Investigating Noise Robustness of Convolutional Neural Networks for Image Classification Using Gabor Filters

By

Sangwon Jeong

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

May 8, 2020

Nashville, Tennessee

Approved:

Frank Tong, Ph.D.

Maithilee Kunda, Ph.D.

To my parents, Hong Jeong and Soohyeon Bae. Thank you for years of unconditional support and encouragement.

ACKNOWLEDGMENTS

I thank my PI Dr. Frank Tong, who had to repeatedly give insights and intuitions about the Neuroscience context regarding the project. Also, working in your lab gave me an extensive inspirations and had a big impact on what I want research next in my academic career. Also, all academic and non-academic interactions with Hojin Jang, David Coggan, and Huiyuan Miao in Tonglab have been a very precious experience for me as I learned the importance of thorough discussion in an interdisciplinary environment, which I know will come in handy in any professional careers up ahead.

I also want to thank my Computer Science advisor, Dr. Maithilee Kunda, for all of the lectures and guidance. Your views and philosophy on academic career and science shaped how I think about Computer Science and Artificial Intelligence. I will always remember you as the very first professor in my Computer Science career.

TABLE OF CONTENTS

Page

DEDICATION ii

ACKNOWLEDGMENTS iii

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTERS .

1 Introduction and Background 1

1.1 Image Classification 3

1.2 Convolutional Neural Networks 4

1.3 Noise in Computer Vision 6

1.3.1 Motivation Behind Noisy Images 7

1.3.2 Noise in Supervised Learning 7

1.3.3 Dealing with Noisy Images 8

1.3.4 An Importance of Improving on Noisy Inputs 9

1.4 Types of Noise 10

1.4.1 Salt-and-Pepper Noise 10

1.4.2 Gaussian Noise 11

1.5 A Vulnerability of CNNs to Noisy Inputs 12

1.6 Gabor Filters 15

1.7 Contribution 16

1.7.1 Means to Improve on a Noisy Dataset 16

1.7.2 Provide Reference to Noisy Input Training with Different Levels of Noise 16

1.7.3 Benefits for Real-World Environment Applications 17

2 Methods 18

2.1 Experiment Overview 18

2.1.1 Gabor Filter Design 19

2.2 Dataset 20

2.2.1 Color 21

2.2.2 Categories 21

2.2.3 Noise Synthesis 22

2.2.4 Noise Injection Methods 22

2.3 CNN Architecture and Training Details 23

2.3.1 Architecture 23

2.3.2 Training Configuration 24

2.4 Evaluation 25

2.4.1 Intra-Noise Generalization 25

2.4.2 Inter-Noise Generalization 25

2.4.3 An Assisting Metric for Evaluation 26

3 Results 29

3.1 Performance 29

3.1.1 Results from ImageNet-16 29

3.1.1.1 Clean dataset 29

3.1.1.2 Uniform injection 30

3.1.1.3 Binary injection 31

3.1.1.4 Single-Level injection 31

3.1.2 Results from ImageNet-1000 33

3.1.2.1 Clean dataset 33

3.1.2.2 Uniform injection 34

3.1.2.3 Binary injection 34

3.1.2.4 Single-level injection 35

3.1.3 Inter-Noise Generalization of All Networks 36

3.1.4 An Effect of Orthogonality in Classification Performance 37

3.2 Qualitative Assessment 38

3.2.1 Learned Convolutional Filters 38

3.2.2 Feature Maps 39

3.2.3 Learning Profile 43

4 Discussions 45

4.1 Gabor Filters Benefit Noise-Robustness in a Dataset with Small Number of Categories 46

4.2 Fixing a Filter Reduces the Network's flexibility 46

4.3 Gabor Filters as an Initialization Method 46

4.4 Towards an Integration of Biological Brain and Artificial System 47

5 Conclusion 48

BIBLIOGRAPHY 50

Appendix A Gabor Filter Full Set 56

Appendix B Accuracy Raw Values 59

LIST OF TABLES

Table Page

1.1 Possible approaches to studying the effect of noise in images 14

2.1 Investigated networks 19

2.2 Gabor Filter Specification 20

3.1 Gabor filter networks compared to investigate the effect of orthogonality . . 38

B.1 Accuracy table of networks trained on ImageNet-16, tested on Gaussian noise 59

B.2 Accuracy table of networks trained on ImageNet-1000, tested on Gaussian noise 59

B.3 Accuracy table of networks trained on ImageNet-16, tested on Salt-and-Pepper noise 60

B.4 Accuracy table of networks trained on ImageNet-1000, tested on Salt-and-Pepper noise 60

LIST OF FIGURES

Figure Page

1.1 Images contaminated with Salt-and-Pepper noise 11

1.2 Images contaminated with Gaussian noise 12

1.3 Performance of publicly available CNN weights on noisy color images . . . 13

2.1 Sampled Gabor Filters 20

2.2 Hypothetical performance of an ideal and a sub-optimal intra-noise generalization capability 26

3.1 Accuracy graph of networks trained on clean ImageNet-16 dataset, tested on Gaussian noise 30

3.2 Accuracy graph of networks trained on ImageNet-16 dataset with uniform injection, tested on Gaussian noise 31

3.3 Accuracy graph of networks trained on ImageNet-16 dataset with binary injection, tested on Gaussian noise 32

3.4 Accuracy graph of networks trained on ImageNet-16 dataset with single-level injection, tested on Gaussian noise 32

3.5 Accuracy graph of networks trained on clean ImageNet-1000 dataset, tested on Gaussian noise 33

3.6 Accuracy graph of networks trained on ImageNet-1000 dataset with uniform noise injection, tested on Gaussian noise 34

3.7 Accuracy graph of networks trained on ImageNet-1000 dataset with binary noise injection, tested on Gaussian noise 35

3.8 Accuracy graph of networks trained on ImageNet-1000 dataset with single-level noise injection, tested on Gaussian noise 36

3.9 Accuracy graph of networks trained on ImageNet-1000 dataset, tested on Salt-and-Pepper noise 37

3.10 Weights of the initial convolutional layer extracted from pytorch pre-trained AlexNet [52]. 39

3.11 Convolutional weights learned from three different noise-injection methods and a clean dataset 40

3.12 An original image without noise containing a shark (a). 42

3.13 Learning curve comparison between B-NG and B-G network 44

A.1 Original Gabor filter. A set of 64. 56

A.2 Orthogonal Gabor filter 1. A set of 64. 57

A.3 Orthogonal Gabor filter 2. A set of 64. 57

A.4 Skewed Gabor filter 1. A set of 64. 58

A.5 Skewed Gabor filter 2. A set of 64. 58

Chapter 1

Introduction and Background

Inputs to a real-world computer vision system can be different from the inputs that were used during its development. During the development, using clean high-quality images under a controlled and replicable environment can be justified because it ensures that the collective effort of the research community can build systems upon systems with a vast number of referable results and details. However, systems that are incubated in a very carefully developed cradle will have to be modified and put out in the real-world for a challenge if they ever wish to see to the end of their potential. Inputs that are gathered from the real-world environment cannot be adequately controlled, nor they cannot be easily predicted in advance. For example, systems that have to deal with blurry or noisy images due to the chaotic nature of the environment (e.g., soldier body camcorder), or a low-resolution image feed (e.g., legacy CCTV cameras) might face a challenge in their robustness to distortions in the information source. Moreover, in some cyber-physical applications like self-driving cars, robustness to distortion is critical for personal safety, making such robustness a crucial feature for successful application of such technology.

Possibly, an image classification system is one of the most popular systems to be deployed in real-world settings. Furthermore, there is an increasing body of such systems employing artificial neural networks thanks to its high capacity in fitting high-dimensional information. More specifically, convolutional neural networks(CNN) are often used in image-related tasks mainly due to their outstanding ability in feature extraction (a process of reducing the dimensionality of the dataset while maintaining meaningful information used for a decision making process). Over the last decade, starting from AlexNet's dramatic performance increase in 2012 [1], image classification systems using convolutional neural networks have gain more and more popularity. Nowadays, CNNs achieve superhuman performance and well over it, reaching 1.3% top-5 error on ImageNet dataset[2] in 2020 [3].

