights what remains to be explored and suggests promising opportunities for future work.

Chapter 2

Background

This thesis builds on existing work on deep perceptual loss and similarity, which is based on other work in machine learning and computer vision. There are also other related subfields from which inspiration is drawn and on which the methods are applied. Before delving into the novelties presented in this thesis, it is, therefore, useful to understand the underlying and related developments. This chapter provides a detailed introduction to the technologies and ideas on which the rest of the thesis builds.

This chapter introduces the concept of perceptual similarity and previous methods for computing the similarity of images. In addition, ways in which perceptual similarity metrics can be evaluated are described, along with the popular datasets for that evaluation. Then the machine learning and deep learning methods that are fundamental to this thesis are introduced, followed by how these methods developed toward deep perceptual loss. Then deep perceptual loss is introduced along with its many applications grouped into larger categories. Building on the methods for calculating deep perceptual loss, deep perceptual similarity is introduced. Finally, some methods that lie beyond the thesis that also use the terms deep perceptual loss and similarity are mentioned.

2.1 Non-deep Similarity Metrics and Loss Functions

As explained in Chapter 1, calculating the similarity of the output of a model to the desired output is the most popular form of loss calculation in machine learning. A similarity metric can be defined as a real-valued function $d(x_1, x_2)$ that computes a distance measure between two vectors x_1 and x_2. Some refer to this as a dissimilarity metric instead, but since measuring dissimilarity can be equated with measuring similarity, this thesis will refer to such functions as similarity metrics.

A naive approach to computing the similarity of two vectors is to calculate the distance between them in Euclidean space. A simple such distance measure is the ℓ_1 norm, also called Manhattan distance, which is used in the loss function for Mean Average Error (MAE). Another commonly used norm is the ℓ_2 norm, calculated as the shortest distance between the vectors in Euclidean space, and is used in the loss function for Mean Squared

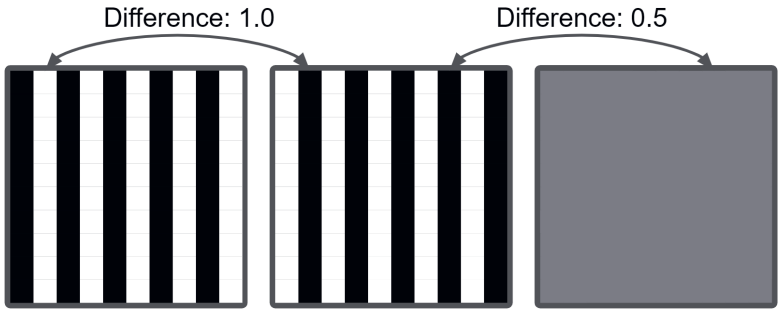


Figure 2.1: An image of black and white stripes, the same image shifted one pixel sideways, and a single-colored gray image with the same average pixel value as the other two. Arrows exemplify how pixel-wise metrics would compare corresponding pixels in the images and show the absolute differences between the compared pixels.

Error (MSE). Using simple distances works well for many tasks when the outputs of the model are independent of one another.

Another common machine learning scenario is classification, where the output of the model is a probability distribution over the potential classes. In cases when the output is a probability distribution, it is intuitive to use some measurement of the distance between distributions. A simple such metric is the Binary Cross-Entropy (BCE), which computes the entropy between the probability distributions. Again, this and similar simple metrics for probability distributions have worked very well on classification tasks.

All of these presented simple distance measures have one thing in common; they are element-wise metrics. This means that they are calculated by comparing the corresponding output values (elements) to each other without considering the other values. For images, these values are pixels, and the metrics are called pixel-wise metrics. Since only the corresponding values are compared, element-wise metrics cannot measure higher-level structures in the data. This is visualized in Fig. 2.1, where two black and white images with the same structure shifted by a single pixel would be considered completely dissimilar by pixel-wise metrics. Meanwhile, a single-colored gray image would be seen as more similar to either striped image despite sharing no structural similarity. Nonetheless, element-wise metrics are useful in many tasks where higher-level structure in the output is non-existent or negligible. This is not the case for images, where the correlations between the individual pixels are often more important than how close the individual pixels are to the desired values.

So what does similarity actually mean? As has been well-established by now, similarity is ambiguous and varies between contexts. The same proofs that are used in the "no free lunch"-theorems to show that all prediction models are equal when averaged over all problems [37] can be used to show that all similarity metrics are equal when averaged over all similarity contexts. The "no free lunch"-theorems are discussed further in Chapter 5. A better-formulated problem would be finding the best metric for some set of tasks. For classification, for example, a loss function is good for a given problem if the

models trained with it perform well. Intuition, theory, mathematics, and experimentation can then be turned to find metrics that are good for relevant tasks in some domain. For this thesis, that domain is images, and the relevant tasks are perceptual similarity.

2.1.1 What is Perceptual Similarity?

What is a good similarity metric for images? For a given task, a good metric is one that provides high performance on that task. Creating a specific metric for every single task is unfeasible, and it seems like a generally good metric should exist for most relevant image similarity tasks. Humans seem to have some shared tendencies to decide which images are more similar, at least when given commonly used images (*i.e.*, images that humans do not consider random noise). A similarity metric that mimics human perception of similarity would therefore seem possible. It would also be useful as there are many different tasks, such as image synthesis and image quality assessment, where the goal is explicitly to optimize for human perception. Metrics that strive to mimic human perception of similarity are called perceptual similarity metrics.

While there is no formalized definition of a good perceptual similarity metric, most perceptual similarity datasets score metrics according to how well they adhere to the average opinion among their human judges [20, 21, 22, 2]. An optimal perceptual metric d^* could therefore be formalized as a similarity metric such that, given a population of humans, it agrees with their average perception of what images are more similar. While there are variations in perception between populations, when averaged over all populations, some metrics perform better than others.

The similarity scores created by metrics do not necessarily have to adhere to a given scale in order to be useful. In many cases, it is enough for the similarity metric to be able to tell if one pair of images are more similar than another. Under those circumstances, the exact scores and their relative values do not matter other than to determine which pairs are more similar. Two metrics, d_1 and d_2, would be equally proficient at measuring perceptual similarity if, for any two image-pairs (x_i, x_j) and (x_a, x_b), it holds that $d_1(x_i, x_j) \leq d_1(x_a, x_b) \iff d_2(x_i, x_j) \leq d_2(x_a, x_b)$. This means that all the metrics presented in Fig. 2.2 are equivalent for evaluating perceptual similarity. If only the ranking of pairs matters, then any metric d which produces the same ranking as d^* would also be considered an optimal metric. Section 2.2 further discusses whether this is a desirable property for perceptual similarity metrics.

2.1.2 Non-deep Metrics of Perceptual Similarity

Pixel-wise metrics fail to capture a human perspective of image similarity [13]. Machine learning for computer vision has mostly kept using pixel-wise loss, despite its known flaws. The field of similarity metrics, however, has developed several types of metrics that outperform pixel-wise metrics for perceptual similarity.

Early works on perceptual similarity metrics were based on the pixel-wise MSE metric [9]. Newer perceptual similarity metrics, on the other hand, tend to use one or more of the following strategies: Subdividing the image into regions that can be compared

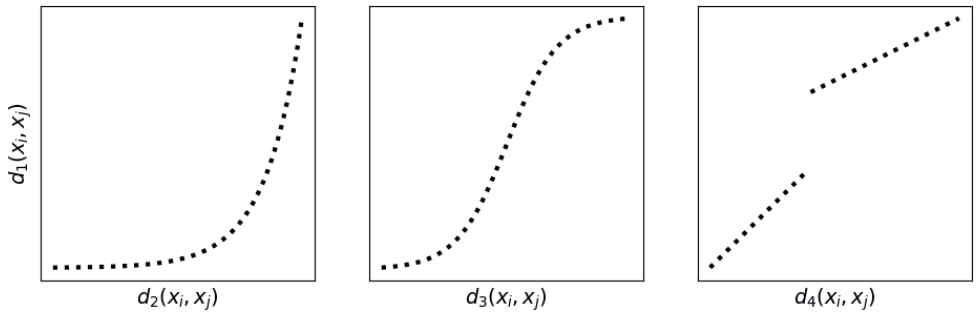


Figure 2.2: Four perceptual similarity metrics that can be considered to have equivalent performance since they uphold the property $d_n(x_i, x_j) \leq d_n(x_a, x_b) \iff d_m(x_i, x_j) \leq d_m(x_a, x_b)$. *I.e.*, for any two image pairs, the metrics agree on which is more similar.

using simple properties such as the mean and variance in color, contrast, and brightness. Using measures of entropy of some image properties in order to quantify the information difference between images. Extracting features of images according to theories of human perception and comparing those features.

One popular image property to use for similarity metrics is the color histograms, which quantify how prominent different colors are in the image. The histogram comparisons were used in IBM's QBIC system for image and video querying [38] and improved by including histograms for other image properties in the joint histogram metric [39]. The joint histogram method has been improved with the use of Taneja entropy [40]. Using histograms on their own has been criticized for lacking spatial knowledge of the image [41].

A method that strives to capture both color and spatial aspects is Color-Spatial Information (CSI) [41]. CSI uses histograms of different regions to try and separate the colors of the background and the subject of the image. The method also uses a measure of entropy to try and divide the image into regions of the identified colors.

A notion of image structure is added in the SSIM metric, which compares the average luminance and contrast of the images, as well as the correlations of the images [9]. To get a stronger notion of space, the metric can be used to compare different regions of the images and then use the average similarity of those regions. SSIM has been a popular perceptual similarity metric and spawned many derivatives. Examples are the multi-scale SSIM which improves the metric by applying it on multiple resolutions [42], and HSSIM, which combines SSIM with the joint histogram method [43].

Another set of metrics is built on the notion of creating rule-based algorithms for extracting image features that can be compared. The Feature Similarity (FSIM) metric uses features justified by physiology and how the human visual system works [44]. FSIM is combined with SSIM in the Feature-Based Structural Measure (FSM) which achieve higher performance on facial recognition. In addition, there are several rule-based feature extraction methods not specifically created for similarity that can be used for this

purpose, including Scale Invariant Feature Transform (SIFT) [45], Speeded Up Robust Features (SURF) [46], and Binary Robus Independent Elementary Features (BRIEF) [47].

Some of these perceptual similarity metrics have also been used to calculate the loss in machine learning applications called perceptual loss. For example, the SSIM metric has been used multiple times for this purpose [24, 25]. Other works have introduced novel perceptual loss functions that can also be used as perceptual similarity metrics [48].

2.2 How Perceptual Similarity is Evaluated

Perceptual similarity can be evaluated on downstream tasks that make use of similarity. Content-based image retrieval is a popular application of similarity metrics that have been used for such evaluation [7]. As will be detailed further, content-based image retrieval is not always a good task for measuring perceptual similarity. Downstream tasks in general measure efficacy on that task, which may not be related to perceptual similarity.

Image datasets with defined similarities can be used to directly evaluate perceptual similarity. Such datasets can be automatically created by distorting images where the degree of distortion is known. However, the ideal way to measure perceptual similarity is to directly use measurements of human perception of similarity. Perceptions are typically gathered by asking human subjects to judge the similarity of different images, which can be done through many different procedures.

2.2.1 Content-Based Image Retrieval

Content-based image retrieval is the task of locating relevant images in a database by using an input query image that is related to those images. Some typical content-based image retrieval datasets define images as similar by containing images of the same object [49], containing objects of the same class [50], containing regions with the same content [51], and having similar styles [52]. As similarity metrics are often used in content-based image retrieval, such tasks can be used as evaluation methods [7]. Image retrieval is a downstream task which makes it similar to evaluating similarity metrics as loss functions, though image retrieval does not require a model to be trained, and the performance is directly linked to the metric. Though image retrieval is not a perfect analogy for similarity. For example, in a database that classifies animals by species, there could exist a species where a human would think the individuals are very dissimilar (*e.g.*, dogs), while there could also be some species that are almost indistinguishable for anyone but an expert (*e.g.*, various species of ants).

2.2.2 Image Distortions

Image distortions can be used to evaluate similarity metrics without requiring expensive labeling. A set of image distortions can be applied with varying severity to images. Metrics can then be evaluated by whether they agree that less distorted versions are more similar to the originals. They can also be evaluated on whether they agree that distorted

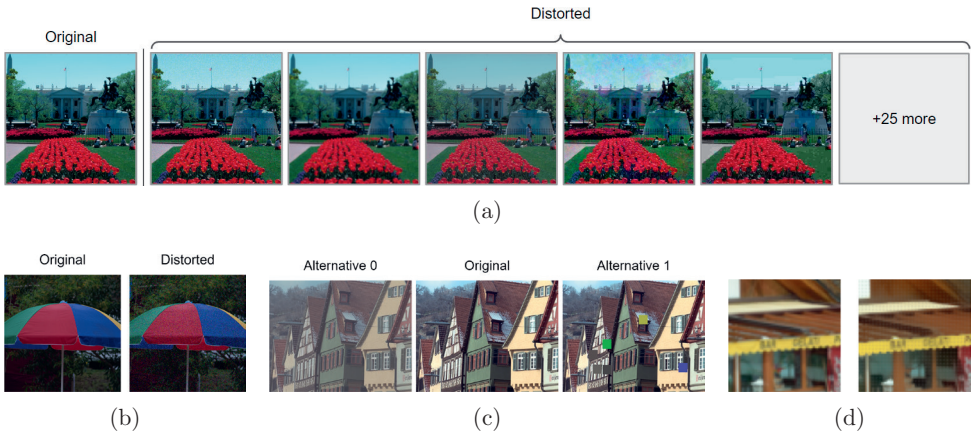


Figure 2.3: Samples of data from the (a) NITS-IQA [56], (b) CSIQ [21], (c) TID2013 [22], and (d) BAPPS [2] datasets similar to how they were presented for labeling. In (a), the human judges were asked to rank the distorted image by their similarity to the original. In (b), they were asked to grade the distorted version by how similar it is to the original on a scale of 1 to 100. In (c), they were asked which alternative they considered more similar to the original. In (d), they were asked if they perceived the image pair as the same after being shown them briefly.

versions of an original image are more similar to that image than completely different images. While this approach is intuitive and easy to apply, it has some significant flaws. First, it cannot be used to evaluate metrics between different types of distortions. Second, it can only use distortion types where perceptual similarity is known to decrease with the severity of distortion. Third, it cannot be used with distortion types where severity is not defined. Despite these flaws, this type of evaluation is often used to highlight problems in metrics and investigate potential improvements [53, 9, 54, 4, 55, 29]. Common distortions to use in these cases include image compression, affine transformation, noise, and blur.

2.2.3 Human Judgements

In Subsection 2.1.1, perceptual similarity is defined as human perception of similarity. Therefore, human judgments are the true metric of perceptual similarity. Several datasets exist that consist of judgments from human subjects on the similarity of images. These datasets can be divided into three categories based on how the judgments were collected; Mean Opinion Score (MOS), Two Alternative Forced Choice (2AFC), and Just Noticable Differences (JND). These categories are further elaborated below, and examples of how judgments can be gathered for each category can be found in Fig. 2.3.

Judgments in MOS dataset are gathered by showing pairs of images to human observers who then ranked them according to a predefined scale such as 1-100 or ”Bad”, ”Poor”, ”Fair”, ”Good”, and ”Excellent [57]. These datasets typically consist of a few

images that are distorted, image pairs are formed from the original and each distorted versions, and many observers give judgments for each pair [2]. Other versions ask participants to rank a number of distorted images by how similar they are to the original image [21]. Metrics are then evaluated based on how well they adhere to the average human scores, called the MOS. This adherence is typically measured by fitting a logistic function to the relationship between the similarity metric and the human judgments. The performance of the metric is then given by the linear correlation coefficient, Spearman rank-order correlation coefficient, the MAE, and the MSE between the logistic predictions and the opinion scores. Other measurements are also often used. Sometimes datasets use other methods, like 2AFC [22], to gather judgments but format those judgments as MOS labels.

2AFC judgments are gathered by presenting a human observer with one image and two distorted versions of that image and asking the observer to select which distorted version is more similar to the original. The image and its distortions are then labeled by what fraction of observers preferred each distortion. Metrics are evaluated on the dataset either in the accuracy of when it agrees with the average population or with a 2AFC score given by averaging what fraction of observers agreed with each of its perception of similarity for each sample. The calculation of the 2AFC score is given in Eq. 2.1, where x is the original image, x_0 and x_1 are the distorted alternatives, J is the fraction of human judges that think x_1 is the most similar alternative, and d is the metric being evaluated. Advantages of 2AFC over MOS include avoiding ambiguity of the average score and drift of opinion over the course of the experiment. A disadvantage is that the relative similarity of pairs outside the triplets is unknown.

$$\text{2AFC}(x, x_0, x_1, J) = \begin{cases} J, & \text{if } d(x, x_1) < d(x, x_0) \\ 1 - J, & \text{otherwise} \end{cases} \tag{2.1}$$

JND datasets consist of pairs containing an image and a barely distorted version of that image. Judgments are gathered by briefly showing the pairs to observers and asking them if they think there is any difference between the two. In order to keep the observers from always giving the same answer, they are also shown pairs of the same images and completely different images as sentinel tasks. The pairs are then labeled by what fraction of observers thought they looked exactly the same. Pairs are then considered more similar if more observers thought they were the same image. Both MOS and 2AFC datasets contain a large degree of subjectivity, where each observer's personal opinion on similarity is reflected in their choices, which may vary over time for a given observer. JND datasets reduce the subjectivity since observers only report what they notice instead of what they think. The primary drawback of JND is that the distorted images have to be very similar to the originals to actually be able to be unnoticed by some fraction of observers. This means that JND data can only cover a small range of human perception. The performance of a metric on a JND dataset can be calculated by how close the metric's ranking of which pairs are more similar aligns with the ranking of the pairs according to the fraction of humans that thought they were the same. The ideal perceptual similarity metric would give a higher similarity to pairs that

(a) 2AFC (b) JND

Figure 2.4: Two samples each from the (a) 2AFC and (b) JND parts of the BAPPS dataset. The 2AFC triplets are labeled with the fraction of human judges that thought alternative 1 is more similar to the original than alternative 0. The JND pairs are labeled by the fraction of human judges that mistook the pair as being the same images when shown briefly.

fooled human judges more often than those that humans could tell apart. The closeness of the two rankings could be calculated with any ranking score, such as the Spearman rank correlation coefficient. In this thesis and related works, a measure called the JND score is used, which is the mean average precision measurement as used in information retrieval to measure the alignment of ranked sequences [58].

2.2.4 Perceptual Similarity Datasets

In this thesis, perceptual similarity datasets refer to datasets that label image similarity with human judgments. Among such datasets, MOS labels that are collected from images with distortions are the most common. The typical distortions consist of compression algorithms, additions of noise or blur, and changes in color, saturation, or contrast. Two popular datasets in this category are LIVE Image Quality Assessment Database (LIVE) [20] and Categorical Image Quality Database (CSIQ) [21]. The recent NITS-IQA [56] dataset is another example of a MOS dataset. Another popular dataset is Tampere Image Database 2013 (TID2013) [22], which refers to the labels as MOS, but they are actually gathered through a 2AFC like procedure. These datasets are also often called IQA datasets since human judgments are often gathered to judge the quality of image compression and transmission.

One perceptual similarity dataset that is particularly important to this thesis and the research on deep perceptual similarity is the Berkeley-Adobe Perceptual Patch Similarity (BAPPS) dataset [2]. The BAPPS dataset is used for evaluation in Chapter 3, 4, and 5, as well as many works on deep perceptual similarity. Samples from the dataset can be found in Fig. 2.4, which is described in detail below.

BAPPS consist of 64×64 image patches obtained from the MIT-Adobe 5k [59], RAISE1k [60], DIV2K [61], Davis Middleburry [62], video deblurring [63], and ImageNet [64] datasets. The dataset also contains distorted versions of those patches obtained by applying several different distortions to them split into six categories. The six categories are (i) Traditional augmentation methods, outputs from (ii) CNN-based autoencoders, (iii) super-resolution, (iv) frame interpolation, (v) video deblurring, and (vi) colorization. The two first categories are created by randomly applying distortions and creating intentionally suboptimal denoising autoencoders. The last four categories are created by applying actual algorithms created to solve the given tasks. For this reason, the first two categories are referred to as the distortion categories, and the last four as the real algorithm categories.

The BAPPS dataset is split into a 2AFC part consisting of triplets with the original patch and two distorted versions and a JND part consisting of the patch and a distorted version. The 2AFC part of the dataset is further split into a training set and a test set. The training set consists of distortions in the distortion categories and some patches with a mix of distortions from the two categories. The test set consists of distortions from all categories. The JND part of the dataset is only for testing and consists of distortions from the distortion categories. The JND part can be used to verify the results of 2AFC part by checking whether the models that perform well on the 2AFC part generalize to another type of perceptual similarity measurement. The dataset contains human judgments for each part according to the earlier described practices for gathering judgments on 2AFC and JND data. While other perceptual similarity datasets have focused on gathering a large number of judgments for a few images, BAPPS has many images with only a few judgments each. The distribution of distortions and judgments across the different parts of the dataset is shown in Table 2.1. Example images from the two parts of the datasets can be found in Fig. 2.4.

2.3 Feature Extraction and Feature Learning

Many tasks in computer vision and other domains rely on extracting useful features from raw data. Such features can be used to create decision-making systems or programs that would be difficult or impossible to use with the raw data. There are many different methods to do this, which are typically grouped into either feature engineering or feature learning.

Feature engineering relies on rule-based methods to extract specific features from data. This can be done using a generic algorithm for the domain, a special algorithm handcrafted by experts for the specific problem, or both. Generic algorithms are typically designed to extract features that have been determined generally important for a given domain. Generic algorithms range from completely domain-independent, such as Principal Component Analysis (PCA) and Independent Component Analysis (ICA), to domain-specific, such as SIFT [45] and SURF [46]. Handcrafted algorithms rely on experts on the domain and specific tasks to decide what features are useful and are then designed to extract those specific features.

Table 2.1: Breakdown of the BAPPS dataset into its constituent parts, the number of samples where each distortion type has been used, and the number of human judgments gathered for each sample.

Split	Distortion	Samples	Judgements/Sample
2AFC Train	Traditional	56.6k	2
	CNN-based	38.1k	2
	Mixed	56.6k	2
	Total	**151.4k**	2
2AFC Test	Traditional	4.7k	5
	CNN-based	4.7k	5
	Superres	10.9k	5
	Frame Interp	1.9k	5
	Video Deblur	9.4k	5
	Colorization	4.7k	5
	Total	**36.3k**	5
JND Test	Traditional	4.8k	3
	CNN-based	4.8k	3
	Total	**9.6k**	3

While feature engineering has been a staple of many early machine learning systems, the approach has some notable disadvantages. Generic algorithms rarely produce good enough features for high-performance downstream applications. Handcrafted algorithms require a domain expert to decide which features to extract and someone to design the algorithm to do this, both of which are resource-intensive tasks. In both cases, good features may be infeasible to extract since the experts might not know how to extract certain features or even think of extracting them in the first place [1]. This has created ample ground for using machine learning to train models explicitly to perform feature extraction.

In recent years, feature learning has become the most popular form of machine learning. Instead of having human experts decide which features to use and how to extract them, machine learning models are used for this purpose. This is often done by stacking models where earlier models learn features useful for the later models, which combine those features into more complex ones, which leads to a hierarchy of increasingly complex features [1]. This stacking of models is called deep learning, and the stack of models is often seen as a single model, with the models composing it referred to as layers.

Deep learning alleviates the problems with feature engineering. The features learned can be specifically adapted to the task at hand by training the final model to solve the final task, ideally forcing the features to be useful for that task. Domain experts are not needed to the same extent as the models themselves to learn what features to use. The

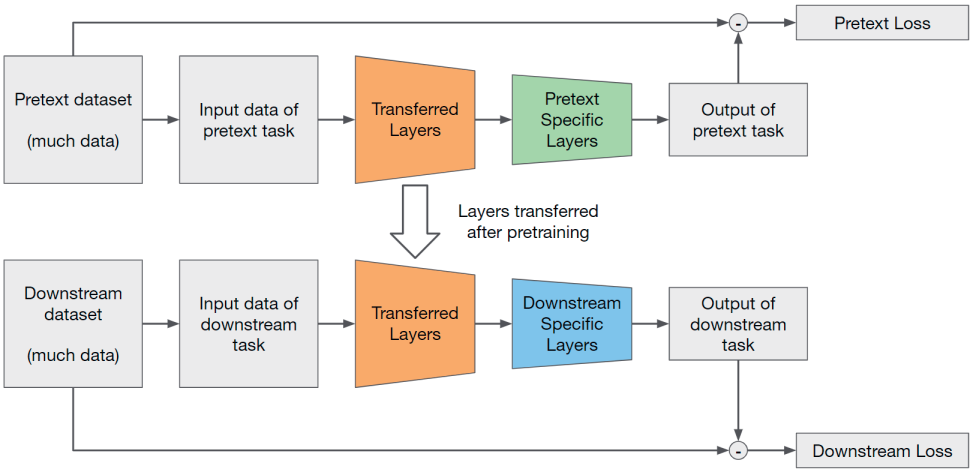


Figure 2.5: An example of a transfer learning setup. The layers from one model are trained on a pretext task and are then used as part of a model being trained for the downstream task.

features that are learned can also include ones that the domain experts did not think about, which sometimes leads to new discoveries in the field.

2.3.1 Transfer Learning

There are downsides to deep learning. Many tasks require learning complex features to solve, which means the deep models need to be large in order to be able to extract such features. However, training large models takes a large amount of data and resources. Some tasks simply do not have enough useful data to train deep models. Though in many of these cases, deep models from similar tasks with a lot of data can be reused and adapted to the specific problem. Since the model has already learned to extract many useful deep features from the previous task, not as much data is needed to adapt it to the new task. This practice is known as transfer learning, as the parameters learned on one task are transferred to and used on another task. Transfer learning works when the deep features learned on one task are also applicable to the other task, which means that transfer learning work best in similar domains. The first task in these scenarios is known as the pretext task, and the second task is known as the downstream task. Training on the pretext task is also known as pretraining (*i.e.*, training prior to training on the downstream task), and the models that have been trained on pretext tasks are referred to as pretrained. An example of a typical transfer learning procedure is visualized in Fig. 2.5.

The effectiveness of training large models and transfer learning has led to the creation of so-called out-of-the-box models. Since training large models on large datasets has

shown to get good performance on the large dataset as well as other related tasks and datasets through transfer learning [65], there is a need to produce and train such large models. This can be prohibitively resource intensive. Even in cases where the resources to do so are available, it could be considered a waste since many other large models have been trained on the dataset before. This issue has been solved through the public sharing of pretrained models. Such models are typically referred to as out-of-the-box models since they can be applied immediately without the need for additional pretraining. Though, additional training and fine-tuning are often performed before applying the model to the downstream task [65].

In computer vision, the de facto default dataset for pretraining large models is ImageNet [64]. ImageNet is a classification dataset consisting of 14197122 images split across 21841 categories. Models are commonly not trained on the entire dataset and instead use a subset of ImageNet used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition. This subset consists of 1281167 images with 1000 labels, which is still a significant amount. The efficiency of new methods and architectures in computer vision are often assessed by their accuracy on ImageNet, and models that achieve high accuracy are often used for transfer learning. A trend in computer vision transfer learning has emerged where models that achieve better pretraining accuracy on ImageNet perform better on downstream tasks after fine-tuning [66, 67].

2.3.2 Convolutional Neural Networks

The most commonly used deep learning architectures in computer vision are CNNs. A CNN are DNN with at least some convolutional layers. Convolutional layers take advantage of spatial dependencies in the data by using kernels that only take inputs from a small region of the input space. Each kernel is then applied like a sliding window over regions covering the entire input space, which computes a feature for each kernel and spatial region as shown in Fig. 2.6. The output space of each kernel is a channel of the next layer, which is called a feature map and has the same spatial structure as the input space since each region is represented. As multiple convolutional layers are applied, deeper features represent larger regions of the input space since they aggregate information that comes from multiple regions. There are multiple benefits to having kernels that do the same computation for many regions of the input space. Firstly, it can capture the natural dependencies in data like images where pixels next to each other likely have a stronger correlation and combine to form local structures that each kernel can learn. These local structures are then present in the feature map of the next layer, and the following convolutions can combine local structures over a larger area. For example, a first-layer kernel might detect sharp edges between dark and light, a second-layer kernel might combine different detected edges into corners or curves, and a third-layer kernel might combine curves into basic shapes. Secondly, it lowers the number of parameters needed as the input space grows because each kernel has the same parameters regardless of input size. Thirdly, there is no maximum size to which a convolutional layer can be applied, and increasing the input size does not require retraining of the kernels. This

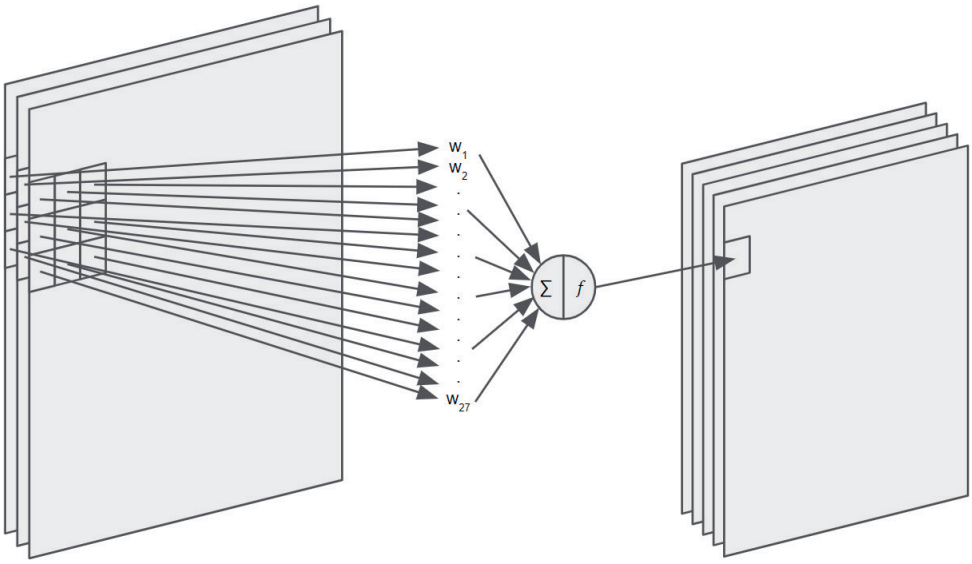


Figure 2.6: The calculation of a single feature of a convolutional layer with 5 kernels and an input layer with 3 channels. The kernel with weights w_1 to w_{27} has size 3×3 and is applied to multiple regions in the input layer to compute the 5^{th} channel of the output layer.

makes CNNs ideal for transfer learning since the pretrained convolutional layers (which typically make up the vast majority of the network) can be applied to any image that is not too small.

