Enhancing Deep Learning Model Robustness: Insights from Out of Distribution Data Augmentation and an Innovative Image Compression Technique

Zahra Golpayegani

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

January 2024

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By:	**Zahra Golpayegani**
Entitled:	**Enhancing Deep Learning Model Robustness: Insights from Out of Distribution Data Augmentation and an Innovative Image Compression Technique**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

Dr. Abdessamad Ben Hamza
Chair

Dr. Abdessamad Ben Hamza
Examiner

Dr. Yong Zeng
Examiner

Dr. Nizar Bouguila
Supervisor

Approved by

Chun Wang, Chair
Department of Concordia Institute for Information Systems Engineering

2024

Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Enhancing Deep Learning Model Robustness: Insights from Out of Distribution Data Augmentation and an Innovative Image Compression Technique

Zahra Golpayegani

Deep learning models excel when tested on images within their training distribution. However, introducing minor perturbations like noise or blurring to the model's input image and presenting it with out-of-distribution (OOD) data can significantly reduce accuracy, limiting real-world applicability. While data augmentation enhances model robustness against OOD data, there is a gap in comprehensive studies on augmentation types and their impact on OOD robustness.

A common belief suggests that augmenting datasets to bias models towards shape-based features improves OOD robustness for convolutional neural networks trained on ImageNet. However, our evaluation of 39 augmentations challenges this belief, showing that an augmentation-induced increase in shape bias does not necessarily correlate with higher OOD robustness. Analyzing results, we identify biases in ImageNet that can be mitigated through appropriate augmentation. Contrary to expectations, our evaluation reveals no inherent trade-off between in-domain accuracy and OOD robustness. Strategic augmentation choices can simultaneously enhance both.

Model performance is influenced not only by perturbations but also by the image compression format. Efficient algorithms for image compression play a crucial role in managing costs associated with data storage. We propose an innovative region-based lossy image compression method named PatchSVD, leveraging the Singular Value Decomposition (SVD) algorithm. Experimental results demonstrate that PatchSVD surpasses SVD-based image compression across three common image compression metrics. Furthermore, we conduct a comparative analysis of compression artifacts between PatchSVD, JPEG, and SVD-based compression, revealing scenarios where PatchSVD artifacts are preferable over both JPEG and SVD artifacts.

Acknowledgments

Completing this thesis has been a journey filled with learning, challenges, and growth. At this juncture, I would like to express my sincere gratitude to those who have been instrumental in making this endeavor possible. First and foremost, I extend my heartfelt appreciation to my thesis supervisor, Professor Nizar Bouguila, whose guidance, expertise, and unwavering support have been indispensable throughout this research journey. He provided valuable insights, constructive feedback, and encouragement, shaping the trajectory of this work. I am grateful to the members of my thesis committee for their time, expertise, and thoughtful contributions. Their feedback and suggestions significantly enriched the quality of this thesis.

Special thanks to all my colleagues at Zetane Systems who supported me with my research, especially Guillaume Hervé and Patrick St-Amant, whose collaboration and dedication were pivotal to the successful execution of the research projects incorporated into this thesis.

My appreciation extends to my family and friends for their unwavering encouragement, understanding, and moral support. Their belief in me has been a constant source of motivation.
Finally, I want to acknowledge the broader academic community and the inspiring researchers whose work laid the foundation for this thesis. To everyone who contributed, directly or indirectly, to the realization of this thesis, I am sincerely thankful for your support and collaboration.

Zahra Golpayegani

Contents

List of Figures **viii**

List of Tables **x**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement and Research Objectives 4

1.3 Contributions 5

2 Literature Review **7**

2.1 Convolutional Neural Networks 7

2.2 Out of Distribution (OOD) Robustness 8

2.3 Distribution Shifts 9

2.4 Data Augmentation 10

2.4.1 Existing Libraries for Data Augmentation 10

2.5 Image Encoding and Image Compression 11

2.5.1 JPEG Image Compression 12

2.5.2 SVD-based Image Compression 13

2.5.3 Other Approaches to Image Compression 16

3 Clarifying Myths About the Relationship Between Shape Bias, Accuracy, and Robustness **19**

3.1 Introduction 19

3.2 Related Works 22

3.3 Method 24

3.3.1 Shape bias 24

3.3.2 OOD robustness 24

3.4 Experiments 25

3.4.1 Dataset 25

3.4.2 Implementation details 26

3.4.3 Augmentations 27

3.5 Results and Discussion 27

3.5.1 Evaluating augmentations on the OOD Benchmark 27

3.5.2 Does data augmentation always improve OOD robustness? 28

3.5.3 Is there a trade-off between in-domain accuracy and OOD robustness? . . . 29

3.5.4 The relationship between shape bias and OOD robustness 30

3.6 Conclusion 30

4 PatchSVD: A Non-uniform SVD-based Image Compression Algorithm 32

4.1 Introduction 32

4.2 Preliminaries 35

4.3 Related Works 35

4.4 Method 37

4.4.1 PatchSVD Algorithm 37

4.4.2 Compression Ratio (CR) 39

4.4.3 Patch Size Lower Bound 40

4.5 Experiments 42

4.5.1 Datasets and Metrics 42

4.6 Results and Discussion 43

4.6.1 PatchSVD Performance Based on Patch Size 43

4.6.2 Image Compression Performance Comparison 43

4.6.3 Compression Artifacts 44

4.6.4 Choice of Score Function 44

4.7 Conclusion 44

5 Conclusion and Future Work **46**

5.1 Conclusion 46

5.2 Future Work 47

Bibliography **48**

List of Figures

Figure 2.1 This figure illustrates the steps involved in JPEG encoding. 13

Figure 2.2 The k-rank SVD approximation of an image, applied for various values of k, illustrates how the quality of the compressed image changes based on k. Note that for $k \geq 260$, indicated by the last row, almost no noise is visible. 16

Figure 3.1 This illustration shows how a ResNet-50 model with 89.06% in-domain accuracy fails when a small perturbation has been applied to the image. The label on top of each image is the name of the perturbation, and the label below each image is the predicted class. ”reference” is the clean image. 20

Figure 3.2 An overview of how different augmentations alter the reference (original) image. Best viewed in color. 27

Figure 3.3 Evaluation of OOD robustness for various augmentations on the OOD Benchmark dataset. The pink bar represents the Vanilla (baseline with no data augmentations) model. 29

Figure 3.4 This figure shows the relationship between the shape bias and the OOD robustness of models trained on augmented datasets. The evaluation has been performed on the OOD Benchmark dataset that contains two groups of datasets, called the Noise and the Style groups. 31

Figure 4.1 JPEG produces more compression artifacts in images containing text compared to the proposed method. The sample is taken from the IAM Handwriting Database and the image has been zoomed in 30 times the original size to better visualize the compression artifacts. 34

Figure 4.2 PatchSVD algorithm first applies low-rank SVD to the original image and subtracts the low-rank approximation from the original image (Figure 4.2(a)) to obtain Δ (Figure 4.2(b)). Then, by applying a score function, PatchSVD calculates the patches that contain more information according to Δ (4.2(c)) to create the final compressed image (see Figure 4.2(d)). 34

Figure 4.3 This figure shows the effect of patch size on the performance of the PatchSVD algorithm across CLIC (first row) and Kodak (second row) datasets. 36

Figure 4.4 This figure compares SSIM, PSNR, and MSE metrics for PatchSVD, JPEG, and SVD image compression algorithms on the CLIC dataset with a patch size of 10 (top row) and the Kodak dataset with a patch size of 16 (bottom row). 39

Figure 4.5 PatchSVD, JPEG, and SVD compression algorithms applied to two image samples from the Kodim dataset show the compression artifacts produced by each algorithm. As you can see, PatchSVD produces more sharp edges which results in perfectly legible text even after the image has been greatly compressed. 41

Figure 4.6 This figure illustrates PatchSVD algorithm performance using different score functions, namely, standard deviation, mean, and max. To compare, SSIM, PSNR, and MSE metrics are used. It is demonstrated that standard deviation has a slightly better performance compared to others. 43

List of Tables

Table 3.1 This table shows a few augmentation examples that can be applied to the ImageNet-100 dataset and increase both in-domain accuracy and OOD robustness, simultaneously. Avg. OOD Robustness is the weighted average of OOD-Noise and OOD-Style robustness values. 30

Chapter 1

Introduction

1.1 Motivation

Deep learning has shown remarkable advancements and continuous improvement since it emerged, particularly in recent years with the advent of technology, the adoption of powerful computation resources such as GPUs and TPUs, the development of social media, and the more common usage of sensors and IoT devices which led to enormous amounts of data. While researchers have designed novel model architectures and training procedures, many tasks are still not solved entirely. We start this work by studying a significant challenge that not only researchers in academia are now facing, but also is keeping industries from pushing the envelope further; model robustness to out-of-distribution (OOD) data.

Supervised image classification is usually treated as a problem that includes two pieces; the dataset and the model. The dataset is a collection of train and test subsets, and it is usually assumed that the train and test subsets are independent and identically distributed. However, in real-world scenarios, this assumption does not always hold.

For example, imagine we are training a model for a self-driving car that classifies the type of car(s) in front of the vehicle for each frame in a video input provided by cameras. If we only train the model on training data collected on a sunny day, the model will undoubtedly perform poorly in rainy weather because the underlying distribution of the target data is now different from the training data. One might argue that we can mitigate this problem by adding data collected in rainy weather to our training set. While this argument is valid to some extent, it does not solve the problem completely. Suppose we augment the training data with samples taken in rainy weather; what should we do in case of snow, fog, frost, tornado, or a blizzard? Do we include all of them in data augmentation? What is the effect of the data augmentation on the model performance? Can augmenting our dataset with samples from snowy weather worsen the performance of the model in rainy conditions? What about the types of data shifts that we cannot even predict from before? What if we train our model on data collected in the streets of Canada, and now we want to use the model in Germany? How much more data do we need to collect? What if we want to use a new camera that has different intrinsic parameters and settings than the camera we used before? What happens if there is a crack on the camera lens that we have not noticed? All of these scenarios and many more are crucial to study thoroughly before a self-driving car on the streets could be considered safe and reliable.

While safety in our example of self-driving cars is paramount, it seems like some have overlooked it. Many predictions were made about when self-driving cars would be on the streets by CEOs of self-driving car companies and none of them turned out to be true. The following quotes are a couple of examples[1]:

> "Five or six years from now [2014] we will be able to achieve true autonomous driving where you could literally get in the car, go to sleep and wake up at your destination.[2]"
>
> CEO, Elon Musk (Tesla)

> "It will take no more than 4 years [since 2017] to have fully autonomous cars on the road.[3]"
>
> CEO, Jensen Huang (NVIDIA)

One of the main reasons why such advancements are impeded is the lack of studies on out-of-distribution data and how researchers can detect and mitigate this challenge. There are many unanswered research questions about common techniques used for OOD data mitigation, such as data

[1]See a full list of predictions at https://www.driverless-future.com/?page_id=384.

[2]https://www.huffpost.com/entry/tesla-driverless-cars_n_5990136

[3]https://www.reuters.com/article/us-nvidia-ai-chips/chipmaker-nvidias-ceo-sees-fully-autonomous-cars-within-4-years-idUSKBN1CV192?feedType=RSS&feedName=technologyNews

augmentation. While data augmentation has been used for over a decade now by many researchers, the effect of one type of data augmentation on the performance of the model on other data distributions, the relationship between accuracy on clean data and OOD data robustness, the effect of introducing bias toward identifying shape rather than texture to a vision model on robustness, and many other questions have not been answered yet. Motivated by the insufficient understanding of this prevalent issue, we conducted studies to address several research questions pertaining to model robustness to out-of-distribution (OOD) data which is presented in detail in Chapter 3.

Aside from the visible changes in image data that are intentionally or unintentionally applied, the image compression format also plays a crucial role in the image appearance. Visible image compression artifacts can have negative effects on how the image is perceived, be it by humans or by machines. Although compression artifacts are inevitable when greater compression ratios are required, having granular control over how the artifacts are distributed would be beneficial. For instance, if we know beforehand that almost all of the information an image contains is concentrated in the 20% of the pixels located in the center of the image, applying compression at lower rates in those 20% pixels would result in fewer artifacts in the area where the information is concentrated, hence, more useful information will be preserved.

An example of this scenario is in medical imaging applications. Compressing diagnostic medical scans is a challenging task because significant information losses could result in incorrect diagnoses. However, by leveraging expert knowledge to identify the specific regions in the image associated with symptoms, we can strategically apply compression at varying rates tailored to the significance of each region.

Nevertheless, expert knowledge is not always readily feasible. It is usually costly to gather such information about images and its usefulness may be limited by the diverse locations of objects across different images. Some deep learning-based methods can extract the important regions in an image to some extent. However, applying deep learning requires sufficient training data, and the model should be trained from scratch per dataset. Moreover, the trained model should be accessible when applying the compression algorithm, which requires even more storage, and computation costs are applicable both during the training phase and the inference phase. Overall, using deep learning is not always cost-efficient.

Fortunately, the important patterns in an image can be extracted using matrix operations, such as Singular Value Decomposition (SVD). SVD provides a cost-effective image compression solution by retaining only the essential components from the input matrix. However, it applies globally to an image, i.e., it compresses all the regions uniformly. In many applications, this approach is less desirable because the information content varies across different regions of the image, and uniform compression leads to a rapid deterioration in image quality as the compression ratio increases. Applying varied levels of image compression through SVD across different regions based on their complexity allows us to preserve more information in intricate patches of the image. This idea was the main inspiration in designing a novel SVD-based non-uniform image compression algorithm, called PatchSVD. In Chapter 4, we discuss PatchSVD in detail and compare it with SVD-based and JPEG image compression.

1.2 Problem Statement and Research Objectives

Given training set and test set form distributions p and q respectively, where $p \neq q$, we study the following questions about model robustness:

(1) Does data augmentation always improve OOD robustness?

(2) Is there a trade-off between in-domain accuracy and OOD robustness?

(3) What is the relationship between shape bias and OOD robustness?

(4) Does augmenting a dataset with one augmentation type have any effect on the robustness of the model to other transformations?

By answering the above questions, we hope we understand the challenge of model robustness deeper and accelerate research in this field. We believe that the lack of study on model robustness keeps industries and researchers from unlocking the true potentials of machine learning and artificial intelligence.

The next research objective we investigate in this study is how can we achieve non-uniform image compression to keep important information in the image without having access to any training data using the properties of Singular Value Decomposition (SVD). We also study the problem of reducing compression artifacts around high-contrast lines in an image, which is a typical issue with JPEG compression. We explore a novel approach to compressing an image using SVD by applying SVD in local patches in an image. This approach is explainable out of the box unlike typical deep learning approaches, requires no training, and works with minimal computation resources using simple mathematical operations.