An image classification system is a core component of many complicated tasks. Also, a classification system itself can be a standalone service to be deployed. Thus, it is essential to develop a core system that is robust to input from an unpredictable source. There are many sources where unpredictability can come from (environment, hardware, and software), but this work focuses on a particular construct of this unpredictability regardless of the source; A disruption in the information. More specifically, we will focus on visual images with noisy input. Also, we try to improve the system's performance by borrowing a simple yet powerful component found from the biological brain: Gabor filters.

The response properties of neurons in the primary visual cortex (or area V1) or the first major cortical area in the visual pipeline of mammalian brains, are very similar to Gabor filters. More precisely, from looking at the responses that these cells emit, researchers found that the Gabor wavelet is suitable for modeling the behavior of V1 cells [4, 5, 6, 7]. V1 area is thought to be selective, meaning that different cells prefer different characteristics. There are many criteria for the selectivity, but the most broadly studied criteria are the orientation tuning and spatial frequency tuning of these neurons [8]. Moreover, research suggests that the Gabor-like tuning of V1 neurons provides a sparse, efficient code for representing information from the natural visual environment [4]. This, makes using Gabor filters for image classification appealing. As will be introduced in the later chapter, we also try to integrate this orientation selectivity by evenly distributing the orientation of Gabor filters.

It is difficult to draw a direct relationship between noise reduction and area V1. However, a modeling study suggested that Gabor filters, combined with spatiotemporal integration and surround suppression can boost noise robustness [9]. Although image classification via CNNs usually does not include temporal information, however, it is also not clear that Gabor filters cannot boost noise robustness to some degree even without temporal information. Moreover, some of the learned filters (initial convolutional layer) in AlexNet[1] architecture resemble Gabor filters, suggesting that Gabor filters might be indeed beneficial to feature extraction, and feature extraction under noisy environment.

Aggregating our intuitions derived from the observations in the Neurophysiological and Artificial Neural Networks contexts, we hypothesize that Gabor filters might also be beneficial to convolutional neural networks for image classification tasks with noisy inputs. Moreover, we think that Gabor filters are more beneficial, mainly when used during the training time. This is drawn from an intuition that we learn with noise in the environment, not without it.

1.1 Image Classification

There is no crystal clear boundary on how people define image classification. However, we use the term image classification and object recognition interchangeably, and we define the problem as a problem of labeling an image by a class or an object category it belongs to. The task has significant importance because many applications and tasks require successful image classification. That is, any computer vision task, at the end of the day, might require an image classification module to provide the user with a readable label for an object in the scene. For instance, in an object detection task where the goal is to find which object resides at a given location in image space, an image classification can play a role in the ”which object” portion.

We studied the performance of convolutional neural networks for image classification in this work. The reason why we used CNN is twofolds. First, its excellent performance. Starting from the AlexNet's superior entry in 2012 ImageNet challenge[1], which outperformed the second best entry by 10.9%, there has been a proliferation of CNN approaches in the domain. Second, a concept of CNNs is modeled after the biological brains [10], which makes the approach interesting and meaningful for applying biological concepts to it.

Unlike most of the neural network approaches for image classification task where one would expect all convolutional filters (or weights) to be subject to an update, in this work, the initial set of convolutional filters are initialized with Gabor filters and are fixed throughout the whole training process. This approach may be conflicting with many views in the machine learning context. For example, [11] emphasizes that the key aspect of deep learning is that weights need not be manually designed. The use of fixed filters is clearly against this statement. However, it is still meaningful because Gabor-like filters in the V1 area are naturally updated filters and they seem to work quite robustly in our visual system. Moreover, the learned filters in AlexNet look somewhat similar to Gabor filters. Based on the intuition that Gabor filters are a naturally developed feature extractors that are robust to perturbation in visual stimuli and convolutional filters might be developing something similar to them, we wanted to evaluate the hypothesis that CNNs with front-end Gabor filters may perform well on noisy images in CNNs.

1.2 Convolutional Neural Networks

A convolutional neural network is a type of artificial neural network where the word *convolutional* comes from the operation used for a feature extraction process. An artificial neural network, in its simplest form, is a set of nodes with an activation function that determines the output from those nodes. Here, the activation function is calculated as a non-linear function (g) applied on a multiplication of weight matrix (W^T) and an input (x) plus a bias term (b).

$$f(x) = g(W^T * x + b)$$

When this set of nodes start to form a hierarchical structure and gets deep (i.e., multiple hidden layers), they are often referred to as a Deep Neural Network.

These nodes can have connections to other nodes. A typical feed-forward CNN has two different parts: convolutional layers and fully connected layers. Convolutional layers are feature extractors and they are connected sparsely, or it can be said that they have local connections. Computation wise, it means that parameter weights are shared throughout a layer, leading to faster computation and less memory requirement. From a Neurophysiological standpoint, local connections corresponds to having an equivalent of receptive fields – making CNNs more similar to human visual system. A fully connected layer is densely connected and they compress and transform extracted features into an output form. The process is equivalent to decision making because with a non-linearity and a high capability of fully connected layers, an arbitrarily complex and high-dimensional function can be learned.

A power to capture the non-linearity is an important aspect of a classification system when the underlying dataset is assumed to be non-linear. This is because not all information can be modeled in a linear fashion. For example, a regression method may be sufficient in capturing some linear relationship between variables with seemingly clear correlation, but it cannot really capture the complexity in the image dataset. Typical artificial neural networks employ methods such as ReLU (Rectified linear unit)[12] to empower the system with a power to deal with this non-linearity. From a Neurophysiological context, ReLU also makes the system comply with a biological neuron because the output from ReLU will be always positive.

Images are not always aligned, nor are they not in the same format. We can perceive a computer monitor wherever in our sight it is positioned. On the other hand, it is not always that straightforward for CNNs. CNNs, along with convolutional layers, use pooling layers to aid the feature extraction process. A pooling layer has two responsibilities. First, a pooling layer passes strong activations and filters weak activations out. This helps in the generalization ability of the network. Intuitively thinking, by pooling the activations, only the general trend of an object's spatial location or features get to be passed onto the next layer. This makes the network decide based on the generic features of an object, not on specific features of a single instance of an object. Second, it downsamples features. As feature space becomes larger towards the later convolutional layers, it is important to make individual features smaller to prevent an explosion in computation.

In an image classification domain, there is no golden rule to use CNNs. For instance, classification on a less complex dataset such as MNIST[13] can easily achieve good performance without convolution. However, as the complexity and the size of the image dataset gets larger, CNNs can become increasingly useful as they can extract features and abstract those features into a less complex dimension with efficiency. Moreover, the use of pooling layers boosts this process by cutting width and height of the information by factors whenever information pass through them. Also, pooling layers make CNNs more robust by making them more invariant to changes in the spatial position of object features.

As described previously, CNNs can be dissected into two major parts. A feature extractor that extracts information that can be used in the second part, a classifier. A classifier takes that information extracted from a feature extractor and processes them to make decisions according to the objective. Intuitively thinking, in an ideal system, it could be said that any information that can negatively affect the process of the decision making should be dealt with before the decision process. In the CNN context, it would boil down to a feature extractor eliminating or manipulating the noise portion of the information that can possibly affect the decision process in a negative way.

1.3 Noise in Computer Vision

Saying that information is distorted can be very vague. It could mean that the information is contaminated with any type of misleading information, it could simply mean that there is a lack of information (i.e., incomplete information), or both. It is not our objective to completely define the types of noise that can arise in computer vision tasks. Below, I survey some of the major types of noise that have been studied in computer vision, and discuss the types of noise that are evaluated in this study.