CNNs are also motivated for image applications by their similarity to the visual cortex in mammals. One of the primary works that inspired the creation of the CNN architecture was explicitly inspired by the visual cortex [68]. The hierarchical structure of CNNs applied to images mimics that of the visual cortex [69]. The similarities extend beyond structural as CNNs trained on images tend to learn kernels that activate strongly to similar patterns that are detected in the early layers of the visual cortex [70, 71].

Many CNNs that have been pretrained on ImageNet are available as out-of-the-box models in different online repositories. Some CNN architectures have become popular to use for transfer learning on computer vision tasks. This thesis and the works it is based on use many different out-of-the-box models pretrained on ImageNet. The out-of-the-box models used in this thesis are listed in Table 2.2 together with some of their attributes. These models are taken from the Torchvision [72] framework version 0.11.3.

Table 2.2: Attributes of the out-of-the-box Torchvision [72] models used in this thesis. Shown are their accuracy on ImageNet, depth, MFLOPS for the forward pass (for a 224×224 pixel image), and whether the architecture has skip-connection, branches, 1×1 convolutions, or batch-normalization.

Architecture	ImageNet Acc. (%)	Depth	MFLOPS	Skip-conn.	Branch	1×1 conv.	Batch norm
VGG Networks							
VGG-11 [73]	69.020	11	7637				
VGG-16 [73]	71.592	16	15517				
VGG-16_bn [73]	73.360	16	15544				✓
VGG-19 [73]	72.376	19	19682				
Residual Networks							
ResNet-18 [74]	69.758	18	1827	✓			✓
ResNet-50 [74]	76.130	50	4143	✓		✓	✓
ResNeXt-50 32x4d [75]	77.618	50	4298	✓	✓	✓	✓
Inception Networks							
GoogLeNet [76]	69.778	22	1516		✓	✓	✓
InceptionNet v3 [77]	77.294	49	2850		✓	✓	✓
EfficientNet							
EfficientNet_B0 [78]	77.692	81	407	✓			✓
EfficientNet_B7 [78]	78.642	271	5308	✓			✓
Uncategorized Networks							
AlexNet [79]	56.522	8	717				
DenseNet-121 [80]	74.434	121	2899	✓		✓	✓
SqueezeNet 1.1 [81]	58.178	18	360		✓	✓	

2.3.3 Self-supervised Learning

While pretraining on ImageNet can often provide increased performance on transfer learning, it is not a universal approach. Some image domains are significantly different from the photographs of natural scenes that make up ImageNet and thus do not benefit as much from the features learned on it. Even if ImageNet pretraining can provide some benefit, other methods can be added to reach state-of-the-art performance [82]. Many of these domains also have limited amounts of labeled data, making supervised pretraining of large models difficult. In these cases, other unsupervised or self-supervised pretraining methods can be applied. One popular such pretraining method is autoencoding, where a model is trained to reconstruct the input [83, 84]. The middle layers of the autoencoder typically have lower dimensionality than the input and output. In those cases, the autoencoder has to learn to reduce the dimensionality of the data by representing it as features that describe the input well enough to reconstruct it.

Other pretraining approaches that have been growing in popularity come from the field of contrastive learning. Contrastive learning covers a set of self-supervised and semi-supervised training procedures that focus on learning deep features through training a

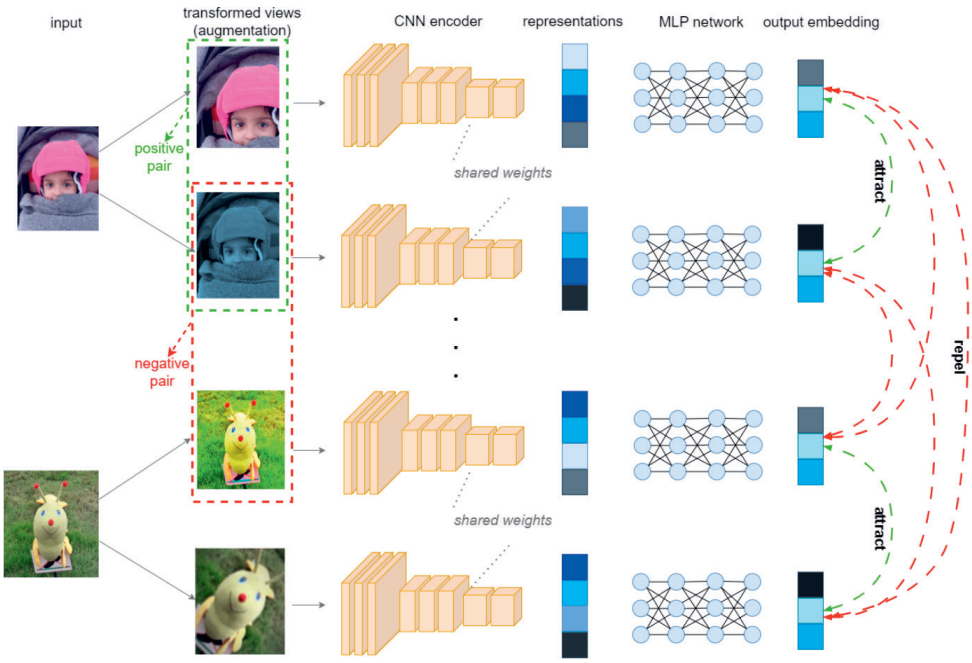


Figure 2.7: An example of a contrastive learning scenario by the work of *Prakash Chandra Chhipa* [82] from which the figure originates. The figure shows how contrastive learning is used to train a model for image embedding. Two images are distorted by different distortions, and each distorted version is then used as input to the same CNN, which produces an internal representation (deep features) and an embedding. The CNN is trained to have similar embeddings for the input from the same image and dissimilar embeddings for input from different images. Reproduced with permission.

network to output similar embeddings for some inputs and dissimilar embeddings for others. For example, an image of a human with and without the color still represents the same person, and the network is trained to produce similar embeddings for the two. The same network is also trained to produce dissimilar embeddings for the image of the human and another image entirely. This can be done without the need for labeling by training the network to output similar embeddings for distorted versions of the same image while it should output dissimilar embeddings between different images. This has shown to be effective in computer vision and particularly in improving the performance of ImageNet transfer learning in domains that are different from that of ImageNet [82]. An example of contrastive learning is provided in Fig. 2.7.

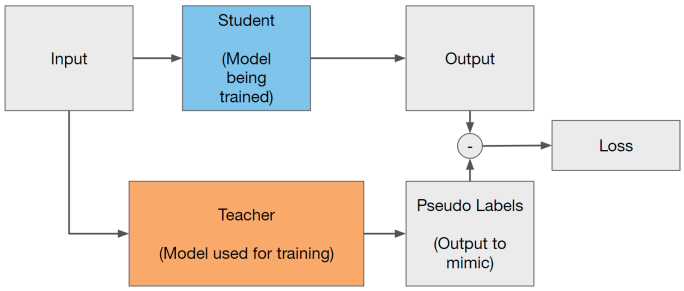


Figure 2.8: An example of a knowledge distillation setup where the outputs of a larger pretrained model are used as the desired outputs for training a smaller model.

2.4 Using Models to Train Other Models

There are many ways beyond transfer learning to reuse trained models and get around the lack of data. One approach that has at times been used to do both is to use one machine learning model as part of the training of another. This practice has been around for a long time. For example, in 1992, it was used in predictability minimization where a single-layer neural network is trained to perform encoding through a loss generated by another single-layer neural network which used that encoding as input [26]. This is an example of actor-critic training; one model is trained to perform a task using loss from another model trained to evaluate that task.

Actor-critic training has been used to get around problems where data is lacking or when it is not easy to come up with the desired output of a model. A famous example of adversarial learning for images is Generative Adversarial Networks (GANs) [27]. GANs consist of two neural networks, a generator that outputs images that should be similar to some given dataset and a discriminator that should determine if a given image was generated or taken from the dataset. The generator takes random input noise and outputs an image, and the discriminator is given that image as input and predicts whether it was generated or not. The generator is trained to fool the discriminator, and the discriminator is trained to classify whether an image is generated or comes from an existing dataset. This powerful combination has been successfully used for many image synthesis and transformation applications [85]. However, there are many potential problems with training GANs and other actor-critic systems, which can cause training to collapse [85]. For example, if the actor or critic becomes much better than the other, they can both become unable to learn further.

Some methods for using one model to train another do not train the models simultaneously. Instead, a pretrained model can be used to train another model in various ways. One example of this is knowledge distillation, through which a smaller "student" model can be trained to achieve similar performance to a larger pretrained "teacher" model by being trained to predict the output patterns of that model [28]. An example of this setup

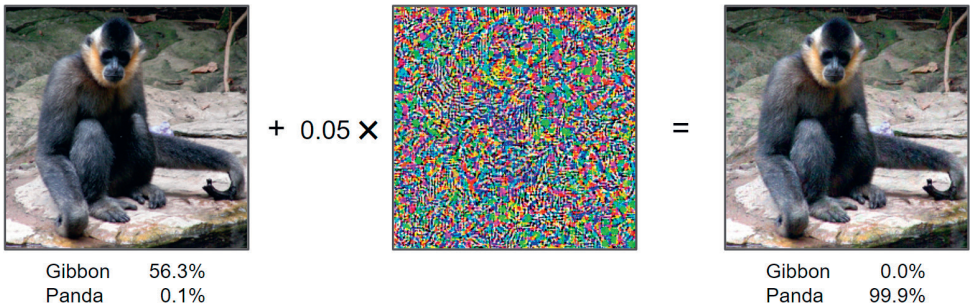


Figure 2.9: An ImageNet [64] photograph of a Gibbon is correctly classified by an out-of-the-box AlexNet [79] model. An adversarial example is created through the addition of a specific and barely visible noise pattern. The adversarial example is, with great confidence, incorrectly classified by the same model despite being very similar to the original.

is visualized in Fig. 2.8. In another version of knowledge distillation, the model can also be trained to predict the deep features of a teacher network [86].

Using a pretrained model for calculating the loss of another has also been useful in the field of explainable AI, which focuses on methods for understanding how different AI models work. A way to visualize what a classification model looks for in a given class is to optimize the output of another model to give the strongest possible prediction for a given class [87]. This same method can also be used to visualize what individual features have learned by instead optimizing for the strongest activation of that feature [88].

Optimizing the input to a model has also been used to locate adversarial examples [89]. An adversarial example is an input to a model that is almost indistinguishable from another input but the model predicts the latter correctly and the former incorrectly. In terms of images, this could take the shape of optimizing for the exact amount of small noise to add to an image for it to be classified incorrectly, as shown in Fig. 2.9. Such adversarial examples can be found for most neural network models [90]. While strategies for defending against adversarial attacks, as they are called, have been developed, new attacks have been proposed to circumvent those defenses, leading to a sort of arms race of defenses and attacks [90]. Currently, it is speculated that the existence of adversarial examples is an inescapable property of DNNs [91], which poses challenges for the field moving forward.

2.5 Development of Deep Perceptual Loss

Developments in perceptual similarity strive to develop new metrics that capture the complex higher-level structure of images. DNNs has been proven effective for learning such complex higher-level structures. Additionally, using models to train other models

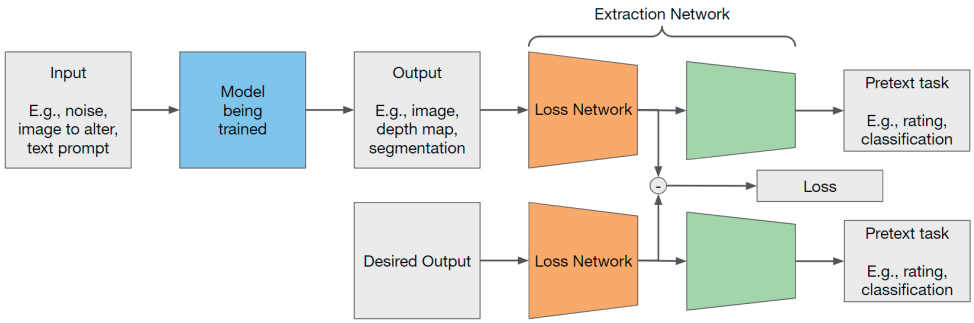


Figure 2.10: An example of training a model with deep perceptual loss by comparing the features extracted by another model from its output with those extracted from the desired output.

has been a popular approach in machine learning. It is then not so surprising that these were eventually combined with the creation of deep perceptual loss.

Deep perceptual loss in this thesis refers to a loss for training a model that strives to mimic human perception by using the deep features of a DNN, called the loss network, as described below. The output image of the model to be trained is used as input to a loss network from which deep features are extracted. Deep features are also extracted from that loss network when it is given the desired output images as input. The loss is generated by comparing the deep features extracted from the model's output and the desired output. An example of this is shown in Fig. 2.10. Beyond this work, deep perceptual loss and similarity are also used to refer to uses outside computer vision. Section 2.8 briefly covers use beyond computer vision.

The most common type of loss network used in deep perceptual loss are CNNs. Likely since CNNs have been among the most popular and successful models for computer vision since the introduction of deep perceptual loss [92]. The most common way to compare the extracted deep features is to use a simple element-wise measurement of the distance between the features, often MSE. This way of comparison is here referred to as the spatial comparison method since the features that are compared against each other correspond to the same spatial region of the two images. This is because the deep features extracted by CNNs correspond to different regions of the image, with earlier layers representing smaller regions and later layers aggregating those smaller regions into large ones. A more detailed description of different comparison methods can be found in Chapter 4.

Deep perceptual loss was first used for neural style transfer [93], which is the process of generating an image with the content of one image and the style of another. In style transfer, deep perceptual loss is used as a stand-in for human judgment of the similarity of an image's style and content to other images. In the first version of neural style transfer, deep perceptual loss with spatial comparisons is used to calculate how similar the generated image is to the desired content of an image. The style loss in this version use is calculated by converting the extracted features to Gram matrices before the difference

is calculated in an element-wise fashion. The loss network used to demonstrate this version was a VGG-19 [73] which had been pretrained on the ImageNet dataset [64].

Deep perceptual loss has also been used for image generation in a GAN scenario. By combining the capability to generate images with specific features from the Variational Autoencoder (VAE) and the ability of the GAN to generate images from a distribution, the VAE-GAN was created [13]. The standard VAE-GAN uses both the typical element-wise reconstruction loss of an autoencoder and the discrimination loss of a GAN but also uses the discriminator of the GAN as a loss network for deep perceptual loss. However, some of the issues with training a GAN also apply to the VAE-GAN. A later work goes around this by removing the discrimination loss and replacing the discriminator network with a pretrained CNN [94]. The use of a pretrained loss network to train a model is the most prolific approach to deep perceptual loss and has been applied to many different problems. In fact, some applications of deep perceptual loss have used a pretrained loss network while simultaneously using a separate discriminator network [95].

The most commonly used loss networks in deep perceptual loss applications are VGG architectures [73] pretrained on ImageNet [64]. Applications also commonly use deep perceptual loss as one of multiple losses used to train models. For example, adversarial losses are commonly included for image synthesis applications [13, 95, 96].

2.6 Applications of Deep Perceptual Loss

The application of deep perceptual loss has grown significantly since its inception. It has now been used in research from theoretical explorations of its capabilities to applications on real-world problems. The potential applications are somewhat limited due to deep perceptual loss being a method for comparing images, meaning the trained model needs to output images or data with the same structure as images. For this reason, the applications are divided into three categories: When the goal is to generate or alter images, when the goal is to generate image-like outputs, or when the image or image-like output is a pretext for training something else. The three categories are detailed in the subsections below.

2.6.1 Image Synthesis

The original application of deep perceptual loss was to create images, which remains popular. The earlier mentioned VAE-GAN can be used to generate images from a learned feature space, where features of different images can be combined or specific features such as “glasses” or “black hair” can be removed or added to generate images described by the altered features [13]. Style transfer has also been mentioned earlier, where an image is generated to combine the content of one image with the style of another [93]. In the former, the image generation is trained by deep perceptual loss but does not use it during inference. In the latter, no training is needed, and deep perceptual loss is used directly for inference. Deep perceptual loss has proven effective enough for pure image generation purposes that it is being used in some of the recently successful diffusion models for high-resolution image generation [97].

Another image synthesis task similar to style transfer is image fusion. Image fusion aims to use the information from multiple images to generate one improved image, typically with better quality. This is, for example, used in cameras to combine many images with low exposure time to estimate one with a long exposure time. Here deep perceptual loss has been applied as well [98].

Probably the most popular application of deep perceptual loss is super-resolution, a task where the goal is to create a higher-resolution version of a given image. Deep perceptual loss is used frequently as part of the training of super-resolution model, for which it is considered to belong to the state-of-the-art [15, 95, 96]. Most super-resolution applications of deep perceptual loss use spatial comparisons and a pretrained loss network. In at least one case, however, the style loss originally from style transfer was used alongside spatial deep perceptual loss [99]. Super-resolution with deep perceptual loss has also been applied in the medical imaging domain, where it has been used to enhance images created by x-rays in hopes of reducing the radiation needed to get the necessary images [100]. There are likely many more interesting uses for super-resolution for which deep perceptual loss has not been tested.

2.6.2 Image-like Outputs

Since deep perceptual loss is founded on approximating human perception of images, it seems like it has to be applied to images. However, many computer vision tasks require outputs that are not strictly images but share structural commonalities with images. Practically, for deep perceptual loss to be applicable, the output of a model need to have the same form as the input of the loss network, or at least it needs to be possible to rearrange it to that form. This typically means that the output needs to be in a 2d-grid as that is the input used for CNNs. Intuitively, the output also needs to adhere to the same perception of similarity as images. The "pixels" of the 2-Dimensional (2D) grid need to have correlations with their neighbors that lead to higher-level structures similar to images. Many computer vision tasks include the creation of such image-like 2D grids, and deep perceptual loss has been applied to some of them, including object detection [17], image segmentation [23], and depth prediction [18].

Object detection is the task of locating the area of an image that contains an object. Machine learning has been successful in improving the performance on these tasks, but the detection of small objects remains a challenge [17]. Deep perceptual loss has been used to try and tackle this challenge by training a model to correlate the feature maps of images where the image is small to those where it is big.

Image segmentation is the task of taking an image and classifying each pixel by which category or instance it belongs to. Since the output is a value of each pixel, it has the same structure as an image, and some structures in the original image should remain in the silhouettes of categories or instances. Image segmentation is sometimes applied to medical imaging, and deep perceptual loss has been used in this case as well [101].

Depth prediction is the task of predicting the depth of each pixel in an image (*i.e.*, how far that part of the image is from the camera). Like image segmentation, this gives

an image-like output with one value per "pixel", the structure of which takes on patterns from the original image.

There is no hard line between image and image-like data. An infra-red photo, for example, could be considered an image outside the human spectrum or a 2D grid of temperature labels. Ultimately there is no hard line, and a lot of data that is not itself images can be visualized with images. For the use of deep perceptual loss, it is likely better to ask if the concept of human perception of visual similarity is applicable to the data.

2.6.3 Feature Learning Pretext Task

Like any loss function, deep perceptual loss may be used while pretraining models on a pretext task. The output of the pretext task must be an image or image-like for the reasons described earlier, but the pretrained model can then be applied to other tasks. For example, the encoder part of the VAE-GAN could be used for dimensionality reduction of images. The most common use of autoencoders trained with deep perceptual loss is for image synthesis and transformation. As part of the research for this thesis, deep perceptual loss has been successfully used to train autoencoders for dimensionality reduction for downstream prediction [14]. Though since the output images are not necessary after training, it could be more effective to directly use the decoder to predict the deep features of the loss network, similar to knowledge distillation. This has also been tested during the research involved in this thesis and was shown to have similar performance with faster training compared to deep perceptual loss [30].

2.7 Deep Perceptual Similarity Metrics

Calculating loss as the difference between the model and desired output is inherently a similarity metric. As such, the methods used in deep perceptual loss can be directly applied to deep perceptual similarity. This idea has been used even before the introduction of deep perceptual loss, as the deep features of neural networks to find similar images have previously been utilized in content-based image retrieval [102]. Later deep perceptual similarity was applied by *Zhang et al.*, [2] directly to perceptual similarity datasets on which it was shown to achieve state-of-the-art performance.

Zhang et al., [2] showed that deep perceptual similarity metrics with ImageNet pretrained CNNs as loss networks can outperform previous rule-based metrics without explicitly training on the task. This is impressive since machine learning models pretrained on one task typically perform poorly on downstream tasks without additional training for the given task. The work also shows that metrics with randomly initialized loss networks can achieve similar performance as rule-based methods. While not competitive with pretrained loss networks, this performance is at least partially related to the CNN structure.

2.7.1 Biological Motivation of Deep Perceptual Similarity

In biology, relations have been found between the ability of machine learning models to classify objects to their ability to classify neural activity when those images are used as stimuli [103]. This implies that the similarity of visual stimuli can perhaps be measured by the similarity of neural activity in the brain. Though, applying this approach to create image similarity metrics is likely unpractical, not to say unethical.

The idea of measuring similarity by neural activity does inspire another potential approach. Since CNNs are known to have similar structure to [68, 69] and learn similar features as the visual cortex [70, 71], perhaps measuring deep features can be used instead of neural activity. Deep perceptual similarity for images could therefore be considered to be a stand-in for measuring the difference in neural activity caused by different visual stimuli.

2.7.2 Training Deep Perceptual Similarity Metrics

Zhang et al., [2] also explicitly train some metrics on the BAPPS dataset. Such trained metrics are referred to in the work as Learned Perceptual Image Patch Similarity (LPIPS) metrics and are what this thesis will call any deep perceptual similarity metrics that have been trained on perceptual similarity data.

Similarity scores in the work are calculated by extracting the deep features from the loss network. The features are multiplied by scalars learned during training and unit-normalizing along the channel dimension. The similarity score is calculated as the MSE of between the extraction from each image and averaging spatially over each feature map. This procedure is formalized in Eq. 2.2, where z and z_0 are the unit-normalized deep features extracted by the loss network from the images x and x_0. N is the set of layers in the loss network from which features are extracted. w are positive scalars which are learned for LPIPS metrics and set to 1 otherwise.

$$d(x, x_0) = \sum_{l \in L} \frac{1}{H_l W_l} w^l \odot ||z^l - z_0^l||_2^2 \tag{2.2}$$

The training is performed on the 2AFC part of the BAPPS dataset. The metric being trained calculates the similarities between the distorted images and the original image as described above. The metric is evaluated with the 2AFC score based on whether the metric agrees with the human judges or not. Since the similarity scores do not have to be exactly the same as the human judgments loss is not calculated at this step. Instead, a three-layer CNN with 1×1 convolutions takes the similarity scores, the difference between the scores, and each score divided by the other as input and outputs which distortion was closer. This CNN is trained with the metric, and loss is calculated by the BCE of the CNN output and the human judgment. The training procedure is visualized in Fig. 2.11.

Three different versions of this procedure are used to train the LPIPS metrics that vary based on how the loss network is trained. The first version, called "scratch", trains the loss networks and scalars from scratch with randomly initialized parameters. The second version, called "fine-tune", trains ImageNet pretrained loss networks as well as

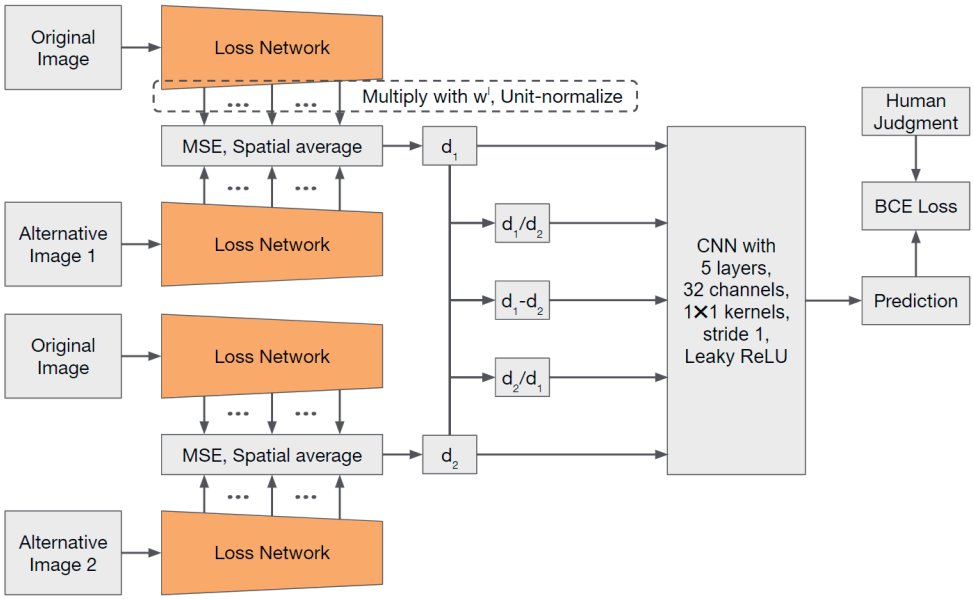


Figure 2.11: Training of an LPIPS metric on a 2AFC dataset as performed by *Zhang et al.*, [2].

randomly initialized scalars. The third version, called "lin", uses ImageNet pretrained loss networks but only trains the randomly initialized scalars. These are compared against a baseline with ImageNet pretrained loss networks, all scalars set to 1, and no additional training.

Interestingly, the LPIPS metrics only perform slightly better than the baseline metrics. Additionally, the LPIPS metrics only perform better on the types of images and distortions for which they have been trained. On other images and distortions, the baseline metrics perform the best. The work, including the results, is further detailed and analyzed in Chapter 3.

Another interesting observation is the similarity between 2AFC training of LPIPS metrics and contrastive learning. In both cases, the models are trained to create deep features that are more similar for some samples and less similar for others. The similarity can be further seen by comparing Fig. 2.7 from earlier to Fig. 2.11, which respectively visualize a contrastive learning scenario and the described 2AFC training of LPIPS metrics. The 2AFC training can be considered a contrastive learning procedure. Though, some differences exist between the 2AFC training performed by *Zhang et al.*, [2] and typical contrastive learning procedures. In contrastive learning, negative pairs that should have dissimilar features are typically created by distorting two different original images, while *Zhang et al.*, [2] only train on pairs created from the same image. Contrastive learning typically also trains to have the same deep features for all distortions of the same image,

while the LPIPS metrics were trained to learn which distorted versions of the same image should have more or less similar deep features.

2.8 Deep Perceptual Similarity Beyond Images

In this thesis, deep perceptual similarity refers to using the deep features of networks to calculate the similarity of images, and deep perceptual loss is when this is used for loss calculation. In the larger field of machine learning, the terms are also used beyond images and the computer vision domain. The method of mimicking human perception of similarity with deep features originated in computer vision for images, and it is often assumed that deep perceptual similarity refers to image similarity.

There are other domains for which human perception of similarity is interesting. Some of these domains have adopted the methodology of deep perceptual loss and similarity, and the term is now also used to refer to these use cases. This section briefly covers how the method has been adapted to these cases.

Perceptual similarity can be used to refer to the similarity between any human (or animal) perception [104]. Perceptual similarity has been widely used in the domain of audio and speech analysis with many rule-based methods existing [105]. The ideas behind deep perceptual loss have also been applied to the audio domain. The basic structure is the same, where the similarities of sounds are measured by comparing the deep features of pretrained models. The typical audio application calculates deep perceptual loss by using a model that has been trained with audio input, such as waveforms [106]. However, CNN architectures designed for images have also been used as loss networks by transforming the sound into an image-like spectrogram [106]. Deep perceptual similarity has also been used in the same way for audio, with contrastive learning used to train loss networks specifically as similarity metrics [107].

A similar approach to deep perceptual similarity has been used in the field of natural language processing. The similarity of words or sentences has been measured by their deep representations as produced by methods such as word2vec [108] or transformers [109]. Such deep features have become integral to the field and are widely used in deep learning applications on text [110]. However, whether or not texts are perceived and, therefore, if this is considered deep perceptual similarity is up for debate. Regardless, since deep features are used for similarity calculations for both image and text, comparing them is likely beneficial.

Chapter 3

Implementing Suitable Loss Networks

The core component of deep perceptual loss and similarity is the loss network which is used for extracting the features used to represent the input data. The loss network can be any DNN that can take images as input, meaning there is an endless variety of networks to choose from. This leads to the research question that this chapter builds upon.

Q1 How does the loss network implementation affect downstream performance?

Addressing Research Question 1, even partially, would benefit further research and applications of deep perceptual similarity and loss. Further research gain motivations for which loss networks to focus on and where new state-of-the-art implementations may be found. Better implementations have a better performance which directly benefits the applications of the methods.

There are many parts to consider in a loss network. Which neural network architecture to use, how to train or otherwise set the parameters of that architecture, and then which features of that network to use for similarity comparison? There are many popular architectures to choose from, and many have not yet been used. The parameters of any selected architecture need to be set, perhaps randomly, but more likely through training. The pretext task for training needs to be decided, what dataset to use, and whether to pretrain the network, train it while using it as a loss network like in an actor-critic setup, or both. Even when an architecture and training method have been chosen, the process for extracting features for comparison need to be selected. These considerations have been used to split Research Question 1 into three sub-questions.

Q1.1 How does the DNN architecture of a loss network affect downstream performance?

Q1.2 How does the pretraining of a loss network affect downstream performance?

Q1.3 How does the feature extraction of a loss network affect downstream performance?

To address these questions, this chapter primarily draws on three works by *Zhang et al.*, [2], *Kumar et al.*, [67], and *Pihlgren et al.*, [32]. The work by *Zhang et al.*, [2]

introduced the use of deep perceptual similarity metrics for perceptual similarity and made some initial tests with different loss networks. The second work by *Kumar et al.*, [2] makes a systematic analysis of a large number of loss networks pretrained on the ImageNet dataset [64] for perceptual similarity. The final work by *Pihlgren et al.*, [32] also makes a systematic analysis of many ImageNet pretrained loss networks but extends the evaluation to cover deep perceptual loss in addition to deep perceptual similarity. Additionally, the final work was produced as part of the research for this thesis.