1.3 Contributions

We study the problem of model robustness to OOD data in image classification through a diverse set of image transformations on a large dataset. To run our experiments, we implemented a toolkit that offers a wide variety of transformations to simulate different scenarios in input images. Using this toolkit, we were able to study some general research questions about model robustness. Our answers to these questions give insights to researchers and developers when they want to design experiments or train robust models.

In our next work, we design a novel image compression algorithm that allows us to have granular control over how much data is retained in each region of an image which enables non-uniform image compression. Through extensive experiments, we compare our algorithm with SVD-based and JPEG image compression on two datasets both in terms of performance and compression artifacts.

We shared our findings with the community in the following two research papers:

- In the first paper, presented in Chapter 3, we designed and ran extensive experiments to see if there is any trade-off between in-distribution accuracy and out-of-distribution robustness. We found out that such a trade-off does not necessarily exist and choosing the correct transformation types can result in a simultaneous increase in both in-distribution accuracy and OOD robustness. Then, we investigated if there are biases in the ImageNet-1K dataset and we suggested that these biases can be mitigated by data augmentation techniques. Moreover, according to our experiment results, we rejected the hypothesis existing in the literature that claimed an increase in shape bias will result in an increase in model robustness. Last but not least, we showed that data augmentation will sometimes harm the model OOD robustness instead of improving it and data augmentation should not be applied blindly. This paper has been published in the Proceedings of the 20th Conference on Robots and Vision (CRV 2023) by IEEE Golpayegani, St-Amant, and Bouguila (2023)[4].

- In the second work, presented in Chapter 4, we designed PatchSVD, a region-based image compression algorithm, to achieve non-uniform image compression based on patch importance without applying deep learning and only using simple mathematical operations. In our experiments, we show that PatchSVD outperforms SVD-based image compression in terms of SSIM, PSNR, and MSE metrics on both Kodak and CLIC datasets. Also, we illustrate that in some applications where compression artifacts need to be minimized in some specific regions, for example, around high-contrast edges, PatchSVD outperforms JPEG. This paper has been accepted to be published in the Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024) and has been nominated for the Best Paper Award.

Finally, in Chapter 5, we conclude this work and discuss some future directions.

[4]The third chapter of this thesis is published in IEEE, 2023 20th Conference on Robots and Vision (CRV). In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Concordia University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

Chapter 2

Literature Review

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were first introduced in LeCun, Bottou, Bengio, and Haffner (1998) as a method that can efficiently be used to detect handwritten digits. Authors showed that CNNs outperform all other methods for handwritten digit recognition on the MNIST LeCun and Cortes (2010) dataset. Although the superior performance of CNNs was demonstrated in this work, training CNNs was not very popular until the late 2000s because it required access to powerful hardware, e.g., GPUs. As GPUs became more accessible to researchers and large-scale datasets such as ImageNet Deng et al. (2009) were introduced, more sophisticated CNN architectures were created in the literature, including ResNet He, Zhang, Ren, and Sun (2016), GoogLeNet Szegedy et al. (2014), and VGGNet Simonyan and Zisserman (2015). In Simonyan and Zisserman (2015), the impact of the network depth on model performance was studied and it was demonstrated that deeper networks outperform shallower ones on ImageNet. Then, He et al. (2016) followed the same principle and introduced Residual Connections to overcome the Vanishing Gradient problem Rumelhart, Hinton, and Williams (1986) and introduced ResNets with over 100 layers.

As the networks become deeper, the number of trainable parameters and the computation costs increase. One approach to train deep models faster while avoiding over-fitting on small datasets is to use Transfer Learning. Using this technique, instead of training a model from scratch, a pre-trained version of the model which has been trained on a large-scale dataset will be used in the initialization stage. The pre-trained model can be used as a feature extractor or fine-tuned further on the target dataset. Many works have used Transfer Learning since the mid 2010s; Simonyan and Zisserman (2015), Zeiler and Fergus (2014), Donahue et al. (2014), Sharif Razavian, Azizpour, Sullivan, and Carlsson (2014), Chatfield, Simonyan, Vedaldi, and Zisserman (2014) are some of the early use-cases.

2.2 Out of Distribution (OOD) Robustness

In supervised learning, we assume that the training dataset includes (x, y) samples from a large set of images and labels (X, Y) with distribution p, and the test dataset contains (x', y') which are in (X', Y') sets of images and labels, respectively, following distribution q. Ideally, $p = q$, i.e., training data and test data are sampled from the same distribution. This is the assumption we make in supervised learning. However, in real-world use cases, this is rarely the case. Since usually $p \neq q$, a model trained on training data with distribution p is not able to generalize well to the test data of distribution q. In this case, the test data that has a different distribution than the training data is called Out-Of-Distribution (OOD) data.

A model that is able to generalize to OOD samples is called a Robust Model and the metric that measures the extent to which the model generalizes to OOD data is Model Robustness. There is no established method that measures OOD robustness in the literature, which makes the research on robustness even more challenging. However, one of the most common methods to measure robustness is calculating the accuracy of the model on a perturbed set of images. It is worth mentioning that if the test data comes from the same distribution as the training data, we call it in-distribution (ID) data. The performance of the model on ID data is usually measured by the classic accuracy metric.

To use a model in real-world applications, it is imperative to ensure the model is robust against OOD samples. There are a few techniques that focus on improving the OOD robustness of models. While some methods focus on detecting OOD data and abstaining from making predictions when OOD data is present Bartlett and Wegkamp (2008); Charoenphakdee, Cui, Zhang, and Sugiyama

(2021); Chow (1970); Franc and Prusa (2019); Schreuder and Chzhen (2021); Thulasidasan, Bhattacharya, Bilmes, Chennupati, and Mohd-Yusof (2019), others improve robustness by introducing defense strategies against OOD data Cohen, Rosenfeld, and Kolter (2019); I. J. Goodfellow, Shlens, and Szegedy (2015); Hendrycks and Dietterich (2019); Kannan, Kurakin, and Goodfellow (2018); Madry, Makelov, Schmidt, Tsipras, and Vladu (2017); S. Zheng, Song, Leung, and Goodfellow (2016) or using data augmentation which is a simple yet effective technique to improve model robustness (see Section 2.4).

2.3 Distribution Shifts

As mentioned previously, the test data is called out-of-distribution or OOD when its distribution is different than the distribution of the training data. In the case of OOD data, the assumption that the samples in the dataset are $i.i.d.$, i.e., are independent and identically distributed, does not hold. Therefore, off-the-shelf models will perform poorly if this challenge is not addressed correctly.

There are three main types of data distribution shifts: covariate shift, label shift, and concept drift. Covariate shift is the most common data distribution shift and is our main focus of study.

Suppose a classical supervised learning problem where we have images and labels from sets X and Y, respectively. Classifying an image can be represented by $P(Y|X)$, and the probability density function of the image (input) and the label (output) can be modeled by $P(X)$ and $P(Y)$. When there is a change in $P(X)$ but $P(Y|X)$ remains unchanged, we face a covariate shift Shen et al. (2021); Shimodaira (2000). An example of this is when we increase the brightness of an image of a “cat”, but the label is still “cat”. Some of the possible reasons why covariate shifts might happen are biases in the sampling process or missing data Kouw and Loog (2018).

Another type of data distribution shift that has gained more attention in recent years is label shift Azizzadenesheli, Liu, Yang, and Anandkumar (2019); Garg, Wu, Balakrishnan, and Lipton (2020); Lipton, Wang, and Smola (2018); Zhao, Liu, Anandkumar, and Yue (2021). Label shift happens when $P(X|Y)$ is unchanged but $P(Y)$ changes. For instance, during a pandemic (e.g., Covid-19), while the symptoms of a patient (e.g., cough) might not change compared to earlier, $P(Y)$ (the patient has Covid-19) will increase for that type of outbreak.

Last but not least, there is concept drift which happens when $P(Y|X)$ changes but $P(X)$ remains unchanged Gama, Žliobaitė, Bifet, Pechenizkiy, and Bouchachia (2014); Schlimmer and Granger (1986). For example, a word may have different meanings today compared to ancient times.

In this work, by out-of-distribution data, we mean distribution shifts that are caused by covariate shifts.

2.4 Data Augmentation

Data augmentation Ciregan, Meier, and Schmidhuber (2012); Krizhevsky, Sutskever, and Hinton (2012); Simard, Steinkraus, Platt, et al. (2003) is one of the most intuitive ways to improve model generalizability and works by artificially increasing the size of the training data. The newly added samples are usually created by applying a transformation on the input image and merging the transformed image with the original data while keeping the image label unchanged.

One can write unlimited functions to perform data augmentation. However, adding more data to the training set increases computation requirements and if the data is irrelevant, it may hinder the learning process by making the training task more difficult for the model.

Ideally, we know what data augmentation functions will benefit the model the most. However, this cannot be readily computed because of the complexity of possible functions and the lack of knowledge about the ways in which the source and target domains are different during training.

Data augmentation can be applied using several techniques including but not limited to geometric, image-level, and patch-level transformation-based algorithms Krizhevsky et al. (2012), style transfer X. Zheng, Chalasani, Ghosal, Lutz, and Smolic (2019), and generative models Bowles et al. (2018).

2.4.1 Existing Libraries for Data Augmentation

Some work has been done on designing and creating data augmentation libraries in the literature. “imgaug”, as introduced in Jung et al. (2020), is a library that provides more than 179 image augmentations with some features including the ability to apply the augmentations to only a subset of images, design augmentation pipelines, and perform augmentations with a given probability. A drawback of imgaug is its inefficient speed performance.

In Buslaev et al. (2020), authors have created a library called “Albumentations” that provides more than 70 image augmentation functions. Albumentations provides most of the features that imgaug has, such as augmentation pipelines and applying augmentations with a given probability. Moreover, authors claim that Albumentations is faster than imgaug and some other commonly used image augmentation tools on the most commonly used image transformations. Albumentations is an augmentation library only and does not provide any insights into the robustness of a model when evaluated against augmentations.

“AugLy” Papakipos and Bitton (2022) is another framework that can be used for data augmentation. An important feature of AugLy is supporting multiple data modalities, including audio, image, text, and video. Some of the augmentation functions, such as emoji overlay, are inspired by those that users apply to their images on social media platforms. Although AugLy supports multiple data modalities, it only provides 16 augmentations for the image domain which makes it insufficient for single-modality augmentation, especially in the image domain.

“torchvision” Francisco Massa (2017) is part of the PyTorch library and consists of image augmentation functions. It provides only 28 image augmentations which come with features such as creating augmentation pipelines. Similar to Albumentations, torchvision does not focus on model robustness and is just limited to data augmentation.

2.5 Image Encoding and Image Compression

Image transformation usually refers to visually distinguishable changes or effects applied to an image. However, there are some other types of transformations that are very commonly applied to images that leave some traces (which are sometimes visible) in the image, yet are largely overlooked. Lossy image encoding, such as JPEG encoding, is one example of such a phenomenon.

To store, transfer, and represent image data, we usually need to encode the raw pixel values of an image into a different format, such as JPEG, PNG, GIF, etc. Encoding image data may result in loss of information, or it may preserve the features of the image depending on the chosen encoding method. Image compression is a form of image encoding, and is categorized into either lossless or lossy compression methods. Lossless image compression algorithms, such as PNG image compression, rely on removing statistical redundancies in an image, therefore, images that are compressed using lossless compression techniques can be restored with zero loss. Nevertheless, the compression ratios achievable by applying lossless image compression are not significant. Lossy image compression methods allow us to achieve higher compression ratios by sacrificing some degree of image quality or fidelity, primarily in terms of perceptually less significant details or information. Whether to choose lossless or lossy image compression depends on the storage limitations and the specific requirements of the application involving image data. Nevertheless, it is noteworthy that lossy image encoding results in changes in the input image that might have an effect on a system as a whole. While information loss might be inevitable when we need to compress images to a greater extent, identifying the regions in the image where more information loss can be tolerated can bring advantages to both human users and computer vision models. By studying how JPEG image compression works, we realized that the local complexity of patches in an image is not taken into account by JPEG, and JPEG performs poorly when there are sharp increases in pixel intensity, which puts JPEG at a disadvantage in certain applications.

2.5.1 JPEG Image Compression

Joint Photographic Experts Group (JPEG) compression Wallace (1991) is a standard method for image compression that has been widely used for many years now. JPEG is designed for both grayscale and colored continuous-tone images, which are images that are not comprised of only black and white dots, but rather, are made up of a wide range of colors. At the heart of the JPEG compression method lies Discrete Cosine Transform (DCT) Ahmed, Natarajan, and Rao (1974), which is a mathematical transform that converts an image, which is essentially a signal, from the spatial domain to the frequency domain. By doing so, the image can be represented using fewer coefficients which allows us to apply quantization without significant loss of fidelity in the image.

More specifically, the JPEG algorithm first groups the image into 8×8 blocks and converts the input pixel values from unsigned integers to signed integers followed by feeding the signed values to the Forward DCT (FDCT). The output of FDCT is the DCT coefficients, which are the relative

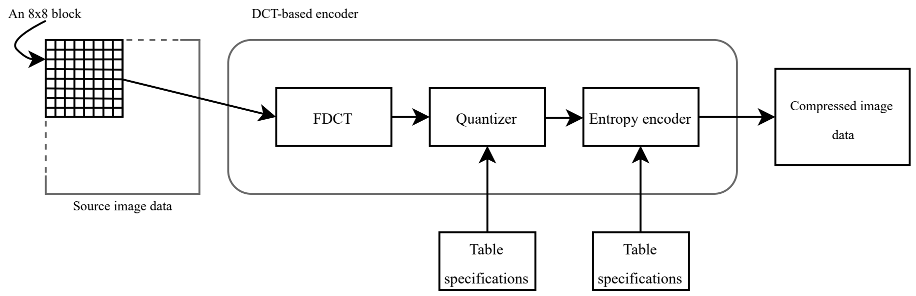


Figure 2.1: This figure illustrates the steps involved in JPEG encoding.

amount of the spatial frequencies contained in the input pixels. A typical 8×8 block usually does not contain sudden changes in pixel intensities, which means that the coefficients for high-frequency signals are negligible in the transformed signal for the typical 8×8 blocks. This allows a large portion of image energy to be concentrated in a small number of coefficients. In the next step, quantization is applied to the DCT output values according to a quantization table. This step plays a major role in the overall compression of the image and most of the data loss occurs in this stage. Finally, the quantized DCT coefficients are further compacted using an entropy encoding algorithm, e.g., Huffman coding Huffman (1952). Figure 2.1 illustrates the steps required in JPEG encoding.

Note that JPEG treats image data as signals, and when encoding or compressing an image using JPEG, no consideration is given to visual features or pattern complexity.

2.5.2 SVD-based Image Compression

Image compression can also be achieved using Singular Value Decomposition (SVD). To have a deeper understanding of how SVD can be applied to image compression, we first overview SVD and its properties, and then discuss SVD when it is applied to image compression.