1.3.1 Motivation Behind Noisy Images

We bound the scope of distorted information to spatially independent pixel noise in this work. However, there still are two major motivations behind the advent of noise in the context of noise-related image classification research. An intention behind noise can either be malicious or non-existent. First, there is a noise with a malicious intention. The motivation behind this noise is to fool the system so that a system will classify an image incorrectly with high confidence. An image with this sort of noise is often called an adversarial image [14, 15, 16]. An adversarial image can be terribly hard for the classification system to work on - In extreme cases, one pixel is all that it takes to break the system [17, 18]. The other type of noise is noise without malicious intent to fool the system, and the type of noise that is used in this work can be categorized under it. We will simply refer to an image contaminated with unintentional randomly generated noise, a noisy image.

1.3.2 Noise in Supervised Learning

A general notion of supervised learning is that data points are pairs of inputs and labels. If we disregard the case where there is a planned effort to generate noise within the learning pipeline, then noise can occur only on the opposite sides of this pair: A noisy label or a noisy image. A noisy label problem [19, 20, 21] is where some labels of images are missing or misleading. Improving on such noisy labels is important, as one can imagine that the success of such approaches can bring about a monumental increase in the information that machine learning researchers can use. Conversely, a noisy image problem is where images themselves are contaminated with some sort of noise, and this is what we are dealing with in this work.

1.3.3 Dealing with Noisy Images

Intuitively put, any system performing a visual task can deal with noise in two different ways. The first approach is to denoise problematic images before feeding it into the core classification system. The second approach is to train the classification system in such a way that it will be able to deal with noisy images and still perform robustly. An approach that is most representative of the first case is to use noise filtering algorithms such as averaging filters. Averaging filter methods take (*n* X *n*) region in a noisy image and average their values using a moving-window, effectively blurring out the outlier pixels while damaging some of the clean pixels. However, noise filtering approach has a key limitation that there is no universal filter that performs well on all types of noise [22].

An alternative to noise filtering algorithms is to use a more powerful denoising module such as neural networks for noise removal [23] or denoising autoencoders [24]. The word module is used rather than an algorithm because attempts using these heavier modules are a learning-based solution where the system learns to remove noise from the input. Unlike the noise-filtering approach, this approach can be universal if the module is complicated enough. In theory, if a neural network in this module is arbitrarily complicated, then it can process any imaginable type of noise (Neural networks can learn a mapping of whatever dimensionality in the noise space). The drawback to this approach is that training such an arbitrarily powerful system only exists in theory, and a denoising procedure within the classification system can be time-consuming.

The second approach is incorporating the ability to classify noisy images into the core classification system, and this is our approach. Instead of trying to eliminate or minimize the magnitude of noise, it is an attempt to let the core classification system learn to classify with noise still in images. In [25], using 12 types of image manipulations and three popular convolutional neural networks, they found that training the network with some type of noise will enhance the network's ability to perform on images infused with the same type of noise. Despite their comprehensive analysis, they were unable to consider different levels of noise due to the increasing complexity of the study such an attempt will cause. [26] Shows that training CNNs on noisy images (Gaussian noise and Fourier noise) improves their performance on respective types of noise. Furthermore, they ran a comparison study with human performance and found that noise-trained CNNs perform better than human subjects. [27] also focuses on fewer types of noise (Gaussian and Salt-and-Pepper) and shows the effect of training a network with differing levels and different types of noise.

Introduced approaches need not be exclusive. For instance, in [27], they contaminate the training dataset with Gaussian and Salt-and-Pepper noise to denoises it before feeding it into the network during training time. They find that denoising images and training those images will increase the network's performance. Likewise, it is an open area where one can creatively construct a system to fight negative effect from the noise.

1.3.4 An Importance of Improving on Noisy Inputs

An accuracy and suitability for real-world applications of an image classification system is an important aspect of its success. A system can perform with an expected accuracy in a controlled environment (desirable lighting conditions, no occlusions, noise-free). However, there is no guarantee that it will behave with the same accuracy when it is put to the test in a real-world environment. There can be many factors that can contribute to the inconsistency of the system's capabilities, but noise, in particular, is by far one of the most critical factors. Unlike lighting conditions (variations in lighting levels) that can be easily pre-processed on a hardware level (sensors), noise cannot be entirely eliminated since it is considered as an absence of information. On the other hand, unlike occlusions that cannot be processed without computationally expensive mechanisms due to a severe loss of information, there is a way that some useful information can be extracted from noisy inputs. This work focuses on types of noise that do not occlude an object to be classified. Also, variations in lighting conditions are not considered as a variable here.

1.4 Types of Noise

Noise in the image domain can be further divided into three major classes: additive noise, multiplicative noise, and impulse noise [22]. Impulse noise refers to a noise that completely disregards an original pixel value, i.e., a noisy pixel is by no means indicative of a true pixel. A good example of an independent noise is a Salt-and-Pepper noise. There are only two noisy values in Salt-and-Pepper noise, meaning that the value is either at the very beginning of a value spectrum or at the very end. For example, if a pixel is an 8-bit pixel (value ranges from 0 to 255 discretely), then the noise will be either 0 or 255. Second, additive noise is a type of noise that can retain some of the original pixel value since noise drawn from some distribution that is not related to the image itself is added to the original image. Gaussian noise is a good example of an additive noise, which will be explained in the next section. Finally, a multiplicative noise is similar to the additive noise, but the operation between an original image and a noise value is multiplication rather than an addition.

1.4.1 Salt-and-Pepper Noise

Salt-and-pepper noise is straightforward to understand. It is an impulse noise that discards all the information that an original pixel had before being contaminated with it. Generally speaking, to generate this noise, only one parameter that controls a ratio of noisy pixels versus original pixels is required. For example, if this parameter is bound between 0.0 and 1.0 and a chosen value is 0.5, then it means that 50% of pixels in an image is either black or white. Example images with Salt-and-Pepper noise is demonstrated in Figure 1.1.

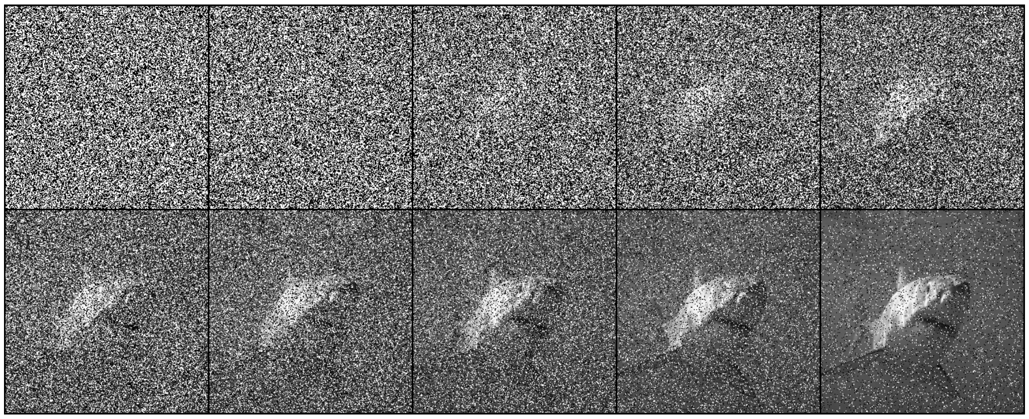


Figure 1.1: Images contaminated with Salt-and-Pepper noise

From top left to bottom right, SSNR value starting from 0.0 increases by 0.1 per image

1.4.2 Gaussian Noise

An image contaminated with Gaussian noise can be described as follows. Signal is an underlying clean image. μ, and σ are the mean and the standard deviation of Gaussian distribution. Value sampled from this Gaussian distribution is statistically independent from the signal, and is added to the signal.

$$Image = S + G(\mu, \sigma) \tag{1.1}$$

The intensity of noise can be adjusted primarily with σ. However, we are using a value called Signal-to-Signal-Plus-Noise Ratio(SSNR), which ranges from 0 to 1.0, to control the intensity of noise in the image. The use of SSNR changes the definition of Gaussian noise slightly different from the basic Gaussian noise. Our noisy image I can be defined as in equation 1.2. Here, S stands for the source image and N stands for the noise sampled from the Gaussian distribution. A_s and A_n are multiplicative factors that sum up to 1, and we use them to bound any resulting pixel values, after adding noise, between 0 and 1.

$$I = S \cdot A_s + N \cdot A_n \tag{1.2}$$

Equation 1.3 describes the SSNR value for an image. If SSNR is 1.0, then it means a factor A_s for signal S becomes 1.0 and a factor A_n for noise becomes 0.0 – resulting in an image without any added noise. On the other hand, if SSNR is 0.0, then A_s for signal S becomes 0.0. This results in a pure noise. Notice that $SSNR = A_s$ and the relationship between SSNR values and resulting noisy images are linear with a slope 1. This property results in SSNR values from 0 to 1 evenly spanning over all possible noisy image space. We are sharing this concept from a noise study [26].

$$SSNR = A_s/(A_s + A_n) \tag{1.3}$$

An example image with Gaussian noise is demonstrated in Figure 1.2.