This chapter first covers the commonalities between the experiments of the three works and places them into a general experimental setup. It then covers how the three works implement the evaluated loss networks. This is followed by the benchmarks that are used for evaluation and how performance is measured. The results and analysis are split between the perceptual similarity benchmark and loss benchmarks. The deep perceptual similarity analysis is based on all three works, while the deep perceptual loss analysis is based on the work by *Pihlgren et al.*, [32]. The analysis of each method ends with a section containing guidelines for implementing suitable loss networks. Finally, there is a joint discussion of the findings, their implications for the larger field, and future directions of research.

Some parts of the research questions are neither addressed in the above-mentioned works nor will be covered by this chapter. Deep learning for computer vision almost exclusively employs CNNs architectures, and even alternatives like Vision Transformers (ViTs) often contain convolutional layers [111, 112]. As such, the work on deep perceptual similarity and loss has almost completely focused on architectures with convolutions, and so too, will this chapter. While loss networks can be trained simultaneously with the model they calculate loss for, this is rarely done, and most applications use pretrained loss networks. This chapter only covers pretrained loss networks and predominantly focuses on loss networks pretrained for classification on ImageNet. Beyond the specifics of the loss network, there are other factors in how deep perceptual loss and similarity are calculated that are not explored in this chapter. More than one loss network could be used at once, and other loss calculations could be used as well, but this chapter does not explore either. This chapter mainly covers the deep perceptual similarity with spatial comparisons, though other ways to compare the extracted features are covered in Chapter 4.

3.1 General Experimental Setup

All three works analyzed in this chapter follow the same framework for running the experiments, which is described here and illustrated in Fig. 3.1. All three works define a set of loss networks that can be split into (i) their composing DNN architectures, (ii) methods for setting the parameters of those architectures, and (iii) features extraction layers. Performance scores for each loss network are gathered by repeating the experiments from previous works with that loss network. All three works use the BAPPS dataset introduced by *Zhang et al.*, [2] to gather performance scores on perceptual similarity. *Pihlgren et al.*, [32] also gather performance scores for deep perceptual loss from experiments from

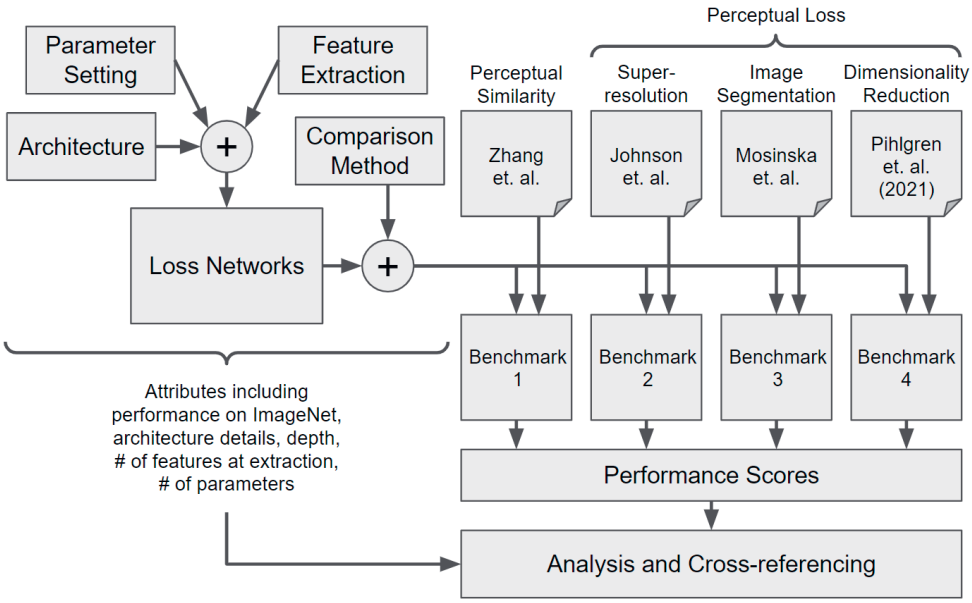


Figure 3.1: Framework for analyzing different aspects of loss networks from *Pihlgren et al.*, [32]. The framework follows the implementation of loss networks with different architectures, parameter settings, and feature extraction methods that are evaluated on four benchmarks based on earlier work to analyze the effect of different loss networks on performance. The previous works used as benchmark experiments are used are those by *Zhang et al.*, [2], *Johnson et al.*, [15], *Mosinska et al.*, [23], and *Pihlgren et al.*, [30]. By analyzing the attributes of the loss networks together with the performance scores, insights are gained regarding the use of deep perceptual loss and similarity. The framework has been adapted to also fit the works of *Zhang et al.*, [2] and *Kumar et al.*, [67] which exclude Benchmark 2, 3, and 4.

three different works, which are elaborated on later. The gathered performance scores are analyzed in conjunction with the attributes of the loss networks to gain insight into the correlations between performance and those attributes.

3.2 Loss Network Collection

This section describes how each of the three works implemented the loss networks used for the experiments. The descriptions are brief and with more details found in the original works.

Table 3.1: The architectures and features extraction layers used by *Zhang et al.*, [2] as loss networks.

Architecture	Feature Extraction Layers
AlexNet [79]	1^{st}, 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} ReLU
SqueezeNet 1.1 [81]	1^{st} ReLU, 2^{nd}, 4^{th}, 5^{th}, 6^{th}, 7^{th} and 8^{th} Fire
VGG-16 [73]	2^{nd}, 4^{th}, 7^{th}, 10^{th}, and 13^{th} ReLU

3.2.1 Loss Networks in *Zhang et al.*

Zhang et al., [2] evaluate three commonly used CNN architectures AlexNet, SqueezeNet and VGG-16. For each architecture, a single feature extraction setup is used, with features extraction from multiple layers spread evenly across the convolutional layers. The architectures and feature extraction layers are shown in Table 3.1.

Parameters were set for each model by pretraining on the ImageNet dataset. Parameters were also set by specifically training LPIPS metrics for perceptual similarity on the 2AFC training set of the BAPPS dataset with three different procedures as explained in Subsection 2.7.2. Additionally, a few other parameter setting methods were tested on variants of the AlexNet architecture. These AlexNet models had parameters set through random Gaussian initialization, stacked k-means initialization [113], and self-supervised learning using puzzle-solving [114], cross-channel prediction [115], motion segmentation [116], generative modeling [117], egomotion prediction [118], inpainting [119], video prediction [120], colorization [121], and sound prediction [122].

3.2.2 Loss Networks in *Kumar et al.*

Kumar et al., [67] evaluate an extensive collection of loss networks. Six ResNet architectures, six EfficientNet architectures, and four ViT architectures are used. The ResNet architectures vary in depth from 6 to 200 layers. The EfficientNet architectures vary in size from the $B0$ to the $B5$ variants. The ViT architectures use two network sizes and two patch sizes for four possible combinations. In addition, these base architectures are modified to test variants with varying widths. For the ViT architectures, depth is also varied.

Features are extracted from 4 different layers of the ResNet and EfficientNet architectures. These features are evenly spaced throughout the networks at places where the spatial dimensions of the features are reduced. Mostly *Kumar et al.*, [67] use all feature extraction layers in the loss networks. However, some evaluations were also made using loss networks where features were extracted from only one of the layers. For the ViT architectures, the CLS tokens generated by the models are used as features.

Kumar et al., [67] evaluate parameter settings through pretraining the architectures on ImageNet images at the non-standard resolution of 64×64. Despite only pretraining on ImageNet, the work evaluates a host of training procedures by changing the parameters of the standard procedure. The pretraining parameters that are altered are the number of

Table 3.2: Pretrained architectures and extraction layers used in the loss networks evaluated by *Pihlgren et al.*, [32].

Architecture	Feature Extraction Layers
	VGG Networks
VGG-11 [73]	1st, 2nd, 4th, or 8th ReLU
VGG-16 [73]	2nd, 4th, 7th, or 13th ReLU
VGG-16_bn [73]	2nd, 4th, 7th, or 13th ReLU
VGG-19 [73]	2nd, 4th, 8th, or 16th ReLU
	Residual Networks
ResNet-18 [74]	1st ReLU, 1st, 2nd, or 4th Block Stack
ResNet-50 [74]	1st ReLU, 1st, 2nd, or 4th Block Stack
ResNeXt-50 32x4d [75]	1st ReLU, 1st, 2nd, or 4th Block Stack
	Inception Networks
GoogLeNet [76]	1st BN, 1st, 3rd, or 9th Inception Module
InceptionNet v3 [77]	3rd BN, 1st, 3rd, or 8th Inception Module
	EfficientNet
EfficientNet_B0 [78]	1st SiLU, 1st, 4th, or 7th MBConv
EfficientNet_B7 [78]	1st SiLU, 1st, 4th, or 7th MBConv
	Uncategorized Networks
AlexNet [79]	1st, 2nd, 3rd, or 5th ReLU
DenseNet-121 [80]	1st ReLU, 1st, 2nd, or 4th Dense Block
SqueezeNet 1.1 [81]	1st ReLU, 1st, 4th, or 8th Fire Module

epochs, central or random crop, weight decay [123], label smoothing [77], dropout [124], number of target classes, learning rate, and input augmentations. Additionally, models pretrained on the standard ImageNet resolution of 224×224 and pretrained ImageNet models with feature scalars trained on BAPPS are evaluated.

Kumar et al., [67] also evaluates comparison methods other than the popular spatial variant. The other functions for comparing deep features that are being evaluated are the style variant introduced in neural style transfer [93] and a novel mean variant. The mean comparison work by averaging the features over the spatial dimensions before calculating the squared difference. Comparison methods are not the focus of this chapter and are explored in Chapter 4 instead.

3.2.3 Loss Networks in *Pihlgren et al.*

Pihlgren et al., [32] evaluate four VGG architectures, three ResNet architectures, GoogLeNet, InceptionNet v3, EfficientNet_$B0$ and _$B7$, AlexNet, DenseNet-121, and SqueezeNet. For each architecture, only a single parameter setting are evaluated. These parameter settings are obtained by using out-of-the-box models from the Torchvision framework [72] pretrained on ImageNet.

Four loss networks with different feature extraction layers are tested for each ar-

chitecture. The feature extraction layers have been chosen, so two are early, one is in the middle, and one is late in the convolutional layers of the architectures. The exact architectures and feature extraction layers used can be found in Table 3.2.

3.3 Benchmark Experiments

The performance scores for the loss networks implemented in the three works are obtained using experiments established in four works, referred to as the Benchmark 1 through 4. Perceptual similarity is evaluated in all three works on Benchmark 1. Benchmark 1 consists of evaluation on the BAPPS dataset which is also introduced by *Zhang et al.*, [2]. *Pihlgren et al.*, [32] also evaluates the implemented loss networks for deep perceptual loss on the remaining benchmarks. Benchmark 2, 3, and 4 cover the three categories of perceptual loss applications established in Section 2.6; image synthesis, image-like outputs, and feature learning pretext tasks. The deep perceptual loss benchmarks consist of the experiments presented in *Perceptual Losses for Real-Time Style Transfer and Super-Resolution* (Benchmark 2) [15], *Beyond the Pixel-Wise Loss for Topology-Aware Delineation* (Benchmark 3) [23], and *Pretraining Image Encoders without Reconstruction via Feature Prediction Loss* (Benchmark 4) [30]. The experimental procedure of the four benchmarks is presented below, with more detailed descriptions found in the benchmark works.

3.3.1 Benchmark 1: Perceptual Similarity

Benchmark 1 evaluates loss networks for perceptual similarity on the BAPPS dataset as presented in Subsection 2.2.4. Since the loss networks can be used directly on the dataset, no further experimental setup is needed. The performance scores for the two parts of the dataset are the 2AFC score for each subdivision, the average over the subdivisions, and the JND score.

3.3.2 Benchmark 2: Super-resolution

Benchmark 2 uses loss networks for training models for style transfer and single-image super-resolution. Only the super-resolution has any performance scores, and as such, only that part is used for loss network evaluation. The benchmark trains an image transformation network to take a downscaled image and output an estimation of the image at its original resolution (288×288). The transformation network consists of residual blocks similar to those used in ResNet architectures.

The transformation network is trained using 288×288 patches cropped from 10000 images from the MS-COCO training set [125]. The patches are blurred using Gaussian blur and then downsampled with bicubic interpolation to the input low-resolution images that are either $\times 4$ or $\times 8$ smaller than the output. Each input patch x is then upscaled by the transformation network τ and compared with the original image X using loss networks with extraction layers L according to Eq. 3.1.

$$\mathcal{L}_{feat}^{\phi,L}(x,X) = \sum_{l \in L} \frac{1}{C_l H_l W_l} ||\phi^l(X) - \phi^l(\tau(x))||_2^2 \tag{3.1}$$

The loss networks are evaluated using the performance of the trained transformation network. The transformation network is tested on the Set5 [126], Set14 [127], and BSD100 [128] (the test set of BSD300) datasets. The images in those test sets have been $\times 4$ and $\times 8$ downscaled with the same procedure as the training patches and are then upscaled by the trained network. The performance scores are the difference between the original and upscaled test images as measured by SSIM and Peak Signal-to-Noise Ratio (PSNR).

The exact implementation and training details can be found in the original work [15].

3.3.3 Benchmark 3: Image Segmentation

Benchmark 3 uses loss networks to train models for delineation, a form of image segmentation where the goal is to segment lines such as edges of objects or roads in satellite images. It trains models with the U-net architecture [16] to output the segmentation map y when given an input image X. The models are trained using the loss described in Eq 3.2, where U is the U-net model being trained and ϕ is the loss network with extraction layers L.

$$\mathcal{L}_{top}^{\phi,L}(x,y) = \sum_{l \in L} \frac{1}{C_l H_l W_l} ||\phi^l(y) - \phi^l(U(x))||_2^2 \tag{3.2}$$

The model is trained and tested for delineating roads on the Massachusetts Road Dataset (MRD) dataset [129]. The performance scores of the loss networks are the performance of the trained U-net on the test set measured by completeness, correctness, and quality of delineation as defined by *Wiedemann et al.*, [130]. Completeness, correctness, and quality for delineation are analogous to recall, precision, and critical success index used in many other applications.

The exact implementation and training details can be found in the original work [23].

3.3.4 Benchmark 4: Dimensionality Reduction

Benchmark 4 uses loss networks to train convolutional autoencoders for dimensionality reduction of images. The reduced features are used as input to small machine learning models that are then trained on classification with a small dataset. The autoencoder architecture used is a modified non-variational version of the network presented by *Ha and Schmidhuber* [131]. The benchmark uses the Street View House Numbers (SVHN) [132] and STL-10 [133] datasets, both of which come split into a training set, a test set, and an auxiliary set. The images from the auxiliary sets are used to train the autoencoder, which is then used to extract reduced features from the training and test sets. The loss for training the autoencoder AE for the input x is calculated using the loss network ϕ with extraction layers L according to Eq. 3.3.

$$\mathcal{L}_{rec}^{\phi,L}(x) = \sum_{l \in L} \frac{1}{C_l H_l W_l} ||\phi^l(x) - \phi^l(AE(x))||_2^2 \tag{3.3}$$

A number of small Multilayer Perceptrons (MLPs) are trained for classification on the training set, using the reduced features as input. The MLP with the best validation performance on the training set is then evaluated on the test set. The performance scores for the loss networks are the test accuracy of the MLP on SVHN and STL-10. The reason for choosing these performance scores is that the performance of a feature extractor (the autoencoder) can be measured by the performance of downstream models (the MLP).

The exact implementation and training details can be found in the original work [30].

3.4 Results and Analysis for Perceptual Similarity

This section will cover the results from Benchmark 1 for each of the three works. First, the results from each work are presented along with the implications of those results. Then the results from all three works are analyzed together, and general findings are presented.

3.4.1 Results from *Zhang et al.*

The 2AFC score of the loss networks from *Zhang et al.*, [2] on Benchmark 1 are summarized in Fig. 3.2 together with the performance of some rule-based methods. The results are shown for the *Distortions* and *Real Algorithms* subdivisions of the 2AFC validation set of BAPPS. Additionally, results from the JND part of BAPPS is shown together with the *Distortions* 2AFC score in Fig. 3.3. The strong correlation between 2AFC and JND scores indicate that deep perceptual similarity metrics generally perform well on perceptual similarity, not just the specific 2AFC part of BAPPS. The distortions used on the JND part is a subset of those used for the 2AFC part, which shows that the two ways of collecting judgments seem to find similar patterns of human perception.

Interestingly the LPIPS metrics have comparable performance to the pretrained networks on *Real Algorithms* but significantly outperform them on *Distrotions*. This is likely because the 2AFC training set uses the same distortion types as the *Distortions* test set. This implies that LPIPS metrics only perform better on the type of distortions for which they have been trained and perform similarly to other pretrained networks otherwise. So unless the typical images or distortions for which you want a similarity metric is known, there is likely no benefit to using LPIPS metrics. Later on in Section 5.6, it is suggested that there may be a flaw in how *Zhang et al.*, [2] train LPIPS metrics and that resolving the flaw may improve generalization.

The randomly initialized loss networks do perform significantly worse than the others but still perform on par with the rule-based methods (called low-level in Fig. 3.3). This likely reflects how CNNs share architectural similarities to the human visual cortex [69]. The similarities captured by the structure lead to the average randomized CNN measuring similarity closer to human perception than not.

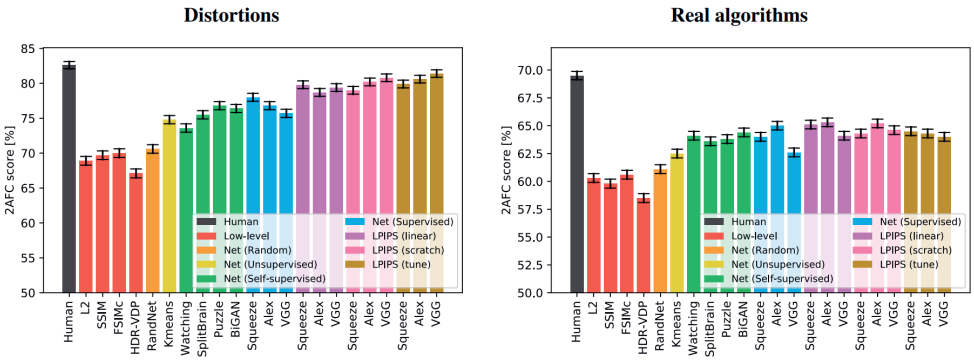


Figure 3.2: The validation 2AFC score on the *Distortions* and *Real Algorithms* parts of BAPPS for the best loss networks evaluated in the work by *Zhang et al.*, [2] from which the image originates. The performance of some rule-based (low-level) metrics and expected human performance are included for reference. Reproduced with permission.

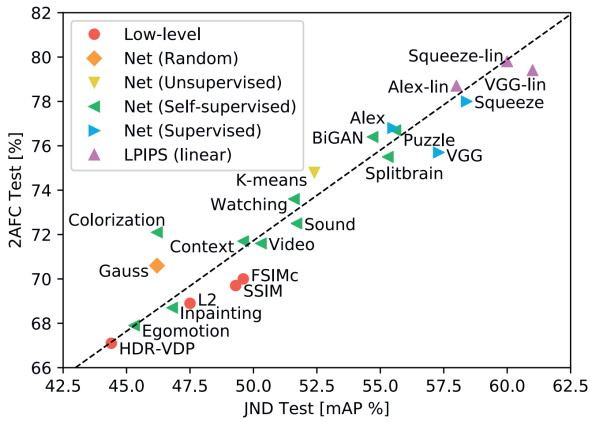


Figure 3.3: The 2AFC and JND scores on the *Distortions* parts of BAPPS for each loss network evaluated in the work by *Zhang et al.*, [2] from which the image originates. Performance of rule-based (low-level) metrics included for reference. Reproduced with permission.

The AlexNet model pretrained with supervised learning on ImageNet performs better than the models pretrained in other fashions. Though some of the other pretraining methods have similar performance, which means that ImageNet pretraining is not strictly superior. The self-supervised methods fill the entire spectrum of performance between the ImageNet pretrained models and the rule-based ones, implying that the choice of self-supervised method is significant to performance.

On the architecture level, AlexNet and SqueezeNet perform better than VGG-16 for

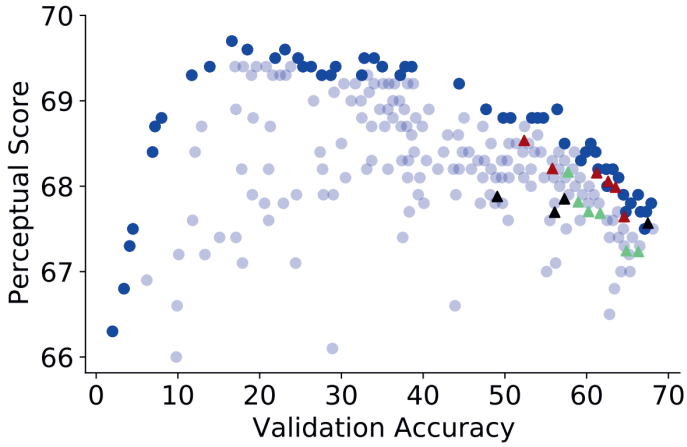


Figure 3.4: The ImageNet validation accuracy and BAPPS 2AFC validation score of all loss networks in the work by *Kumar et al.*, [67] from which the image originates. Each point represents a loss network used in the work. Dark blue indicates the best 2AFC score for that region of ImageNet accuracy. EfficientNets (▲), ResNets (▲), and Vision Transformers (▲) trained with their standard hyperparameters have been specifically marked. Reproduced with permission.

pretrained models, but the difference is negligible for the LPIPS metrics. The poor performance of VGG-16, which has significantly better ImageNet accuracy, is interesting as typically good performance on ImageNet pretraining implies good performance on downstream transfer learning tasks [67]. This lack of obvious correlation between ImageNet accuracy and perceptual similarity score was a major motivation of the work by *Kumar et al.*, [67].

Some of the loss networks are also evaluated on the TID2013 dataset. The evaluation confirms that the metrics work beyond the BAPPS dataset by achieving competitive performance. However, rule-based metrics still outperform the deep perceptual similarity metrics on TID2013. This implies that there is more to explore and improve in deep perceptual similarity. Some such improvements are detailed in Chapter 4.

3.4.2 Results from *Kumar et al.*

The primary finding of *Kumar et al.*, [67] is a Pareto frontier between ImageNet pretraining accuracy and BAPPS 2AFC score. This front means that loss networks with greater ImageNet accuracy have a greater potential 2AFC score up to a certain turning point accuracy, after which the potential is inversely correlated with accuracy. This is made shown in Fig. 3.4 which is originally from the work by *Kumar et al.*, [67].

The best loss networks for perceptual similarity discovered by *Kumar et al.*, [67] can be found around the 20-40% accuracy range. That a loss network pretrained on ImageNet

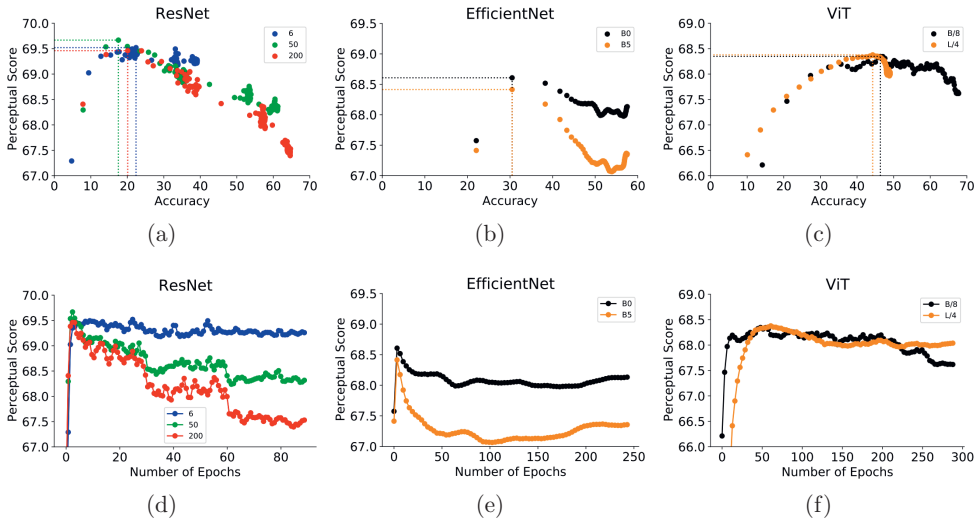


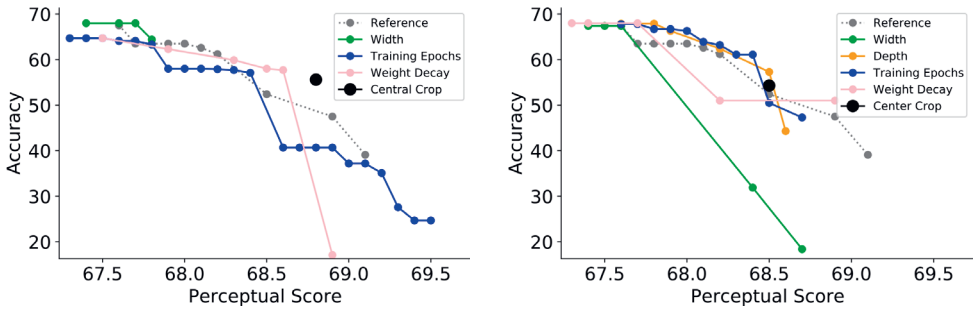
Figure 3.5: The 2AFC score compared to the ImageNet accuracy and number of training epochs for some ResNet (a and d), EfficientNet (b and e), and ViT (b and f) loss networks. Subfigures orignally from the work by *Kumar et al.*, [67]. Reproduced with permission.

lies in this region is not a guarantee of good performance since the Pareto frontier only implies an upper bound to the 2AFC score. The turning point ImageNet accuracy also varies for a given architecture or training procedure.

The turning point accuracy can be found for a given architecture and training procedure by varying the number of epochs during training to find at which accuracy the 2AFC peaks. *Kumar et al.*, [67] evaluated how the number of training epochs affected accuracy and 2AFC score for a few architectures under their otherwise standard training procedures. The results of this evaluation are visualized in Fig. 3.5. It is notable that similar architectures have similar turning point accuracy even with vastly different network sizes. Additionally, the turning point accuracy varies between architecture types. This means that the 20-40% accuracy range identified in Fig. 3.4 is not universal. Instead, this range is where the turning point accuracy for ResNet loss networks tends to be, and those perform the best on perceptual similarity among the evaluated loss networks.

Since the turning point accuracy varies for a given architecture and training procedure, training to reach a specific accuracy is only useful if the turning point accuracy is known. Computing the validation 2AFC score on BAPPS is significantly faster than training an epoch of ImageNet. Therefore, it is comparably inexpensive to keep track of the epoch with the best 2AFC score during pretraining. However, training on ImageNet in general is resource intensive. As such, it is not always feasible to train many models on ImageNet to find the best architecture and training procedure.

Further results by *Kumar et al.*, [67] show that scaling down the loss networks by using less depth or width can increase the performance of perceptual similarity. Training



(a) ResNet-200 (b) ViT B/8

Figure 3.6: The Pareto frontiers of ImageNet accuracy and 2AFC score on BAPPS found by *Kumar et al.*, [67], from which the images originate, as different training hyperparameters are changed. The frontiers are shown for models with (a) ResNet-200 and (b) ViT B/8 architectures. Reproduced with permission.

hyperparameters can also be changed to improve performance, with dropout and label smoothing being the only hyperparameters with similar optimal values for accuracy and perceptual similarity. This is illustrated in Fig. 3.6, which shows Pareto frontiers between ImageNet accuracy and 2AFC score for two architectures for altering several hyperparameters. This can be interpreted in two ways. First, the turning point accuracy might exist since the hyperparameters that give good perceptual similarity are different from those that give high ImageNet accuracy. Second, the hyperparameters that give poor ImageNet accuracy produce models that are closer to the turning point accuracy.

Further experiments were conducted to verify that the turning point accuracy was not just an artifact of the specific implementation. The results from these experiments also provide some additional information when it comes to performance. Feature extraction from the later layers gives better performance than the early layers. Increasing the number of classes in the pretraining data can decrease the performance of larger metrics. *lpips* metrics trained on BAPPS do not have as noticeable turning point accuracy but still display the trend that lower accuracy loss networks perform better. Using the mean comparison method can slightly increase performance but seemingly does not affect the turning point accuracy. It is also shown that there is no correlation between the distortion vulnerability and perceptual similarity performance of loss networks. Both comparison methods and distortion vulnerability are further examined in Chapter 4. Finally, the findings are shown to generalize beyond BAPPS on the TID2013 dataset.

3.4.3 Results on Benchmark 1 from *Pihlgren et al.*

Pihlgren et al., [32] only evaluate out-of-the-box models pretrained on ImageNet and therefore do not provide insight into parameter setting. Instead, the work evaluates

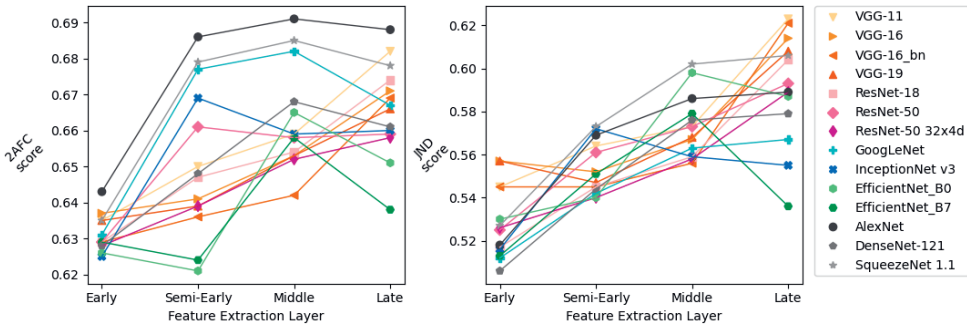


Figure 3.7: The validation 2AFC and JND scores of all loss networks evaluated by *Pihlgren et al.*, [32]. The scores are organized according to how deep into the convolutional layers the deep features were extracted.

many types of CNN architectures and how the choice of feature extraction layer affects performance.

The primary finding of *Pihlgren et al.*, [32] from Benchmark 1 is that for out-of-the-box models, the feature extraction layer has as much impact on performance as architecture. In Fig. 3.7, the 2AFC and JND scores of each evaluated loss network are organized according to how deep into the convolutional layers of the model features are extracted. For any architecture, both scores could be significantly increased by selecting the most suitable layer for feature extraction. The figure also shows that the middle to later convolutional layers is typically significantly better to use than the earlier ones. The decrease in performance between the middle to later layers that some models experience could be due to the later layers being specialized for prediction on the pretraining dataset. For architectures, the simpler architectures tend to outperform the more complex ones. Additionally, within each architecture type, the deeper architectures perform worse when averaged across layers.