Singular Value Decomposition (SVD) Overview and Properties

Given an input matrix $A_{m \times n}$ in $\mathbb{R}^{m \times n}$ [1], SVD decomposes A into three matrices as follows:

$$A = U \Sigma V^T \tag{1}$$

[1]SVD is also valid for matrices in $\mathbb{C}$, but it is out of the scope of this thesis because SVD will be applied only to image data which are matrices in $\mathbb{R}$.

where the columns of $U_{m\times m}$ and $V_{n\times n}$ are orthonormal and $\Sigma_{m\times n}$ is a diagonal matrix with non-negative real values.

This decomposition has some interesting properties:

- Every matrix A can be decomposed into U, Σ, and V^T matrices using SVD, and the decomposition may not be unique.
- The columns of U are the left singular vectors.
- The columns of V are the right singular vectors.
- The diagonal entries of Σ are the singular values of A in descending order.
- The i^{th} entry of the diagonal of Σ is the i^{th} singular value of A, σ_i for $i = 1, 2, \ldots, r$, where r is the rank of A. For $i > r$, the diagonal entries of Σ are equal to zero.

We can provide a geometric representation for a better understanding of SVD. Suppose that A is a linear transformation that maps vectors from the input space to the output space. SVD explains the linear transformations in A in terms of a sequence of simpler operations, namely, rotation, reflection, and scaling. The matrices U and V explain the rotation or reflection transformations, and Σ includes the amount of scaling. The columns of U form an orthonormal basis for the input space, and each column is a unit vector. U explains the direction of the rotation or reflection by A. The rows of V^T are the orthonormal basis for the output space, and each row is again a unit vector. V^T represents the transformed direction in the output space. The singular values in Σ are the scaling factor for the rotation or reflection of which the direction is specified by U.

Calculating SVD

To calculate SVD, we should first calculate the eigenvectors of A^TA which give us the columns of U. Then, the eigenvectors of AA^T are calculated to make up V^T. The square roots of the eigenvalues of A^TA, or AA^T equivalently, are the singular values of A that form the diagonal elements of Σ when sorted in descending order. Calculating these three matrices gives us the full SVD decomposition of the given matrix A.

Different Forms of SVD Compression

Compression with SVD can be achieved in three general forms: 1) Thin SVD, 2) Compact SVD, and 3) Truncated SVD.

Thin SVD is when we decompose $A_{m\times n}$ into $A_k = U_k\Sigma_k V_k^T$, where $k = min(m, n)$. If we choose to only keep the vectors corresponding to non-zero elements in Σ, we will have the Compact SVD. **Compact SVD** represents A using only the first r column vectors of U and r row vectors of V^T, corresponding to the first r non-negative singular values on the diagonal of Σ. In this representation, r is the rank of matrix A. The final form, which is the **Truncated SVD**, results in further compression by calculating only $t \ll r$ singular values in Σ and their corresponding vectors in U and V^T. In this form, A can only be approximated to $\tilde{A}$, since some information is lost. However, Eckart and Young (1936) show that this approximation minimizes the Frobenius norm of $A - \tilde{A}$ when the rank of $\tilde{A}$ is fixed.

SVD Applied to Image Data

Image data can be represented using matrices. A two-dimensional matrix $A_{m\times n}$ represents a grayscale image of height m and width n, and a color image can be represented using a three-dimensional matrix, like $A_{m\times n\times c}$, where c represents the number of color channels (in the case of RGB, $c = 3$). To compress a color image, we first decompose it into its color channels and then apply SVD per channel.

Compressing an image of size $m \times n$ translates to compressing the matrix $A_{m\times n}$. While applying Thin SVD to A does not necessarily save storage (think of square images), Compact SVD and Truncated SVD both can represent an image using less data. Note that Truncated SVD is an approximation that is not exact, contrary to Compact SVD.

To compress an image $A_{m\times n}$ into $A^{SVD}_{m\times n}$ using SVD, we first need to decide on the rank k of the compressed image A^{SVD}. The chosen rank determines which reduced form of SVD will be used. As the rank becomes less, more information is lost, and the quality of A^{SVD} deteriorates more significantly. Figure 2.2 illustrates the application of SVD to an image for different values of k. To compress an image into its k-rank approximation, we first apply SVD, and then only retain

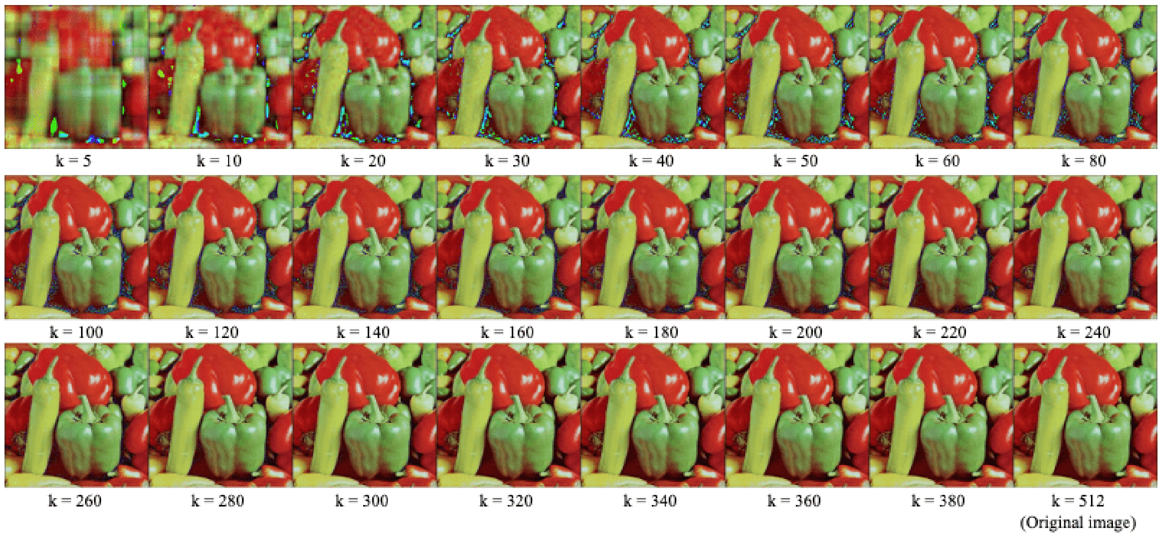


Figure 2.2: The k-rank SVD approximation of an image, applied for various values of k, illustrates how the quality of the compressed image changes based on k. Note that for $k \geq 260$, indicated by the last row, almost no noise is visible.

the first k columns, k rows, and k entries on the diagonal of Σ. This will retain $m \times k$ entries from U, k entries from Σ, and $k \times n$ values from V, which amounts to $k(m + n + 1)$ values in total. Approximating A will then be possible by multiplying the reduced U_k, Σ_k, and V_k matrices.

2.5.3 Other Approaches to Image Compression

Many lossless and lossy image compression algorithms exist in the literature, and each one has its own advantages and disadvantages. For instance, Predictive Coding Kobayashi and Bahl (1974) can be used for image compression which is based on predicting each pixel value based on the value of the neighboring pixels, followed by quantizing and compressing the prediction error. Other forms of predictors may also be used in Predictive Coding. For instance, in video compression, the information from one frame can be used to predict the content of the next frame. However, Predictive Coding is highly dependent on the predictor function, does not perform well in complex scenes containing intricate patterns, and suffers from error accumulation when subsequent poor predictions are made.

In Linde, Buzo, and Gray (1980), the Linde-Buzo-Gray (LBG) algorithm is introduced that undertakes another approach to image compression using vector quantization. LBG algorithm is a codebook design technique that segments images into blocks, assigning each vector, representing a block, to a corresponding code word. Through iterative refinement, the algorithm optimizes the codebook. In the final compression step, each block in the image is replaced with its reference in the codebook. While LBG is generalizable to many data types, it is computationally expensive and its performance is heavily dependent on the initialization of the codebook.

Following previous methods, Fractal Image Compression Barnsley and Hurd (1993) was introduced. The main idea behind using fractals was that some patterns repeat in different parts of an image, and fractals can be used to represent those patterns. To achieve image compression using fractals, a set of mathematical transformations, called Iterated Function Systems (IFS) Barnsley, Ervin, Hardin, and Lancaster (1986) were introduced that map domain blocks to range blocks. Domain blocks closely resemble the range blocks, and both domain and range blocks are regions in the image. While fractal image compression has proven valuable in specific niche applications Bhavani and Thanushkodi (2013); Liu, Zhang, Qi, and Ma (2016), particularly when self-similarity is anticipated, its limited adoption is attributed to its dependence on initial conditions and the computational cost, which is not always balanced against the achievable compression ratio.

Deep learning approaches, including using autoencoders Kramer (1991), generative adversarial networks I. Goodfellow et al. (2014), convolutional neural networks LeCun et al. (1989), and residual networks He et al. (2016) have also been applied to image compression. For example, in L. Zhou, Cai, Gao, Su, and Wu (2018), variational autoencoders are utilized to create an end-to-end image compression framework, consisting of a non-linear encoder transform, a uniform quantizer, a non-linear decoder transform, and a post-processing module that aims to reduce the compression artifacts for low bit-rate images. Torfason et al. (2018) aimed to reduce computation costs by integrating image compression using deep neural networks into image classification and semantic segmentation tasks. By jointly training the image compression (using autoencoders) and image classification models (using residual networks), they achieved performance and accuracy improvements. For image compression at extremely low bit-rates, Agustsson, Tschannen, Mentzer, Timofte, and Gool (2019) introduced a generative adversarial network (GAN) based approach that was trained with mean squared error, adversarial loss, and entropy loss, and would synthesize details when they could not be stored due to the storage limit. While these deep learning-based approaches sometimes outperform conventional image compression methods, such as JPEG, they do not apply to every use case because they require sufficient training data and computational resources. Also, the model should be accessible during inference, which calls for additional storage.

Chapter 3

Clarifying Myths About the Relationship Between Shape Bias, Accuracy, and Robustness

3.1 Introduction

While computer vision models have been applied to different areas of research to this date, developing models that can perform well when perturbations are present in the input image is still a challenging quest. To use computer vision in practice, we need reliable models that are robust against major and minor image perturbations. Nevertheless, even the most commonly used models fail to perform well when minor perturbations in light, color, etc. are present in the input image Borkar and Karam (2019); Ghosh, Shet, Amon, Hutter, and Kaup (2018); Roy, Ghosh, Bhattacharya, and Pal (2018); Y. Zhou, Song, and Cheung (2017). Fig. 3.1 illustrates some examples that can easily fool a trained ResNet-50 He et al. (2016) model with 89.06% in-domain accuracy. To push the boundaries of computer vision, more work still needs to be done to make models reliable enough to be used even in use cases, such as in autonomous vehicles, where a faulty decision can lead to disastrous consequences.

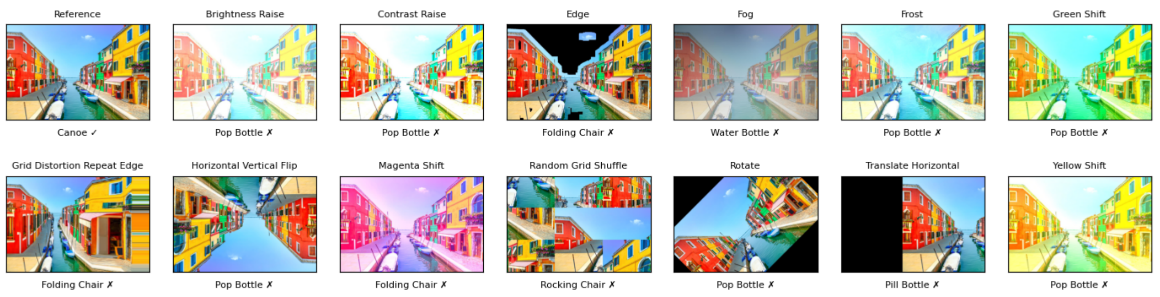


Figure 3.1: This illustration shows how a ResNet-50 model with 89.06% in-domain accuracy fails when a small perturbation has been applied to the image. The label on top of each image is the name of the perturbation, and the label below each image is the predicted class. ”reference” is the clean image.

The image perturbations can even be undetectable to a human observer and yet fool the model. A well-studied instance of this phenomenon is adversarial examples Szegedy et al. (2013) which are inputs that are intentionally designed to fool the model and are indistinguishable to human eyes. Different methods have been proposed to make models robust against adversarial attacks Bashivan et al. (2021); T. Chen et al. (2020); Herrmann et al. (2022); Kurakin, Goodfellow, and Bengio (2016); Tzeng, Hoffman, Saenko, and Darrell (2017); H. Wang and Yu (2019). In addition to the adversarial examples, other changes can be applied to images, especially perturbations that are more common in nature, such as changes in light, color shifts, and different weather conditions. These types of image perturbations are the main focus of study in this work since they are present in many real-world applications.

Data augmentation is a well-practiced method to improve model robustness. Some data augmentation libraries already exist from previous research work, including imgaug Jung et al. (2020), Albumentations Buslaev et al. (2020), AugLy Papakipos and Bitton (2022), and torchvision Francisco Massa (2017). In none of the above frameworks, however, authors propose a concrete solution based on which users can find which augmentations are best to use to improve their model performance. Some previous works focus on finding the optimal augmentation automatically. For instance, in Cubuk, Zoph, Mane, Vasudevan, and Le (2018), a reinforcement learning-based framework has been proposed that searches for the best augmentation policies in a discrete space. However, because of the high complexity of the search space, when the number of augmentations that are required to be investigated is more than a few, especially in cases where the resources are limited, this method cannot be used at all. Developing a fast and efficient solution to find which augmentations fit a specific learning problem is crucial as currently many data scientists rely on trial and error to choose an augmentation which requires a lot of resources.

Based on the experiments in Geirhos et al. (2018), it is hypothesized that by adding more shape bias to a network, model robustness will be improved. While some works are based on the correctness of this hypothesis Lee, Hwang, Kang, and Zhang (2022), there is not any extensive study in the literature that examines this claim in practice. However, there is a growing belief that using augmentations that add shape bias to the network will improve model robustness.

In this paper, by evaluating a diverse set of augmentations, including simulating weather conditions, occlusions, noise, color changes, etc., we study how each data augmentation function affects the model's robustness to out-of-distribution (OOD) data. Furthermore, we use a shape bias metric introduced in Geirhos et al. (2021) to measure the shape bias imposed by each augmentation function on the trained model, and we conclude that although in some cases there is a correlation between the model's shape bias and its robustness to OOD samples, there is no causal relationship that indicates an increase in shape bias results in improved OOD robustness. Therefore, more focus should be on improving the OOD robustness directly, instead of trying to achieve higher robustness by aiming to increase a model's shape bias.