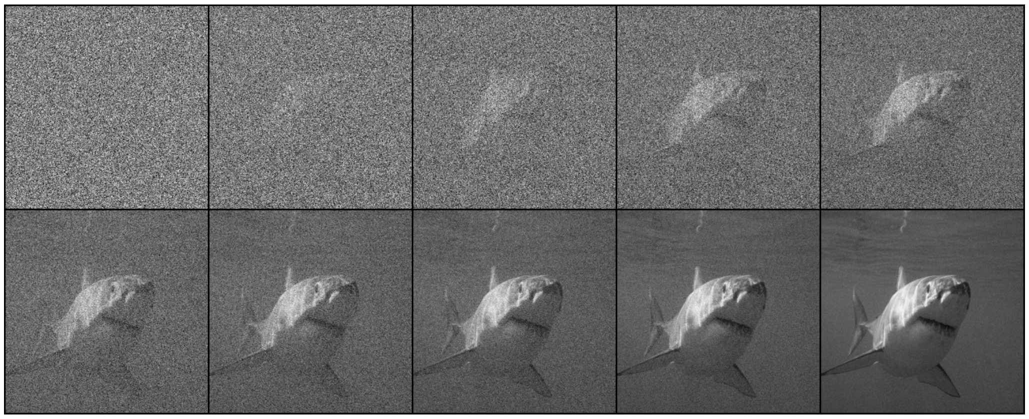


Figure 1.2: Images contaminated with Gaussian noise

From top left to bottom right, SSNR value starting from 0.0 increases by 0.1 per image

1.5 A Vulnerability of CNNs to Noisy Inputs

Undoubtedly, convolutional neural networks are a powerful method for image classification tasks. For a concrete reference on how well they perform, its top-5 error (4.94%) [28] surpassed that of a human-level top-5 error (5.1%) [29] in 2015. The growth did not stop there and is on an increasing track, reaching 1.3% top-5 error in 2020 [3]. Despite its ever-growing performance, CNNs can sometimes be vulnerable to the changes in the testing environment.

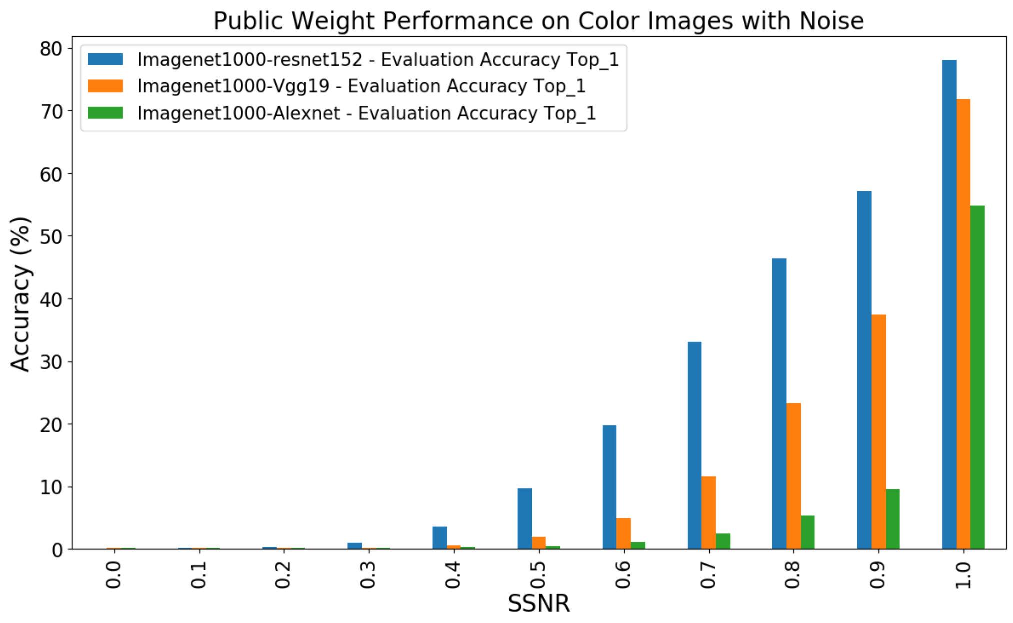


Figure 1.3: Performance of publicly available CNN weights on noisy color images

When noisy images are presented to a network that was trained on clean images, they tend to show a sub-optimal performance. Along with many similar works, [30] demonstrates that the accuracy of a network (trained on clean images), which once showed 60%, retreated back to 15% when tested on images where only 10% of pixels are contaminated with a Salt-and-Pepper noise. As another demonstration of such behavior, Figure 1.3 shows performances of publicly offered pre-trained weights for AlexNet[1], VGG19[31], and Resnet152[32] on varying levels of noisy images. Although deeper and newer networks(Resnet152) tend to do better than shallow networks (AlexNet), all of the networks suffer from noisy inputs.

In previous approaches, an effect of noise on image classification was investigated with some combination of experiments specified in Table 1.1 (Abbreviated notation borrowed from [30]). Studies involving denoising algorithms as a means of pre-processing noisy images mostly focus on a model tested on denoised images(N2D, C2D) [30, 27, 33, 22, 34]. This effort also includes studies involving denoising autoencoders [35], or feedforward networks for denosing [23]. These efforts suggest solutions to dealing with noisy images but require an additional noise-related module in the system. On the contrary, studies involving training a network with noisy input focus their investigation on cases N2N, C2N, N2D [36, 37, 25, 38, 26]. Our approach is in line with training a network directly on noisy images to brew the network's capability to perform on noisy inputs, without having to add a denoising algorithm or a module.

-	Trained on Noise	Trained on Clean	Trained on Denoised
Tested on Noise	N2N	C2N	D2N
Tested on Clean	N2C	C2C	D2C
Tested on Denoised	N2D	C2D	D2D

Table 1.1: Possible approaches to studying the effect of noise in images

Breaking CNNs is not entirely a bad thing. Adversarial images or noisy images can be used during training time with the hope that the model will learn to perform better when faced with disruptive inputs. Noisy images can indeed benefit the performance of a network. Following from the example shown from the last section, which was from [30], if the same network that showed 15% accuracy on 10% of salt and pepper noise were trained on images mixed with salt and pepper noise, the accuracy goes back to about 55%. The trend holds true for all of their experiments.

CNNs are sometimes easy to break. However, the very fact that they can be broken can be used in their favor, and this is what future research will have to care for in addition to just enhancing performance. A robust image classification system demands multiple properties: training efficiency, online inference, scalability, compact architecture, robustness to changes in the environment, etc. Not all of them can be solved overnight, but they can be conquered one-by-one. Similar to the studies introduced above, this study focuses on investigating CNN's robustness to changes in the input image. We do so by training networks on noisy images that would have made them vulnerable otherwise.

1.6 Gabor Filters

Gabor filters or Gabor wavelets (Examples in Figure A.1), are used widely in the vision domain. Although the trend in computer science seems like that they are being replaced by freely learned feature extractors, due to their ability to encode orientation information[39], they can be used for texture-based feature extraction [40] and can play a crucial role in most of the computer vision tasks such as classification, segmentation[41], and edge detection[42]. More generally, since the responses of Gabor filters are similar to the responses that of the V1 cells in the human visual system [43], it can be used for image representation study [43, 44]. Studying image representation is important because it can reveal both high and low level description of an image. As a simple example of image representation, extracted features from convolutional layers can be thought of as one kind of low level description of an image. On a more practical and application side, Gabor filters can be used in computer vision algorithms for face recognition [45, 46], fingerprint identification, or fingerprint scan denoising [47], to name a few.