Using deeper layers for feature extraction or larger architectures also increases the computation needed to compute similarity scores. In Fig. 3.8, this computational demand is visualized as the FLOPS needed for the forward pass of each loss network is shown in comparison to their 2AFC and JND scores. The FLOPS needed for the forward pass of some loss networks are orders of magnitude greater than others that have better or comparable performance. For some architectures, choosing an earlier layer can decrease the needed computation by an order of magnitude with only small losses in performance. All of this motivates choosing less computationally expensive options, especially since the best-performing loss networks are found among those that use fewer FLOPS.

The work also analyzed the effect on the performance of a number of architecture and feature extraction attributes. Table 3.3 shows the Pearson linear correlation coefficients between many tested attributes and performance. Since the extraction layer correlates with perceptual similarity performance, many attributes that correlate with extraction layer depth also do so.

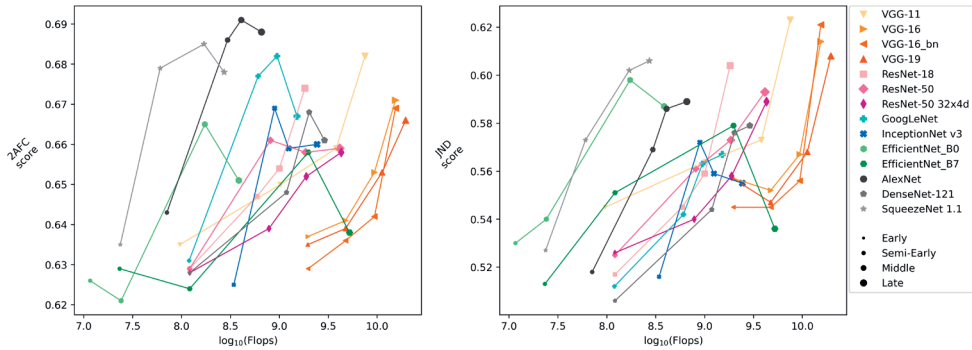


Figure 3.8: The 2AFC and JND scores on BAPPS of the metrics with ImageNet pretrained loss networks evaluated by *Pihlgren et al.*, [32] compared to the $\log_{10}$(FLOPS) used in the forward pass of those loss networks.

Table 3.3: The Pearson correlation coefficient between some architecture attributes and the 2AFC score on BAPPS of metrics based on pretrained loss networks with the given architecture attributes. Some of the attributes depend on at which layer the features are extracted and have been marked.

Attribute	2AFC Corr. Coef.	Depend on Layer?	Attribute Description
Layer	0.669	✓	Extraction layer as the numbers 1, 2, 3, and 4.
%Depth	0.554	✓	% of total depth at extraction layer.
%Depth (CNN)	0.617	✓	% of total conv. depth at extraction layer.
#Parameters	0.098	✓	#Parameters up to extraction layer.
%Parameters	0.219	✓	% of total parameters up to extraction layer.
#Channels	0.356	✓	#Channels at extraction layer.
#Features	-0.505	✓	#Features extracted at extraction layer.
View size	-0.162	✓	Pixels of the input that affects each feature.
%FLOPS	0.646	✓	% of total computing up to extraction layer.
Depth	0.030		#Matrix multiplication up to extraction layer.
Skip-connections	-0.301		Does the architecture have skip-connections.
Branching	0.163		Does the architecture have branching.
1×1 convs.	0.132		Does the architecture have 1×1 convolutions.
BatchNorm	-0.285		Does the architecture use batch normalization.
ImageNet acc.	-0.485		The ImageNet accuracy of the model.

3.4.4 Analysis of Results for Perceptual Similarity

Together the three works [2, 67, 32] show that parameter setting, architecture, and feature extraction all have a significant impact on the performance of loss networks on

perceptual similarity. Several interesting trends that go against common machine learning conventions and intuition emerge. There is no need to fine-tune the loss networks as they have impressive performance on perceptual similarity directly after training on the pretext task. Pretraining seems to have as much impact on performance as training on the actual task, and such training does not seem to generalize well to similar tasks. Loss networks with better ImageNet performance do not necessarily achieve higher perceptual similarity performance, and after a certain turning point, improved ImageNet performance is detrimental. Newer architectures with better optimal ImageNet performance do not outperform older architectures. The best feature extraction layer is not necessarily later in the network. These notable outcomes will be further elaborated on in this section.

Most tasks require at least some tuning of the pretrained model for it to perform well on the task. For some tasks, it is practically impossible for transfer learning to perform well without fine-tuning, as the pretrained model needs to be retrofitted with another output layer to have correctly formatted output. Any layers added to the end of the model in this way would then require training [65]. For models to be able to learn one task by training on another is a goal of entire subfields of machine learning, including multi-task learning and meta-learning. How, then, are old architectures designed specifically for image classification able to compete with old metrics without any retraining at all?

The likely answer to the question is how the calculation of deep perceptual similarity differs from most transfer learning cases. Transfer learning from pretrained models is typically performed by swapping some of the last layers to fit the output of the target task and training either those layers or all layers on a task-specific dataset [65]. In deep perceptual similarity, the pretrained model is instead often used without additions, some layers are chosen for feature extraction, and those features are then compared against each other and aggregated to get the similarity score. In the former case, the model features are used as input to train layers, and in the latter case, the similarity is measured by the difference in activations. Similarity calculation is a simpler task from some perspectives since even just comparing the two inputs with pixel-wise metrics gives a flawed but functional metric. The same would likely give close to random performance for many other computer vision tasks or even be impossible. Adding, as described in Subsection 2.7.1 that CNNs have similarities to the visual system in their structure and that trained CNNs have similar features, it is not surprising that they work better than pixel-wise metrics without change. There is also a significant difference between using the difference between features directly compared to learning a model on top of them. When they are used directly without retraining, the assumption is that each feature is relevant and equally important for the task. A retrained system does not need to rely on each feature being relevant and can even contain random and contra-productive features since the model can learn to ignore those features or adapt them into something useful for the task.

The difference between deep perceptual similarity and other transfer learning tasks is likely part of the explanation for all the previously mentioned quirks. When loss networks are applied directly without retraining, some of the features that are highly

tuned to ImageNet will not be beneficial to perceptual similarity, but the metrics cannot learn not to use them. In other domains having a breadth of features that are useful in some contexts and useless in others is beneficial since the models can learn to use them when appropriate. Such highly tuned features are less likely to appear in smaller models, models with lower accuracy, and models that are earlier in training. Thus better ImageNet models are, in general, good for transfer learning since the downstream models can learn to use the relevant features, but detrimental to perceptual similarity beyond a certain point since the features that are not relevant will still be considered. This also explains why older and smaller models tend to have better performance on perceptual similarity relative to their larger and more modern counterparts. The same argument explains the significance of pretraining. Later layers also tend to be more specific to the training data [65], which is likely why the last layers are not always the best choice for deep perceptual similarity.

3.5 Selecting a Loss Network for Similarity

Research Question 1 asks how the implementation of a loss network affects performance. The sub-questions further specify how the specifics of architecture, pretraining (parameter setting), and feature extraction is influential. The results and analysis from the three works [2, 67, 32] address these questions with respect to perceptual similarity. The knowledge gained from addressing the questions can be used to decide which loss networks to use to achieve better performance. This section details what choices to make and what trade-offs exist to select the best loss network for perceptual similarity. Following the division of the sub-questions, the section is split between selecting architecture, parameter setting, and feature extraction.

3.5.1 Selecting Architecture

Of the out-of-the-box Torchvision models that have not been pretrained specifically for perceptual similarity, the shallower models perform better. SqueezeNet 1.1, AlexNet, ResNet-18, and VGG-11 all achieve top scores on both the 2AFC and JND parts of BAPPS. With VGG-11 performing best on average and SqueezeNet 1.1 coming close with many fewer FLOPS needed for the forward pass. These four networks are notably the four with the worst performance on ImageNet out of all evaluated by *Pihlgren et al.*, [32]. Therefore picking pretrained models whose ImageNet performance is closer to the 20-40% turning point accuracy range identified by Kumar *et al.*, [67]. While the turning point accuracy of an architecture is not necessarily in that exact range, all the architectures evaluated by *Kumar et al.*, [67] have a turning point accuracy below 50%, which is well below most out-of-the-box models. Surprisingly selecting architectures with low ImageNet accuracy is likely a good way to select out-of-the-box models. Among those, one could favorably select architectures that require fewer FLOPS in the forward pass to minimize resource consumption.

However, as the next subsection covers, the best performance is achieved by loss networks that have been pretrained specifically to have accuracy close to the turning point. There is a lot less information to go on regarding which architectures are good in this scenario. However, ResNet architectures outperformed both EfficientNets and ViTs, so going for older architectures seems to still be a decent strategy. Additionally, the 50-layer ResNet has the best optimal performance but is only slightly ahead of the 6- and 200-layer architectures. This seems to imply that the exact depth does not matter as much as long as the accuracy is close to the turning point.

3.5.2 Setting Parameters and Pretraining

Five different types of parameter setting are considered between the three works; random initialization, self-supervised pretraining, pretraining on ImageNet for accuracy (out-of-the-box models), pretraining on ImageNet for perceptual similarity, and training LPIPS metrics on BAPPS. Out of these pretraining on ImageNet for accuracy followed by training on BAPPS achieved the best results. While similar results are likely achievable by LPIPS metrics pretrained for perceptual similarity, this is still uncertain. Additionally, it is possible that LPIPS metrics are less susceptible to decreased performance beyond the turning point accuracy as the metrics can learn which features are beneficial, though this too is uncertain.

While pretraining on ImageNet to find the turning point accuracy performs well, it is also the most resource-intensive parameter setting for slightly improved performance over out-of-the-box models. If resources are constrained, it is recommended to use an out-of-the-box ImageNet pretrained loss network and, if resources allow it, to train it as an LPIPS metric. Only if resources are plentiful would it be worth it to attempt ImageNet pretraining for perceptual similarity since those resources could likely be better spend testing a few different architectures and feature extraction layers instead. If such pretraining is conducted, it is better to train with images the same size as is used during inference.

3.5.3 Where to Extract Features

Both *Kumar et al.*, [67] and *Pihlgren et al.*, [32] identify the middle to later layers as the best for perceptual similarity. *Zhang et al.*, [2] also present the scalars for each feature of the pretrained loss networks learned by the LPIPS metrics (in the appendix of the preprint version of the work [134]). These scalars can be analyzed to see to what degree the features of each layer are utilized by the metrics trained on perceptual similarity. Such analysis reveals that the early layers tend to have a few features with large scalars and the remainder close to 0. The later layers have more features with non-zero scalars but not as many large scalars. This implies that a few features of the early layers are important, and many features in the later layers have moderate importance.

In general, feature extraction from the last quarter of the loss networks seems to be a good approach when no additional training is performed. When LPIPS metrics are trained, features should be extracted from many layers spread across the loss network

since the training can learn to ignore the disruptive features. Some other works have also shown that including pixel-wise metrics in the calculation can improve performance [3, 4].

While not considered a part of the feature extraction in this thesis, the comparison method used is also an important consideration. Of the three works, only *Kumar et al.*, [67] consider non-spatial comparison methods, and in the limited evaluations, they perform averaging the features of each channel before distance calculation seem to perform better. The issue of which comparison methods are good and why is analyzed in detail in Chapter 4.

3.6 Results and Analysis for Deep Perceptual Loss

Pihlgren et al., [32] evaluate loss networks for a number of deep perceptual loss applications in addition to the evaluation of deep perceptual similarity. The evaluation of deep perceptual loss was performed in Benchmark 2, 3, and 4, each of which trains models to perform tasks by using the various loss networks as part of the loss calculation. The performance scores for Benchmark 3 and 4, as well as all the Benchmark 2 scores from BSD100, are shown in Fig. 3.9. The figure orders the performance by architecture and extraction layer.

Interestingly, there is a strong correlation between the performance scores of each loss network, except for in the case of dimensionality reduction. The likely difference is that the calculation of the different performance scores in Benchmark 2 and 3 depends on the same input, which is not the case for Benchmark 4. All scores from Benchmark 3 are calculated on the same datasets; the three evaluation datasets in Benchmark 2 are very similar, while the two datasets used in Benchmark 4 are quite different. Additionally, the scores for dimensionality reduction are the performance of models that are two steps removed from the actual deep perceptual loss. In Benchmark 2 and 3, the loss is used to directly train the models being assessed, while in Benchmark 4, the loss trains autoencoders, and the features of those autoencoders are used as input to train the assessed models. So it is not surprising to see a lack of correlation in that specific task. In general, this implies that a loss network that works well for a certain task and dataset will not necessarily perform well on another dataset for the same task if that dataset is too dissimilar to the first one.

It is interesting that the performance of a loss network on one benchmark is seemingly not correlated to its performance on another. Though, the VGG networks without batch normalization perform well on all benchmarks. Additionally, there is some agreement within each performance score for which extraction layer gives the best performance. Notably, the preferred depth of extraction seems to be relative to the depth of the entire network rather than absolute depth. This trend also goes beyond specific architecture types. Likely the models extract similar features at the layers with similar relative depth.

The depth that works best for most loss networks seems to depend on what the task is. The clearest example of this is the super-resolution experiments in Benchmark 2, which favor the earlier layers for most architectures. Since the super-resolution performance scores are low-level and mostly depend on the exact values of the pixels, it is not sur-

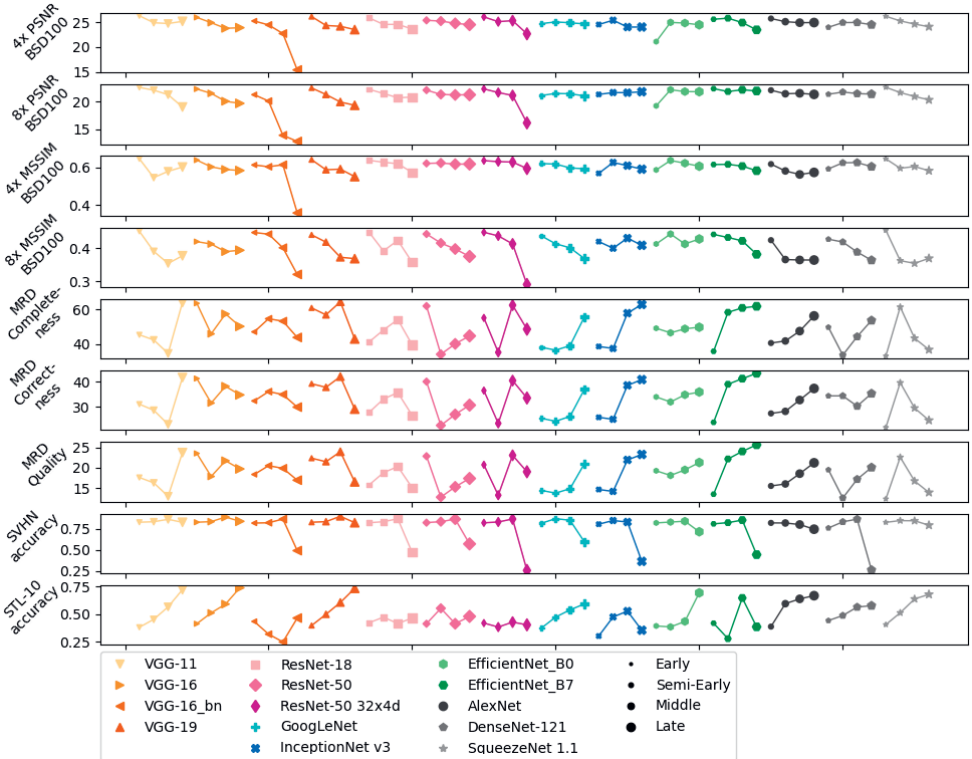


Figure 3.9: The performance scores of all out-of-the-box models for all feature extraction layers explored by *Pihlgren et al.*, [32]. The performance scores shown include all scores from Benchmark 3 and 4 as well as all BSD100 scores from Benchmark 2. Each model is represented by a color and symbol, and the extraction layers for each are shown from early to late, with the symbols growing in size for later extraction layers.

prising that the earlier layers closer to the input pixels perform better. Additionally, the trained transformation network gets a lower-resolution version of the image as an input which already contains information about the image's higher-level structures, making the transformation network less dependent on having to learn to generate such structures. On the other hand, the delineation experiments in Benchmark 3 generally show better performance in the later layers. Since the performance scores for delineation allow for leeway for where the lines are drawn, the exact pixels matter less, and the overall structure of the generated road network is more important. Both of these factors favor deeper layers. In Benchmark 4, the dimensionality reduction experiments mostly perform better with the later layers but with a notable difference between the two datasets. The SVHN accuracy performs better in the middle layers and then drops off in the later layers for most architectures. In some cases, this drop-off is significant. Meanwhile, the STL-10

accuracy is at its highest in the latest layer for most architectures. Since classification cares about the higher-level structures, it is not surprising that the later layers are important. STL-10 consists of images from ImageNet, which is what the loss networks were pretrained on, so the features in the last layers should be well-suited to the structures found in STL-10. However, SVHN consists of cropped photos of house number digits, which are quite different from most images in ImageNet. Therefore, the deep features of the last layers are likely too specific to ImageNet and thus do not work well on SVHN. To summarize, it seems like understanding the task itself and the structure of the dataset can indicate at which layers in the architecture the feature extraction should be performed.

Beyond the choice of layer, there are no simple correlations between the attributes of the architecture and its performance on any of the deep perceptual loss benchmarks. While a few architectures, like the VGG architectures without batch-normalization, generally perform well, there is no clear indicator as to why this is. Even the inverse correlation between ImageNet accuracy and performance scores that were present in Benchmark 1 is not there for the deep perceptual similarity tasks.

Better ImageNet performance does not imply that a loss network will give a better performance score for the deep perceptual loss experiments in Benchmark 2, 3, and 4. This is also the case in the deep perceptual similarity experiments in Benchmark 1. For deep perceptual similarity, the optimal performance scores were gained at a specific turning point accuracy for a given architecture. Perceptual similarity performance would decrease as the ImageNet accuracy moved away from that point. Such a correlation cannot be shown or rejected with the deep perceptual loss experiment. To do so would likely require an order of magnitude more experiments. Though, it seems likely that the same correlations should exist for deep perceptual loss, with the turning point accuracy being specific for each architecture and task.

3.7 Selecting a Loss Network for Loss Calculation

Like with deep perceptual similarity, addressing Research Question 1 gives insight into how to implement loss networks for the best performance on deep perceptual loss applications. Since deep perceptual loss covers a wide variety of tasks, each implementation will have to consider the specific task, making the procedure more complex. Additionally, evaluating loss networks for deep perceptual loss is often significantly more resource intensive than for deep perceptual similarity. Evaluating for deep perceptual loss entails training a model for the specific task using the loss network and then evaluating this model. The training of an extra model for a given task can take orders of magnitude more time than a simple evaluation on the test set of a perceptual similarity dataset like BAPPS. The loss network selection for perceptual similarity also benefited from analysis in all three works [2, 67, 32] instead of just one [32]. For these reasons, the implementation suggestions for deep perceptual loss are less detailed.

3.7.1 Selecting Architecture

The VGG architectures without batch-normalization all perform well on all three deep perceptual loss benchmarks. They take two out of the top three spots on all three benchmarks when averaged over the performance scores for the best layer of each architecture. VGG-19 does particularly well with the top spot on delineation and dimensionality reduction and takes third place on super-resolution. Unfortunately, it is difficult to extrapolate why these out-of-the-box models do so well since the analyzed attributes do not seem to correlate with the performance scores.

3.7.2 Setting Parameters and Pretraining

No parameter setting beyond ImageNet pretraining was analyzed for Benchmark 2, 3, and 4, which means that there is little to go on. It seems likely that evaluation during ImageNet pretraining to find the turning point accuracy for a given architecture and task would likely improve performance. However, this would be incredibly resource intensive. While evaluation on BAPPS would take a fraction of the time of a training epoch on ImageNet, doing the equivalent evaluation for deep perceptual loss would require completely training and evaluating at least one downstream model every epoch. This could increase the computation needed for the already expensive ImageNet pretraining by orders of magnitude. Therefore, such pretraining is likely a waste of resources compared to other methods of training.

It is also likely that using a pretraining dataset that is more similar to the actual data of the downstream task would be beneficial if the dataset is large enough. Training on such a dataset could be performed on a loss network that has already been pretrained on ImageNet to benefit from both datasets. While training LPIPS metrics for deep perceptual loss has not been evaluated, it would likely not be beneficial unless the model output is something where human perception is important. Examples of tasks with such outputs are style transfer and generation of natural images.

3.7.3 Where to Extract Features

The effect of feature extraction from different layers on the performance of different tasks was analyzed in length in the previous section. While there is no depth at which feature extraction is always better, there seem to be correlations between which layers would work better and the task. Evaluating different layers is a good place to spend additional resources that can be used. The motivation for this is that layer selection has shown to be so impactful on performance. Identifying good layers for one architecture generalizes to some degree to others, and the average best layer is less consistent between tasks than the average best architecture.

Earlier layers will likely perform better for tasks where performance is measured at the pixel level. Though, models trained purely with pixel-wise metrics may struggle to learn to produce the larger structures on which pixel-level improvements can be made [14]. For these reasons using deep perceptual loss can improve performance even in cases where

performance is measured in a pixel-wise fashion. Earlier layers are also likely to perform better where models do not have to represent higher-level structures from the data. An example of where models do not have to represent higher-level structures is when the structure is provided as an input, such as downscaled images for super-resolution.

The dataset used for the particular task is also relevant to the extraction layer. If the dataset images are similar to those in the pretraining dataset of the loss network, later layers will likely work better, and earlier layers otherwise. This will likely only matter if the most prominent extraction layers are later in the architecture. For example, if the other recommendations indicate that an early layer should be used, the dataset does not matter as much as if the recommendations indicate the middle to later layers. It is also uncertain how to handle cases when the dataset has similar higher-level structures to the pretraining dataset but is otherwise dissimilar. For example, should earlier or later layers be used in semantic segmentation of photographs? The segmentation maps are very dissimilar from the natural photographs of ImageNet, which would favor earlier layers, but they contain structures of the same objects that are found in ImageNet, which suggests later layers should be better.

Although the comparison method is not part of the feature extraction, it is included here. Only spatial comparisons are considered for Benchmark 2, 3, and 4, which makes conclusions difficult to draw. However, for most applications of deep perceptual loss, the exact spatial position of the output matters, which suggests that keeping spatial information is important. Though, global comparison methods could be used jointly with them to represent global information in the image. This is done in neural style transfer, where a spatial comparison method is used to represent the image content, and a global comparison method is used to represent style [93]. Chapter 4 goes into more detail regarding the strengths and weaknesses of different comparison methods.

3.8 Discussion

A lot of the questions that remain could likely be addressed by an extensive ablation study. How do different architecture attributes affect performance and the turning point accuracy for optimal perceptual similarity? Does the same correlation between ImageNet accuracy and downstream performance exist in deep perceptual loss? How do different parameter setting methods affect deep perceptual loss? How does the choice of pretraining dataset affect performance on different tasks? However, many of these questions would require extensive resources to answer through an ablation study. So such studies might not be an efficient use of scientific resources. Instead, surveys of the existing research on deep perceptual loss and similarity might provide answers.

The most interesting of the findings is the ways deep perceptual loss and similarity break from two established transfer learning conventions. The first such convention is that models with higher ImageNet accuracy typically provide better downstream transfer learning performance [67]. The second is that feature extraction is typically best performed at the last layers [65].

It is not necessarily the case that better pretraining performance leads to better

downstream performance. For example, the pretraining task might be unrelated to the downstream task. Even when pretraining benefits performance, this does not always hold [135]. Nevertheless, it seems like downstream performance has increased in many tasks as researchers have been applying ImageNet pretrained models with increasing ImageNet accuracy to them [67]. Deep perceptual loss and similarity do not follow this trend. Deep perceptual similarity even favors models with accuracy far below the optimal found for those architectures. Subsection 3.4.4 propose that this might be due to some fundamental differences between deep perceptual similarity and the typical transfer learning applications. Fundamentally, the proposed difference is that the typical transfer learning application trains a model on top of the extracted features, which can learn how to utilize them. If this is the case, then the identified correlation between ImageNet accuracy and perceptual similarity should not be as prominent for LPIPS metrics. *Kumar et al.*, [67] show that this is the case, though the turning point accuracy still exists, which means that the hypothesis is still an interesting proposition for future research.

It is possible that the lack of correlation between ImageNet accuracy and downstream performance is not isolated to deep perceptual loss and similarity. The convention that the two are positively correlated is often justified by showing that studies that use models with better pretraining performance do better than previous studies with lower pretraining performance. Such reasoning can be dangerous as there are many things that might cause a difference in performance between studies. Results on a task often improve as a combination of many factors. Improved downstream performance of newer studies sometimes comes from general improvements in methodology rather than utilizing newer architectures with better ImageNet accuracy. Similar issues have previously caused false trends of improvement. In the field of metric learning, it was assumed that the newer methods improved performance as performance had improved over time. This was called into question as a systematic study showed no significant difference between newer and older methods [136].

A systematic study of ImageNet pretrained models for general transfer learning applications has found that, in general, better models improve performance [66]. However, the study also found that this is not the case for all transfer learning tasks. It is also possible that it is not the improved ImageNet accuracy that is beneficial but that the architectures that achieve the performance do well for general tasks after fine-tuning. Furthermore, it is possible that the turning point accuracy for general transfer learning applications is found around very high accuracies making the trend difficult to uncover. Regardless, deep perceptual similarity still seems to stand out among transfer learning applications as a field with such a clear decrease in performance as accuracy increases beyond a specific range.

Feature extraction for transfer learning is typically performed in the later layers, and often by specifically replacing the last layer with a task-specific one [65]. It is often assumed that this is the best place for feature extraction to be performed, but for deep perceptual loss and similarity, this is not always the case. Instead, the optimal layer for feature extraction depends on the downstream task and dataset. This is not surprising since neural networks tend to learn features at different abstraction levels at varying

depths, with deeper layers containing more high-level abstract features [1]. As such, tasks where low-level features of the output are more important benefit from earlier layers, and when the high-level features are important later layers should be used. This poses the question of why transfer learning is mostly performed in the later layers. Since most transfer learning applications allow for retraining the neural network, the extracted features can be updated to better suit the task at hand. Additionally, tasks that require deep neural networks, such as object classification and segmentation, often benefit from higher-level structures. Despite this, this convention might need a reassessment in the larger field of transfer learning.

It is also interesting that the performance of loss networks is not correlated between different benchmarks. Part of this lacking correlation is the different preferences of feature extraction layers between tasks. The trend persists even when this is accounted for by comparing the average or maximum performance scores per architecture. There are only a few weak positive correlations between the performance scores of different benchmarks, which can likely be attributed to random chance (*i.e.*, compare enough scores, and some are bound to be incidentally correlated). For the deep perceptual similarity scores measured in Benchmark 1, it seems like the performance of a loss network is correlated to how close its ImageNet accuracy is to its turning point accuracy. Since most out-of-the-box models likely have accuracy significantly above the turning point, perceptual similarity scores are negatively correlated to ImageNet accuracy for those models. This is not the case for the deep perceptual loss benchmarks. There are many possible answers to why this is. It might be that the turning point accuracy of an architecture varies from task to task and that Benchmark 2, 3, and 4, in general, have higher turning point accuracies for the same architectures. To confirm this would require an extensive study. It might also be that deep perceptual loss is significantly different from deep perceptual similarity. While deep perceptual similarity directly uses the difference between deep features as a similarity metric, deep perceptual loss uses that similarity as a loss to train a model. Training with deep perceptual loss has similarities with knowledge distillation, where one pretrained teacher model is used to train a student model. In knowledge distillation, the student model often gets higher performance from training with the teacher compared to training directly on the dataset, even if the teacher model has significant errors on the dataset. This might also occur in deep perceptual loss, where models with flaws in their similarity measurements can still train downstream models with high performance. Understanding why similarity metrics perform differently under different circumstances and being able to create a metric for a specific circumstance is an interesting proposition. This goal is aligned well with Research Question 3, which is addressed in Chapter 5.

This chapter has considered architecture, parameter setting, and feature extraction. However, the comparison methods have been mostly left unexplored. The impact of using correct comparison methods is further explored in Chapter 4.

Chapter 4

Investigating Deep Perceptual Similarity

Deep perceptual similarity has been established as the state-of-the-art in perceptual similarity [2, 3, 67]. Deep perceptual loss is also considered part of the state-of-the-art for certain image synthesis tasks [97] and is under investigation for various applications. Most work on these methods has been focused on evaluating them on new tasks and applications. A smaller body of work has focused on the methods themselves and how to improve them. The direction for understanding and improving deep perceptual similarity that is the focus of this chapter is captured in Research Question 2.

Q2 What significant flaws do deep perceptual similarity metrics have, and how can they be mitigated?

To address Research Question 2 question, this chapter builds on the work by *Sjögren et al.*, [29], which was produced as part of the research for this thesis. This chapter complements that work with insights from other works devoted to improving deep perceptual similarity. By cross-referencing the various works that strive to improve deep perceptual similarity, a bigger picture of the method is formed, and directions for further improvements can be identified.

One example of a flaw that affects almost all deep learning models, including deep perceptual similarity, is adversarial examples. The chapter covers how this flaw affects deep perceptual similarity before moving on to flaws and improvements specific to the method itself. First, scenarios where the method succeeds where pixel-wise metrics do not are analyzed to understand how the method is successful. This knowledge is then used to infer the weaknesses of existing approaches and which other approaches can mitigate them. These other approaches are then similarly analyzed for flaws. The possibility of combining different approaches to alleviate the flaws in each is analyzed. Finally, future directions for further improvements of deep perceptual similarity are discussed, along with how existing and future improvements might impact deep perceptual loss.

4.1 Adversarial Attacks

Adversarial examples, as explained in Section 2.4, are inputs to a model that is almost imperceptibly different from another input but for which the model produces a completely different output [89]. For instance, a model can accurately classify an image of a Gibbon, but when less than a percent of adversarial noise is added, it classifies the image as something completely different. Creating adversarial examples for a model is typically called an attack, and many such attacks exist [90]. A common attack strategy is to use optimization to find a small amount of noise that creates an adversarial example when added to the input. Adversarial examples seem so far to be ubiquitous to neural network models [91], and most strategies for defending against them are vulnerable to other methods of attack [90]. Additionally, adversarial examples have been shown not only to be able to alter the output completely but the deep features of a DNN as well [137].