There is a dispute on whether there is a trade-off between in-domain accuracy and OOD robustness Herrmann et al. (2022); Kurakin et al. (2016). If such a trade-off exists, focusing on improving model robustness to OOD examples comes at the cost of losing in-domain accuracy. By analyzing the results we obtain from studying the in-domain accuracy and OOD robustness of models trained on various augmented datasets, we find that both in-domain and OOD accuracies can be improved at the same time, and the trade-off does not always exist.

As we focus on improving model robustness to OOD samples, we should note that a very common threat to model robustness is the presence of bias in the dataset. Once identified, dataset bias can be partially reduced using data augmentation. ImageNet-1K Deng et al. (2009) is known to be biased towards representing texture-related features Geirhos et al. (2018). By analyzing our results on the OOD robustness of models trained using augmented datasets, we find three sources of bias in the ImageNet-1K dataset that can be readily mitigated by applying data augmentations.

The main contributions of this work are the following:

- We apply 39 data augmentations to ImageNet-100, a fair representative subset of ImageNet-1K, and evaluate the effect of each data augmentation on the OOD robustness of a ResNet-50 model.

- We find some biases in the ImageNet-1K dataset and suggest that by using simple data augmentations, most of these biases can be reduced. Moreover, we show that not all data augmentations improve the OOD robustness.

- We demonstrate that there is not necessarily a trade-off between in-domain and OOD robustness, and by choosing the right augmentations, both the in-domain and OOD accuracies can be improved.

- By investigating the relationship between shape bias and OOD robustness, we show that it is not good practice to focus on improving the model's shape bias in order to achieve higher OOD robustness. This is because an increase in a model's shape bias does not necessarily improve the model's robustness to OOD samples.

3.2 Related Works

In Geirhos et al. (2018), a stylized version of the ImageNet dataset, called Stylized ImageNet (SIN), has been created. Moreover, it is shown that a SIN-trained ResNet outperforms an ImageNet-trained ResNet on most of the image perturbations considered in their work. According to this result, it seems since SIN-trained ResNets learn a shape-based representation of ImageNet, they are more robust than ImageNet-trained ResNets that mostly focus on texture information. Therefore, the authors claim that ImageNet data induce texture bias in a network, and they hypothesize that by adding more shape bias to a network, model robustness will be improved.

This hypothesis forms the basis of some of the following related works in the literature. For example, in Lee et al. (2022), a shape-based augmentation method has been proposed that applies different augmentations from a set of three augmentations to the image foreground and background. The idea behind this is to reduce the texture bias and increase the shape bias to improve the model's robustness to OOD data. While this method has achieved acceptable results, it is limited to only three shape-based augmentations, including Color Jitter, Random Grayscale, and Random Gaussian Blur. Moreover, the questions of why these augmentations were selected and what makes them "shape-based" remain unanswered.

Improving model robustness using shape-focused augmentations has also been observed in self-supervised learning methods. In Hendrycks, Mazeika, Kadavath, and Song (2019) it is shown that using a rotation-predictor auxiliary head along with a classification network can improve model robustness since the model needs to learn the shape representations in order to minimize the loss function.

Other than data augmentation, the choice of the network affects the shape bias, as well. Vision Transformers Dosovitskiy et al. (2020) are less biased towards texture compared to Convolutional Neural Networks (CNNs) Naseer et al. (2021). However, data augmentation has a much larger effect on shape bias compared to the network type or the training objective function Geirhos et al. (2021); Hermann, Chen, and Kornblith (2020).

Although a correlation exists between using data augmentation and improving model robustness, it should not be concluded that such an increase in robustness is a direct result of selecting augmentations that impose shape bias on the model. In other words, *an increase in shape bias results in an increase in model robustness* is only a hypothesis. To evaluate this hypothesis, in Mummadi et al. (2021), a method is introduced to increase a model's shape bias by keeping the shape-related information in a training dataset while suppressing texture-related features. The authors challenged the hypothesis and concluded that there is no clear causal relationship between shape bias and model robustness. However, this method is limited because it only keeps the edge information in the input image, which cannot introduce meaningful shape information to the model when partial object occlusions exist in the image, for instance. Furthermore, they evaluated the OOD robustness of their models on only few image perturbations that are not diverse enough to represent a large variety of perturbations and do not include those that are commonly found in nature (except Frost).

3.3 Method

We are interested in exploring the characteristics of different augmentations and investigating if there is a relationship between shape bias and a model's in-domain accuracy and OOD robustness. To this end, we performed an extensive set of experiments on a fairly representative subset of the ImageNet-1K dataset. The reason why we did not use ImageNet-1K was the significant computation requirements. More specifically, we needed to train a model for each augmented dataset with a training size two times larger than the original dataset. This required a huge amount of computation power if we wanted to use the whole ImageNet-1K dataset, especially in our case since we aimed to evaluate various augmentation types for the purpose of our studies. We used the experimental results to analyze biases in the ImageNet-1K dataset by examining those augmentations that affect the model OOD robustness the most.

3.3.1 Shape bias

We calculate shape bias using a measure defined in Geirhos et al. (2021) as described in (2). This measure is applied to a dataset containing images with a shape corresponding to one class, and a texture corresponding to another. For example, it contains an image with the texture of an elephant that shows the shape of a cat. If in this case, the model detects a cat, then, it has a stronger bias towards shape than texture. Therefore, in order to calculate shape bias using this method, we count the number of correct shape decisions and divide that by the number of correctly classified images (either shape or texture).

$$\text{shape bias} = \frac{\text{\# correct shapes}}{\text{\# correct shapes + \# correct textures}} \tag{2}$$

3.3.2 OOD robustness

To evaluate the OOD robustness of a model, first, we trained the model on the source dataset augmented using a selected augmentation type. Then, we evaluated the performance of the trained model on a target dataset with images that were obtained by applying 17 various perturbations to the images from the source dataset. We used the model's average top-1 classification accuracy on the target dataset as a measure to describe the OOD robustness in our experiments.

3.4 Experiments

3.4.1 Dataset

OOD Benchmark Dataset

We used the OOD Benchmark dataset designed in Geirhos et al. (2021) to evaluate the effect of each augmentation function on the OOD robustness. The OOD Benchmark dataset is a collection of 17 datasets that are obtained by applying different types of image perturbations on a subset of ImageNet-1K. Based on the perturbation type, these 17 datasets are categorized into two groups; five datasets belong to the Style group and the rest of the datasets belong to the Noise group. Each dataset in the Style group is created by applying a specific effect on a subset of the ImageNet-1K dataset. These effects include "sketch", "edge filter", "silhouette", "cue-conflict" (images that have a texture corresponding to a class with a shape from a different class), and "stylized" (images that are stylized using painting styles). In the Noise group, images are obtained by applying different types of digital degradations to images, including noise and blur. Each image in these 17 datasets corresponds to a label from a set of 16 human-recognizable labels ("airplane", "bear", "bicycle", "bird", "boat", "bottle", "car", "cat", "chair", "clock", "dog", "elephant", "keyboard", "knife", "oven", "truck") introduced in the same work. To obtain these labels, a mapping has been introduced from ImageNet-1K classes to 16 broader categories[1]. For example, a few classes of different cat types, including "tabby cat", "Persian cat", and "Siamese cat" are mapped to one class, called "cat".

ImageNet-100

For training purposes, we used a subset of 100 classes from the ImageNet-1K dataset, called ImageNet-100. In order to build ImageNet-100 from ImageNet-1K, we needed to pick 100 classes from the 1000 classes in ImageNet-1K. Since we wanted to perform OOD robustness testing on the

[1] https://github.com/bethgelab/model-vs-human

OOD Benchmark dataset, we needed labels that could be converted to the 16 human-recognizable labels used in the OOD Benchmark dataset. Therefore, we examined the above mentioned mapping from ImageNet-1K to 16 human-recognizable categories to find those classes that were mapped to only few classes from ImageNet-1K. For example, “knife” was only mapped to one class in ImageNet-1K, whereas “dog” corresponded to 110 classes. To make ImageNet-100 balanced, we prioritized these classes over the ones that corresponded to a higher number of classes in ImageNet-1K. Then, according to the mapping, we started selecting ImageNet-1K classes until we reached 100 classes. This approach allowed us to end up with classes that can be mapped to the 16 human-recognizable labels, and at the same time, represent a balanced set of classes in ImageNet-1K. ImageNet-100 represents ImageNet-1K; because by design, for each class in ImageNet-1K, there is a broader class in ImageNet-100 that represents the selected ImageNet-1K class.

Cue-conflict

To calculate shape bias, we used the cue-conflict dataset which is one of the datasets in the Style group from the OOD Benchmark dataset. Cue-conflict is a collection of images that have the shape corresponding to class A, but texture corresponding to class B, where $A \neq B$.

3.4.2 Implementation details

We used the PyTorch framework for our experiments. For every augmentation, we trained a ResNet-50 model with an initial learning rate of 0.1 for 40 epochs and used the ReduceLROnPlateau scheduler that multiplies the learning rate by a factor of 0.1 whenever the validation accuracy has not increased for 5 epochs. The batch size and weight decay were set to 256 and 0.0005, respectively. We also trained a baseline model, called Vanilla, that is trained on the ImageNet-100 training set without adding any augmentations for comparison purposes. For each of the 39 models, one for each augmentation type, we evaluated the in-domain accuracy on clean data, as well as the model’s OOD robustness when evaluated on the OOD Benchmark dataset. Furthermore, we used the cue-conflict dataset from the OOD Benchmark dataset to measure the shape bias of each model.

3.4.3 Augmentations

We studied 39 types of augmentations that span a diverse set of image perturbation types. The intensity of these augmentations is balanced in a way that the transformed image is still easily recognizable to an observer, while they can instantly realize that some known kind of perturbation has been applied to the image. Fig. 3.2 illustrates different types of augmentations and how they change a reference image.

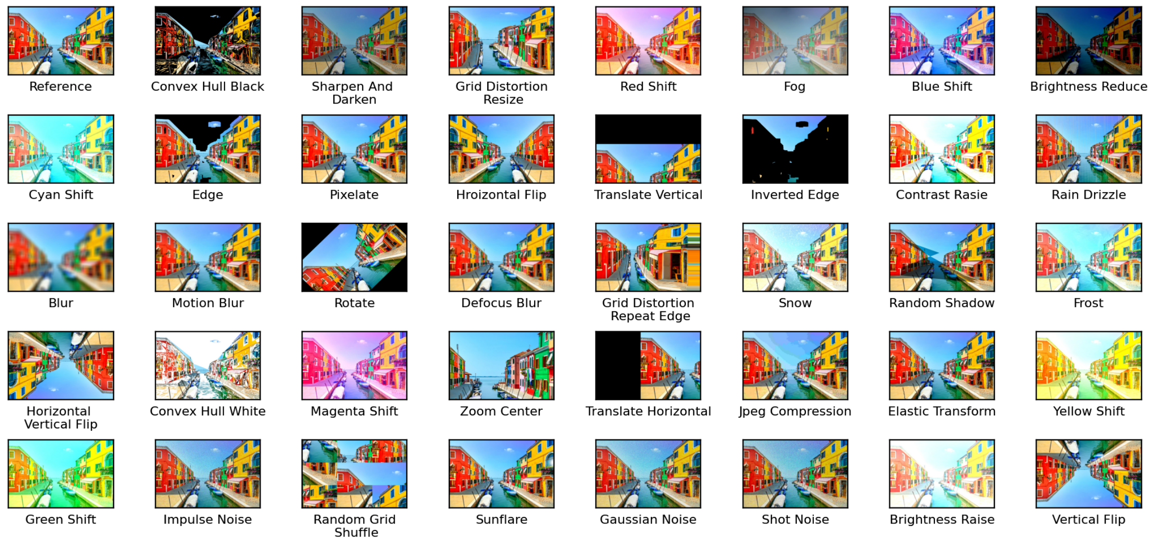


Figure 3.2: An overview of how different augmentations alter the reference (original) image. Best viewed in color.

3.5 Results and Discussion

3.5.1 Evaluating augmentations on the OOD Benchmark

By evaluating ResNet-50 models trained on augmented datasets using various image augmentations, we calculated the OOD robustness on the OOD Benchmark dataset. We measured this factor by considering a diverse set of 39 augmentations across various perturbation groups, including blur, noise, distortion, occlusion, color shift, lightness, weather, and geometry. We also measured the OOD robustness for the Vanilla (baseline) model, which is trained on ImageNet-100 without applying any data augmentation to compare the results. Fig. 3.3 shows how OOD robustness varies across different augmentations.

As we mentioned in Section 3.4.1, ImageNet-100 fairly represents ImageNet-1K by design. Therefore, by choosing the augmentations that have the highest effect on model robustness according to Fig. 3.3 and augmenting ImageNet-1K using those augmentations, we can improve model robustness in situations where we are using the ImageNet-1K dataset. Fig. 3.3 illustrates that "Brightness Reduce" augmentation contributed the most to OOD robustness. On the other hand, "Brightness Increase" holds the 8th rank out of 40. This means that most of the images are in the same range of light settings in the ImageNet-1K dataset and a strong light augmentation can increase OOD robustness significantly.

The second most effective augmentation is "Grid Distortion Resize" which performs grid distortion on the input image and then resizes the output to the original image size. This augmentation alters the visual representation of the image in a way that objects seem more narrow or wide on different sides. This is similar to viewing an object from different angles. Since applying this augmentation has a high effect on model robustness, it can be concluded that the ImageNet-1K dataset images mostly represent objects from limited angles. To improve OOD robustness against other points of view, one cost-efficient approach is to use geometric augmentations, like the "Grid Distortion Resize" augmentation.

Last but not least, "Translate Horizontal" stands third in the ranking, which indicates that ImageNet-1K objects are usually located in the center of the image; whereas in real-world applications, this is not always the case. By applying a simple translate augmentation, OOD robustness can improve significantly.

3.5.2 Does data augmentation always improve OOD robustness?

According to Fig. 3.3, some of the augmentations rank lower than the baseline Vanilla model. For instance, "Elastic Transform" which adds a wavy effect on the input image has a significantly lower OOD robustness (-3.5%) compared to the Vanilla model. This observation is an example of a case where data augmentation does not improve model robustness. Since such cases exist, we should always perform OOD robustness evaluations on augmented datasets to make sure we are achieving results that are better than the baseline.