It was previously mentioned we will probe noise robustness of networks with Gabor filters versus networks with learned filters. Additionally, we will also test the hypothesis that evenly distributed (i.e., evenly spaced orientations) Gabor filters will be most likely to have the upper hand in noisy image classification when compared to unevenly distributed Gabor filters. The rationale behind this hypothesis has two interpretations. Evenly distributed orientations can lead to more balanced responses. On the other hand, unevenly distributed filters can lead to correlated responses that hurt information gain. Another way to see this is that evenly oriented filters are more likely to capture a larger hypothesis space as a function. The hypothesis will be tested by comparing CNNs with different front-end Gabor filters defined by different orientation sets, which is further described in Table 2.2.

1.7 Contribution

1.7.1 Means to Improve on a Noisy Dataset

Noise can be smoothed or eliminated via noise removing filters such as mean filters, median filters, or adaptive filters [48, 49]. However, there are some shreds of evidence that the different types of noise require different kinds of treatment [22]. Predicting the type of noise that a classification system will have to deal with is a difficult task. Thus having a pre-processing step to treat noisy images might not be a generic approach. Although it does not necessarily work better than treatments with pre-processing steps, the idea of training the system with noisy input in the first place does not require an assumption of having to know the type of noise in advance, so it can be used with more freedom regardless of the type of noise. Multiple studies such as [26, 25] show that convolutional neural networks can improve on noise if trained on noisy input images.

Especially in [26], which set a foundation for this study, AlexNet, variants of VGG network, GoogLeNet, and ResNet-152 were trained on images corrupted with Gaussian and Fourier phase-scrambled noise noise. They found that training CNNs on noisy images improved the networks' noise-robustness. More specifically, they show that noise-robustness can be improved both for spatially correlated and uncorrelated noise (Fourier and Gaussian noise respectively). Moreover, they found that CNNs can attain and sometimes surpass human-level noise robustness.

1.7.2 Provide Reference to Noisy Input Training with Different Levels of Noise

There are multiple previous studies introduced throughout this paper that provide observations of behaviors from different convolutional neural networks, different datasets, and different noise types. These observations can play a critical role in future studies involving convolutional neural networks and noise. This study also provides one such observation on the behavior of convolutional neural networks and noise with the hope that some future studies can make use of these findings.

1.7.3 Benefits for Real-World Environment Applications

One of the goals for an image classification system is to be successfully deployed in the real-world environments after being developed. However, these systems sometimes perform sub-optimally due to unexpected circumstances present in the deployment environment. An approach developed here might find a place in some real-world applications.

Chapter 2

Methods

The main purpose of this study is to see how Gabor filters, that are thought to be residing in the V1 cortex of the human brain's visual pipeline, affect the performance of CNNs in a noisy image classification task. First, we hypothesize that the use of Gabor filter in the initial convolutional layer of AlexNet during training will enhance the network's robustness to noise. To explore this hypothesis, we trained networks on two datasets: ImageNet with 16 categories and ImageNet with 1000 categories. Training on a smaller dataset enabled us to quickly train networks to get preliminary results. Also, we used grayscale images to eliminate the influence of color information during classification. This lets us solely focus on the texture, the shape, and the noise itself.

To focus on the effect that the Gabor filters give, we limit the use of noise to two different types of noise; a Gaussian noise and a Salt-and-Pepper noise. Furthermore, Salt-and-Pepper noise was not used as a noise for the training images. We employed several different ways of introducing noisy images during training – we call them noise injection methods. By having different ways of injecting noise, which will be described in a moment, an effect that training with noise will be contrasted in more depth compared to prior works that use a sole method of injecting noise.

2.1 Experiment Overview

All networks trained for the experiment are shown in 2.1. All of them are trained on Gaussian noise. There are six different types of filters (Non-Gabor, Gabor, Gabor-Orthogonal 1, Gabor-Orthogonal 2, Gabor-Skewed 1, Gabor-Skewed 2). Networks with six different filters are then trained with different ways of presenting noisy images (noise injection methods), which gives 24 different networks per dataset. We experimented with two datasets (ImageNet 1000 categories and ImageNet 16 categories), so the number of networks trained in total is 48.

	Clean	Uniform	Binary	Single-Level
Non-Gabor	C-NG	U-NG	B-NG	S-NG
Gabor	C-G	U-G	B-G	S-G
Gabor Orthogonal 1	C-GO1	U-GO1	B-GO1	S-GO1
Gabor Orthogonal 2	C-GO2	U-GO2	B-GO2	S-GO2
Gabor Skewed 1	C-GS1	U-GS1	B-GS1	S-GS1
Gabor Skewed 2	C-GS2	U-GS2	B-GS2	S-GS2

Table 2.1: Investigated networks
Abbreviated {noise-injection method}-{filter type}

These networks are then tested on two types of noise; Gaussian noise and SaP noise; to investigate an how well a network perform on unseen levels of noise and how well a network generalizes on unseen type of noise (intra-noise generalization ability and an inter-noise generalization ability).

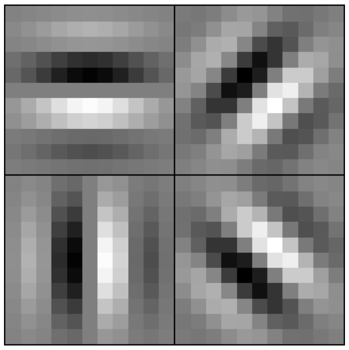
2.1.1 Gabor Filter Design

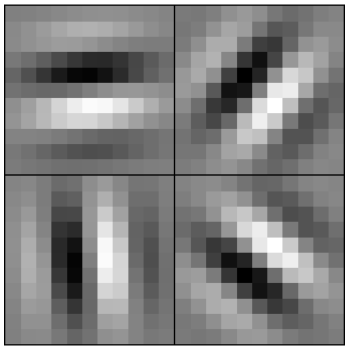
As summarized in Table 2.1, there are five different types of Gabor filters. These five different filters with differing orientations will provide one additional mini-study that investigates whether evenly distributed orientations for Gabor filters will benefit classification performance. However, while orientations for five filters are a variable, phase and frequency remain the same across all filters as specified in Table 2.2. There are 64 convolutional filters for the initial layer in the version of AlexNet that we use. Each of these 64 filters are an exclusive combination of the parameter sets in the table.

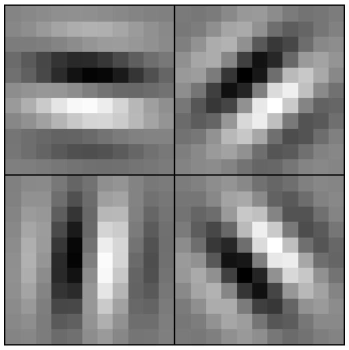
Figure 2.1 shows some of the filters sampled from the set of Gabor filters. Five respective images contain four filters. Each filter in these sample images has a common frequency of 2, and phase 0. Orientation differences are described in the figure.