Deep perceptual similarity metrics are built around loss networks and are, therefore, vulnerable to adversarial examples. Since the deep features of the loss networks can be completely altered, this means that there are images that humans would perceive as almost exactly the same that deep perceptual similarity metrics would consider very dissimilar. However, rule-based metrics also suffer from the existence of counter-examples where the metrics go against human perception. Generating these counterexamples can be even easier to generate than adversarial examples, as the rules for those metrics are typically simpler than DNNs. Pixel-wise metrics, for example, will perform poorly with color shifts and inversions since it cares for the exact values of pixels. SSIM has also been shown to have counter-examples [138].

While using DNNs makes deep perceptual similarity vulnerable, it also means that it can utilize the same defenses against adversarial examples as other DNNs. One such defense strategy relies on using multiple models and input distortions [90]. Distorting the input might make it different enough that the deep perceptual similarity metrics no longer produce the desired results. *Kettunen et al.*, [3] investigated this and showed that not only can the defense be used, but it also provides a performance boost to LPIPS metrics.

It is also worth noting that the existence of adversarial examples does not necessarily pose as much of a problem for the use of deep perceptual loss. Most opportunities to attack models with adversarial examples exist during inference [139], but loss calculation is typically only used during training. Adversarial examples are additionally rare, so the likelihood that they would be randomly generated by the model being trained is low, having negligible training.

While the existence of adversarial examples is certainly a flaw with deep perceptual similarity metrics, this issue is also prevalent in all other areas of deep learning. In some ways, deep perceptual loss and similarity are better situated to avoid issues with adversarial examples since many applications are used during training and not inference. New defenses developed against general adversarial attacks can likely be implemented for similarity metrics as shown by *Kettunen et al.*, [3]. Additionally, similar flaws can often

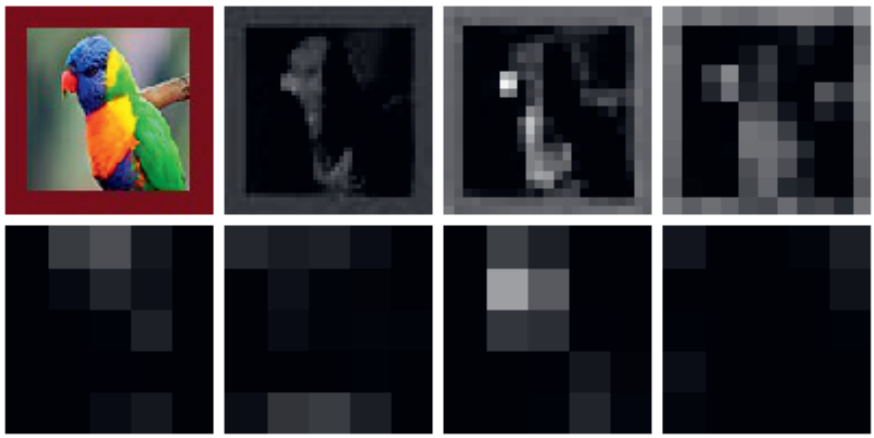


Figure 4.1: An image of a parrot used for analysis by *Sjögren et al.*, [29] (top-left) along with feature maps from channels of SqueezeNet 1.1 [81] extracted at the (top) 1st ReLU, 2nd, 4th, (bottom) 5th, 6th, 7th and 8th Fire modules. Brighter colors indicate higher activations.

be easier to find and exploit in rule-based metrics [138]. Instead of focusing on adversarial examples, the chapter will look for simpler flaws that are related to the method itself.

4.2 Analyzing Loss Networks with Feature Maps

To improve deep perceptual similarity metrics, it is important to understand how they work and why they are successful. By analyzing the process through which similarity scores are successfully created, it might be possible to glean insight into what makes the process successful and when it fails. This sentiment is formalized in the first sub-question of Research Question 2.

Q2.1 Why is deep perceptual similarity able to solve perceptual similarity tasks where pixel-wise metrics fail?

To address both the sub-question and research question, the process of how images are compared by deep features needs to be understood. To achieve this, a qualitative approach is taken where the process of comparing a small set of images is analyzed in depth. The extracted deep features are analyzed to understand how deep perceptual similarity works. For CNNs, such analysis is often done through visualization of feature maps.

Feature maps are the 2D grids of activations produced by each channel in a CNN. By plotting them as an image with color or brightness determined by the strength of activations, the regions of the image for which that channel is activated are visualized.

Figure 4.2: Three pairs of black-and-white images and distortions (top) and the seven reference images with single colors and random noise (bottom) used by *Sjögren et al.*, [29] for analysis of deep perceptual similarity metrics.

An example of this can be seen in Fig. 4.1, where an image of a parrot is shown together with feature maps extracted from the image by an ImageNet pretrained SqueezeNet 1.1 at different depths. In the deeper layers, the spatial resolution of each channel is lowered, which aligns with each pixel representing a larger region of the input image. Feature maps are commonly used as ways to understand CNNs and how it works [140].

CNNs are the most common architectures used as loss networks for deep perceptual loss and similarity. Moreover, the deep features of these CNNs are compared to calculate the loss or similarity. Using feature maps to understand how deep perceptual similarity works, then, is a reasonable approach.

Feature map analysis could be used to understand how deep perceptual similarity can handle cases which other metrics struggle with. Understanding how deep perceptual similarity handles those cases can give insight into what cases it cannot handle. This approach is a starting point for the qualitative analysis of deep perceptual similarity followed in this chapter.

For the initial analysis, a set of 11 black-and-white images with different patterns were created. For each image, a distorted version is created, with inverted colors. To have something to compare the similarity of the black-and-white images against, a series of reference images are created. These reference images consist of black, gray, white, red, green, and blue single-colored images as well as one with random noise. Examples of the black-and-white image pairs and the reference images are shown in Fig. 4.2.

A good metric, one can assume, should consider the inverted versions more similar to the originals than any of the reference images, as they have the same structure. This criterion is used during the analysis to evaluate the success of metrics. A metric has correctly identified an image pair if it considers the original more similar to the distorted version than any of the reference images. Black-and-white inversions were chosen for initial analysis since pixel-wise metrics fail on all those pairs as they, by definition, find them as dissimilar as possible.

The implementation of deep perceptual similarity metrics that are used for the analysis is that by *Zhang et al.*, [2], which is detailed in Section 2.7. These metrics are defined in Eq.4.1, where z^l are the features extracted from the loss network, $l \in L$ are the feature

extraction layers with width W_l, height H_l and channels $1 \leq c \leq C_l$. The features z are unit-normalized along the channel dimension in the original work, meaning that all features that share spatial positions in the feature maps are unit-normalized. Since normalization along the channel dimension is not commonly performed in deep perceptual loss, a version without normalization is also tested. These metrics use the spatial comparison method which compares deep features against each other by comparing features in the same spatial position in each channel against each other. The spatial comparison method is the most commonly used in deep perceptual loss and similarity.

$$d_{spatial}(x, x_0) = \sum_{l \in L} \frac{1}{C_l H_l W_l} ||(z_x^l - z_{x_0}^l)||_2^2 \tag{4.1}$$

Three ImageNet pretrained models were used as loss networks throughout the analysis; AlexNet [79], SqueezeNet 1.1 [81], and VGG-16. The metrics using AlexNet and SqueezeNet consider all 11 inversions as more similar than the reference images, while VGG-16 gets a single image wrong, both with and without unit-normalization. A summary of the performance of each metric can be found later in Section 4.3 and Table 4.1. This shows that deep perceptual similarity can mostly handle inversions, and analysis of the feature maps gives insight as to how. An image and its inversion from the set are shown in Fig. 4.3 along with the feature maps from the two images. The feature maps are extracted from SqueezeNet at various layers of increasing depth. The rightmost column of each subfigure shows the difference between the left and middle columns, which visualizes how the similarities and differences between feature maps as measured by spatial similarity.

Fig. 4.3 shows that the earliest layers detect many differences between the images, while the differences are barely visible in later layers. Additionally, the deeper layers do not activate strongly from the image, likely because the image is quite plain and does not have any higher-level features that make up the general images in ImageNet. As the deeper layers locate features that are more agnostic to color, such as shapes and contours, deep perceptual similarity will detect similar features in the inverted images, which also have these structures. Since the features in the deeper layers take information from many pixels in a region, they also abstract away some of the pixel-perfect positioning that pixel-wise metrics rely on. Similar patterns were observed in the other image pairs in the set. However, it is notable that even at deeper layers, the feature maps still correspond with spatial regions of the image.

A set of images for which pixel-wise metrics are unable to correctly predict the perceptual similarity can be created by understanding how the similarity is calculated. It might be possible to create a similarly difficult set for deep perceptual similarity by understanding how it works. This idea is formalized in the second sub-question of Research Question 2. The observation that the features detected, even in later layers, are strongly spatially correlated provides a starting point for finding such counterexample images.

Q2.2 How can counterexamples be generated where current deep perceptual similarity metrics fail to assess perceptual similarity correctly?

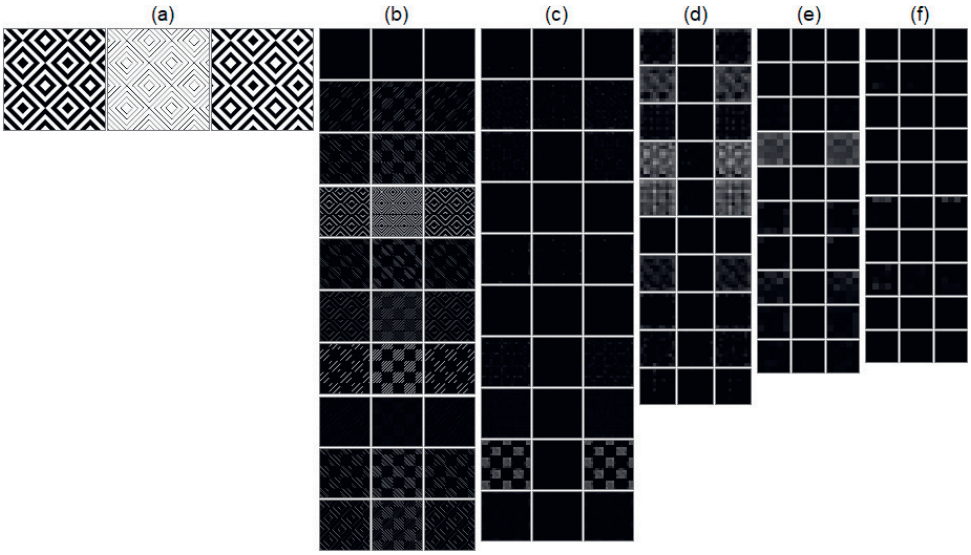


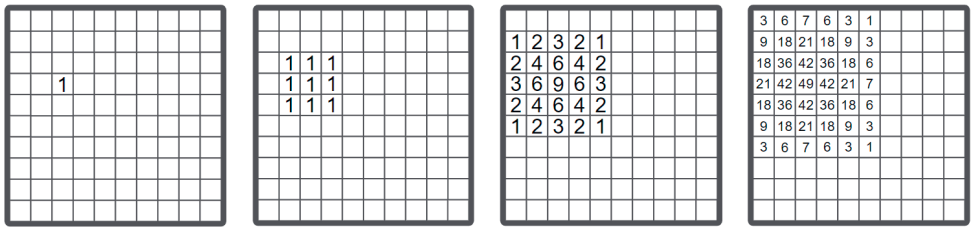
Figure 4.3: An image (left) and its inversion (right) from the black-and-white images analyzed by *Sjögren et al.*, [29] (a) along with some feature maps from both extracted from an ImageNet pretrained SqueezeNet after the 1st ReLU (b) and 2nd (c), 4th (d), 6th (e), and 8th fire block. Brighter colors indicate stronger activations in the feature maps. The middle column contains the absolute difference between the images or feature maps.

4.3 Flaws with the Spatial Comparison Method

The strong spatial correlations between input images and feature maps are not surprising since CNN architectures are typically designed such that each feature only depend on a small specific region of the input or previous layer. Even in deeper layers, which aggregate information from almost the entire image, the pixels in the central part of those regions have more impact on the features as visualized in Fig. 4.4.

The strong spatial correlation might indicate flaws in the spatial comparison method for deep perceptual similarity since a part of the image that has been moved would now cause activations that do not overlap in the feature maps. Such non-overlapping activations might result in a lower deep perceptual similarity evaluation than simply comparing to an image that does not cause those activations at all. To evaluate this, *Sjögren et al.*, [29] create two more sets of image pairs consisting of images that have been translated and rotated, distortions which cause regions of the image to change position.

To explicitly evaluate the possible issues with features not overlapping in the feature maps, the images in the translated set have a region with much structure and are otherwise plain. The image is then paired with a distorted version where the structured region has been translated so it is in a completely different part of the image. The image pairs in the rotate set consist of images and distorted versions that have been rotated in



(a) Input image (b) Layer 1 (c) Layer 2 (d) Layer 3

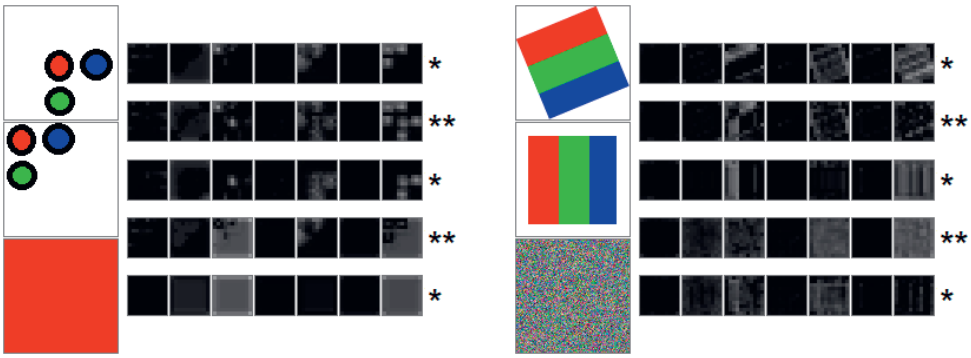
Figure 4.4: The number of times the marked pixel in the input image (a) comes into consideration for calculating the features centered at a given pixel in three consecutive convolutional layers (b-d) in a CNN. The CNN has kernels of size 3×3 and stride 1. Even though later features take pixels further away into account, they weigh the impact of spatially close pixels higher.

Table 4.1: The number of distorted image pairs in the qualitative analysis by *Sjögren et al.*, [29] that each perceptual similarity metric correctly identifies as being more similar than random noise and single-color images.

Method	Channel Norm	Network	Invert	Translate	Rotate
Pixel-wise			0/11	0/5	17/30
Spatial	✓	AlexNet	11/11	0/5	26/30
	✓	SqueezeNet	11/11	0/5	25/30
	✓	VGG-16	10/11	0/5	25/30
Spatial		AlexNet	11/11	0/5	11/30
		SqueezeNet	11/11	0/5	20/30
		VGG-16	10/11	0/5	6/30

steps of 22.5 degrees up to 90 degrees, as well as one rotated 180 degrees. A good metric would consider the original image to be more similar to the distorted versions compared to some reference images consisting of single colors or noise. Examples of an image pair from the two sets and reference images are shown in Fig. 4.5b, along with their feature maps and the absolute difference between those feature maps.

The same models (AlexNet, SqueezeNet, and VGG-16) with the same spatial comparisons as before (Eq. 4.1) perform poorly on these sets. For translation, all three models fail to consider any of the five translated images to be more similar than the reference images. For rotation, only one of the loss networks performs better than SqueezeNet and only barely. The results from the two sets and the inversions can be found in Table 4.1, which shows how many image pairs from each set the spatial metrics identified as more similar than the reference images with pixel-wise results for comparison.



(a) (b)

Figure 4.5: An image (middle), a distorted version of that image (top), and a reference image (bottom) from the translate (a) and rotate (b) sets used for analysis by *Sjögren et al.*, [29]. To the right of the images are rows of their feature maps (*) from the 4$^{\text{th}}$ layer of an ImageNet pretrained SqueezeNet. Between the rows of feature maps are rows of the absolute difference between those feature maps (**). The difference between the feature maps of the image and its distortion does not seem less compared to the feature maps of the references.

Feature map analysis of the images in the translate and rotate sets confirm that the lack of overlap between parts of the image that cause activations will cause the difference between the feature maps to be greater. In some cases, this difference is greater than compared to images that cause weak or strong activations in the entire feature map, such as single-colored images or random noise. This can be seen in Fig. 4.5b, where the feature maps of two images are compared against those of the translated and rotated versions. In addition, the feature maps of the images are compared against plain and noisy reference images. The difference between the feature maps seems to be about the same when compared to the distorted version, as they are against the references. This is likely why the images and their distortions are seen as less similar than the images and references by all six tested deep perceptual metrics.

The inability of deep perceptual similarity with spatial comparisons to handle image translations has also been observed by *Ghildyal and Liu* [55]. The work builds on a longer trend of research into making CNN architectures shift invariant. Spatial LPIPS metrics would change which distorted image was perceived as the most similar to the original after being shifted by a single pixel, which is almost imperceptible to humans. This aligns with previous research that has found CNNs to be vulnerable to small transformations [141]. To verify this, a dataset of JND human judgments was gathered on whether humans could notice small shifts from 0 up to 9 pixels of an image. The findings showed that in a majority of cases, the subjects would fail to notice shifts of 1 or 2 pixels. Another dataset was then created by taking the 2AFC part of the BAPPS dataset and shifting the distorted image that had been judged most similar by 1, 2, and 3 pixels. The LPIPS

metrics used by *Zhang et al.*, [2] were shown to have significantly reduced performance on the shifted 2AFC dataset.

The issues with spatial metrics, even if they account for higher-level structures, have also been noted by *Ding et al.*, [4]. The work shows how spatial metrics lack a general description of textures. This is highlighted by showing how images, in which a patch of a texture has been replaced by a different patch of the same texture, are considered less similar by spatial metrics than images with more visible distortions.

The existence of flaws in the most common implementations of deep perceptual loss and similarity leads one to wonder if those flaws could be mitigated. This is what is asked by the third sub-question of Research Question 2.

Q2.3 How can deep perceptual similarity be improved to mitigate existing flaws?

The works by *Ghildyal and Liu* [55] and *Ding et al.*, [4] both identify flaws with the CNN architectures as causes of these issues, and both suggest some changes to fix them. Kernels with large stride (> 1) can cause the sampling rate of the channel to fall under the Nyquist criterion, which means that some information is lost even before it is analyzed. MaxPooling is also a practice that can cause the sampling rate to fall, which is why both works adopt other pooling operations such as ℓ_2 pool [142] and BlurPool [143]. In addition, *Ghildyal and Liu* [55] evaluate full convolution [144], where the input is padded so that each kernel element is applied to each input at least once. *Ding et al.*, [4] also introduced changes to the comparison methods used to evaluate the difference between the extracted features to address the texture replacement issue. These comparison methods and others are introduced and analyzed in the following section.

4.4 Flaws with Non-spatial Comparison Methods

Several comparison methods exist beyond the spatial one, some of which have even been introduced specifically to handle the shortcomings of spatial comparisons. *Sjögren et al.*, [29] propose that global methods that aggregate the features of each channel in ways that remove spatial information could remove the issues of similar but non-overlapping regions. For this purpose, they propose to use the mean comparison method (Eq. 4.2) that was evaluated previously by *Kumar et al.*, [67] as well as introduce a new comparison method where the features of each channel are sorted by activation strength before comparison (Eq. 4.3). The mean and sort comparison methods are shown in equations Eq. 4.2 and 4.3, where z_x^l are the activations, which may have been normalized along the channel dimension, in layer l from a loss network with input x and extraction layers $l \in L$. $\overline{z}$ and $z^{\downarrow}$ are the average and descending reordering of the channels in z respectively.

$$d_{mean}(x, x_0) = \sum_{l \in L} \frac{1}{C_l} ||\overline{z}_x^l - \overline{z}_{x_0}^l||_2^2 \tag{4.2}$$

$$d_{sort}(x, x_0) = \sum_{l \in L} \frac{1}{C_l} ||z_x^{l\downarrow} - z_{x_0}^{l\downarrow}||_2^2 \tag{4.3}$$

The mean comparison uses the difference of the average activation for each channel rather than each individual feature in that channel. Even if a region of the image has been translated, it still impacts the average in the same way, which should result in images with the same content at different places getting similar average channels. However, by averaging the channels, information is lost, which can result in two dissimilar feature maps having the same average. Sort comparisons are proposed to handle this issue. By sorting the channel and using the difference between the strongest activations, 2^{nd} strongest activations, etc., the distribution of activations is maintained, but where those activations occur in the feature map does not matter for similarity calculation.

Sjögren et al., [29] analyze these two comparison methods with the previously mentioned invert, translate, and rotate sets of images. Both methods have significant improvements over spatial comparisons and have close to perfect recognition on the three sets. However, analysis of the feature maps reveals that many channels activate strongly for many types of structures, color changes, and textures. Since the global comparison methods do not consider the exact position of activations, objects with similar overall structures and textures could give rise to similar activation distributions. Also, noise and color added to an image could give rise to activations that would disrupt the distribution of activations. This could lead to cases where global comparison methods would consider two images to be similar because they exhibit similar styles and textures and are dissimilar simply due to altered background colors or light stains.

To test whether the mean and sort comparison methods are vulnerable to changes in background color and stains *Sjögren et al.*, [29] created a color stain image set. This set consists of images with a region containing structures, such as a photo of an animal or some icons, and two distorted versions of that image, similar to a 2AFC dataset. Examples of these triplets are shown in Fig. 4.6. Instead of gathering human judgments on which of the two should be considered more similar, one of the images is intentionally made to be dissimilar. The version of the image that could be considered more similar to the original has the same region of interesting structure but with a change in background color and possibly added stains. The other distorted versions, which are made to be dissimilar, have the same unstained background, but the interesting background has been changed to something different, such as random noise or another icon entirely. Table 4.2 shows how many image pairs of each set are identified as more similar to each other than the references of that set for spatial, mean, and sort comparison methods, with and without channel-normalization.

The results show that while the mean and sort comparison methods can handle the translate and rotate cases well, they struggle with the color stain images. It is notable that using normalization along the channel dimension causes all three methods to perform worse on the color stain dataset. Likely this is due to normalization taking the stain activations into account, which means that the activations from actual similarities are lowered in one image compared to the other. This can cause differences even when the features give similar activations for similar regions.

Other comparison functions have been proposed by a number of works. *Gatys et al.*, [93] introduced both the spatial comparison method for content loss and a style

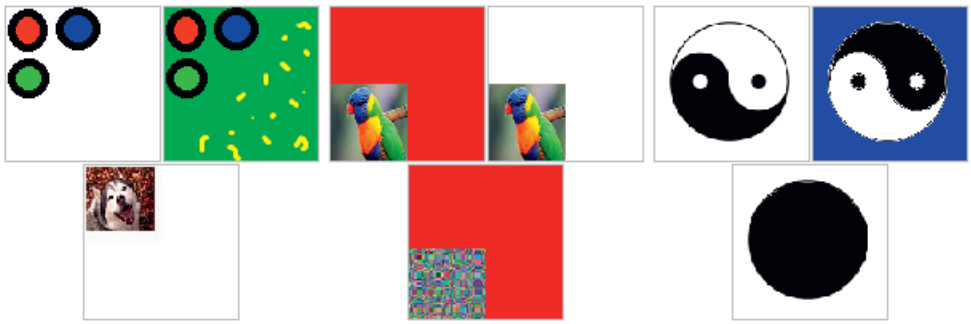


Figure 4.6: Three image triplets used by *Sjögren et al.*, [29], from which the figure originates, to analyze the vulnerability of deep perceptual loss to color stains. The triplets consist of an image with a visually distinct region (left), an image with a similar distinct region but with different color or noise in the background (right), and a reference image (bottom) with a different distinct region in the same place with the same background. Metrics are considered successful if the first image (left) is considered more similar to the second (right) than the reference (bottom). Reproduced with permission.

comparison method for style loss. The style comparison method aggregates channel activations by calculating the inter-channel cross-correlation matrices for the extracted features of each layer and comparing those. The style comparison method is described by Eq. 4.4, where z_x^l are the activation in layer l of a loss network with input x and extraction layers $l \in L$.

$$d_{style}(x, x_0) = \sum_{l \in L} \frac{1}{C_l^2} ||z_x^l \cdot (z_x^l)^T - z_{x_0}^l \cdot (z_{x_0}^l)^T||_2^2 \tag{4.4}$$

Ding et al., [4] also split the deep perceptual similarity metric into one comparison method for structure and another for content. This is similar to how *Gatys et al.*, [93] split it into one for content and another for style. In fact, the style comparison method explicitly builds on a method aimed at capturing texture [145]. However, *Ding et al.*, [4] argue that the huge number of statistics that are being compared against each other in the cross-correlations of the style method is excessive for the task of texture similarity. Instead, Simply comparing the means of the channels as in the mean comparison method is shown to be an effective metric for texture similarity. It is suggested that the structure similarity can be measured using the variances of the features in each channel along with the covariances between the features of the corresponding channels for the two images. Furthermore, it is argued that using ℓ_p norms to compare the features, as previous works have done, is not suitable since those norms assume that the features are independent of each other, which is not the case for CNN features. Instead, inspiration is taken from the rule-based SSIM for the formulas of texture and structure similarity. The texture and structure comparison methods are described in Eqs. 4.5 and 4.6, where $\overline{z}_x^{lc}$ and $(\sigma_x^{lc})^2$ are the average and variance of activations in layer l and channel c from a loss network

Table 4.2: The number of distorted image pairs for each set in the qualitative analysis by *Sjögren et al.*, [29] that each perceptual similarity metric correctly identifies as being more similar than the reference images of that set.

Method	Channel Norm	Network	Invert	Translate	Rotate	Color stain
Pixel-wise			0/11	0/5	17/30	0/5
Spatial	✓	AlexNet	11/11	0/5	26/30	0/5
	✓	SqueezeNet	11/11	0/5	25/30	0/5
	✓	VGG-16	10/11	0/5	25/30	0/5
Spatial		AlexNet	11/11	0/5	11/30	3/5
		SqueezeNet	11/11	0/5	20/30	5/5
		VGG-16	10/11	0/5	6/30	4/5
Mean	✓	AlexNet	11/11	5/5	30/30	1/5
	✓	SqueezeNet	11/11	5/5	30/30	1/5
	✓	VGG-16	11/11	5/5	30/30	2/5
Mean		AlexNet	11/11	5/5	28/30	2/5
		SqueezeNet	11/11	5/5	29/30	3/5
		VGG-16	10/11	5/5	30/30	1/5
Sort	✓	AlexNet	11/11	5/5	30/30	0/5
	✓	SqueezeNet	11/11	5/5	30/30	1/5
	✓	VGG-16	10/11	5/5	30/30	0/5
Sort		AlexNet	11/11	5/5	30/30	2/5
		SqueezeNet	11/11	5/5	28/30	4/5
		VGG-16	11/11	5/5	30/30	3/5

with input x and extraction layers $l \in L$. σ^{lc}_{x,x_0} is the covariance between the activations extracted from x and x_0 at layer l and channel c. w^{lc} is a positive scalar that can be learned through training or set to 1, and *epsilon* is a small constant to avoid issues when the denominator is close to 0.

$$d_{texture}(x, x_0) = \sum_{l \in L} \sum_{c=1}^{C_l} w^{lc} \frac{2\overline{z}^{lc}_x \overline{z}^{lc}_{x_0} + \epsilon}{(\overline{z}^{lc}_x)^2 + (\overline{z}^{lc}_{x_0})^2 + \epsilon} \tag{4.5}$$

$$d_{structure}(x, x_0) = \sum_{l \in L} \sum_{c=1}^{C_l} w^{lc} \frac{2\sigma^{lc}_{x,x_0} + \epsilon}{(\sigma^{lc}_x)^2 + (\sigma^{lc}_{x_0})^2 + \epsilon} \tag{4.6}$$

While structure comparison does not strictly work the same as the previously introduced spatial comparison, it does compare each feature element-wise by spatial position and is therefore susceptible to some of the issues as spatial comparisons. On the other

hand, the global methods described in this section are not free from flaws. The color stain problem showed earlier for mean, and sort comparisons are likely to extend to any global metrics where entire channels can get strongly activated by noise that has little impact on similarity. While it is possible that the style and texture comparisons are less vulnerable to color stains, there are other issues with making global comparisons. For example, while the spatial metrics are oversensitive to translation, the global metrics are, in theory, invariant instead. However, invariance is not desired either, as humans do actually notice if images move [55] and likely consider images to be less similar the more they have been shifted. Then, if both spatial and global metrics seem to have complementary flaws, why not try to use them together?

4.5 Using Multiple Comparison Methods

Both *Gatys et al.*, [93] and *Ding et al.*, [4] use a combination of spatial and global methods. *Gatys et al.*, [93] make the difference between the two explicitly by using the spatial method to determine the content of the image and the global method for the style. This seems intuitive, as content depends on the exact objects and their position, while style is typically uniform over an image. *Ding et al.*, [4] combine the texture and structure comparisons that they introduce by adding them together to form the so-called Deep Image Structure and Texture Similarity (DISTS) metric. They show that the combined method is competitive for the classification and retrieval of textures. Importantly, DISTS metrics are shown to be insensitive to small distortions, such as translations and rotations, that humans do not always notice. This suggests that DISTS metrics can handle distortions that would typically be problematic for spatial methods but does not provide the same assurance for distortions that would be difficult for global methods.

Sjögren et al., [29] also analyze combining spatial and global methods by evaluating the combinations of the spatial method with each of the mean and sort methods. The methods are combined by a simple sum and are each referred to as spatial+mean and spatial+sort. These combined metrics, with and without unit-normalization along the channel dimension, are assessed on the same set of analysis images as the others, the result of which can be found in Table 4.3. The combined metrics that use normalization perform poorly on the color stain set, which is not surprising as none of the stand-alone metrics with normalization could either. Besides that, the combined metrics seem to act as spatial metrics when normalization is used and as global metrics without it. This binary behavior means that the benefits of combination do not materialize. Likely this happens since either the spatial or global metric has a larger amplitude than the other and thus dominates it when it comes to the actual similarity calculation.

The issues with combined metrics encountered by *Sjögren et al.*, [29] could possibly have been mitigated by multiplying one of the metrics with a scalar to balance the two metrics. Using scalars to weigh the different terms of the sum is common in machine learning when summing different model outputs, losses, or other measurements. These scalars are ideally tuned to find a good balance between the terms and achieve better

Table 4.3: The number of distorted image pairs for each set in the qualitative analysis by *Sjögren et al.*, [29] that each combined metric correctly identifies as being more similar than the reference images of that set.