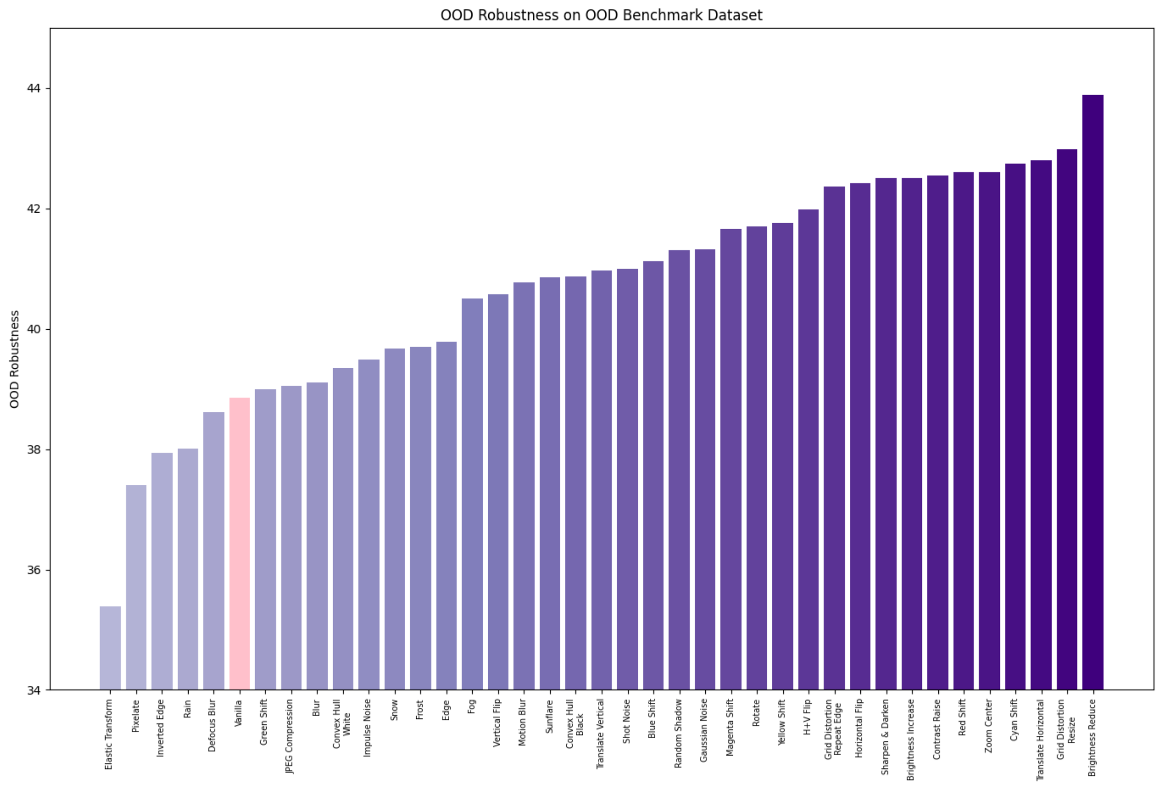


Figure 3.3: Evaluation of OOD robustness for various augmentations on the OOD Benchmark dataset. The pink bar represents the Vanilla (baseline with no data augmentations) model.

3.5.3 Is there a trade-off between in-domain accuracy and OOD robustness?

In Kurakin et al. (2016), it is claimed that there is a trade-off between in-domain accuracy and OOD robustness. In Herrmann et al. (2022), however, it has been shown that this trade-off can be broken for Vision Transformers, at least. We study if such a relationship exists for CNN-based models by exploiting the in-domain accuracy and OOD robustness of our trained models on the OOD Benchmark dataset. Table 3.1, demonstrates a few examples for which this trade-off breaks. A few other augmentations meet these conditions as well, which are not listed in this table for brevity. According to these examples, it cannot be concluded that increasing the OOD robustness comes at the cost of a decrease in in-domain accuracy, necessarily. An ideal augmentation increases both in-domain accuracy and OOD accuracy.

Table 3.1: This table shows a few augmentation examples that can be applied to the ImageNet-100 dataset and increase both in-domain accuracy and OOD robustness, simultaneously. Avg. OOD Robustness is the weighted average of OOD-Noise and OOD-Style robustness values.

Augmentation Name	In-domain Accuracy	OOD-Noise Robustness	OOD-Style Robustness	Avg. OOD Robustness
Vanilla (Baseline)	89.06	42.99	28.92	38.85
Grid Distortion Resize	94.92	48.67	29.33	42.98
Grid Distortion Repeat Edge	93.75	47.47	30.11	42.3
Rotate	93.75	46.27	30.74	41.70
Translate Vertical	92.57	44.44	32.64	40.96
Translate Horizontal	92.18	46.8	33.19	42.79
Zoom Center	91.01	47.16	31.66	42.60

3.5.4 The relationship between shape bias and OOD robustness

An increase in the shape bias does not *result* in an increase in the OOD robustness. However, as the shape bias increases, OOD robustness against the augmentations in the Style group *usually* increases at the cost of a decrease in the OOD robustness against the augmentations in the Noise group. This claim can be observed in Fig. 3.4, where the data points are estimated using a second-degree polynomial for each OOD group (Style and Noise). Since the OOD robustness for the Noise group reduces with the increase in shape bias, we cannot claim that a higher shape bias results in a higher OOD robustness. Therefore, it is not good practice to try to achieve improved model robustness by focusing on improving the model's shape bias as seen in the literature.

3.6 Conclusion

In this work, we evaluated the OOD robustness of 39 different types of augmentations on the OOD Benchmark dataset using ResNet-50 models trained on various augmented versions of the ImageNet-100 dataset. By studying the experimental results carefully, we found some biases in the ImageNet-1K dataset that can be reduced by applying data augmentation. Moreover, we explained that not all data augmentations improve model robustness, and it is necessary to evaluate whether

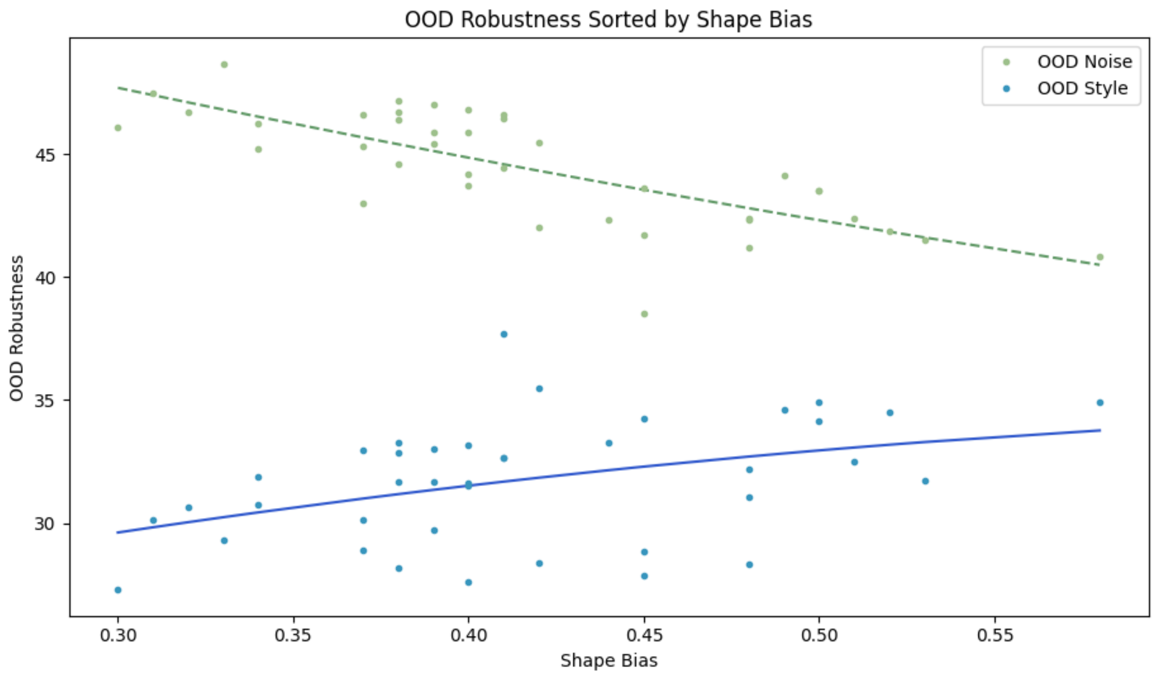


Figure 3.4: This figure shows the relationship between the shape bias and the OOD robustness of models trained on augmented datasets. The evaluation has been performed on the OOD Benchmark dataset that contains two groups of datasets, called the Noise and the Style groups.

augmenting a dataset using a specific type of data augmentation contributes to an increase in OOD robustness. Furthermore, the relationship between in-domain accuracy and OOD robustness was studied, and it was shown that there is not necessarily a trade-off between in-domain accuracy and OOD robustness. Finally, it was concluded that an increase in shape bias does not always result in an increase in OOD robustness.

Chapter 4

PatchSVD: A Non-uniform SVD-based Image Compression Algorithm

4.1 Introduction

Enormous[1] amounts of data are generated every day by various sources, including social media, sensors on wearable devices, and smart gadgets. In many cases, the data needs to be stored on a small device to serve a specific purpose. For instance, a real-time defect detector stores images of its camera feed and sends them to a lightweight model to catch possible flaws in a production line Pham, Chang, et al. (2023). Such small devices are limited in storage capacities; therefore, it is essential to design efficient data storage algorithms capable of storing the data without significant loss in quality.

Images are one of the most commonly used data formats, and storing an image file on a digital device can require anywhere from kilobytes to megabytes of storage space, depending on the complexity of the image and the storage technique. Image compression algorithms can be categorized into lossless and lossy compression methods. In lossless methods, such as PNG image compression, the original input can be reconstructed after compressing because compression was achieved by removing statistical redundancies. However, only low compression ratios are achievable through

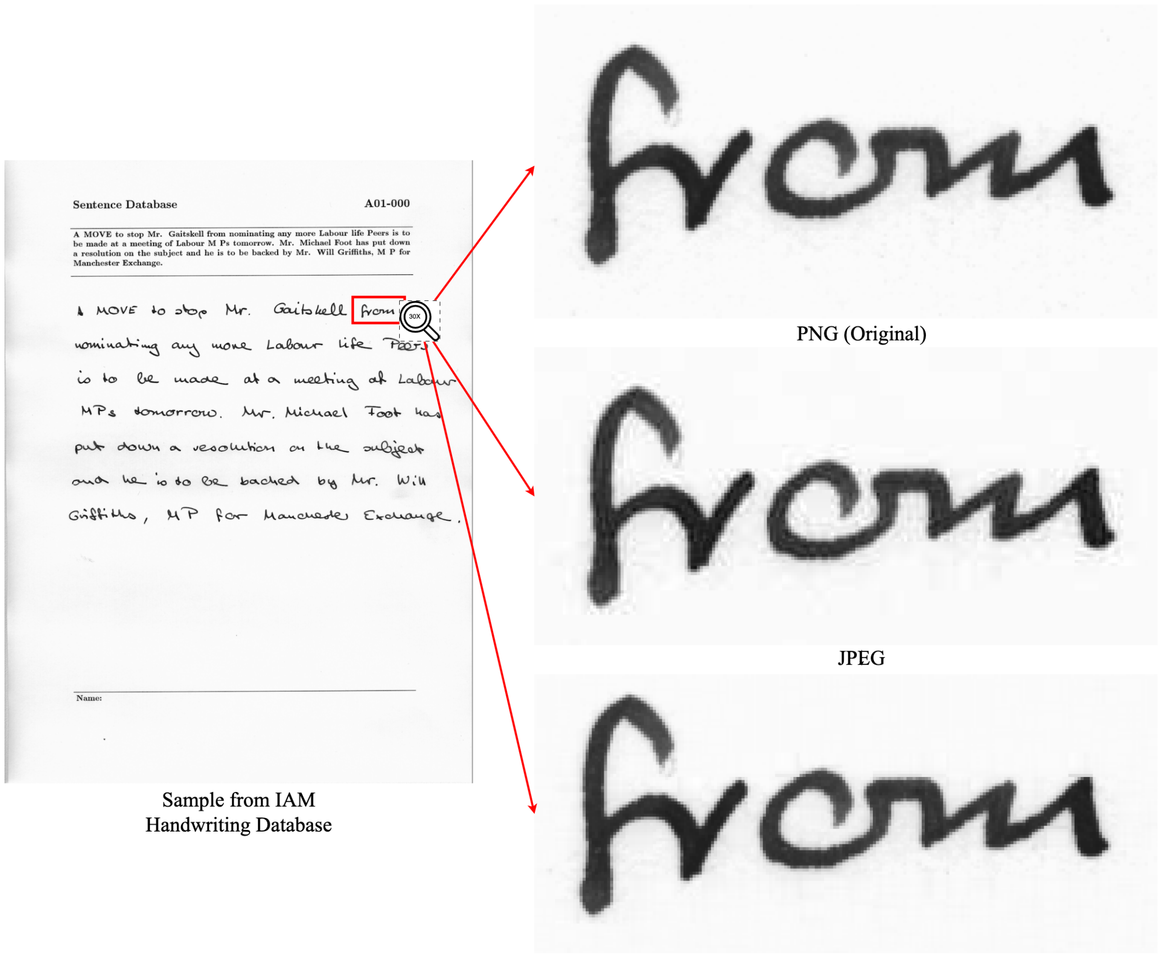
[1]This work has been accepted to be published in the Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2024) `https://icpram.scitevents.org/`

lossless image compression, and image compression is not always guaranteed. On the other hand, lossy compression techniques, such as JPEG compression Wallace (1991), achieve higher compression ratios by allowing more information loss, but the reconstructed image sometimes contains visible compression artifacts. Specifically, JPEG image compression is based on the assumption that within an 8×8 pixel block, there are no sharp changes in intensity. However, in some use cases, such as compressing images of text or electronic circuit diagrams, this assumption does not hold, and JPEG creates visible compression artifacts around the drawn lines. Figure 4.1 compares the JPEG compression artifacts with those of the proposed method presented in this paper, using an example image containing text sourced from the IAM Handwriting Database Marti and Bunke (2002).

To improve the compression results, some traditional methods compress regions of interest (ROIs) with a higher bit-rate than the rest of the regions Christopoulos, Skodras, and Ebrahimi (2000), however, the ROIs are not detected automatically and should be specified by the user. Deep learning-based approaches have also been applied to image compression Agustsson et al. (2017); Cheng, Sun, Takeuchi, and Katto (2018, 2019); Li, Zuo, Gu, Zhao, and Zhang (2018); Prakash, Moran, Garber, DiLillo, and Storer (2017); Toderici et al. (2017). Nevertheless, training deep learning models require high computational power and the models are not readily explainable. Moreover, during inference, the model has to exist on the device that runs the compression code, which calls for additional storage compared to traditional methods.

In this paper, we propose a new image compression algorithm based on SVD, called PatchSVD[2]. First, we explain how PatchSVD works, what is the compression ratio we can achieve using PatchSVD, and what are the required conditions to compress an image using PatchSVD. Then, through extensive experiments, we demonstrate that PatchSVD outperforms SVD in terms of SSIM, PSNR, and MSE metrics. Moreover, through examples, we show that PatchSVD is more robust against sharp pixel intensity changes, therefore, it is preferable over JPEG in some use cases, including compressing images of text, where high changes in pixel intensity exist.