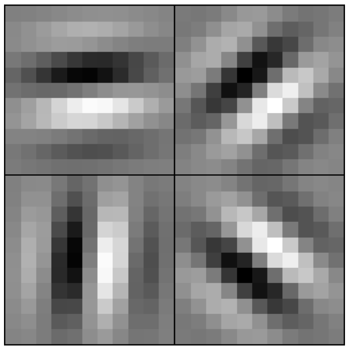
	Phase Set	Spatial Frequency Set	Orientation Set
	(π)	(cycles/filter)	(Degrees)
Gabor	0, 0.5, 1, 1.5	0.5, 1, 2, 4	0, 45, 90, 135
Gabor Orthogonal 1	0, 0.5, 1, 1.5	0.5, 1, 2, 4	5, 40, 85, 140
Gabor Orthogonal 2	0, 0.5, 1, 1.5	0.5, 1, 2, 4	-5, 40, 85, 130
Gabor Skewed 1	0, 0.5, 1, 1.5	0.5, 1, 2, 4	5, 40, 85, 140
Gabor Skewed 2	0, 0.5, 1, 1.5	0.5, 1, 2, 4	-5, 50, 95, 130

Table 2.2: Gabor Filter Specification


(a) Orientation 0, 45, 90, and 135 degrees


(b) Orientation 5, 40, 85, and 140 degrees


(c) Orientation -5, 40, 85, and 130 degrees


(d) Orientation 5, 40, 85, and 140 degrees

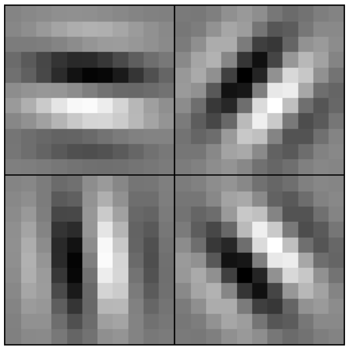

(e) Orientation -5, 50, 95, and 130 degrees

Figure 2.1: Sampled Gabor Filters
All orientations descriptions are in left to right, top to bottom order. (a) is original Gabor filters and orientations are evenly spaced(45 degrees) and this holds for (b) and (c) as well. However, orientations for filters (d) and (e) are not evenly spaced(35, 45, 55 and 55, 45, 25 degrees apart, respectively). Under the assumption that evenly spaced filters will perform better, filters (a), (b), (c) should do better than filters (d) and (e).

2.2 Dataset

We are using ImageNet(ILSVRC2012)[2] for training and testing. The diversity of the object classes it contains makes it appealing for our image classification study. Moreover, it can be imagined that a construct of biological evolution, Gabor filter, might benefit from the dataset's emphasis on natural objects(fauna and flora). In this section, how our manipulation of images will be explained along with the necessary concepts behind it.

2.2.1 Color

We used grayscale images to train and test the convolutional neural network. The choice was made to investigate the effect of noise more clearly. One might ask whether it is appropriate to use grayscale images to test recognition performance, rather than color images, as color can provide some semantic information with respect to the object category. However, there is evidence that using grayscale has minimal effect on the performance of image classification networks [25]. pre-training a network with grayscale images can sometimes produce better weights for transfer learning in certain domains such as medical imaging [50].

There are numerous ways to convert a color image to a grayscale image. [51] shows that different conversion methods give different performance in image classification. According to [51], a method called *Gleam* and a variant method of *Intensity* had the best performance. However, we used a method called *Luminance* that ranked the fourth least-performing method, where a grayscale value is obtained by taking a sum of R, G, B value multiplied by $0.299, 0.587, 0.114$ respectively. The choice was inherent to the PyTorch library [52] that we used to implement experiments.

2.2.2 Categories

One smaller variant of ImageNet was used alongside an original ImageNet dataset. We will refer to the smaller dataset as ImageNet-16 because it has 16 categories. It is comprised of 8 animate objects and 8 inanimate objects, and is the same dataset as the one used in [26]. They chose these 16 categories in a way that different categories would be somewhat confusable with one another. The primary purpose of the smaller dataset is to get preliminary results to the main experiment(1000 categories). The original ImageNet dataset will be referred to as ImageNet-1000.

2.2.3 Noise Synthesis

By noise synthesis, we refer to an activity of adding noise into an image. An important aspect to note is that we synthesized noise on-the-fly, meaning that the noise image dataset was not constructed prior to the training process. Our approach, opposed to constructing a dataset with noise prior to training, might give the network a harder challenge as the dataset is not deterministic. For example, over the span of 100 epochs, the same image can be shown with a differing noise level. Even if somehow the noise level was the same from the other, the actual noise pattern will most certainly be different because noise is newly sampled every time it is generated.

2.2.4 Noise Injection Methods

We devised three ways to show noisy input to the network, and we refer to these different noise injection conditions as *injection*. The three types are uniform noise injection, binary noise injection, and single-level noise injection. It is meaningful to train networks with different injections because it can give a more holistic view on the effect of noise. For example, [26] found that training a network using a particular SSNR level led the network to overfit on images synthesized with SSNR levels around the trained SSNR. On the other hand, they also found that training a network with a combination of noisy images and noise-free images led to networks performing better on a wider range of SSNR levels – a higher intra-generalization ability. By having different injection methods, we can obtain a better understanding of an effect that noise have on classification. This section is dedicated to explaining these methods and what to expect from these different types of injection.

First, Uniform noise is the case when a probability of a certain noise level arising during training time is identical across all noise levels. If a low boundary and an upper boundary of

SSNR is chosen to be respectively 0.9 and 1.0, then any floating value in between these two numbers will have an equal chance of being drawn. This injection method is the baseline for our noise study, which will reveal an overall ability of a network to classify noisy images. In the uniform noise injection experiments, SSNR 0.2 and SSNR 1.0 is chosen to be the lower bound and the upper bound.

Second, Binary noise is called binary because images mixed with only two levels of noise are introduced to a network during training time. For example, if the lower SSNR is 0.9 and upper SSNR is 1.0, then half of the training images will contain 10% noise, and the other half will be a clean image. This injection method will reveal how well the network generalizes on unseen levels of noise. In the binary noise injection experiments, we chose SSNR 0.4 and SSNR 1.0.

Lastly, in single-level noise injection experiments, only one level of noise is used during training. This method is an extreme case of binary injection, and the purpose of the study is to see if the network has the ability to start learning from noisy images to start with. We chose SSNR 0.4 in our single-level noise injection experiments.

Besides the aforementioned three injection methods for training, networks are trained on untouched images to see the effect of noise on the networks' classification accuracy. This experiment will serve as a reference to a performance reduction caused by noisy images, and also as an indicator of the network's ability to generalize to noisy input without having been trained on noisy input.

2.3 CNN Architecture and Training Details

2.3.1 Architecture

Although there are architectures with much better performance and better noise generalization ability, we used AlexNet [1]. AlexNet was chosen for multiple reasons. First, AlexNet is the first widely known CNN that later contributed to a large body of trailing researches and has been used as a benchmark in variety of studies. Second, the use of large convolutional filters (11x11) at the initial convolutional layer makes it appealing for the study, because Gabor filters can be more effectively characterized if the kernel size is larger. For example, if you want filters with high frequency (more strokes), then smaller filters, such as (3x3), cannot completely convey boundary information of different phases, thus failing to characterize this Gabor filter. For the full set of Gabor filters, see Figure A.1. A structure of AlexNet is straightforward, and is consisted of 5 convolutional layers followed by 3 fully-connected layers, and an average pooling layer in between.

AlexNet was modified slightly in two ways. First, for grayscale training an input dimension for the initial layer of the feature extractor was altered to 1 from 3. Second, specific to the experiment with ImageNet-16, an output dimension of the last layer in a classifier was changed from 1000 to 16. Additionally, when training a network with custom Gabor filters, bias was initialized to 0.01 for all filters rather than random initialization. Such a decision was made to see which set of filters would be preferred by a network after the training was over.

2.3.2 Training Configuration

Stochastic gradient descent (SGD) with a learning rate (0.01), Nesterov momentum (0.9) and weight decay (1e-6) was used as an optimizer. Also, a learning rate was reduced by a factor of 10^{-1} on a plateau (where evaluation accuracy starts to increase slower). For the batch size, we chose to use 256 images per batch. We ran 60 epochs for every network training on the 1000-category ImageNet dataset and 150 epochs for all network training on the 16-category ImageNet dataset. We didn't do a model selection for evaluation since the final testing was done on the validation set that was used during a training-time for validation step. As for the model selection, to make the condition fair for every network, the last learned weight was chosen for testing.