Method	Channel Norm	Network	Invert	Translate	Rotate	Color stain
Spatial+mean	✓	AlexNet	11/11	3/5	29/30	1/5
	✓	SqueezeNet	11/11	4/5	29/30	0/5
	✓	VGG-16	11/11	4/5	30/30	0/5
Spatial+mean		AlexNet	11/11	0/5	15/30	2/5
		SqueezeNet	11/11	0/5	22/30	5/5
		VGG-16	10/11	0/5	9/30	4/5
Spatial+sort	✓	AlexNet	11/11	2/5	28/30	0/5
	✓	SqueezeNet	11/11	2/5	27/30	1/5
	✓	VGG-16	10/11	3/5	29/30	0/5
Spatial+sort		AlexNet	11/11	0/5	19/30	4/5
		SqueezeNet	11/11	0/5	22/30	5/5
		VGG-16	11/11	1/5	21/30	4/5

results. While the scalars are sometimes tuned by hand in many cases they are learned during training of the model. The DISTS metric employs this strategy by training the scalars w present in both the texture (Eq. 4.5) and structure (Eq. 4.6) metrics. The scalars learned for the DISTS metrics do not only scale texture and structure metrics against each other but are used to also scale each channel in the loss network as well which gives more weight to channels that are better aligned with perceptual similarity as defined by the training set. Since a set of scalars for each channel is learned for both composing metrics, channels that are beneficial in one of the metrics can be scaled up for that one specifically. This type of training is similar to what is done with LPIPS metrics [2]. The downside of learning scalars is, of course, that a dataset is required and that the metric learned will be specific to that dataset. Though, as is covered in Chapter 5, any metric will always perform better on some tasks than others.

Another practice used by *Ding et al.*, [4] is to compare the pixels in addition to the features. In the work, it is considered as feature extraction from the 0^{th} layer of the loss network, and then the comparison methods are applied as normal for that layer. This practice is already commonplace in the application of deep perceptual loss [13, 14, 97] where it is used to ensure that the output resembles the desired output on the pixel level in addition to the higher level structures that are captured by the deep features.

Table 4.4: The 2AFC score on BAPPS and its subdivisions of the deep perceptual similarity metrics with different comparison methods and loss networks evaluated by *Sjögren et al.*, [29].

		Distortions			Real Algorithms					2AFC	JND
Method	Network	Traditional	CNN-based	Avg.	Super-res	Video Deblur	Colorization	Frame Interp	Avg.	Avg.	JND
Human	-	80.8	84.4	82.6	73.4	67.1	68.8	68.6	69.5	73.9	-
Spatial	Squeeze	73.3	82.6	78.0	70.1	60.1	63.6	62.0	64.0	68.6	60.2
	AlexNet	70.6	**83.1**	76.8	**71.7**	**60.7**	65.0	62.7	65.0	68.9	57.6
	VGG-16	70.1	81.3	75.7	69.0	59.0	60.2	62.1	62.6	67.0	59.1
Mean	Squeeze	77.1	82.3	79.7	69.9	60.0	65.2	63.1	64.5	69.5	63.6
	AlexNet	73.9	82.8	78.4	71.4	**60.7**	**65.5**	**63.5**	**65.3**	**69.6**	60.2
	VGG-16	77.9	81.8	**79.8**	68.9	59.5	64.0	63.0	63.8	69.2	**65.2**
Sort	Squeeze	76.8	82.0	79.4	69.8	60.1	64.6	61.9	64.1	69.2	62.0
	AlexNet	73.3	82.8	78.0	71.1	60.6	64.6	62.6	64.7	69.2	58.5
	VGG-16	**78.1**	81.5	**79.8**	68.1	59.2	62.7	61.5	62.9	68.5	64.8
Spatial+mean	Squeeze	75.0	82.5	78.8	69.9	60.1	64.5	62.1	64.2	69.0	61.5
	AlexNet	71.8	83.0	77.4	71.6	**60.7**	**65.5**	62.7	65.1	69.2	58.5
	VGG-16	73.4	81.9	77.7	69.3	59.4	64.5	62.5	63.9	68.2	61.0
Spatial+sort	Squeeze	75.5	82.5	79.0	70.0	60.1	64.4	61.9	64.1	69.1	61.2
	AlexNet	72.2	**83.1**	77.7	71.3	60.6	64.9	62.8	64.9	69.2	58.5
	VGG-16	74.9	81.9	78.4	69.4	59.4	62.3	62.1	63.3	68.4	61.9

4.6 Evaluation on Human Judgements

Sjögren et al., [29] also investigate how the insights gained from the analysis impact performance on actual human judgments of similarity. To do this, the three stand-alone metrics and two combined metrics with normalization along the channel dimension are evaluated on the BAPPS dataset. The evaluation is performed on both the 2AFC and JND parts of the dataset. The scores for each subdivision of the 2AFC part are gathered to analyze the impact on different distortions since the 2AFC part of the BAPPS dataset is subdivided according to which types of distortions are applied. The results of this evaluation can be found in Table 4.4.

Despite the flaws shown throughout this chapter, spatial deep perceptual similarity metrics still achieve good performance on BAPPS. Averaged over the 2AFC part, the spatial metrics only trail the mean and sort metrics by about 1.0. This can, to a large degree, be attributed to that a few of the distortions applied in the dataset significantly affect the position of the features in the image. The assumption that the images that are being compared are aligned is something that many existing metrics rely on for performance [146] and which many similarity datasets provide. Looking at which subdivision of the 2AFC data that the spatial metrics perform relatively worse reveals the previously observed flaws. On the traditional distortions, which include distortions like translation

and rotation, the spatial metrics are around 4.0 points worse than the global metrics. The other subdivisions are outputs of algorithms which likely have spatially aligned outputs.

The use of unit-normalization along the channel dimension for the evaluations on BAPPS might have made the mean and sort metrics worse. Both metrics performed better without normalization on the previous analysis, and this might have translated to even better performance on BAPPS. It is also worth mentioning that the combined metrics outperform spatial metrics but not mean and sort.

4.7 Discussion and Future Developments

While the global comparison methods outperform their spatial counterparts on BAPPS, they still have flaws and perform worse than spatial metrics in some cases. This discrepancy between the performance of different comparison methods shows that there is plenty remaining to explore when it comes to the calculation of deep perceptual similarity. Some flaws of spatial comparison methods have been solved by changing the architecture instead of the method itself [55]. This shows that further improvements can be made both within the comparison methods and the entire makeup of deep perceptual similarity.

The primary strategy adopted to deal with the flaws of spatial and global comparison methods is to combine the two in hopes that each will make up for flaws in the other. This strategy seems to be working in some cases but requires either careful balancing between the two metrics, either through engineering efforts or training. It is interesting to consider whether a deep perceptual similarity metric could exist that is neither spatial nor global. Instead of combining the two, find a metric that solves the desired problem on its own. In a sense, this is already how spatial metrics work. Deep features of CNNs aggregate pixels covering a larger space of the input image, which means that spatial comparison methods, to some degree, do not care for the exact spatial positions. However, as shown in Section 4.3, deep features, even in later layers, are more affected by a small spatial region of the image. Perhaps modifying the typical CNN architecture to be less spatially dependent or using another architecture altogether will alleviate some of the flaws with spatial metrics.

Beyond improving deep perceptual similarity, it is interesting to consider how these flaws impact deep perceptual loss. Beyond style transfer, deep perceptual loss currently uses almost exclusively spatial comparison methods. Despite the known flaws and additions of other comparison methods, these developments have not been introduced to deep perceptual loss. However, many deep perceptual loss applications like segmentation, super-resolution, style transfer, and depth prediction are highly dependent on the exact spatial positioning of the output. For this reason, some such applications also use pixel-wise losses in addition to deep perceptual loss [13, 3, 4]. On these problems using only global comparison methods would likely be problematic as the models might not be able to learn the correct spatial placement of the output. On the other hand, combined metrics can potentially be successful, as they have been for style transfer [93]. Other applications of deep perceptual loss, such as image synthesis from text or features, are not as dependent on spatial information and might therefore be possible to learn with only

global comparison methods. Though, applications of deep perceptual loss, in general, seem to benefit from adding spatial and even pixel-wise losses to refine the output [14].

Another issue in evaluating comparison methods is that most datasets for evaluating deep perceptual similarity are, to a large degree, spatially aligned. Such alignment benefits spatial metrics as they would have significantly decreased performance on images where the corresponding regions are not overlapping. In reality, it would seem like images would not tend to overlap, but many of these datasets use images from real-world applications like image compression and algorithm outputs, which tend to be spatially aligned. The existing datasets are also likely to benefit global metrics in other ways. The dataset tends to compare images with a slightly altered version of the same image. As indicated by the color stain analysis, global metrics are more likely than spatial metrics to mistake completely different images as being more similar. This flaw is not as prominent when evaluating metrics on datasets that do not compare images with different content to each other.

Perhaps a new set of datasets that evaluate deep perceptual similarity metrics under new circumstances are needed. Even if other such datasets are used, it would not change the fact that some metrics are better than others on the existing datasets and the applications they evaluate. Instead, using different metrics in different contexts is a potential path forward. New datasets and metrics for particular contexts are both analyzed further in Chapter 5.

Chapter 5

Overcoming Ambiguity in Similarity

It has long been established that the notion of perceptual similarity is inherently ambiguous [104]. Even a single individual changes their perception of similarity to the context of the situation. Whether two images are similar can have different answers if the question is posed when discussing artistic style or when examining life-likeness. This is visualized in Fig. 5.1, where the middle image has a similar style to the right image and portrays a similar subject to the left image. Any metric, by definition, will give an answer to which image is more similar. If the metric gives the same answer, regardless of context, it would seem that in certain scenarios any metric would have to go against the human perception. This is the foundation for the third research question of this thesis.

Q3 To what extent can deep perceptual similarity handle the inherent ambiguity of similarity?

5.1 No Free Lunch in Similarity

The idea that for any similarity metric a context can be found in which it performs poorly is analogous to the arguments made in the "no free lunch"-theorems [37]. The "no free lunch"-theorems establish that, when averaged over the set of all optimization problems, all methods have equal performance. The proof of this builds on the set of all problems containing every permutation of desired predictions and therefore any model will be wrong on as many of those permutations as all others. For example, a model might learn to predict "Monday" on the problem of predicting the first day of the week, but the set of all problems also contains problems where the desired prediction for the first day of the week is any of the other days. Many of the problems contained in the set of all problems are not relevant to society in anything but a theoretical capacity.

The "no free lunch"-theorems show that no optimization method is inherently better than another. Instead, the usefulness of a method depends on the purpose of its use and the problems that it is applied to. The purpose of using a given method is therefore equally relevant to its actual performance.

Figure 5.1: Three images from the GPR1200 dataset [52] where the middle image is a black-and-white sketch portraying a bird. The left image is a photograph of a bird and the right image is a black-and-white sketch of a platypus. In the context of content, the left image would be considered more similar to the middle, while in the context of style, the right image would be considered more similar.

The theorems have the same implications in the domain of similarity metrics. On the set of all possible definitions of what is similar, all similarity metrics have the same average performance. Similarly, it also demands that the purpose of a metric is defined to be able to assess its actual usefulness. Which definitions of similarity are relevant depends on that purpose.

Measuring similarity according to human perception, perceptual similarity is one such purpose. Having metrics that achieve that purpose with high performance is also likely to be beneficial to society and, therefore, a useful endeavor.

Limiting the similarity definitions to those that align with human perception for some populations and contexts means that it is possible to find metrics that outperform all others on the specific problems. However, as was shown in Fig. 5.1, ambiguity still exists, and a metric that works well for one subdivision of perceptual similarity might be terrible at another. Recall the example of the model predicting the first day of the week. Even when limited to what humans think of as the first day of the week, any day from Friday through Monday might be correct, depending on who you ask. While a model that is best in the average context can be found, selecting a model appropriate to the given context would seem better. A model predicting "Sunday" would work better in North America, and a model predicting "Monday" would work better in Europe.

While there exist metrics that perform better on the average perceptual similarity problem, it also seems like high performance in some contexts leads to low performance in others. Further improvements to perceptual similarity metrics could be possible if the issues related to the ambiguity of similarity were better understood. This leads to the first sub-question of Research Question 3.

Q3.1 How does ambiguity in similarity pose a problem for perceptual similarity metrics?

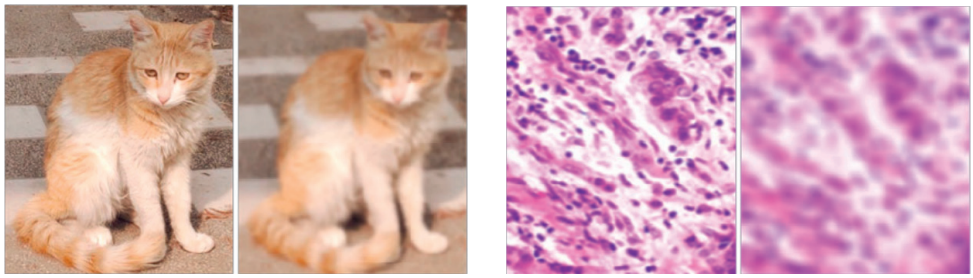
5.2 Ambiguity and Adaptability

To tackle the issue of ambiguity over all the tasks that belong to perceptual similarity, specific subsets could be defined for which metrics could be created. To be useful the subsets would need to represent useful real-world problems, to be specific enough that performance could be improved on that subset, and general enough that a metric that performs well on the problems is likely to perform well on unseen data related to the real world problems. Such subsets will be referred to as contexts, and potentially interesting contexts to solve for perceptual similarity include style similarity, event similarity, content similarity, etc.

One context that is well-represented in perceptual similarity datasets is the context of difference under algorithmic distortions. Such datasets compare images to themselves under different distortions that mostly preserve the original image. In most samples of such datasets, it is easy for humans to tell that the alternative images are distortions of the same original image. These datasets represent the majority of human judgment datasets of perceptual similarity. As such it is difficult to analyze perceptual similarity metrics for other contexts.

One field that often makes use of perceptual similarity where the issue of ambiguity has been raised is content-based image and video retrieval [147, 148]. Query by example, where an image or video is searched for using an example image, often uses similarity metrics to compare the example to the entries in the database. Here the user's intent can result in different desired outcomes from the same query. For example, a user might be looking for images with the same subject or style as in the query image. In text-based querying, which also has to handle ambiguity in queries, efforts to solve this ambiguity by using user-specific data have been undertaken [149]. For image queries, interactive systems that help the user clarify their intent and thereby define the similarity context have been developed [147]. Another retrieval application where ambiguity is specifically problematic is query by sketch, in which users make quick sketches of what they are interested in finding. In query-by-sketch, for example, most people would probably agree that another sketch is the most similar image, but in the context where the user can only query using sketches a better similarity metric would likely be looking for the most similar non-sketch images. A work on video retrieval through query-by-sketch has suggested the approach of retrieving content for all likely contexts for the user to choose from [148]. However, all of the mentioned approaches for handling ambiguity in querying are rule-based and specifically designed for their given applications. This might make them unsuitable for use beyond image and video retrieval applications and likely difficult to adapt to new contexts.

Deep perceptual loss is another application where having metrics adapted to specific contexts could be beneficial. As was covered in Chapter 3, it has been shown that a loss network that performs well on perceptual similarity does not necessarily perform well as a loss network [32]. It is better to evaluate loss functions by the performance of the models trained with them than by how well they agree with human perception of similarity. Thus being able to adapt metrics to the tasks for which the model is being



(a) (b)

Figure 5.2: The original and blurred versions of (a) a cat photograph and (b) a medical microscopy image. While the cat is still identifiable after blurring, the details of the microscopy image needed for diagnosis are almost completely removed. The images originate from the work by *Prakash Chandra Chhipa* [82]. Reproduced with permission.

trained is likely beneficial. For example, in the analysis of medical images, the human perception of similarity between animals might not apply. An image of a blurred cat will likely be recognized as a cat, but blurring a medical image could completely remove the relevant structures needed to detect the presence of certain diseases [82].

Since there exist applications where the ambiguity of perceptual similarity is an issue, developing metrics that can handle those issues is beneficial The definition of a similarity metric as a function $d(x, x_0)$ that takes two input images and gives a similarity score is insufficient for a single metric to be able to handle different similarity contexts. Two immediate solutions to this problem are available. The first is to alter the similarity metrics to also take the context into consideration as a new input or parameter. The second is to create one metric for each relevant context. The latter option is simpler since there is no need to formalize how to give context as input. Instead, each context needs an evaluation method to assess each metric for that context. Such evaluation methods are already known as the evaluation used for perceptual similarity datasets could be used. Though gathering such datasets can be time-consuming, especially if there are many different relevant contexts. Additionally, if the problem is not limited to a discreet number of known contexts, this approach is impossible.

For the approach of using one metric for each context to be viable, though, the ability to create metrics for those contexts is relevant. Rule-based metrics would likely be a poor choice as each metric would have to be engineered specifically for the given context. While many existing rule-based methods have hyperparameters that could be altered to fit a given context, their hyperparameters can be limited and unable to handle certain contexts. Additionally, such hyperparameters would also have to be decided through engineering or found through resource intense exhaustive search. Deep perceptual metrics, on the other hand, seem specifically suited to this task. They build on neural networks which are highly adaptable and can, under the correct circumstance, estimate any function with real-valued inputs and outputs [150]. It is feasible that a deep perceptual

similarity metric could be adapted to a given context by swapping which loss network is used or by fine-tuning the existing network. As such, deep perceptual similarity is a good candidate for creating adaptable perceptual similarity metrics.

That the loss networks should, in theory, be able to adapt to a given context does not necessarily mean that this is practical or even achievable. As covered in Chapter 4 CNN, deep perceptual loss and similarity have known flaws that make them vulnerable to certain distortions. While some of these flaws can be mitigated, there may exist contexts that cannot be handled by training a specific metric.

Another problem with training a metric for each context is that this could be very resource intensive. Deep perceptual loss and similarity commonly use loss networks pretrained with ImageNet [64]. Training a new metric on ImageNet for each context would consume considerable resources. Though, it is possible that no major retraining is needed. The loss networks learn a large number of features during pretraining and different subsets of those may be useful for different contexts. The issue then is instead that all features are applied with the same weight to all contexts. In this case, rather than retraining the loss network entirely, a layer of scalars could be learned for the extracted features to adapt the metric for the given context as was described in Subsection 2.7.2. *Zhang et al.*, [2] showed that training such scalars can improve performance on perceptual similarity. They also showed that training scalars generalize better to new data distributions than training the entire loss network. Though, the trained metrics only barely outperform baseline metrics using pretrained loss networks without extra training. Additionally, the extra training was performed in the same context that the metrics already worked well on. As such, the results do not show if deep perceptual similarity metrics can be adapted to new contexts. Especially those contexts in which the baseline metrics would struggle, which are the cases where adaption could be most beneficial.

The issue of whether there is a practical method for adapting deep perceptual similarity metrics to a given context is captured in the second sub-question of Research Question 3.

Q3.2 How can deep perceptual similarity metrics be adapted to different definitions of similarity?

5.3 Proof-of-concept of Adaptability

This thesis addresses Research Question 3.2 based on the work by *Pihlgren et al.*, [33], which was produced as part of the research for this thesis. *Pihlgren et al.*, [33] performs proof-of-concept experiments to evaluate whether deep perceptual similarity metrics can be adapted to different contexts. The experiments build on those by *Zhang et al.*, [2] which train LPIPS metrics to improve their performance as explained in Subsection 2.7.2. *Zhang et al.*, [2] show that the metrics got increased performance on the specific distortion types that they were trained on, but that increase only barely affected performance on other distortions. This is promising since it shows that existing metrics can be improved

for a specific task by training on that task. The lack of impact on the performance of other distortion types can be seen as both a downside and an upside. It is a downside since it means that the metrics cannot generalize human perception from one set of distortions to another. It is an upside since it implies that adapting a metric to some specific data will not significantly reduce performance on another. The training of LPIPS metrics is a possible method to adapt metrics to a new context where similarity is defined by 2AFC triplets defining what images are considered more similar than others.

Training metrics on BAPPS do not necessarily show that the metrics are adaptable. While the performance of the metrics did improve for the distortions they were trained on, the improvement was slim, and the metrics already performed well for that task. To show that metrics are adaptable would at least require showing that they can learn to predict which images should be considered more similar in many different contexts beyond those typically tested in perceptual similarity datasets.

The experiments of *Pihlgren et al.*, [33] that are analyzed in this thesis follow the approach of learning positive scalars to adapt deep perceptual similarity metrics to a context defined by 2AFC triplets. This training procedure is mostly the same as the one called "lin" in Subsection 2.7.2 with a few exceptions presented later. The reason for only learning the scalars and not also fine-tuning or training the model from scratch is twofold, it requires fewer resources, and it allows evaluation of the features learned from pretraining. While any deep perceptual similarity metric will consider one pair of images more similar than another, perhaps the deep features extracted by the loss networks contain the features needed to make either judgment; they just need to be scaled to the given context. Since ImageNet pretrained loss networks are already attuned to human perception to a large degree, they would seem like a good starting point for adapting to human perceptions in other contexts. They might already have learned the features needed to handle those contexts and only need to balance those features with respect to the others. This is also the reason for keeping the scalars positive, as it means the features are only rebalanced and not inverted. As will be shown later, it also allows analysis between adapted and non-adapted (baseline) metrics since a baseline metric that performs worse than random could be improved by inverting its predictions. Such inversion could be learned by scalars if they were allowed to be negative, meaning that adapted metrics can perform better simply by inverting the features rather than finding what features are relevant to the context.

To create different contexts to adapt to, six different commonly used distortion types were chosen and then randomly ranked according to which distorted image should be considered more similar to the original. Each such ranking then presents a different context where the distortions are to be considered more or less disruptive to the original image. Through this procedure, metrics will have to adapt to rankings that the baselines are already in agreement with, rankings that go against the perception of the baselines, and ranking where distortions that the baselines consider similar are put far apart. However, adapting to such rankings could lead to the metrics only learning the ranking on similar images and failing to generalize to different images. To test this, the adapted metrics can be evaluated on a dataset with significantly different images in addition to the dataset

Table 5.1: The different components that are evaluated in the adaption experiments. Each combination of loss network, comparison method, and adaption training is evaluated on each of the random rankings. The adapted metrics are trained and tested four times for each ranking to provide average and variance of performance.

Loss Networks	Comparison Methods	Training	20 random rankings of:
AlexNet [79]	Spatial	Pretrained	Rotating
SqueezeNet 1.1 [81]	Mean	baseline	Translating
VGG-16 [73]	Sort		Lowering brightness
	Spatial+mean	Pretrained+	Shifting hue
	Spatial+sort	adapted	Gaussian blurring
			Zooming in

they were trained on. Furthermore, it is possible that the metrics will overfit to the specific rankings leading to decreasing performance on estimating human perception on unrelated perceptual similarity datasets. This is tested by comparing the performance of adapted metrics to the baseline metrics on such a dataset. How these experiments were carried out in practice is detailed in the next section.

5.4 Experimental Setup

This thesis evaluates the adaptability of deep perceptual similarity metrics. The experiments are composed of many different components that are briefly described below and then expanded on in the following subsections. The same ImageNet pretrained AlexNet [79], SqueezeNet 1.1 [81] and VGG-16 [73] loss networks and feature extraction are used as by *Zhang et al.*, [2]. The five feature comparison methods are the same as those used by Sjögren *et al.*, [29]. Each combination of loss networks and comparison methods was adapted to 20 different random rankings of six different distortions by training scalars in an altered version of the training procedure used by *Zhang et al.*,. Each metric is adapted four times to get the variance of performance over multiple adaptions. The images that are being distorted during training are taken from the training set of the SVHN dataset. For evaluation of adaption to each ranking, the images from the test sets of SVHN and STL-10 are both used in order to evaluate how the adapted metrics perform when the images are different from those that they were adapted on. Additionally, the metrics are evaluated on the BAPPS dataset to see how adaption affects the performance on an unrelated perceptual similarity task. The combination of three loss networks, five comparison methods, 20 different rankings, four different runs of each adaption, and evaluating the baseline metrics results in 1500 different rows of data. The different parameters that have been combined for each of these combinations are detailed in Table 5.1.

Table 5.2: The ImageNet pretrained architectures and feature extraction layers are used for the loss network for adaption training and evaluation.

Architecture	Feature Extraction Layers
AlexNet [79]	1^{st}, 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} ReLU
SqueezeNet 1.1 [81]	1^{st} ReLU, 2^{nd}, 4^{th}, 5^{th}, 6^{th}, 7^{th} and 8^{th} Fire
VGG-16 [73]	2^{nd}, 4^{th}, 7^{th}, 10^{th}, and 13^{th} ReLU

5.4.1 Loss Networks

The three loss networks (AlexNet, SqueezeNet 1.1, and VGG-16) come from the out-of-the-box models available in the Torchvision [72] framework version 0.11.3. The features were extracted from layers throughout the convolutional parts of the models as detailed in Table 5.2.

5.4.2 Similarity Calculations

Similarity scores are calculated in the same way as by *Zhang et al.*, [2] with the addition of more comparison methods. The spatial, mean, sort, spatial+mean, and spatial+sort methods that were analyzed in Chapter 4 are evaluated.

The similarity metrics are altered to use spatial, mean, and sort comparison methods as detailed in Eq. 5.1 to 5.3 below. In the equations, z^l_x are the activations, which may be normalized along the channel dimension, in layer l from a loss network with input x and extraction layers $l \in L$. $\overline{z}$ and $z^{\downarrow}$ are the average and descending reordering of the channels in z respectively. w_l are the scalars for the features of layer l, which are set to 1 in the baseline cases and adapted to be positive values during adaption training, as explained later.

$$d_{spatial}(x, x_0) = \sum_{l \in L} \frac{1}{C_l H_l W_l} ||w_l \odot (z^l_x - z^l_{x_0})||_2^2 \tag{5.1}$$

$$d_{mean}(x, x_0) = \sum_{l \in L} \frac{1}{C_l} ||w_l \odot (\overline{z}^l_x - \overline{z}^l_{x_0})||_2^2 \tag{5.2}$$

$$d_{sort}(x, x_0) = \sum_{l \in L} \frac{1}{C_l} ||w_l \odot (z^{l\downarrow}_x - z^{l\downarrow}_{x_0})||_2^2 \tag{5.3}$$

5.4.3 Datasets

three datasets are used; SVHN for training and evaluation of adaption to each ranking, STL-10 for evaluating adaption to each ranking on different images, and BAPPS for evaluating how adaption affects the performance on human judgments of similarity. The BAPPS dataset consists of human judgments on 2AFC and JND data, all of which are described earlier in Section 2.2. SVHN and STL-10 are both labeled image classification

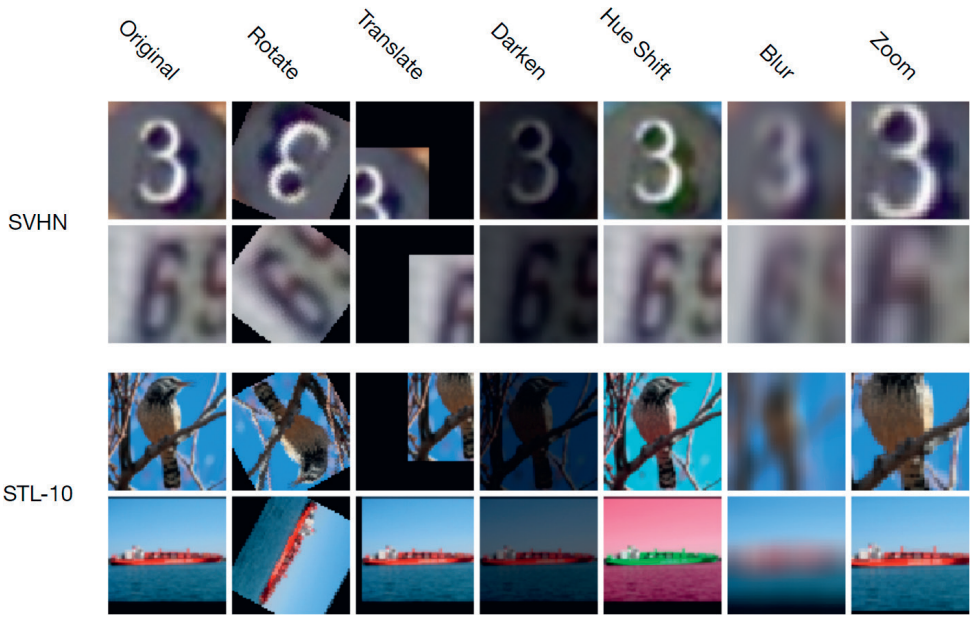


Figure 5.3: Example images from the test sets of SVHN [132] and STL-10 [133] along with versions that have been distorted by the six different distortions that are used by *Pihlgren et al.*, [33] from which the figure originates. Reproduced with permission.

datasets, though only the images are used for adaption training and evaluation. The former consists of 32×32 images of digits cropped from photos of house number signs split into 73257 training and 26032 testing images that are used for adaption training and evaluation. The latter consists of 96×96 images of animals and vehicles taken from the ImageNet [64] dataset with 8000 testing images that are used for adaption evaluation. Sample images from the two datasets and versions of those images distorted with the six distortion types are shown in Fig. 5.3. As can be seen in the figure, the two datasets contain images with very different content.

5.4.4 Distortions

The adaptability of the metrics is evaluated by training them on 20 different random rankings of six different distortions types. The six distortion types are rotating, translating, lowering brightness, shifting hue, Gaussian blurring, and zooming in on the image, all of which are commonly used in image augmentation for machine learning [151]. The implementation of the distortions comes from the Torchvision [72] framework, and when used on an image, they are uniformly randomly applied within the parameter intervals specified in Table 5.3. Only a single distortion is ever applied at the same time (*e.g.*, no

Table 5.3: The distortions that are used for training and evaluation of adaptive metrics and the intervals within which the parameters of each distortion were randomized as it is applied to an image.

Distortion	Parameters and intervals
Rotating	30 to 330 degrees
Translating	-0.5 to 0.5 of the image size in each direction
Lowering brightness	0.1 to 0.5 of the original brightness
Shifting hue	-0.5 to 0.5 hue factor (all possible hues)
Gaussian blurring	11 to 21 kernel size 4 to 10 std. dev. for generating kernel values
Zooming in	1.1 to 2 scale of zoom

distorted image will be both rotated and blurred). Application of each distortion type to images from the SVHN and STL-10 datasets are shown in Fig. 5.3.