[2]PatchSVD source code is available at `https://github.com/zahragolpa/PatchSVD`


30X
PNG (Original)
JPEG
PatchSVD (Proposed)
Sample from IAM
Handwriting Database

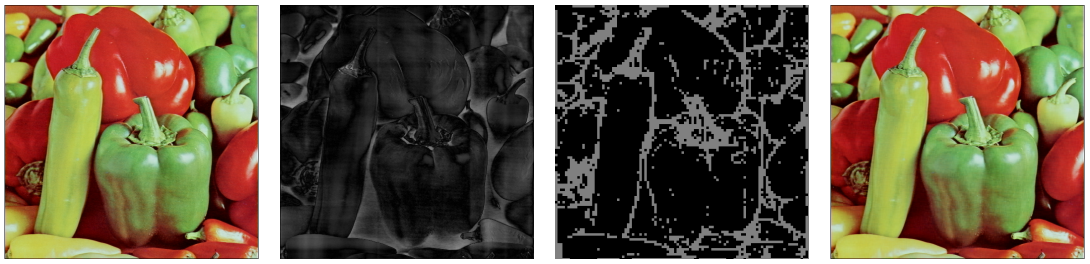
Sentence Database A01-000

A MOVE to stop Mr. Gaitskell from nominating any more Labour life Peers is to be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot has put down a resolution on the subject and he is to be backed by Mr. Will Griffiths, M P for Manchester Exchange.

A MOVE to stop Mr. Gaitskell from nominating any more Labour life Peers is to be made at a meeting of Labour MPs tomorrow. Mr. Michael Foot has put down a resolution on the subject and he is to be backed by Mr. Will Griffiths, MP for Manchester Exchange.

Name:

Figure 4.1: JPEG produces more compression artifacts in images containing text compared to the proposed method. The sample is taken from the IAM Handwriting Database and the image has been zoomed in 30 times the original size to better visualize the compression artifacts.



(a) Original image (b) Difference image (Δ) (c) Complex (in gray) and simple (in black) patches (d) PatchSVD output

Figure 4.2: PatchSVD algorithm first applies low-rank SVD to the original image and subtracts the low-rank approximation from the original image (Figure 4.2(a)) to obtain Δ (Figure 4.2(b)). Then, by applying a score function, PatchSVD calculates the patches that contain more information according to Δ (4.2(c)) to create the final compressed image (see Figure 4.2(d)).

4.2 Preliminaries

In this section, we briefly overview SVD and its basic properties. Singular Value Decomposition or SVD is an algorithm that factorizes a given matrix $A \in \mathbb{R}^{m \times n}$ into three components $A = USV^T$, where $U_{m \times m}$ and $V^T_{n \times n}$ are orthogonal matrices and $S_{m \times n}$ is a diagonal matrix containing singular values in descending order. The singular values are non-negative real numbers that represent the magnitude of the singular vectors; i.e., larger singular values indicate directions in the data space where there is more variability. Therefore, by keeping the larger singular values and their corresponding vectors, the original matrix can be approximated with minimum information loss.

The maximum number of linearly independent rows or columns of a matrix is called the rank (r) of that matrix. When a matrix A is decomposed using SVD, the number of non-zero elements on the diagonal of S is equal to r. By retaining only the first r elements from U, S, and V^T we get an r-rank approximation $A_r = U_r S_r V_r^T$ for the matrix A.

If we use k-rank SVD to compress an image represented by matrix $A_{m \times n}$, the number of elements we need to store to represent the compressed image is calculated by summing up the number of elements from each SVD component. Therefore, storing the k-rank version of $A_{m \times n}$ with $m \times n$ elements only requires $S_{SVD} = k(m+n+1)$ values. Note that the k-rank approximation of matrix A keeps most of the useful information about the image while reducing the storage requirements for the input, especially if the rows and columns in the image are highly correlated ($r \ll min(m,n)$). However, when higher compression ratios are required, it is beneficial to sacrifice more information to save more storage, which is achievable by choosing $k_{SVD} < k$.

4.3 Related Works

SVD has been used before in literature for image compression Akritas and Malaschonok (2004); Andrews and Patterson (1976); Cao (2006); Kahu and Rahate (2013); Prasantha, Shashidhara, and Murthy (2007); Tian, Luo, and Liao (2005). In Ranade, Mahabalarao, and Kale (2007), a variation on image compression using SVD has been proposed that applies a data-independent permutation on the input image before performing SVD-based image compression as a preprocessing step. In Sadek (2012), a forensic tool is developed using SVD that embeds a watermark into the image noise

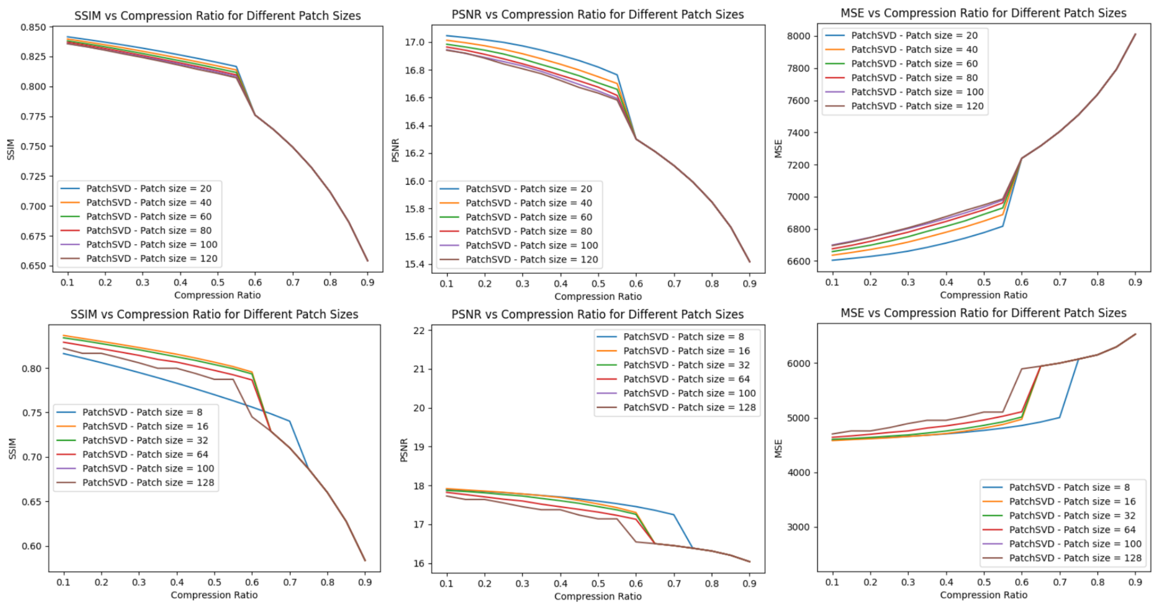


Figure 4.3: This figure shows the effect of patch size on the performance of the PatchSVD algorithm across CLIC (first row) and Kodak (second row) datasets.

subspace. Another application of SVD applied to image data is image recovery discussed in Z. Chen (2018) where authors used SVD for matrix completion.

Few studies have investigated region-based or patch-based image compression using techniques similar to SVD. In Lim, Yap, and Manap (2014), a GUI system is designed that takes Regions of Interest (ROI) in medical images to ensure near-zero information loss in those regions compared to the rest of the image when compressed. However, users need to select the ROIs manually, and Principle Component Analysis (PCA) is used instead of SVD. Lim and Abd Manap (2022) used a patch-based PCA algorithm that eliminates the need to manually select the ROIs in brain MRI scans using the brain symmetrical property. However, their approach has very limited use cases because of the assumed symmetrical property in the images.

Joint Photographic Experts Group (JPEG) compression Wallace (1991) is one of the most commonly used image compression methods, applicable to both grayscale and color continuous-tone images. JPEG works by transforming an image from the spatial domain to the frequency domain using Discrete Cosine Transforms (DCTs) Ahmed et al. (1974). Each image is divided into 8×8 pixels blocks, transformed using DCT, and then quantized according to a quantization table, followed by further compression using an entropy encoding algorithm, such as Huffman coding Huffman

(1952). While JPEG is used in many use cases, it fails to perform well in examples where sudden changes in intensity exist within the blocks.

We aim to extend the previous works by automatically selecting the complex patches and utilizing SVD with respect to the image context to compress images with minimum information loss and achieve significant reductions in storage without any training required using simple mathematical operations.

4.4 Method

4.4.1 PatchSVD Algorithm

To compress $A_{m\times n}$ using PatchSVD, we first compute the SVD of the image and take the first k_{SVD} singular values to reconstruct the k_{SVD}-rank approximation. Then, we subtract the k_{SVD}-rank approximation from the original image to get matrix Δ (see Figure 4.2(b)). In our experiments, we selected the initial $k_{SVD} = 1$. Based on the properties of SVD, the k_{SVD}-rank approximation captures most of the image information; therefore, high values in Δ (white pixels) indicate pixels that were not captured by the first singular values, and low values in Δ (black pixels) would give us those locations were almost all the information in the original image was captured by the k_{SVD}-rank approximation. Therefore, to distinguish the complex patches that SVD missed from the simpler ones, we investigate the values in Δ and determine if a patch is complex or simple (see Figure 4.2(c)). In other words, the Δ matrix helps us find the areas that would introduce large compression errors if we used the standard SVD for image compression. We utilize Δ as a heuristic function to minimize the compression error by applying non-uniform compression.

More specifically, we split the image corresponding to the Δ matrix into patches of size $P_x \times P_y$. If the image is not divisible by the patch dimensions, we add temporary margins to the sides with pixel values equal to the average pixel value in the image. Then, we loop over the patches and assign a score to each patch according to a score function. Next, we sort the patches based on the score and select the top n_c complex patches. The number of complex patches is determined based on the desired compression ratio, following Equation 4.

After we find the complex patches, we run SVD for each patch and take the k-rank approximation with two different constants; k_c for complex patches and k_s for simple patches where $k_s \leq k_c$. When we encounter the patches that contain the extra margin, we remove the margin before performing k-rank approximation using SVD. Finally, we put the compressed patches together to form the compressed image (see Figure 4.2(d)). PatchSVD algorithm is described in detail in Algorithm 1. We argue that PatchSVD is a more flexible image compression algorithm compared to SVD and JPEG because it allows you to assign non-uniform importance to each patch in the image according to a customizable function. While our method relies on the 1-rank SVD to calculate the Δ matrix and employs the standard deviation score function to sort patches, it's noteworthy that various techniques, such as graph-based approaches, gradient-based methods, edge detection, and expert knowledge, can also be employed to detect and sort complex patches. The choice may depend on the specific requirements of the application.

Algorithm 1 PatchSVD

```
Require: $A_{m\times n}, k_s, k_c, P_x, P_y, \texttt{CR}$
Ensure: $\texttt{cmpr\_image}$
 1: $\frac{n_c}{t} \leftarrow \left(\frac{P_x P_y (1-\texttt{CR})}{P_x+P_y+1} - k_s\right)\frac{1}{k_c - k_s}$
 2: $\Delta \leftarrow A - \text{k\_rank\_SVD}(A, 1)$
 3: $\Delta_{\texttt{patches}} \leftarrow \text{patch\_and\_add\_margin}(\Delta, P_x, P_y)$
 4: if $\frac{n_c}{t} \times \texttt{num\_patches} < 1$ then
 5:     $\texttt{k\_SVD} \leftarrow \texttt{int}((1-\texttt{CR}) \times \frac{m\times n}{m+n+1})$
 6:     return $\text{k\_rank\_SVD}(A, \texttt{k\_SVD})$
 7: end if
 8: for $\texttt{patch in } \Delta_{\texttt{patches}}$ do
 9:     $\Delta_{\texttt{scores}}[\texttt{patch}] \leftarrow \text{score}(\texttt{patch})$
10: end for
11: $\Delta_{\texttt{scores}} \leftarrow \text{sort}(\Delta_{\texttt{scores}})$
12: for $\texttt{patch in } \Delta_{\texttt{patches}}$ do
13:     $U, S, V^t \leftarrow \text{SVD}(\texttt{patch})$
14:     if $\text{index}(\texttt{patch}) \leq n_c$ then
15:         $k \leftarrow k_c$
16:     else
17:         $k \leftarrow k_s$
18:     end if
19:     $\texttt{cmpr\_patch} \leftarrow \text{k\_rank\_SVD}(\texttt{patch}, k)$
20:     $\texttt{cmpr\_patches}.\text{append}(\texttt{cmpr\_patch})$
21: end for
22: $\texttt{cmpr\_image} \leftarrow \text{arrange\_patches}(\texttt{cmpr\_patches})$
23: return $\texttt{cmpr\_image}$
```

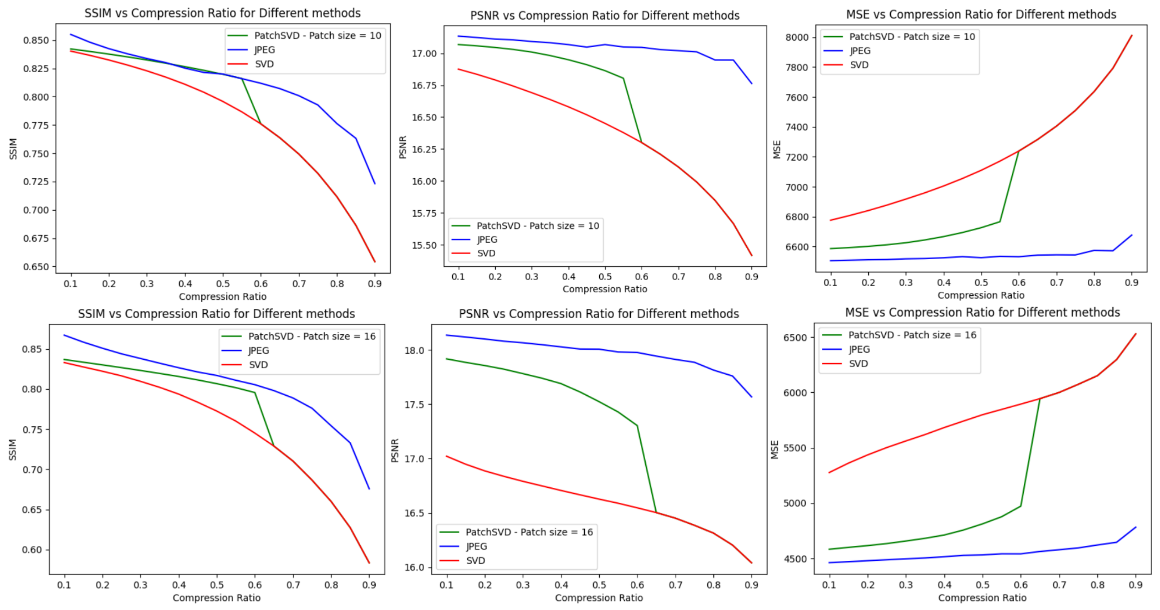


Figure 4.4: This figure compares SSIM, PSNR, and MSE metrics for PatchSVD, JPEG, and SVD image compression algorithms on the CLIC dataset with a patch size of 10 (top row) and the Kodak dataset with a patch size of 16 (bottom row).