Some standard augmentation methods(random cropping and flipping) were bringing additional difficulty to the network during the training process, so we removed all of the augmentation methods. This does not necessarily mean that the dataset will be any easier than other convolutional neural network research because we are effectively augmenting the dataset via noise synthesis.

2.4 Evaluation

Whether or not a model performs well on noise can be answered through a question ”How well does it perform on the images contaminated with the same type of noise that the network was trained on?” and ”How well does it perform on the images contaminated with a different type of noise that the network was trained on?”. Yet, there is no unified way of measuring such capabilities. In this study, we try to measure noise generalization with three quantitative metrics and one qualitative property. We start by defining intra-noise generalization and inter-noise generalization.

2.4.1 Intra-Noise Generalization

As was briefly touched while explaining binary noise injection and single-level noise injection, intra-noise generalization ability is to see whether a network performs well in unseen levels of noise within the same class of noise. Since we are training the network on a Gaussian noise, this investigation is only done through testing networks on images with Gaussian noise. For concrete understanding through an example, Figure 2.2 shows a hypothetical situation between an optimal network with good intra-noise generalization and a sub-optimal network with poor intra-noise generalization capability.

2.4.2 Inter-Noise Generalization

One can test a network that was trained on one type of noise with a different type of noise, to determine how well the network generalizes on an unseen type of noise. Inter-

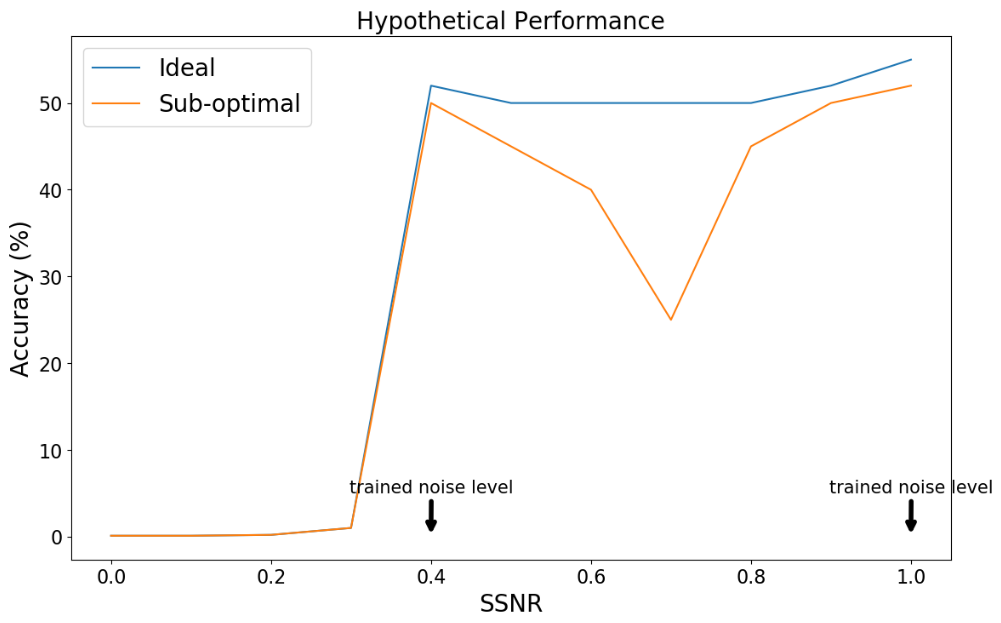


Figure 2.2: Hypothetical performance of an ideal and a sub-optimal intra-noise generalization capability

A hypothetical network trained with a binary-injection method with SSNR 0.4 and SSNR 1.0. An ideally-optimized network should be able suffer less on unseen levels of noise, in this case from SSNR 0.5 to SSNR 0.9.

noise generalization capability is crucial to the network, especially when it is deployed because the type of noise that might occur is hard to predict. Although there are a number of possible distinctive noises, we explored this capability by testing a network trained on Gaussian noise on a Salt-and-Pepper noise.

2.4.3 An Assisting Metric for Evaluation

The problem is that there is no unified way of measuring such generalization abilities, let it be within the same noise type or across different noise types. It is hard to enforce one unified metric, since noise characteristics can differ from one study to another, and from noise to noise. What is even worse, ways of generating noise can differ. So we devised a metric that can be used uniformly across different studies. To start with, a generic way of measuring noise robustness at any single noise level can be stated as in 2.1.

$$R_\alpha = A_\alpha * D(\alpha, \beta) \tag{2.1}$$

Where R_α is a network's robustness at noise level α, A_α is an accuracy at noise level α, and $D(\alpha, \beta)$ is a distance between the tested noise level α and trained noise level β. For example, if a network was trained on a clean dataset, then this β will become 1.0. However, this alone cannot be descriptive of a network's robustness to noise at every tested level of noise. So, we you can sum all R_α for all $\alpha \subset A$, where A is a set of all levels of noise tested.

$$R = \Sigma_{\alpha \in V} \left[\frac{A_\alpha * D(\alpha, \beta)}{|V|} \right] \tag{2.2}$$

Where V is a set of all tested SSNR levels, and $|V|$ is the size(or length) of the set.

Alternative distance measures can be used in place of $D(\alpha, \beta)$. Choosing a good distance measure will make the metric more feasible in different applications, and some distance measure will capture noise-robustness better. However, in this study, we used a Euclidean distance that becomes a simple arithmetic difference.

$$D(\alpha, \beta) = \min_{\beta \in T} f(\alpha, \beta) \tag{2.3}$$

Here in 2.3, T is a set of all trained SSNR levels. The minimum distance is multiplied to the accuracy to account for a case where a network is trained on multiple levels of SSNRs, such as binary or uniform injection. In uniform injection case, we make T discrete by rounding the values to n integers(e.g. T={0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0} if SSNR was drawn from a distribution(0.4, 1.0)). Throughout this study, this value will be denoted as an R-value. By discretizing, an upper bound for this value becomes 550 and a lower bound becomes 0.

However, the metric still cannot capture all of the aspects of noise-robustness. First, it does not take into account the increasing difficulty of the task as the noise level becomes more severe. For example, a relationship between a model that learned on SSNR 1.0 and was tested on SSNR 0.9 and a model that learned on SSNR 0.3 and tested on SSNR 0.2 should be different. It should be different because the latter model had to learn with a severe noise in the first place, making the learning process more obscuring than the former model. One possible fix would be to introduce another distance function that scales up severe noise levels, but it is also hard to assume the difficulty in the first place. The second problem is that it is very hard to measure the difference between two different types of noise. One noise can be harder for any system to study than the other. In our experiment, Salt-and-Pepper noise is harder than Gaussian noise because Salt-and-Pepper noise does not retain any information from the original pixel values. However, how much that different characteristic for each noise types affects the learner is almost impossible to be modeled quantitatively. For these reasons, we only use this metric as a support metric aside from a qualitative assessment in this study.

Chapter 3

Results

3.1 Performance

3.1.1 Results from ImageNet-16

A smaller dataset makes it possible to observe the tendency of network behaviors before delving into a bigger dataset. Overall, we found that networks with Gabor filters performed better than networks without them when they are trained on a noisy dataset. Moreover, the non-Gabor filter networks failed to converge if they were trained with noisy images only (single-level injection). However, networks trained without Gabor filters generalized better to images synthesized with severe noise, i.e., they tended to have higher accuracy in lower SSNR levels. On the other hand, networks with Gabor filters worked better with clean images regardless of how they are trained. Observations are explained in the following sections.

3.1.1.1 Clean dataset

We started by investigating networks trained on clean images. Intuitively thinking, if networks are trained on clean images, it is likely that their performance would gradually decrease as the noise level rises, and this characteristic is well portrayed in Figure 3.1. The result showed that a network trained without Gabor filters performs well on rather severe noise levels(SSNR 0.4 - SSNR 0.7). However, networks trained with Gabor filters do better with less noisy images(SSNR 0.8 - SSNR 1.0). This observation suggests that the Gabor filters does not benefit CNNs in classifying noisy images when the network was trained exclusively on clean images. Hence, in a clean dataset case, a network without Gabor filters is the most performant as it has the highest R-value(9.98). The least performant network's R-value is 7.98, and it is trained with GO1 filters.