5.4.5 Adaption Training

The metrics are trained to adapt to each ranking using the images in the training set of SVHN, 20% of which are used for validation. Every epoch, two of the six distortions are randomly applied to each image to create 2AFC triplets for training. The triplets are labeled with which of the distortions are earlier in the ranking and, thus, which should be seen as more similar (0 if the first distortion is earlier, 1 otherwise). The metrics are trained to adapt to the ranking with the same procedure described in Subsection 2.7.2. Training is performed for 10 epochs, and during the last 5 epochs, the learning rate decays linearly towards 0.

The CNN that performs prediction during training can learn to predict equally well with metrics that predict the opposite of the desired output. This sometimes leads to metrics that learn the opposite ranking of what is desired. To prevent this, an additional loss $\mathcal{L}_{sync}$ is added to synchronize the prediction with the metric similarity. The synchronization loss trains the metric to higher similarity for the distortion that is earlier in the ranking. However, forcing the metrics to have a fixed difference between the similarity scores puts unnecessary constraints on them. For this reason, the synchronization is only applied to every epoch until the validation 2AFC score of the metric is higher than random chance. The loss is described in Eq. 5.4 where d is the metric being trained, x is the original image, x_0 and x_1 are the distorted versions of x, σ is the sigmoid function, and J is 1 if the distortion of x_1 is earlier in the ordering and 0 otherwise.

$$\begin{aligned} &\mathcal{L}_{sync}(x, x_0, x_1, J) = \\ &10 \cdot \max(0, \mathrm{BCE}(\sigma(d(x, x_0) - d(x, x_1)), J)) \end{aligned} \tag{5.4}$$

For each baseline metric and ordering, four different adaptions are trained to measure the variance of training.

Additionally, a small sample of metrics was adapted by fine-tuning the parameters of the loss network in addition to learning positive scalars like the other adapted metrics. These additional adapted metrics are intended to evaluate the benefits of training just scalars in comparison to fine-tuning. They also have the advantage of inverting the features which cannot be done using only positive scalars. These fine-tuned adapted metrics were only trained for with AlexNet as a loss network with spatial comparison. They are referred to as fine-tuned adapted metrics to separate them from the other adapted metrics being evaluated.

5.4.6 Evaluation

For each ranking, the adapted metrics for that ranking, as well as all baseline metrics, are evaluated on the test set of SVHN and STL-10. Test set triplets have been formed in the same way as for the training set; by randomly applying two of the six distortions. The performance scores of the two datasets are the 2AFC scores according to how well the metric follows the ranking.

The metrics are also evaluated on the BAPPS dataset to evaluate how adaption to the ranking affects performance on the previous performance. The performance scores for BAPPS are the 2AFC and JND scores on the test sets.

5.5 Results of Adaption

The 2AFC scores for each metric and ranking on SVHN and STL-10 are visualized in Fig. 5.4. The scores presented in the figure show the fraction of images test set where the metrics agreed with the particular ranking (which the adapted metrics had been trained on) of which distorted version of the image should be more similar. The figure shows, for all 20 rankings, each combination of loss networks, comparison methods, and whether the metric is a baseline (•) or adapted (×) to the ranking. The data points are colored according to the average STL-10 score of the baseline metrics, with red being lower and blue being higher. The figure shows that there is a correlation between the performance of a metric on a particular ranking on SVHN and the same metric and ranking on STL-10. There is a significant correlation between which rankings the baseline and adapted metrics perform well on. Specifically, the Spearman rank correlation coefficient of the average performance on each ranking between baseline and adapted metrics are 0.66 and 0.72 for SVHN and STL-10, respectively. Additionally, there is a strong correlation between the performance on each ranking between all loss networks as well as between all comparison methods. The Spearman rank correlation coefficient is above 0.9 between all pairs of loss networks as well as between all pairs of comparison methods.

Figure 5.5 shows the SVHN and STL-10 scores for each combination of loss network, comparison method, and whether the metric has been adapted (above) or not (below). The figure shows the average and variance scores across all rankings.

The decrease in 2AFC and JND score on BAPPS for adapted metrics compared to their baseline counterparts is around ~ 0.01 for both scores averaged over all rankings and

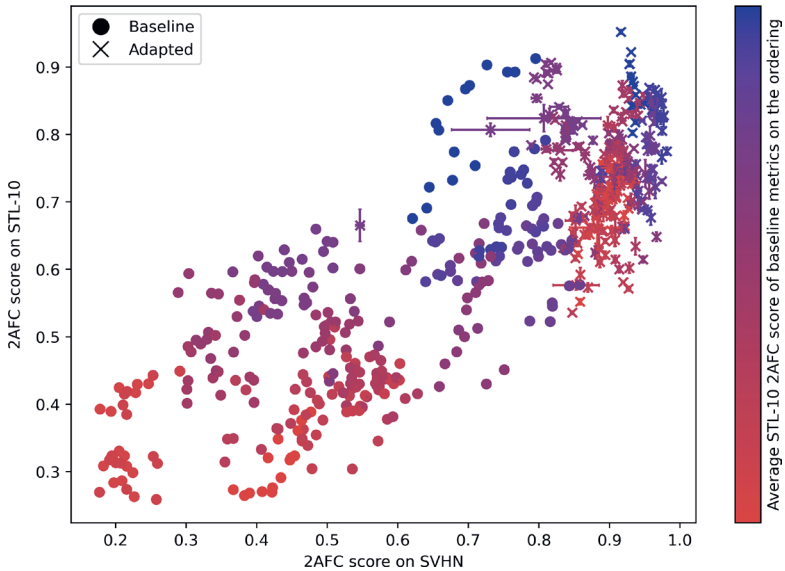
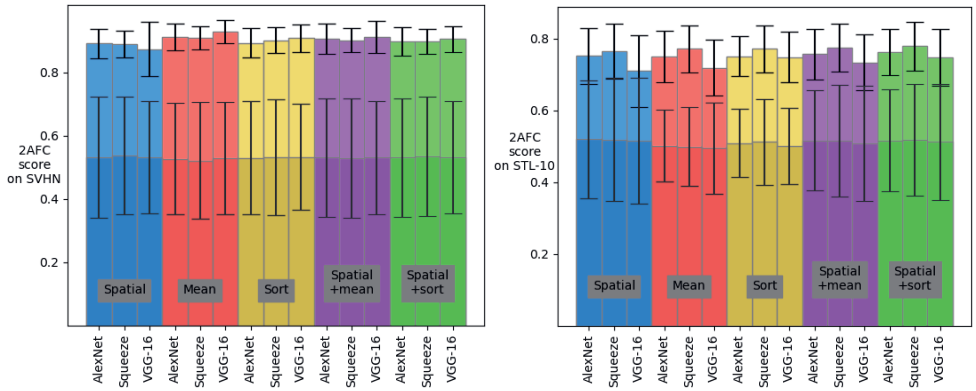


Figure 5.4: The 2AFC scores for all adapted (×) and baseline metrics (•) on the test images of SVHN and STL-10 for all 20 rankings of distortions in the work by *Pihlgren et al.*, [33] from which the image originates. The scores measure the fraction of cases where the metrics agree with the ranking on which distorted version of the image should be more similar. The color of the data points indicates which ranking the score is from, where each ranking is placed on a gradient from low (red) to high (blue) according to the average STL-10 score of baseline metrics on that ranking. Reproduced with permission.

metrics. The baseline metrics outperformed their adapted counterparts on the BAPPS metrics for all rankings. The small sample of adapted spatial AlexNet metrics that were fine-tuned showed even worse deterioration with average performance decreases of ~ 0.03 and ~ 0.02 respectively for the 2AFC and JND scores. For comparison, when adapting only scalars, the scores decrease by ~ 0.01 averaged over all tests using a spatial AlexNet metric.

The fine-tuned adapted metrics also showed a significant improvement of 0.10 and 0.05, respectively, on the SVHN and STL-10 scores compared to the other adapted metrics. Though likely, much of this improvement is due to the ability to invert the features to perform better on rankings with low baseline scores. This is supported by the lower correlations for the different training procedures to the baselines between which rankings they perform worse on. The Spearman rank correlation coefficient is 0.75 between the scalar-only adapted metrics and the baselines, while it is only 0.30 between the fine-tuned adapted metrics and the baselines. This implies that the baselines and scalar-only metrics perform well on the same rankings, and the fine-tuned metrics perform well on other rankings.



(a) (b)

Figure 5.5: The 2AFC scores on SVHN (a) and STL-10 (b) from the work by *Pihlgren et al.*, [33] from which the figures originates. The scores are the fraction of test images where each metric agreed with the ranking on which of two randomly distorted versions should be considered more similar. The scores are averaged over all rankings for each combination of loss networks, comparison methods, and whether the metric is baseline (below) or adapted to the ranking (above). Reproduced with permission.

5.6 Analysis and Implications of Results

The results in Section 5.5 show that adaption is possible to contexts defined by which distortions should be considered more similar. However, the adapted metrics are unable to reach the performance that a classifier likely could reach given the task of classifying which distortion has been applied. The stricter learning scenario of a metric can partly explain this. Instead of training to identify each distortion, the metrics are trained to produce features such that when features extracted from two different images are compared, the outcome is a distance that should correlate with similarity. In this case, the measurement of similarity is defined by a ranking. Additionally, the adapted metrics are limited to learning positive scalars, while learning negative scalars could further improve performance, especially for the worst-performing adapted metrics, some of which would have been improved by simply learning all scalars to be -1. Another potential issue that metrics suffer from compared to classifiers is the intensity of distortions. Fig. 5.3 shows that a given distortion type can sometimes significantly alter the image and, other times, barely affect it. For example, the translation in the figure moves removes most of the original image in one case and only barely moves it in another. The rankings in the current evaluation do not capture these differences and only care that one distortion type is seen as more or less similar to some others. Regardless, adaption is possible for the given scenarios despite some distortion rankings being extremely difficult for the baseline metrics. This shows that, to some degree, the features learned on ImageNet pretraining are enough for at least partial adaption to contexts defined in similar ways.

This is additionally significant since there is likely a significant overlap between which features activates more strongly between various distortions. For example, translation and rotation both move the image around and produce black-colored regions. Despite this, the metric can adapt to seeing one of these as the most and the other as the least similar.

Compared to the baselines, the performance is improved on both SVHN and STL-10 images for the adapted metrics. This shows that adaptions generalize to images of a different type to those that the adaption was learned on. However, most adapted metrics outperform even the best baseline metric on SVHN, but on STL-10, the worst adapted metrics are on par with the average baselines, and some baselines are comparable to the best-adapted metrics. It should be noted that only two out of the 300 combinations of loss networks, comparison methods, and rankings do the baseline metrics outperform their adapted counterparts. The reduced performance when evaluating on STL-10 instead of the dataset that adaption training was performed on is not surprising. Fig. 5.3 showed how different some of the distortions affected the two datasets, with hue shift, in particular, being barely noticeable in the subdued colors of SVHN while having a significant impact on the colorful images of STL-10. Still, the reduced performance is a potential issue for this method of adapting to a context.

Interestingly the performance of adapted metrics on different rankings is correlated to the performance of the baseline metrics on the same ranking. This means that even though the metrics are adaptable, the rankings that are challenging for baseline metrics are also difficult to adapt to. Partly, this is due to the adapted metrics only using positive scalars, which means that the features that predict the opposite ranking cannot be inverted for the given ranking. If negative scalars were allowed, then the performance of adapted metrics on these rankings could at least reach the inverted performance. In this scenario, the rankings where baseline metrics have close to random performance would likely be the most difficult to adapt to. This implies that the features learned through pretraining are not necessarily favorable for learning in all contexts. Since the adapted performance on the rankings where baseline metrics have close to random performance is decent, allowing negative scalars would somewhat alleviate the issue. Though, perhaps improved pretraining could produce deep features that are more generally applicable.

Interestingly, no combinations of loss networks and comparison methods significantly outperformed any other. That the choice of loss network had little impact on the results is consistent with prior results [2]. This is surprising because, as Chapter 4 covered, previous works have shown that spatial metrics are poor at handling translation and rotation distortions [55, 29]. The high correlation between performance on different rankings of the various comparison methods also means that comparable performance mostly holds for individual rankings. This means that their comparable performance does not come from spatial metrics being better at some rankings and worse at others. A potential answer is that the metrics have adapted to some quirks of how the distortions are performed rather than actually adapting to actually consider the distortion itself more or less similar. For example, as can be seen in Fig. 5.3, the implementation of translation and rotation used is such that the parts where the image has been removed

are filled with black. In these cases, the metrics might learn that when there is black at the edges or in the corners, rather than when the same image is moved, should be seen as more or less similar. This might actually mean that the metrics have learned to look for translation, regardless of image content, which could mean that the metric would consider a translated version of another image to be more similar than another distortion of the same image. A potential fix for this that could be implemented in future works is to train the metrics and also to consider any distortion of the original image to be more similar than any distortion of a completely different image. This should also be used in evaluation to ensure that the metrics have learned to see image similarity and not simply the artifacts of distortion.

Training LPIPS metrics by not only comparing each image to distorted versions of itself but also to other images and their distortions might improve the training procedure as a whole. This would generalize better because it would make it more difficult for the metrics to overfit to the specific distortions. This could lead to improve perceptual similarity performance overall. Such updates to the training procedure would make it even more similar to contrastive learning, as negative pairs are now created from distortions on different images.

Despite the potential issues of learning artifacts of distortions rather than the similar features in images, the adapted metrics have only slight decreases in their performance on the BAPPS datasets. Interestingly, there were no rankings for which the adapted metrics increased the BAPPS performance compared to the baselines. It would seem possible that some distortion rankings would better align with the human judgments in the datasets and that training on those could therefore improve performance, but this is not the case for the conducted experiments. Instead, it is likely that overfitting on the rankings and learning that extreme distortions of one type are more similar than mild distortions of another is behind the deterioration on the BAPPS dataset. The possibility of learning seems to be supported by the further decreased BAPPS performance for the fine-tuned adapted metrics, as allowing fine-tuning would enable the metric to overfit further. This also enables an interesting trade-off between adaption and maintaining previous properties depending on how much of the model is retrained. Additionally, the low deterioration is positive since it suggests that metrics can be adapted to a context without losing previous desirable properties unless those properties go against what the metric is being adapted for. This issue, known as catastrophic forgetting, where neural networks have a significant decrease in performance on previous data after training on new data, can often be problematic in other domains [152]. Since catastrophic forgetting does not seem to be an issue in the explored settings, this implies that adaption could be performed on specific issues for a otherwise good metric. For example, an ImageNet pretrained deep perceptual similarity metric that has decent performance on medical images could be further improved by adapting it to consider blurring as a more severe distortion without losing its good properties.

These initial findings by *Pihlgren et al.*, [33] seem to show that adapting deep perceptual similarity metrics to different contexts is possible. Though there are many things to further explore, not just in regarding how to train for adaption, but how to handle

ambiguity in general and how adaptability could be useful. Such thoughts are contained within the third and final sub-question of Research Question 3.

Q3.3 What are the benefits of having deep perceptual similarity metrics able to adapt to different definitions of similarity?

5.7 Future Work on Ambiguity and Adaptability

Sub-question 3.3 builds directly on both sub-questions 3.1 and 3.2. The former asks how ambiguity poses a problem, and the latter asks if metrics can be adapted to handle the varying contexts that cause ambiguity. Addressing 3.1 raised some issues caused by ambiguity, and addressing those issues could therefore be a benefit for adaptable deep perceptual similarity metrics. However, 3.2 was only addressed in a proof-of-concept manner, with more developments needed for adapted metrics to actually be beneficial in those applications. This section covers both potential future development for adaptive deep perceptual similarity metrics as well as where such metrics could be put to use on practical tasks.

The proof-of-concept work by *Pihlgren et al.*, [33] provides initial support for deep perceptual similarity metrics being adaptable to given contexts. However, there are many potential further developments to the implementation used in the. The previous section has already mentioned improving the adaption training by incorporating triplets where one of the distortions is applied to another image entirely. The previous section also suggested training on BAPPS during adaption to avoid catastrophic forgetting. Taking this suggestion one step further, it could be possible to integrate the adaption training entirely into the pretraining. As discussed in Chapter 3, *Kumar et al.*, [67] have shown that the pretraining procedure of deep perceptual similarity metrics is one of the most influential factors for performance. Thus, getting both the performance boost from pretraining specifically for perceptual similarity and the context-specific improvement from adaption could massively improve performance. Since pretraining is now performed for the metric, datasets from the target domain could be used, which can improve performance even further. However, pretraining for the neural networks used in deep perceptual similarity is typically very resource intensive. Pretraining specifically for each context is likely wasteful of resources when adapting already pretrained models is sufficient.

Another issue is how to adapt to contexts for practical applications. The proof-of-concept work has shown that adapting to consider certain distortions more or less similar is possible, and this can perhaps be directly used for some applications. Experts in certain domains could define distortions that models should be mostly invariant to and those which would make the image very different in that domain. Blurring in medical images has already been given as an example of such a distortion that severely affects images. The training procedure could also be expanded beyond simply using a linear ranking of distortions to other methods of creating 2AFC triplets from the relevant distortions. Perhaps all the distortions with low impact on similarity are only put in triplets with high impact distortions.

The approach of learning on distorted triplets is limited to using distortions that can be automatically applied to the data, though, and not all contexts are easily defined by distortions. For example, similarity of painting style might be difficult to adapt to using specific rankings or distortions since what causes styles to be more or less similar depends on higher-level structures than those affected by common computer vision distortions. Another way forward is to create the triplets by other means than distortions. The triplets in the BAPPS dataset was gathered by using human judgments, and a similar process could be used to gather human judgments on similarity for many different contexts. This could be beneficial both for training and evaluation of adapted metrics. While constructing datasets is a resource-intensive undertaking, for perceptual similarity, human judgments are the ideal method for evaluation. A concern with learning high-level human perceptions of similarity from a dataset that goes beyond simple distortions is that the adapted metrics might not generalize to that context for data not included in the training. This was the case for the implementation by *Zhang et al.*, [2] where learned metrics did not improve their performance in the same context but with distortions for which they were not trained. However, in that case, the metric already performed well, and the improvement on the distortions for which the metrics were trained was relatively small.

There are many comparisons to draw between the training of similarity metrics used by *Zhang et al.*, [2] and *Pihlgren et al.*, [33] to practices used in contrastive learning. In fact, the training used can be considered a form of contrastive learning with the original image and the distortion judged as more similar being used as a positive pair and the image and less similar distortion being a negative pair. For this reason, it is likely beneficial to take inspiration from successes in contrastive learning to improve the training of adaptive metrics. The field of contrastive learning has also made use of distortions to learn features for data that are supposed to be considered more or less similar. It has also been noted that the use of some distortions which are beneficial for training models for photographs do not perform well in other domains such as medical imaging [82]. Instead, using information from the data to determine which images to consider similar is explored as an alternative [36]. Loss networks trained with contrastive learning have already been used for deep perceptual loss to improve image generators [153, 154]. Furthermore, some contrastive learning approaches use similarity in the feature space to define if two items are similar, which is analogous to deep perceptual similarity [155]. There are likely many more interesting developments in contrastive learning that could be used for perceptual similarity metrics in general and adaptive metrics in particular.

In order to assess whether adaption actually provides any tangible benefits, it has to be evaluated on datasets representing a practical task to perform. Likely the two most accessible approaches for this are deep perceptual loss and content-based image retrieval. Deep perceptual loss is suitable since it can be applied to a host of tasks, which can see improved performance. Content-based image retrieval is suitable since it is a task domain that has already acknowledged the issues caused by ambiguity [147, 148].

Chapter 6

Conclusion and Future Work

This chapter summarizes and discusses the contributions of the presented research from a broader perspective. It covers the general conclusions of the thesis as a whole and how that ties into the specific fields of perceptual similarity and deep perceptual loss, as well as machine learning in general. The ethical considerations of the research are covered, as well as what ethical issues exist within deep perceptual loss and similarity specifically and machine learning generally. Finally, the future of deep perceptual similarity and loss is discussed, including remaining issues, promising directions for future research, and what those suggestions might bring.

6.1 Conclusion

The research for this thesis has been carried out to further the understanding of deep perceptual loss and similarity. Three areas where knowledge of the methods was lacking were identified, and research questions were posed to address this. Each of these research questions has been, at least partially, answered in this thesis.

Q1 How does the loss network implementation affect downstream performance?

Q2 What significant flaws do deep perceptual similarity metrics have, and how can they be mitigated?

Q3 To what extent can deep perceptual similarity handle the inherent ambiguity of similarity?

The first research question is addressed in Chapter 3, which shows how the choice of architecture, parameter setting, and feature extraction affects deep perceptual loss and similarity by aggregating three different works [2, 67, 32]. The findings show that deep perceptual loss and similarity neither adhere to the convention that better ImageNet pretraining accuracy leads to better downstream performance nor that feature extraction is best performed in the later layers. Additionally, the performance of a loss network

on a given deep perceptual loss or similarity task does not predict its performance on a different task. Instead, more complex correlations were found. These correlations include that certain relative depths in CNNs are suitable for certain tasks and that those depths seem to be predictable based on the task and dataset. Based on this, guidelines for implementing loss networks with better performance for a given task are provided. The contributions towards addressing Research Question 1 consist of the cross-analysis of the three works as well as the research that resulted in one of those works [32].

The second research question is addressed in Chapter 4, which covers some flaws inherent to deep perceptual similarity as well as flaws related to CNNs and the methods used for comparing the deep features extracted from images. The chapter analyzes of the strengths and weaknesses of different comparison methods. The findings support combining spatial and global comparison methods to improve deep perceptual similarity, but more research is needed, especially regarding how this can be applied to deep perceptual loss. The contributions towards addressing Research Question 2 include the aggregation of different comparison methods and their applications along with the research that is presented in the earlier mentioned work [29].

The third research question is addressed in Chapter 5, which covers the ambiguity inherent in perceptual similarity and the ability of deep perceptual similarity to adapt to different similarity contexts. The chapter covers proof-of-concept experiments to investigate the capability of deep perceptual similarity metrics to adapt to different contexts [33]. The results show that the metrics can adapt to different contexts without significant detriment to their prior performance and that the CNNs used already have learned features useful to many different contexts. The contributions towards addressing Research Question 3 include the previously mentioned work as well as a brief analysis of the issues related to ambiguity and the benefits of adaptable metrics.

6.2 Ethical Considerations

Almost all ethical concerns in research can be divided into one of three categories: (i) concerns regarding how the research is performed, (ii) concerns regarding how the research findings will be applied, and (iii) concerns regarding the products and results of the research. This section covers the ethical concerns of the research for this thesis, split into three categories. The first category covers concerns that are present while the research is being conducted, such as researcher bias, treatment of subjects and personal data, and cost of the research. The second category covers concerns not with the research itself but with how it might be utilized in ways that unintentionally or intentionally harm people. An example of unintentional harm is training models that learn unwanted discriminatory biases against certain groups of people. Another example, this time of a malicious case, could be the use of machine learning to create scams. The third category falls in between the other two as the issues might neither be with how the research is performed nor with how it is applied, but instead with products of research beyond the knowledge itself. In machine learning, such results are typically the creation of datasets or training of large models. Examples of such issues are biased datasets and bigoted models.

6.2.1 Concerns with Performing the Research

This thesis and the research for it have not used subjects or data with sensitive information. Datasets with human data and labels have been used along with models pretrained on such datasets. As the research has not concerned the human data and could have been conducted on datasets without such data, this has not impacted the research itself. The models used in the research might have learned unwanted biases from the data, which is covered in Subsection 6.2.2 and 6.2.3. As such, the primary concerns regarding how the research was carried out are the cost of the research. The cost of the research comes in two parts, financial cost to society and energy consumption.

Ph.D., student positions typically come with a high financial cost for the hosting institution. This cost includes reimbursement of the student, time from supervisors, rent of offices, research equipment, and more. Funding for these costs usually comes from either industrial partners or the government. The value of a doctorate then has to be weighed against these costs. This is complicated as valuing research, especially research that is not directly applied to a real-world problem, is notoriously difficult [156]. While the research for this thesis is not directly applied, it builds on machine learning and computer vision, both of which have vast potential to improve society in the years to come. Specifically, deep perceptual loss and similarity have been applied to actual problems, including medical imaging, which can have direct societal benefits. There might also be additional unknown benefits as it can be difficult to predict what future research or applications might be inspired by the findings in this thesis. However, in spite of all of this, the cost of Ph.D., students is not typically viewed as investments into the published theses but rather into the education of new researchers that might offset the cost through future research.

Energy consumption is another relevant cost since the research falls in the field of deep learning, which is notorious for energy-intensive models [157]. One of the key enablers of deep learning has been the increase in availability and performance of computer hardware, particularly in the form of Graphics Processing Units (GPUs) and similar parallel computing hardware [5]. The success of large models and datasets has led to a field where the creation of such large models consumes vast amounts of electricity [158]. This increasing electricity consumption occurs while society is trying to minimize its strain on the world's resources. Specifically, the reduction of carbon emissions, 40% of which come from electricity and heat production [159]. This has led to an increasing interest in measuring not only the performance of models but also their resource consumption.

Measuring how much electricity is consumed by the training of models can be difficult. Many different projects might use the same computing hardware, which consumes different amounts of electricity from different sources over time. Monitoring this is another task for researchers to weigh against other uses of their limited time and resources. However, an estimate of the order of magnitude can be produced. The most energy-intensive experiments performed as part of the research for this thesis, by an order of magnitude, are the ones presented in Chapter 3. The GPU power consumption, which stands for the vast majority of the electricity consumed throughout those experiments, was around 500W. The duration of those experiments was around 100 days of processing

(with several experiments running in parallel), or 1056 hours, which gives a lower bound of at least 1.2 MWh of electricity consumed by the experiments. Since those experiments were by far the most electricity consuming and the GPU consumption stood for the most of that consumption, an upper bound can be estimated by rounding up to 2 MWh. This is a lot of electricity, roughly equivalent to what Sweden (where the research and experiments have been conducted) consumes per capita every month (2019) [160]. Though, the amount is still smaller than what the researchers involved consumed in their day-to-day for the duration of the experiments. Additionally, the experiments were conducted in Sweden, where around 1% of the energy consumed is produced with fossil fuels, and an additional 9% through waste heat (2019) [160].

Connecting back to deep perceptual loss and similarity, these methods replace previous less energy-intensive rule-based calculations. Thus using them will increase the energy intensity of the computations. For loss calculation, this additional cost is mostly negligible as the loss network is often smaller than the model being trained. Additionally, the loss network is only needed during training and not inference where, in real-world use cases, a majority of the energy consumption takes place [157]. For deep perceptual similarity, though, the energy consumption can be orders of magnitude larger than the rule-based methods that are being replaced. This is why the analysis of the computation required for different loss networks from Chapter 3 is relevant. By selecting the right loss network, energy consumption can be lowered by orders of magnitude.

In general, the field of machine learning needs to focus more on the energy consumption of the research and models being produced. While it seems like ever-larger models will produce better results, there also seem to be diminishing returns, with significant increases in energy consumption returning slight increases in performance. The question then is when the added cost will start outweighing the benefits. Thankfully, energy-efficient machine learning is a growing field of research [161]. More and more research is considering not only the performance of models but the energy consumption as well [162].

6.2.2 Concerns with Applications of the Research

This thesis builds on and promotes the use of deep perceptual loss and similarity. However, these techniques have flaws that could cause harm if the technology is applied without addressing them.

Section 4.1 covered how neural networks, and therefore deep perceptual loss and similarity, are susceptible to adversarial attacks. If deep perceptual similarity metrics were applied in a real-world setting, there is a risk that malevolent attackers could use adversarial attacks to trick the metrics. This risk must be considered in actual applications. However, this is not a new risk, as most rule-based metrics are likely susceptible to similar attacks. Additionally, adversarial attacks are very unlikely to happen by accident, meaning that they are only an issue in settings where malicious actors have access to the system.

Another potential issue that has not been brought up before is potential bigotry in the models. The data used for training machine learning models are often labeled by humans

who then put their own, sometimes harmful, biases into the data [163]. Machine learning models are well-known to be able and likely to pick up and learn such biases [163]. In general, the loss networks used for deep perceptual loss and similarity have been pretrained on ImageNet [64], which is a dataset collected from crawling the internet for images and labeled by humans. The images that are accessible on the internet are subject to societal bias, and the labels are subject to individual biases.

Models pretrained with ImageNet have been shown to learn human-like biases, some of which resemble the harmful biases which exist in society [164]. Similarity metrics that use ImageNet pretrained loss networks would also likely have these biases, which need to be considered before applying them. In deep perceptual loss, the loss networks are only used for training models and are not actually involved in any decision-making. However, the downstream model might learn biases from the loss network used to train it. Training with deep perceptual loss can be likened to knowledge distillation which is explicitly used for transferring the knowledge of one model to another. This fear is, therefore, reasonable.

There are several technical approaches being investigated to minimize the effects of unwanted bias [163]. One such approach trains DNNs on a given task while simultaneously training to be unable to predict sensitive attributes from the deep features [165]. Ideally, the result is a model with deep features that are invariant to the sensitive attributes and therefore do not bias the results based on them. These approaches can not only be used to create more fair machine learning models but could theoretically also be used to reduce unwanted bias in human decision-making as well. Though, care must be taken not to allow poorly understood models to override human decision-making.

There are likely other potentially harmful issues besides those mentioned above that will need to be considered before the research is applied. The responsibility to minimize the potential harm caused by applications of research falls both on the researchers and the individuals or organizations that apply the research. The research community has a joint responsibility to be vigilant for the potential harm that applications of the research might cause. This responsibility includes keeping society informed of the potential harm to minimize the risk of accidental harm and defend against malicious use. The community has developed guidelines to steer research and its applications away from such harm. For the field of machine learning and AI, one such set of guidelines is the *Ethics Guidelines for Trustworthy Artificial Intelligence* presented by the *EU's High-Level Expert Group on AI* [166].

6.2.3 Concerns with Products of the Research

The research behind this thesis has given rise to a few products. The primary products of the research are this thesis and the other publications in which the research is presented. The implementations of the research have also been made accessible as public code repositories. Making such code repositories available supports reproducible research and allows other researchers and the public to build on and scrutinize implementations directly. This is especially crucial in the field of machine learning, where describing every

detail of the implementation can more than double the size of a publication. Some even argue that this practice is needed in the field [167], and perhaps it should be considered part of best practices. On the other hand, if access to such repositories can allow careless or malicious actors to cause harm, perhaps there is justification for not making them public. Though, since scientific publications should be reproducible malicious actors might create their own implementations based on the publication.