4.4.2 Compression Ratio (CR)

Suppose we want to compress an input image of size $m \times n$ using PatchSVD and we choose the patch size to be $P_x \times P_y$. The amount of compression we get depends on the values we choose for k_c and k_s, i.e., lower-rank approximations result in higher amounts of image compression. More specifically, the number of digits we need to save, S_{PatchSVD}, to represent an image using PatchSVD is $S_{\text{PatchSVD}} = n_c k_c(P_x + P_y + 1) + n_s k_s(P_x + P_y + 1)$, where n_c and n_s are the number of complex and simple patches, respectively. Taking $k_c = k_s = k_{\text{SVD}}$ would result in the storage required by the SVD algorithm (S_{SVD}) which is equivalent to the storage needed by the PatchSVD algorithm when $P_x = n$, $P_y = m$, $n_c = 1$, and $n_s = 0$ which is equal to $S_{\text{SVD}} = k_{\text{SVD}}(m + n + 1)$.

Therefore, the Compression Ratio (CR) can be calculated using the following formula:

$$\begin{aligned} \text{CR} &= \frac{S_{\text{Original}} - S_{\text{PatchSVD}}}{S_{\text{Original}}} \\ &= 1 - \frac{(P_x + P_y + 1)(n_c k_c + n_s k_s)}{P_x P_y (n_c + n_s)} \end{aligned} \tag{3}$$

where S_{Original} is the storage needed for the original image. From 3, we can calculate the number of complex patches n_c when the required compression ratio is known:

$$\begin{aligned} \text{CR} &= 1 - \frac{(P_x + P_y + 1)(n_c k_c + n_s k_s)}{P_x P_y (n_c + n_s)} \\ \implies \frac{n_c}{t} &= (\frac{P_x P_y (1 - CR)}{P_x + P_y + 1} - k_s)\frac{1}{k_c - k_s} \end{aligned} \tag{4}$$

where t is the total number of patches in the image. We need to ensure $CR \geq 0$ to have a compressed image; therefore, the following should hold:

$$0 \leq \frac{n_c}{n_c + n_s} \leq \frac{\frac{P_x P_y}{P_x + P_y + 1} - k_s}{k_c - k_s} \tag{5}$$

which requires two conditions to be true:

$$\frac{P_x P_y}{P_x + P_y + 1} \geq k_s \tag{6}$$

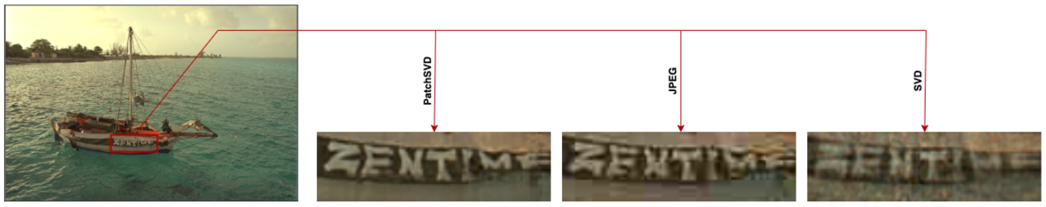
and

$$k_c \geq k_s \tag{7}$$

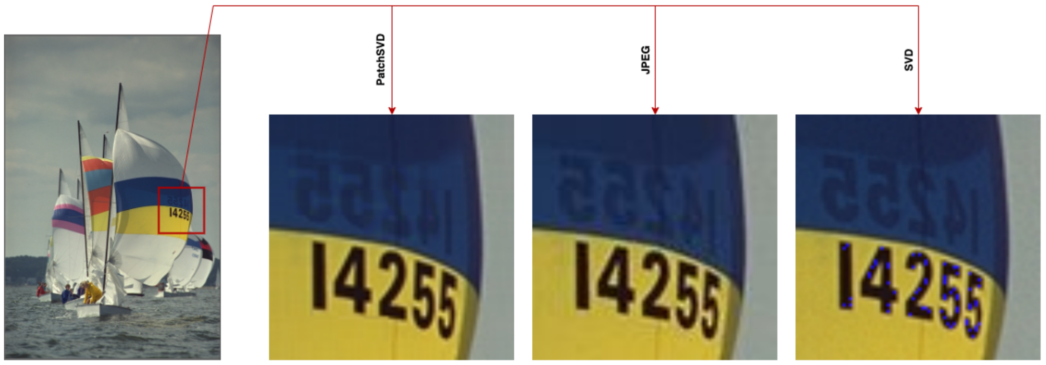
Moreover, we can observe from the condition in 5 that there is a trade-off between the proportion of the complex patches and the values we choose for k_c and k_s. For instance, if we want to keep a higher proportion of complex patches, the difference between k_c and k_s should be higher and k_s should be much smaller than k_c. Note that since $k_c \geq k_s \geq 0$, the proportion of complex patches cannot be higher than a threshold. On the other hand, if the proportion is too small, the number of complex patches may end up being less than 1, in which case PatchSVD will fall back to the standard SVD-based image compression algorithm.

4.4.3 Patch Size Lower Bound

Not every arbitrary patch size works with the PatchSVD algorithm. While the image size is a trivial upper bound for P, it should be noted that greater patch sizes will result in less accurate scoring. More specifically, if the patch size is too large, the score function will lose its sensitivity to complex versus simple patches because of its less local domain. To find a lower bound for patch



(a) Image compression at CR = 85% applied to ”kodim06.png” from the Kodak dataset.



(b) Image compression at CR = 20% applied to ”kodim09.png” from the Kodak dataset.

Figure 4.5: PatchSVD, JPEG, and SVD compression algorithms applied to two image samples from the Kodim dataset show the compression artifacts produced by each algorithm. As you can see, PatchSVD produces more sharp edges which results in perfectly legible text even after the image has been greatly compressed.

size, we simplify Equation 6 by assuming that we are using a square patch with $P_x = P_y = P$. Then, we will have:

$$\frac{P^2}{2P+1} \geq k_s \tag{8}$$

By simplifying this inequality further, we get a lower bound on the patch size as follows:

$$P \geq k_s + \sqrt{k_s^2 + k_s} \geq 0 \tag{9}$$

4.5 Experiments

With PatchSVD, we compress images from two datasets using different patch sizes to evaluate the effect of patch size on the performance of the algorithm. Then, we pick the best patch size for each dataset and compare PatchSVD with SVD and JPEG according to three metrics and by visually comparing the compression artifacts. We also briefly discuss some choices for PatchSVD score functions and how they compare with each other.

4.5.1 Datasets and Metrics

We evaluated our method on Kodak Kodak (1999) and CLIC Toderici et al. (2020) datasets because they contain original PNG format images that were never converted to JPEG. Kodak contains 24 full-color (24 bits per pixel) images that are either 768x512 or 512x768 pixels large. The images in this dataset capture a variety of lighting conditions and contain different subjects. The CLIC dataset was introduced in the lossy image compression track for the Challenge on Learned Image Compression in 2020 and includes both RGB and grayscale images. For all the experiments, we utilized the test split, which comprises 428 samples.

We use traditional image compression metrics, including Mean Squared Error (MSE), Peak Signal-to-noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) Z. Wang, Bovik, Sheikh, and Simoncelli (2004) to evaluate the image compression performance of each method. Lower MSE means better performance, indicating reduced pixel deviations. Higher PSNR and SSIM values signal superior image quality with less distortion and increased similarity between original and processed images.

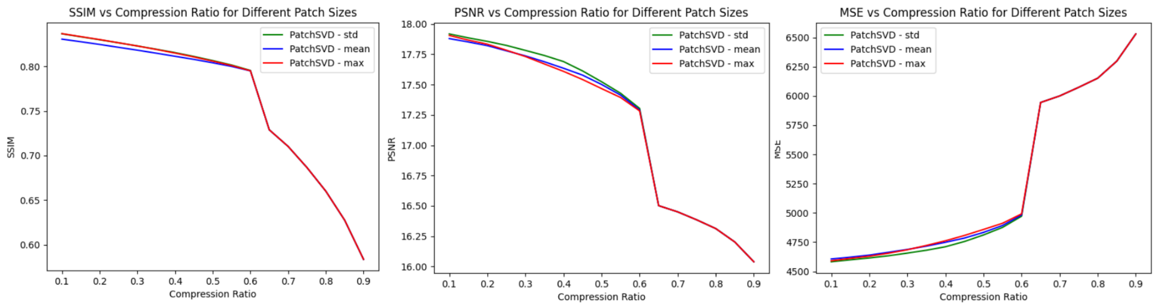


Figure 4.6: This figure illustrates PatchSVD algorithm performance using different score functions, namely, standard deviation, mean, and max. To compare, SSIM, PSNR, and MSE metrics are used. It is demonstrated that standard deviation has a slightly better performance compared to others.

4.6 Results and Discussion

4.6.1 PatchSVD Performance Based on Patch Size

Figure 4.3 depicts the impact of patch size on the performance of the PatchSVD algorithm across the Kodak and CLIC datasets, as measured by SSIM, PSNR, and MSE metrics. The findings indicate that opting for excessively large patch sizes is not advisable, and the effectiveness of compression may be compromised with patch sizes that are too small, contingent on the characteristics of the dataset. Note that the algorithm falls back to SVD for patch sizes that are too large which is why the plots overlap for some compression ratios.

4.6.2 Image Compression Performance Comparison

To better demonstrate the performance of PatchSVD, we compared the performance of PatchSVD with a fixed patch size against JPEG and SVD on CLIC and Kodak datasets. According to the experiment results in Section 4.6.1, patch sizes 10 and 16 were selected for CLIC and Kodak, respectively. Figure 4.4 illustrates the performance comparison with respect to SSIM, PSNR, and MSE. PatchSVD outperforms SVD in all three metrics on both datasets, although JPEG still performs better than PatchSVD. This is expected because neither of the three metrics are context-aware. Nevertheless, PatchSVD may still be preferred over JPEG in some use cases as explained in Section 4.6.3.

4.6.3 Compression Artifacts

While the compression artifacts of PatchSVD and SVD are usually in the form of colored pixels (sometimes called ”stuck pixels”, see SVD output in Figure 4.5(b)), for JPEG, these artifacts take the forms of a general loss of sharpness and visible halos around the edges in the image. In higher compression ratios, the edges of blocks that PatchSVD and JPEG use become visible, too. However, for use cases where sharpness should be maintained locally, PatchSVD is preferable. For example, in Figure 4.5, for both samples, the text written on the boat is more legible when the image is compressed with PatchSVD.

4.6.4 Choice of Score Function

We compared various score functions, including taking the maximum value, averaging, and calculating the standard deviation of the pixel values present in the input patch, as shown in Figure 4.6. The performance of all the score functions is almost similar, but standard deviation yields better results in terms of SSIM, PSNR, and MSE. Hence, the default score function for this algorithm is the standard deviation. The intuition behind this is that standard deviation introduces sensitivity to deviations from the mean which is usually where more complex patterns are present.

4.7 Conclusion

In this work, we introduced PatchSVD as a non-uniform image compression algorithm based on SVD. Through experiments, we demonstrated that the patch size in the PatchSVD algorithm affects the compression performance. Also, we compared the performance of PatchSVD with JPEG and SVD with respect to SSIM, PSNR, and MSE. We compared the compression artifacts that each algorithm introduced to images and illustrated examples of the cases where PatchSVD was preferable over JPEG and SVD because it produced less destructive artifacts in regions that contained information that would have been lost if we applied standard SVD-based image compression. Studying the impact of PatchSVD as an image compression algorithm on the downstream tasks is an interesting future work. Moreover, applying PatchSVD to medical images is a prospective extension because, in medical images, higher resolution is required in the pixels containing diagnostic information compared to the rest of the image and non-uniform local compression could be beneficial. Expert knowledge can lead us to more customized score functions which makes this application even more interesting.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we have studied the relationship between the shape bias of models and their accuracy and robustness. In previous works, it was hypothesized that a model with a higher shape bias would have better robustness to out-of-distribution data. Also, the relationship between accuracy and robustness was somewhat unclear and understudied. To find answers to these questions, we devised experiments on a subset of ImageNet-1K, called ImageNet-100, and demonstrated that first, increasing the shape bias of a model does not necessarily result in a higher OOD robustness. Second, no trade-off exists between in-domain accuracy and out-of-distribution robustness. In fact, we showed some cases where we improved out-of-distribution robustness and in-domain accuracy simultaneously. Moreover, through experiment results, we illustrated that not all data augmentations are beneficial to the model that is being trained. Rather, in some cases, data augmentation can hurt model robustness. In our experiments, we applied 39 different data augmentations to ImageNet-100 and trained various ResNet-50 models on each separately augmented dataset to conduct comparisons.

Last but not least, we studied image encoding as an image transformation and closely investigated one of the most commonly used image encoding methods, namely, JPEG. To mitigate some drawbacks of JPEG, such as compression artifacts around high-contrast edges in images, we introduced a novel image compression algorithm based on Singular Value Decomposition (SVD), called PatchSVD. PatchSVD applies non-uniform compression to an image by using the information retained by SVD as a heuristic to differentiate complex patches from simpler ones. PatchSVD outperforms SVD-based image compression on Kodak and CLIC datasets in terms of SSIM, PSNR, and MSE, which are the commonly used image compression metrics. Moreover, it does not produce JPEG-like compression artifacts around the high-contrast edges which is useful in some use cases, including compressing images of text documents. Also, in cases where domain knowledge allows us to understand the local importance of patches in an image, PatchSVD allows us to perform customized image compression and retain the most important information in the image by taking a non-uniform compression approach.

5.2 Future Work

We propose to study the effect of augmenting a dataset with its PatchSVD-compressed version and compare the robustness results with JPEG compression. In light of this experiment, we can show if keeping edge information with as little noise or artifacts as possible can improve the performance of the model. Furthermore, future studies can compare PatchSVD applied to each RGB channel with PatchSVD applied to other channels, such as the YCbCr color space where the luminance (brightness) and chrominance (color) information are stored as two different components. Applying PatchSVD to YCbCr channels instead of RGB channels could be beneficial because the human eye is more sensitive to changes in luminance than color Tkalcic and Tasic (2003). Therefore, the information can be more heavily compressed in the color channel compared to the luminance channel using PatchSVD, and this may improve compression fidelity. Another direction could be devoted to the integration of finite mixture models McLachlan, Lee, and Rathnayake (2019) to tackle the problems investigated in this thesis Bouguila and Elguebaly (2012); Elguebaly and Bouguila (2010, 2015); Fan and Bouguila (2012); Fan, Sallay, and Bouguila (2017); Mashrgy, Bdiri, and Bouguila (2014); Oboh and Bouguila (2017).

References

Agustsson, E., Mentzer, F., Tschannen, M., Cavigelli, L., Timofte, R., Benini, L., & Gool, L. V. (2017). Soft-to-hard vector quantization for end-to-end learning compressible representations. *Advances in neural information processing systems*, *30*.

Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., & Gool, L. V. (2019). Generative adversarial networks for extreme learned image compression. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 221–231).

Ahmed, N., Natarajan, T., & Rao, K. R. (1974). Discrete cosine transform. *IEEE transactions on Computers*, *100*(1), 90–93.

Akritas, A. G., & Malaschonok, G. I. (2004). Applications of singular-value decomposition (svd). *Mathematics and computers in simulation*, *67*(1-2), 15–31.

Andrews, H., & Patterson, C. (1976). Singular value decomposition (svd) image coding. *IEEE transactions on Communications*, *24*(4), 425–432.

Azizzadenesheli, K., Liu, A., Yang, F., & Anandkumar, A. (2019). Regularized learning for domain adaptation under label shifts. *arXiv preprint arXiv:1903.09734*.

Barnsley, M. F., Ervin, V., Hardin, D., & Lancaster, J. (1986). Solution of an inverse problem for fractals and other sets. *Proceedings of the National Academy of Sciences*, *83*(7), 1975–1977.

Barnsley, M. F., & Hurd, L. P. (1993). *Fractal image compression*. AK Peters, Ltd.

Bartlett, P. L., & Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, *9*(8).

Bashivan, P., Bayat, R., Ibrahim, A., Ahuja, K., Faramarzi, M., Laleh, T., … Rish, I. (2021). Adversarial feature desensitization. *Advances in Neural Information Processing Systems*, *34*, 10665–10677.

Bhavani, S., & Thanushkodi, K. G. (2013). Comparison of fractal coding methods for medical image compression. *IET image Processing*, *7*(7), 686–693.

Borkar, T. S., & Karam, L. J. (2019). Deepcorrect: Correcting dnn models against image distortions. *IEEE Transactions on Image Processing*, *28*(12), 6022–6034.

Bouguila, N., & Elguebaly, T. (2012). A fully bayesian model based on reversible jump MCMC and finite beta mixtures for clustering. *Expert Syst. Appl.*, *39*(5), 5946–5959.

Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., … Rueckert, D. (2018). Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: fast and flexible image augmentations. *Information*, *11*(2), 125.

Cao, L. (2006). Singular value decomposition applied to digital image processing. *Division of Computing Studies, Arizona State University Polytechnic Campus, Mesa, Arizona State University polytechnic Campus*, 1–15.

Charoenphakdee, N., Cui, Z., Zhang, Y., & Sugiyama, M. (2021). Classification with rejection based on cost-sensitive classification. In *International conference on machine learning* (pp. 1507–1517).

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., & Wang, Z. (2020). Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 699–708).

Chen, Z. (2018). Singular value decomposition and its applications in image processing. In *Proceedings of the 2018 1st international conference on mathematics and statistics* (pp. 16–22).

Cheng, Z., Sun, H., Takeuchi, M., & Katto, J. (2018). Deep convolutional autoencoder-based lossy image compression. In *2018 picture coding symposium (pcs)* (pp. 253–257).

Cheng, Z., Sun, H., Takeuchi, M., & Katto, J. (2019). Deep residual learning for image compression. In *Cvpr workshops* (p. 0).

Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, *16*(1), 41-46. doi: 10.1109/TIT.1970.1054406

Christopoulos, C., Skodras, A., & Ebrahimi, T. (2000). The jpeg2000 still image coding system: an overview. *IEEE transactions on consumer electronics*, *46*(4), 1103–1127.

Ciregan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 ieee conference on computer vision and pattern recognition* (pp. 3642–3649).

Cohen, J., Rosenfeld, E., & Kolter, Z. (2019). Certified adversarial robustness via randomized smoothing. In *international conference on machine learning* (pp. 1310–1320).

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning* (pp. 647–655).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., … others (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218.

Elguebaly, T., & Bouguila, N. (2010). Bayesian learning of generalized gaussian mixture models on biomedical images. In F. Schwenker & N. E. Gayar (Eds.), *Artificial neural networks in pattern recognition, 4th IAPR TC3 workshop, ANNPR 2010, cairo, egypt, april 11-13, 2010. proceedings* (Vol. 5998, pp. 207–218). Springer.

Elguebaly, T., & Bouguila, N. (2015). Simultaneous high-dimensional clustering and feature selection using asymmetric gaussian mixture models. *Image Vis. Comput.*, *34*, 27–41.

Fan, W., & Bouguila, N. (2012). Variational learning of dirichlet process mixtures of generalized dirichlet distributions and its applications. In S. Zhou, S. Zhang, & G. Karypis (Eds.), *Advanced data mining and applications, 8th international conference, ADMA 2012, nanjing, china, december 15-18, 2012. proceedings* (Vol. 7713, pp. 199–213). Springer.

Fan, W., Sallay, H., & Bouguila, N. (2017). Online learning of hierarchical pitman-yor process mixture of generalized dirichlet distributions with feature selection. *IEEE Trans. Neural Networks Learn. Syst.*, *28*(9), 2048–2061. Retrieved from https://doi.org/10.1109/TNNLS.2016.2574500 doi: 10.1109/TNNLS.2016.2574500

Franc, V., & Prusa, D. (2019). On discriminative learning of prediction uncertainty. In *International conference on machine learning* (pp. 1963–1971).

Francisco Massa, N. H., Vasilis Vryniotis. (2017). *torchvision.* https://github.com/pytorch/vision. GitHub. (Online; accessed 26-Sep-2022)

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 1–37.

Garg, S., Wu, Y., Balakrishnan, S., & Lipton, Z. (2020). A unified view of label shift estimation. *Advances in Neural Information Processing Systems*, *33*, 3290–3300.

Geirhos, R., Narayanappa, K., Mitzkus, B., Thieringer, T., Bethge, M., Wichmann, F. A., & Brendel, W. (2021). Partial success in closing the gap between human and machine vision. *Advances in Neural Information Processing Systems*, *34*, 23885–23899.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2018). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*.

Ghosh, S., Shet, R., Amon, P., Hutter, A., & Kaup, A. (2018). Robustness of deep convolutional neural networks for image degradations. In *2018 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 2916–2920).

Golpayegani, Z., St-Amant, P., & Bouguila, N. (2023). Clarifying myths about the relationship between shape bias, accuracy, and robustness. In *2023 20th conference on robots and vision (crv)* (pp. 281–287).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, *27*.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). *Explaining and harnessing adversarial examples.*

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

Hendrycks, D., Mazeika, M., Kadavath, S., & Song, D. (2019). Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, *32*.

Hermann, K., Chen, T., & Kornblith, S. (2020). The origins and prevalence of texture bias in convolutional neural networks. *Advances in Neural Information Processing Systems*, *33*, 19000–19015.

Herrmann, C., Sargent, K., Jiang, L., Zabih, R., Chang, H., Liu, C., ... Sun, D. (2022). Pyramid adversarial training improves vit performance. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 13419–13429).

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, *40*(9), 1098–1101.

Jung, A. B., Wada, K., Crall, J., Tanaka, S., Graving, J., Reinders, C., ... others (2020). *imgaug.* https://github.com/aleju/imgaug. GitHub. (Online; accessed 26-Sep-2022)

Kahu, S., & Rahate, R. (2013). Image compression using singular value decomposition. *International Journal of Advancements in Research & Technology*, *2*(8), 244–248.

Kannan, H., Kurakin, A., & Goodfellow, I. (2018). *Adversarial logit pairing.*

Kobayashi, H., & Bahl, L. R. (1974). Image data compression by predictive coding i: Prediction algorithms. *IBM Journal of Research and Development*, *18*(2), 164–171.

Kodak. (1999). *Kodak lossless true color image suite.* Retrieved from http://r0k.us/graphics/kodak (Last accessed on 2023-11-14)

Kouw, W. M., & Loog, M. (2018). An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, *37*(2), 233–243.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., & Cortes, C. (2010). *MNIST handwritten digit database.* http://yann.lecun.com/exdb/mnist/. Retrieved 2016-01-14 14:24:11, from http://yann.lecun.com/exdb/mnist/

Lee, S., Hwang, I., Kang, G.-C., & Zhang, B.-T. (2022). Improving robustness to texture bias via shape-focused augmentation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 4323–4331).

Li, M., Zuo, W., Gu, S., Zhao, D., & Zhang, D. (2018). Learning convolutional networks for content-weighted image compression. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3214–3223).

Lim, S. T., & Abd Manap, N. B. (2022). A region-based compression technique for medical image compression using principal component analysis (pca). *International Journal of Advanced Computer Science and Applications*, *13*(2).

Lim, S. T., Yap, D. F., & Manap, N. (2014). A gui system for region-based image compression using principal component analysis. In *2014 international conference on computational science and technology (iccst)* (pp. 1–4).

Linde, Y., Buzo, A., & Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on communications*, *28*(1), 84–95.

Lipton, Z., Wang, Y.-X., & Smola, A. (2018). Detecting and correcting for label shift with black box predictors. In *International conference on machine learning* (pp. 3122–3130).

Liu, S., Zhang, Z., Qi, L., & Ma, M. (2016). A fractal image encoding method based on statistical loss used in agricultural image compression. *Multimedia Tools and Applications*, *75*, 15525–15536.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Marti, U.-V., & Bunke, H. (2002). The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, *5*, 39–46.

Mashrgy, M. A., Bdiri, T., & Bouguila, N. (2014). Robust simultaneous positive data clustering and unsupervised feature selection using generalized inverted dirichlet mixture models. *Knowl. Based Syst.*, *59*, 182–195.

McLachlan, G. J., Lee, S. X., & Rathnayake, S. I. (2019). Finite mixture models. *Annual review of statistics and its application*, *6*, 355–378.

Mummadi, C. K., Subramaniam, R., Hutmacher, R., Vitay, J., Fischer, V., & Metzen, J. H. (2021). Does enhanced shape bias improve neural network robustness to common corruptions? *arXiv preprint arXiv:2104.09789*.

Naseer, M. M., Ranasinghe, K., Khan, S. H., Hayat, M., Shahbaz Khan, F., & Yang, M.-H. (2021). Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, *34*, 23296–23308.

Oboh, B. S., & Bouguila, N. (2017). Unsupervised learning of finite mixtures using scaled dirichlet distribution and its application to software modules categorization. In *IEEE international conference on industrial technology, ICIT 2017, toronto, on, canada, march 22-25, 2017* (pp. 1085–1090). IEEE.

Papakipos, Z., & Bitton, J. (2022). Augly: Data augmentations for robustness. *arXiv preprint arXiv:2201.06494*.

Pham, D.-L., Chang, T.-W., et al. (2023). A yolo-based real-time packaging defect detection system. *Procedia Computer Science*, *217*, 886–894.

Prakash, A., Moran, N., Garber, S., DiLillo, A., & Storer, J. (2017). Semantic perceptual image compression using deep convolution networks. In *2017 data compression conference (dcc)* (pp. 250–259).

Prasantha, H., Shashidhara, H., & Murthy, K. B. (2007). Image compression using svd. In *International conference on computational intelligence and multimedia applications (iccima 2007)* (Vol. 3, pp. 143–145).

Ranade, A., Mahabalarao, S. S., & Kale, S. (2007). A variation on svd based image compression. *Image and Vision computing*, *25*(6), 771–777.

Roy, P., Ghosh, S., Bhattacharya, S., & Pal, U. (2018). Effects of degradations on deep neural network architectures. *arXiv preprint arXiv:1807.10108*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Sadek, R. A. (2012). SVD based image processing applications: State of the art, contributions and research challenges. *CoRR*, *abs/1211.7102*. Retrieved from http://arxiv.org/abs/1211.7102

Schlimmer, J. C., & Granger, R. H. (1986). Incremental learning from noisy data. *Machine learning*, *1*, 317–354.

Schreuder, N., & Chzhen, E. (2021). Classification with abstention but without disparities. In *Uncertainty in artificial intelligence* (pp. 1227–1236).

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 806–813).

Shen, Z., Liu, J., He, Y., Zhang, X., Xu, R., Yu, H., & Cui, P. (2021). Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, *90*(2), 227–244.

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *Icdar* (Vol. 3).

Simonyan, K., & Zisserman, A. (2015). *Very deep convolutional networks for large-scale image recognition.*

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2014). *Going deeper with convolutions.*

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Thulasidasan, S., Bhattacharya, T., Bilmes, J., Chennupati, G., & Mohd-Yusof, J. (2019). Combating label noise in deep learning using abstention. *arXiv preprint arXiv:1905.10964*.

Tian, M., Luo, S.-W., & Liao, L.-Z. (2005). An investigation into using singular value decomposition as a method of image compression. In *2005 international conference on machine learning and cybernetics* (Vol. 8, pp. 5200–5204).

Tkalcic, M., & Tasic, J. (2003). Colour spaces: perceptual, historical and applicational background. In *The ieee region 8 eurocon 2003. computer as a tool.* (Vol. 1, p. 304-308 vol.1). doi: 10.1109/EURCON.2003.1248032

Toderici, G., Shi, W., Timofte, R., Theis, L., Balle, J., Agustsson, E., … Mentzer, F. (2020). *Workshop and challenge on learned image compression (clic2020).* Retrieved from http://www.compression.cc (Last accessed on 2023-11-14)

Toderici, G., Vincent, D., Johnston, N., Jin Hwang, S., Minnen, D., Shor, J., & Covell, M. (2017). Full resolution image compression with recurrent neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5306–5314).

Torfason, R., Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., & Van Gool, L. (2018). Towards image understanding from deep compression without decoding. *arXiv preprint arXiv:1803.06131*.

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7167–7176).

Wallace, G. K. (1991). The jpeg still picture compression standard. *Communications of the ACM*, *34*(4), 30–44.

Wang, H., & Yu, C.-N. (2019). A direct approach to robust deep learning using adversarial networks. *arXiv preprint arXiv:1905.09591*.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment:

from error visibility to structural similarity. *IEEE transactions on image processing*, *13*(4), 600–612.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer vision–eccv 2014: 13th european conference, zurich, switzerland, september 6-12, 2014, proceedings, part i 13* (pp. 818–833).

Zhao, E., Liu, A., Anandkumar, A., & Yue, Y. (2021). Active learning under label shift. In *International conference on artificial intelligence and statistics* (pp. 3412–3420).

Zheng, S., Song, Y., Leung, T., & Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4480–4488).

Zheng, X., Chalasani, T., Ghosal, K., Lutz, S., & Smolic, A. (2019). Stada: Style transfer as data augmentation. *arXiv preprint arXiv:1909.01056*.

Zhou, L., Cai, C., Gao, Y., Su, S., & Wu, J. (2018). Variational autoencoder for low bit-rate image compression. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 2617–2620).

Zhou, Y., Song, S., & Cheung, N.-M. (2017). On classification of distorted images with deep convolutional neural networks. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 1213–1217).