6.3 Future of the Field

Deep perceptual loss and similarity have seen growing research interest over the last few years and have been applied to some real-world tasks. At the same time, there is much that remains unexplored. This final section of the thesis presents what is missing from the field and opportunities for future research to further the understanding and usefulness of deep perceptual loss and similarity.

6.3.1 Expanded Evaluation

This thesis tackles three very different challenges of deep perceptual loss and similarity. These challenges were formalized into Research Question 1, 2, and 3. Through addressing the research questions, several different methods for potentially achieving better performance on perceptual similarity have been identified or suggested. Chapter 3 covers how to implement loss networks for deep perceptual loss and similarity and gives suggestions for how to achieve high performance. Chapter 4 enumerates many different comparison methods with their own advantages, shows that some of them lead to improved performance, and suggests that others could also add benefits. Chapter 5 points out flaws in the LPIPS training used to achieve the best results on the BAPPS dataset and suggests improvements inspired by contrastive learning. Most of these improvements have not been properly evaluated for how they affect performance on BAPPS and other datasets. A combination of them might be able to improve performance close to the theoretical maximum on some of the datasets. In that case, perhaps new and more challenging datasets are needed. On the other hand, plenty of datasets are used to evaluate rule-based metrics on which deep perceptual similarity has not yet been evaluated.

Expanding evaluation to existing perceptual similarity datasets would be beneficial if nothing else than to confirm the efficacy of the method. Both *Zhang et al.*, [2] and Kumar *et al.*, [67] evaluate their metrics on the TID2013 dataset [22]. In both cases, they achieve performance similar to state-of-the-art rule-based metrics instead of significantly outperforming them as on the BAPPS dataset. This motivates the evaluation of deep perceptual similarity on further datasets and also to keep comparing against rule-based methods, which remain competitive. *Ding et al.*, [4] evaluate their alteration to deep perceptual similarity on the TID2013 [22], LIVE [20], and CSIQ [21] datasets. Here too, the deep perceptual similarity metrics fail to outcompete their rule-based counterparts. An interesting question to ponder is why these datasets favor the rule-based methods while

BAPPS favor deep perceptual similarity. The answer to that question might also provide the insight needed to make deep perceptual similarity competitive in these contexts.

The above-mentioned perceptual similarity datasets are all focused on assessing how similar different distorted versions of an image are to the original. This is quite a limited context since humans also judge the similarity of images with completely different content, something these datasets do not evaluate. This can make fitting metrics to these datasets problematic as the metrics might learn that certain distortions should be considered similar regardless of which images they are applied to. These datasets also tend to have the images spatially aligned with the exception of a few distortions, which is not necessarily the case with images in the wild. New datasets that cover these issues would certainly be beneficial, if nothing else, to evaluate whether these issues are significant. However, new datasets might not be needed to evaluate this as content-based image retrieval datasets already cover many of these issues.

Content-based image retrieval was proposed in Chapter 5 as a good test bed for perceptual similarity in other contexts. Often the images to be retrieved have different content than the query image. The images might not even have the same content at all and be similar in other aspects such as style, color scheme, and class of content. The similarities of the images are not necessarily spatially aligned and might be either spread across the image or focused in a particular region. These varying contexts are facilitated by the large variety of content-based image retrieval datasets such as object category, scene, digit, apparel, and landmark datasets [168]. This is a domain that has also had deep perceptual similarity applied to it previously [102], and using the knowledge gained from those applications could perhaps be used to further improve performance on the perceptual similarity datasets.

While existing datasets for perceptual similarity might seem limited in scope, it is worth noting that these datasets usually use distortions that feasibly would be encountered in real-world scenarios. Such distortions include artifacts from compression and transmission as well as outputs of algorithms and machine learning models. These are the scenarios in which perceptual similarity is typically applied. Expanding datasets beyond domains in which the metrics are actually applied might be a waste of resources. On the other hand, knowledge gained from such datasets might be used to improve performance in the actual use cases.

6.3.2 Improving Deep Perceptual Loss

Most improvements to deep perceptual similarity remain untested for deep perceptual loss. Implementation of ideas from deep perceptual similarity, such as how to train loss networks and what comparison methods to use, could lead to further improvements of the method. Deep perceptual loss is especially ripe for improvements with regard to inspiration from contrastive learning and adapting to given contexts. Given its wide array of applications, further in-depth analysis of how to implement deep perceptual loss for a given application could definitely be beneficial. Especially since it is part of some of the largest image generation models [97], which require orders of magnitude of more

computation than investigations into deep perceptual loss would use. Findings from such works could quickly pay back their cost in terms of performance and computation improvements for the large models.

Chapter 3 covers the lack of strong correlations in the performance of loss networks between deep perceptual loss and deep perceptual similarity. The chapter also speculates that deep perceptual loss and similarity might benefit from comparing deep features in different ways. Therefore, the improved performance for both methods might be related to the general transfer learning advantage pretraining gives rather than deep perceptual similarity metrics giving an inherent advantage to loss calculations. If this is the case, improvements to one might not translate to the other. This is even more likely in scenarios where human perception of similarity is less relevant.

Finally, it is interesting that no works surveying the uses of deep perceptual loss exist. With its widespread and popular use, such a survey could benefit future developments by aggregating the findings from the many different works. Perhaps a survey of deep perceptual similarity could also be of use, though there is a significantly smaller body of work to draw on for such a survey.

6.3.3 Inspiration from Other Fields

Many other fields within machine learning utilize methods and techniques related to those used in deep perceptual loss and similarity. Chapter 5 covers how training of LPIPS metrics is similar to many contrastive learning methods and how ideas from those methods could be used to fix potential flaws in current training procedures. A similarity metric can be viewed as a one-shot learner that functions by comparing the similarity of an instance to the labeled example. Furthermore, domains beyond computer vision use perceptual similarity or related methods, all of which might be connected back to the image origins of the method. Each of these examples will be expanded on below.

The training of LPIPS metrics share similarities to contrastive learning methods. These methods learn to place the features of some images close to each other while others should be distant. This is similar to the 2AFC triplets used to train metrics, where the metric is supposed to consider one image more similar to the reference image than the other. Chapter 5 also covers how potential flaws in current training of LPIPS metrics could be improved by incorporating ideas from contrastive learning. The field of contrastive learning also has to deal with contexts, as methods that work in one domain do not necessarily generalize to others. Inspiration from how these cases are handled could be incorporated to improve the adaption of metrics to specific contexts.

Contrastive learning methods might also benefit from deep perceptual loss and similarity. Typically, negative pairs are formed by combinations of different images. Though some pairs of images might be similar enough that using them as negative pairs could lead to decreased efficacy of the learned features. In these cases, deep perceptual similarity might be used to avoid forming negative pairs of similar images. A version of this is already utilized in soft-contrastive loss, where the strength at which a pair is attracted or repulsed is determined by how similar their learned deep features are [155]. This could

be especially useful in multi-modal cases such as image captioning, where the number of image captions is usually small, and each caption is specific to a given image. A caption might describe more than just a single image in the dataset, and similarity metrics might be used to find more images that they are applicable to.

Another place from which inspiration might be drawn is few-shot learning. In few-shot learning problems, the goal is to create a model, often with extensive pretraining, that can learn to predict with only a handful of labeled samples per class [169]. Similarity metrics could be used as few-shot learners by classifying each unlabeled sample with the class that has the most similar samples. An approach comparable to deep perceptual similarity has seen recent success in few-shot learning by using nearest-neighbor classification in deep feature space [170, 171]. Metrics have also been used in a way analogous to few-shot learning in content-based image retrieval tasks where a given query image (unlabeled sample) should be paired with the best image in the database (labeled sample). Some content-based image retrieval datasets are actually directly built on image classification datasets [168]. For this reason, it is likely that the research on few-shot learning includes knowledge relevant to improving deep perceptual similarity.

This thesis is focused on deep perceptual loss and similarity in the image domain from which it originates, but, as stated in Section 2.8, these methods have been applied in other domains. The most similar applications of deep perceptual loss and similarity can be found in the domain of audio and speech processing, which sometimes even uses architectures intended for image processing [106]. The similarity of text is also typically calculated with deep features. Examining these domains for potential improvements in deep perceptual loss and similarity is therefore prudent.

Beyond comparing samples for a single domain, deep features are also used to compare samples between completely different domains through so-called joint embedding spaces. This is a common tactic to both combine different modalities for joint prediction or to generate one modality based on another [172]. Joint embedding spaces are already used for purposes that could be considered deep perceptual loss between modalities. For example, in some text-to-image applications, an image is optimized to have similar deep features as those generated by a text prompt [173]. These multimodal problems can both be an inspiration for how similarity metrics can be trained as well as a potential application of deep perceptual loss and similarity. Multimodal loss networks have already been used for deep perceptual loss to a lesser degree. An image embedder learned using contrastive language image pretraining [173] has been used as a loss network for sketch generation due to better representing semantic contexts than ImageNet pretrained models [154]. This improved representation of semantics is likely due to images sharing embedding space with text which cannot as strongly represent the exact content of an image. For example, there are many images that could fit a single short text description, and those images would therefore have similar deep features since they all map to the same description in the embedding space.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. https://www.deeplearningbook.org.

[2] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[3] Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. E-LPIPS: robust perceptual image similarity via random transformation ensembles. arXiv preprint, 2019.

[4] Keyan Ding, Kede Ma, Shiqi Wang, and Eero P. Simoncelli. Image quality assessment: Unifying structure and texture similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2567–2581, 2022.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[6] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, pages 1–26, 2020.

[7] Maria Hatzigiorgaki and Athanassios N. Skodras. Compressed domain image retrieval: a comparative study of similarity metrics. In *Visual Communications and Image Processing 2003*, volume 5150, pages 439 – 448. International Society for Optics and Photonics, SPIE, 2003.

[8] Ming Li, Xin Chen, Xin Li, Bin Ma, and P.M.B. Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[9] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[10] Bradley J. Erickson, Panagiotis Korfiatis, Zeynettin Akkus, and Timothy L. Kline. Machine learning for medical imaging. *RadioGraphics*, 37(2):505–515, 2017.

[11] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[12] Vedang Chauhan and Brian Surgenor. Fault detection and classification in automated assembly machines using machine vision. *The International Journal of Advanced Manufacturing Technology*, 90:2491–2512, 2017.

[13] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1558–1566. PMLR, June 2016.

[14] Gustav Grund Pihlgren, Fredrik Sandin, and Marcus Liwicki. Improving image autoencoder embeddings with perceptual loss. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.

[15] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[17] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] Xiaoyu Liu, Huachen Gao, and Xiaowen Ma. Perceptual losses for self-supervised depth estimation. *Journal of Physics: Conference Series*, 1952(2):022040, jun 2021.

[19] A.M. Eskicioglu and P.S. Fisher. Image quality measures and their performance. *IEEE Transactions on Communications*, 43(12):2959–2965, 1995.

[20] Hamid Sheikh, Alan Bovik, Lawrence Cormack, and Zhou Wang. Live image quality assessment database.

[21] Eric Cooper Larson and Damon Michael Chandler. Most apparent distortion: full-reference image quality assessment and the role of strategy. *Journal of Electronic Imaging*, 19(1), 2010.

[22] Nikolay Ponomarenko, Oleg Ieremeiev, Vladimir Lukin, Karen Egiazarian, Lina Jin, Jaakko Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, and C.-C. Jay Kuo. Color image database tid2013: Peculiarities and preliminary

results. In *European Workshop on Visual Information Processing (EUVIP)*, pages 106–111, 2013.

[23] Agata Mosinska, Pablo Marquez-Neila, Mateusz Koziński, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3136–3145, 2018.

[24] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.

[25] Jake Snell, Karl Ridgeway, Renjie Liao, Brett D. Roads, Michael C. Mozer, and Richard S. Zemel. Learning to generate images with perceptual similarity metrics. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4277–4281, 2017.

[26] Jürgen Schmidhuber. Learning Factorial Codes by Predictability Minimization. *Neural Computation*, 4(6):863–879, 11 1992.

[27] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[29] Oskar Sjögren, Gustav Grund Pihlgren, Fredrik Sandin, and Marcus Liwicki. Identifying and mitigating flaws of deep perceptual similarity metrics. In *Proceedings of the Northern Lights Deep Learning Workshop 2023*, 2023.

[30] Gustav Grund Pihlgren, Fredrik Sandin, and Marcus Liwicki. Pretraining image encoders without reconstruction via feature prediction loss. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4105–4111, 2021.

[31] Gustav Grund Pihlgren. *Deep Perceptual Loss for Improved Downstream Prediction*. Licentiate thesis, Luleå tekniska universitet, 2021.

[32] Gustav Grund Pihlgren, Konstantina Nikolaidou, Prakash Chandra Chhipa, Nosheen Abid, Rajkumar Saini, Fredrik Sandin, and Marcus Liwicki. A systematic performance analysis of deep perceptual loss networks breaks transfer learning conventions. arXiv preprint, 2023.

[33] Gustav Grund Pihlgren, Fredrik Sandin, and Marcus Liwicki. Deep perceptual similarity is adaptable to ambiguous contexts. arXiv preprint, 2023.

[34] Kumar Shridhar, Ayushman Dash, Amit Sahu, Gustav Grund Pihlgren, Pedro Alonso, Vinaychandran Pondenkandath, György Kovács, Foteini Simistira, and Marcus Liwicki. Subword semantic hashing for intent classification on small datasets. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2019.

[35] Johan Edstedt, Amanda Berg, Michael Felsberg, Johan Karlsson, Francisca Benavente, Anette Novak, and Gustav Grund Pihlgren. Vidharm: A clip based dataset for harmful content detection. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1543–1549, 2022.

[36] Prakash Chandra Chhipa, Richa Upadhyay, Gustav Grund Pihlgren, Rajkumar Saini, Seiichi Uchida, and Marcus Liwicki. Magnification prior: A self-supervised method for learning representations on breast cancer histopathological images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2717–2727, January 2023.

[37] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[38] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Qian Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *Computer*, 28(9):23–32, 1995.

[39] Greg Pass and Ramin Zabih. Comparing images using joint histograms. *Multimedia systems*, 7:234–240, 1999.

[40] Mohammed Abdulameer Aljanabi, Noor Abdalrazak Shnain, and Song Feng Lu. An image similarity measure based on joint histogram — entropy for face recognition. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1626–1631, 2017.

[41] Wynne Hsu, ST Chua, and HH Pung. An integrated color-spatial approach to content-based image retrieval. In *Proceedings of the third ACM international conference on Multimedia*, pages 305–313, 1995.

[42] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003.

[43] Asmhan F Hassan, Dong Cailin, and Zahir M Hussain. An information-theoretic image quality measure: comparison with statistical similarity. 2014.

[44] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. Fsim: A feature similarity index for image quality assessment. *IEEE Transactions on Image Processing*, 20(8):2378–2386, 2011.

[45] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[46] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[47] Michael Calonder, Vincent Lepetit, Christoph Strecha, and FP Brief. Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision*, pages 778–792, 2010.

[48] Feng Qi, Debin Zhao, and Wen Gao. Reduced reference stereoscopic image quality assessment based on binocular perceptual information. *IEEE Transactions on Multimedia*, 17(12):2338–2344, 2015.

[49] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[50] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[51] Silvan Heller, Luca Rossetto, and Heiko Schuldt. The PS-Battles Dataset – an Image Collection for Image Manipulation Detection. 2018.

[52] Konstantin Schall, Kai Uwe Barthel, Nico Hezel, and Klaus Jung. Gpr1200: A benchmark for general-purpose content-based image retrieval. In *MultiMedia Modeling: 28th International Conference, MMM 2022, Phu Quoc, Vietnam, June 6–10, 2022, Proceedings, Part I*, page 205–216, Berlin, Heidelberg, 2022. Springer-Verlag.

[53] Christian J. Van den Branden Lambrecht and Olivier Verscheure. Perceptual quality measure using a spatiotemporal model of the human visual system. In *Digital Video Compression: Algorithms and Technologies 1996*, volume 2668, pages 450 – 461. International Society for Optics and Photonics, SPIE, 1996.

[54] Eric A. Silva, Karen Panetta, and Sos S. Agaian. Quantifying image similarity using measure of enhancement by entropy. In Sos S. Agaian and Sabah A. Jassim, editors, *Mobile Multimedia/Image Processing for Military and Security Applications 2007*, volume 6579, page 65790U. International Society for Optics and Photonics, SPIE, 2007.

[55] Abhijay Ghildyal and Feng Liu. Shift-tolerant perceptual similarity metric. In *Computer Vision – ECCV 2022*, pages 91–107, Cham, 2022. Springer Nature Switzerland.

[56] Jayesh Ruikar and Saurabh Chaudhury. Nits-iqa database: A new image quality assessment database. *Sensors*, 23(4), 2023.

[57] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, 2006.

[58] Mu Zhu. Recall, precision and average precision. Technical report, Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, 2004.

[59] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR 2011*, pages 97–104. IEEE, 2011.

[60] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. Raise: A raw images dataset for digital image forensics. In *Proceedings of the 6th ACM multimedia systems conference*, pages 219–224, 2015.

[61] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1122–1131, 2017.

[62] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, 2001.

[63] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 237–246, 2017.

[64] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: a large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[65] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.

[66] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[67] Manoj Kumar, Neil Houlsby, Nal Kalchbrenner, and Ekin Dogus Cubuk. Do better imagenet classifiers assess perceptual similarity better? *Transactions on Machine Learning Research*, 2022.

[68] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[69] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[70] Radoslaw Martin Cichy, Aditya Khosla, Dimitrios Pantazis, Antonio Torralba, and Aude Oliva. Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific reports*, 6(1):1–13, 2016.

[71] Michael Eickenberg, Alexandre Gramfort, Gaël Varoquaux, and Bertrand Thirion. Seeing it all: Convolutional network layers map the function of the human visual system. *NeuroImage*, 152:184–194, 2017.

[72] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1485–1488. Association for Computing Machinery, 2010.

[73] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations ICLR*, 2015.

[74] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[75] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[76] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[77] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[78] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[79] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. arXiv preprint, 2014.

[80] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[81] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv preprint, 2016.

[82] Prakash Chandra Chhipa. *Self-supervised Representation Learning for Visual Domains Beyond Natural Scenes*. Licentiate thesis, Luleå tekniska universitet, 2023.

[83] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[84] Dana H Ballard. Modular learning in neural networks. In *AAAI*, pages 279–284, 1987.

[85] Zhengwei Wang, Qi She, and Tomás E. Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Comput. Surv.*, 54(2), feb 2021.

[86] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[87] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at 2^{nd} International Conference on Learning Representations ICLR*, 2014.

[88] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *Deep Learning Workshop, 32^{nd} International Conference on Machine Learning*, 2015.

[89] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint, 2013.

[90] Alex Serban, Erik Poll, and Joost Visser. Adversarial examples on object recognition: A comprehensive survey. *ACM Comput. Surv.*, 53(3), jun 2020.

[91] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *7^{th} International Conference on Learning Representations, ICLR*, 2019.

[92] Xin Feng, Youni Jiang, Xuejiao Yang, Ming Du, and Xin Li. Computer vision algorithms and hardware implementations: A survey. *Integration*, 69:309–320, 2019.

[93] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[94] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in neural information processing systems*, pages 658–666, 2016.

[95] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[96] Mohammad Saeed Rad, Behzad Bozorgtabar, Urs-Viktor Marti, Max Basler, Hazim Kemal Ekenel, and Jean-Philippe Thiran. SROBB: targeted perceptual loss for single image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[97] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[98] Dongdong Xu, Yongcheng Wang, Xin Zhang, Ning Zhang, and Sibo Yu. Infrared and visible image fusion using a deep unsupervised framework with perceptual loss. *IEEE Access*, 8:206445–206458, 2020.

[99] Mehdi S. M. Sajjadi, Bernhard Scholkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[100] Qingsong Yang, Pingkun Yan, Yanbo Zhang, Hengyong Yu, Yongyi Shi, Xuanqin Mou, Mannudeep K. Kalra, Yi Zhang, Ling Sun, and Ge Wang. Low-dose CT image denoising using a generative adversarial network with wasserstein distance and perceptual loss. *IEEE Transactions on Medical Imaging*, 37(6):1348–1357, 2018.

[101] Zhenjie Chai, Kang Zhou, Jianlong Yang, Yuhui Ma, Zhi Chen, Shenghua Gao, and Jiang Liu. Perceptual-assisted adversarial adaptation for choroid segmentation in optical coherence tomography. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 1966–1970, 2020.

[102] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *Computer Vision – ECCV 2014*, pages 584–599. Springer International Publishing, 2014.

[103] Alice J. O'Toole, Fang Jiang, Hervé Abdi, and James V. Haxby. Partially distributed representations of objects and faces in ventral temporal cortex. *Journal of Cognitive Neuroscience*, 17(4):580–590, 04 2005.

[104] Linda B Smith and Diana Heise. Perceptual similarity and conceptual structure. In *Advances in psychology*, volume 93, pages 233–272. Elsevier, 1992.

[105] Francesc Alías, Joan Socoró, and Xavier Sevillano. A review of physical and perceptual feature extraction techniques for speech, music and environmental sounds. *Applied Sciences*, 6(5):143, May 2016.

[106] Ya-Liang Chang, Kuan-Ying Lee, Po-Yu Wu, Hung-yi Lee, and Winston Hsu. Deep long audio inpainting. arXiv preprint, 2019.

[107] Pranay Manocha, Zeyu Jin, Richard Zhang, and Adam Finkelstein. Cdpam: Contrastive learning for perceptual audio similarity. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 196–200, 2021.

[108] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations ICLR*, 2013.

[109] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[110] Qilu Jiao and Shunyao Zhang. A brief survey of word embedding and its recent development. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 5, pages 1697–1701, 2021.

[111] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR*, 2021.

[112] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollar, and Ross Girshick. Early convolutions help transformers see better. In *Advances in Neural Information Processing Systems*, volume 34, pages 30392–30400. Curran Associates, Inc., 2021.

[113] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. In *4th International Conference on Learning Representations ICLR*, 2016.

[114] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[115] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017.

[116] Deepak Pathak, Ross Girshick, Piotr Dollar, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[117] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *5th International Conference on Learning Representations ICLR*, 2017.

[118] Pulkit Agrawal, João Carreira, and Jitendra Malik. Learning to see by moving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 37–45, 2015.

[119] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016.

[120] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2794–2802, 2015.

[121] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *Computer Vision – ECCV 2016*, pages 649–666, Cham, 2016. Springer International Publishing.

[122] Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H. Adelson, and William T. Freeman. Visually indicated sounds. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2405–2413, 2016.

[123] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[124] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[125] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014.

[126] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012.

[127] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, 2010.

[128] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.

[129] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013.

[130] Christian Wiedemann, Christian Heipke, Helmut Mayer, and Olivier Jamet. Empirical evaluation of automatically extracted road axes. *Empirical evaluation techniques in computer vision*, 12:172–187, 1998.

[131] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2450–2462. Curran Associates, Inc., 2018.

[132] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[133] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[134] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. arXiv preprint, 2018.

[135] Michele Alberti, Mathias Seuret, Rolf Ingold, and Marcus Liwicki. A pitfall of unsupervised pre-training. arXiv preprint, 2017.

[136] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020.

[137] Fabio Carrara, Rudy Becarelli, Roberto Caldelli, Fabrizio Falchi, and Giuseppe Amato. Adversarial examples detection in features distance spaces. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[138] Jim Nilsson and Tomas Akenine-Möller. Understanding ssim. arXiv preprint, 2020.

[139] Michele Alberti, Vinaychandran Pondenkandath, Marcel Wursch, Manuel Bouillon, Mathias Seuret, Rolf Ingold, and Marcus Liwicki. Are you tampering with my data? In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[140] Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen. How convolutional neural networks see the world—a survey of convolutional neural network visualization methods. *Mathematical Foundations of Computing*, 1(2):149, 2018.

[141] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 20(184):1–25, 2019.

[142] Olivier Hénaff and Eero Simoncelli. Geodesics of learned representations. In *4th International Conference on Learning Representations ICLR*, 2016.

[143] Richard Zhang. Making convolutional networks shift-invariant again. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7324–7334. PMLR, 09–15 Jun 2019.

[144] Osman Semih Kayhan and Jan C. van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[145] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[146] Kede Ma, Zhengfang Duanmu, and Zhou Wang. Geometric transformation invariant image quality assessment using convolutional neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6732–6736, 2018.

[147] Sabyasachi Saha and Sandip Sen. Agent based framework for content based image retrieval. In *Papers from the 2004 AAAI Spring Symposium*. AAAI Press, 2004.

[148] Luca Rossetto, Claudiu Tănase, and Heiko Schuldt. Dealing with ambiguous queries in multimodal video retrieval. In *MultiMedia Modeling*, pages 898–909, Cham, 2016. Springer International Publishing.

[149] Cheng Luo, Yiqun Liu, Min Zhang, and Shaoping Ma. Query ambiguity identification based on user behavior information. In *Information Retrieval Technology*, pages 36–47, Cham, 2014. Springer International Publishing.

[150] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[151] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1), 2021.

[152] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[153] Alex Andonian, Taesung Park, Bryan Russell, Phillip Isola, Jun-Yan Zhu, and Richard Zhang. Contrastive feature loss for image prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1934–1943, 2021.

[154] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics*, 41(4), jul 2022.

[155] Chen Feng and Ioannis Patras. Adaptive soft contrastive learning. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2721–2727, 2022.

[156] Sandra Rousseau, Giuseppe Catalano, and Cinzia Daraio. Can we estimate a monetary value of scientific publications? *Research Policy*, 50(1):104116, 2021.

[157] Joseph McDonald, Baolin Li, Nathan Frey, Devesh Tiwari, Vijay Gadepally, and Siddharth Samsi. Great power, great responsibility: Recommendations for reducing energy for training language models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1962–1970, Seattle, United States, July 2022. Association for Computational Linguistics.

[158] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696, April 2020.

[159] B.W. Ang and Bin Su. Carbon emission intensity in electricity production: A global analysis. *Energy Policy*, 94:56–63, 2016.

[160] Camilla Dellby and Susanne Enmalm. Årlig energistatistik (el, gas och fjärrvärme) 2019. Technical report, Statistiska centralbyrån, 2020.

[161] Vanessa Mehlin, Sigurd Schacht, and Carsten Lanquillon. Towards energy-efficient deep learning: An overview of energy-efficient approaches along the deep learning lifecycle. arXiv preprint, 2023.

[162] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134:75–88, 2019.

[163] Jake Silberg and James Manyika. Notes from the ai frontier: Tackling bias in ai (and in humans). https://www.mckinsey.com/featured-insights/artificial-intelligence/tackling-bias-in-artificial-intelligence-and-in-humans, 2019.

[164] Ryan Steed and Aylin Caliskan. Image representations learned with unsupervised pre-training contain human-like biases. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 701–713, 2021.

[165] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, page 335–340, New York, NY, USA, 2018. Association for Computing Machinery.

[166] High-Level Expert Group on Artificial Intelligence. Ethics guidelines for trustworthy ai, 2019.

[167] Supatsara Wattanakriengkrai, Bodin Chinthanet, Hideaki Hata, Raula Gaikovina Kula, Christoph Treude, Jin Guo, and Kenichi Matsumoto. Github repositories with links to academic papers: Public access, traceability, and evolution. *Journal of Systems and Software*, 183:111–117, 2022.

[168] Shiv Ram Dubey. A decade survey of content based image retrieval using deep learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2687–2704, 2022.

[169] Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. arXiv preprint, 2022.

[170] Yan Wang, Wei-Lun Chao, Kilian Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. arXiv preprint, 2019.

[171] Imtiaz Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed. Laplacian regularized few-shot learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11660–11670. PMLR, 13–18 Jul 2020.

[172] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019.

[173] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.

Acronyms

2AFC Two Alternative Forced Choice. 20–24, 36, 37, 42, 44, 46–52, 54, 70–72, 77, 86, 88, 90–93, 96, 106

2D 2-Dimensional. 34, 35, 65

AI Artificial Intelligence. 2, 31, 103

BAPPS Berkeley-Adobe Perceptual Patch Similarity. 22–24, 36, 40, 44, 46–50, 52, 54, 55, 58, 59, 70, 77, 78, 86–88, 91, 92, 95–97, 104, 105

BCE Binary Cross-Entropy. 16, 36

BRIEF Binary Robus Independent Elementary Features. 19

CNN Convolutional Neural Network. 6, 26, 27, 29, 32–36, 38, 40, 42, 46, 51, 53, 65, 66, 68–71, 73, 78, 85, 90, 100

CSI Color-Spatial Information. 18

CSIQ Categorical Image Quality Database. 22, 104

DISTS Deep Image Structure and Texture Similarity. 75, 76

DNN Deep Neural Network. 2, 9, 26, 31, 32, 39, 40, 64, 103

FSIM Feature Similarity. 18

FSM Feature-Based Structural Measure. 18

GAN Generative Adversarial Network. 30, 33

GPU Graphics Processing Unit. 101, 102

ICA Independent Component Analysis. 23

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 26

IQA Image Quality Assessment. 4, 22

JND Just Noticable Differences. 20–24, 44, 46, 47, 51, 52, 54, 70, 77, 88, 91, 92

LIVE LIVE Image Quality Assessment Database. 22

LPIPS Learned Perceptual Image Patch Similarity. 36–38, 42, 46, 48, 55, 59, 61, 64, 70, 76, 85, 86, 95, 104, 106

MAE Mean Average Error. 15, 21

MLP Multilayer Perceptron. 46

MOS Mean Opinion Score. 20–22

MRD Massachusetts Road Dataset. 45

MSE Mean Squared Error. 15, 17, 21, 32, 36

PCA Principal Component Analysis. 23

PSNR Peak Signal-to-Noise Ratio. 45

SIFT Scale Invariant Feature Transform. 19, 23

SSIM Structural Similarity Index Measure. 6, 18, 19, 45, 64, 73

SURF Speeded Up Robust Features. 19, 23

SVHN Street View House Numbers. 45, 46, 57, 58, 87–94

TID2013 Tampere Image Database 2013. 22, 48, 50, 104

VAE Variational Autoencoder. 33

ViT Vision Transformer. 40, 42, 49, 50, 55

Department of Computer Science, Electrical and Space Engineering
Division of EISLAB

ISSN 1402-1757
ISBN 978-91-8048-325-4 (print)
ISBN 978-91-8048-326-1 (pdf)

LULEÅ
UNIVERSITY
OF TECHNOLOGY