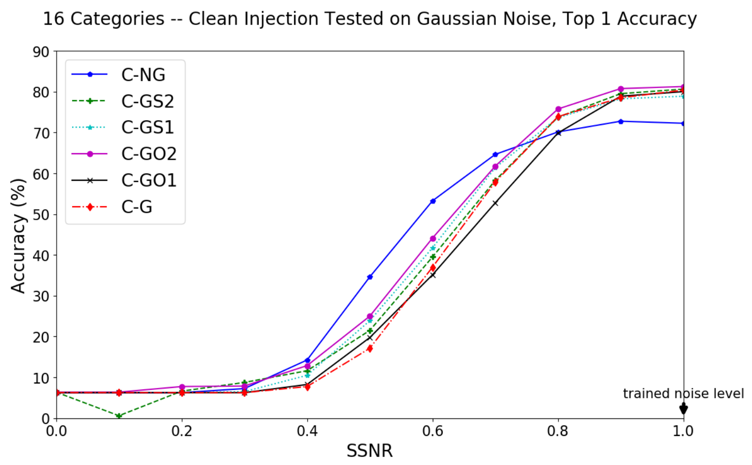


Figure 3.1: Accuracy graph of networks trained on clean ImageNet-16 dataset, tested on Gaussian noise

Raw values in Appendix Table B.1

3.1.1.2 Uniform injection

We predicted that training with the uniform injection method should lead to the greatest noise robustness since the network will learn to process the same images on multiple occasions with varying levels of noisiness. Figure 3.2 reveals that the overall accuracy of the network is superior to those trained on the clean dataset case. Here, networks with Gabor filters tend to perform better than the network without it, across the full range of noise levels used for training. However, this improvement did not appear to lead to an overall leftward shift of the accuracy by SSNR curve, implying that both types of networks were similarly impaired by images presented at SSNR levels lower than those used for training. This observation displays the simple fact that networks with Gabor filters are more performant than the non-Gabor network. To be more specific, an R-value for the network with GO2 filters is 1.79, and an R-value for the network with freely learned filters is 1.71. The least performant network was the one with the GO1 filters.

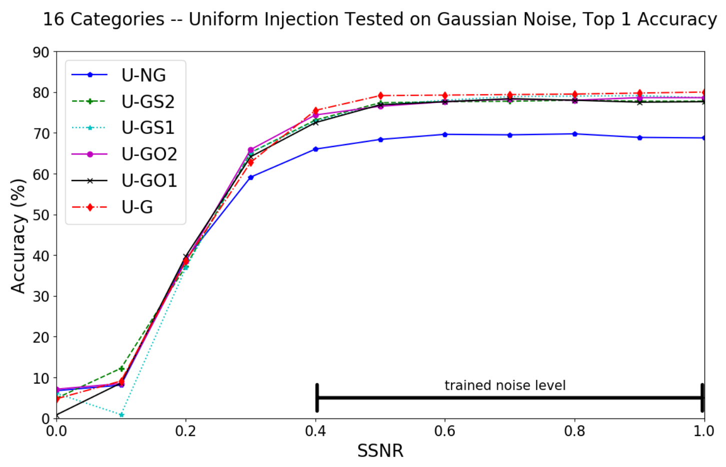


Figure 3.2: Accuracy graph of networks trained on ImageNet-16 dataset with uniform injection, tested on Gaussian noise
Raw values in Appendix Table B.1

3.1.1.3 Binary injection

Intra-noise generalization ability cannot be readily addressed in the previous two cases, it can be better addressed by the binary injection experiment. This is because a network trained on clean images was never exposed to noisy images, and a network trained on images with uniformly drawn noise is overexposed to varying levels of noise. In comparison, the binary injection method exposed the network only to a narrowly-controlled noise space, i.e., the network has more unseen levels of noise. Under the binary injection method, the network with GO2 filters is most successful with an R-value of 7.66. Here, the network without Gabor filters did the worst with an R-value of 7.00. Network comparisons are shown in Figure 3.3.

3.1.1.4 Single-Level injection

Being an extreme case of binary injection, networks trained with a single-level injection method unveiled interesting behaviors. Laid out in Figure 3.4, networks now perform worse on clean images and do best on the noise level for which they were trained on (SSNR 0.4). This is because networks were only exposed to noisy images and not clean images. These

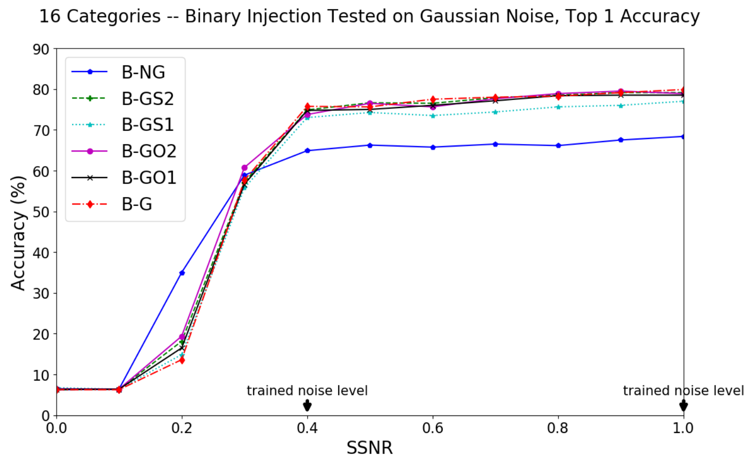


Figure 3.3: Accuracy graph of networks trained on ImageNet-16 dataset with binary injection, tested on Gaussian noise

Raw values in Appendix Table B.1

findings are consistent with those reported by [26]. Also, here we found that a network without front-end Gabor filters could not converge well. We can say that the network learned to classify under the noise from the fact that the accuracy is above chance-level, but the performance still lingers around 12%. The network with Gabor filters was a winner with an R-value of 13.79.

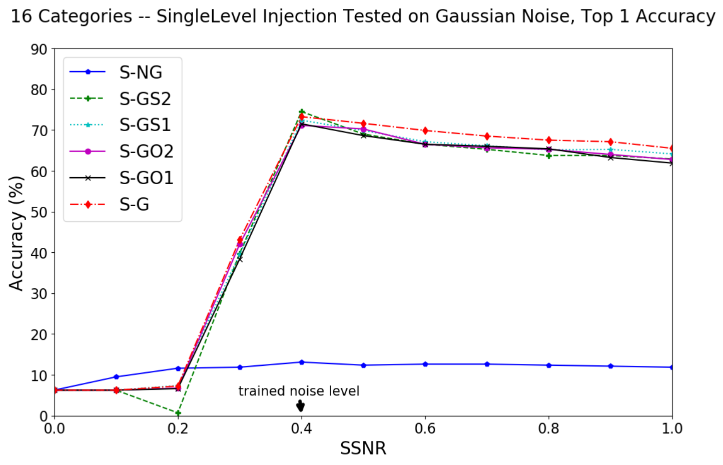


Figure 3.4: Accuracy graph of networks trained on ImageNet-16 dataset with single-level injection, tested on Gaussian noise

Raw values in Appendix Table B.1

3.1.2 Results from ImageNet-1000

In the case of training on ImageNet with 1000 object classes, we found that networks with Gabor filters performed sub-optimally overall, when compared to networks that could freely learn their filters. Although the general trend in performance is almost identical for the two types of networks, the previously seen noise-robustness superiority for binary injection and uniform injection method for the ImageNet-16 study is no longer observed when we train with the full 1000-category dataset.

3.1.2.1 Clean dataset

It is clear that C-NG modestly but consistently network outperforms all other networks in all noise levels according to Figure 3.5. From this experiment, we can conclude that Gabor filters do not benefit CNNs in noise robustness if trained on clean images; that is, if the network is trained on clean images, then randomly initializing the weight values and letting the network learn from clean images will lead to somewhat greater benefit with respect to noise-robustness.

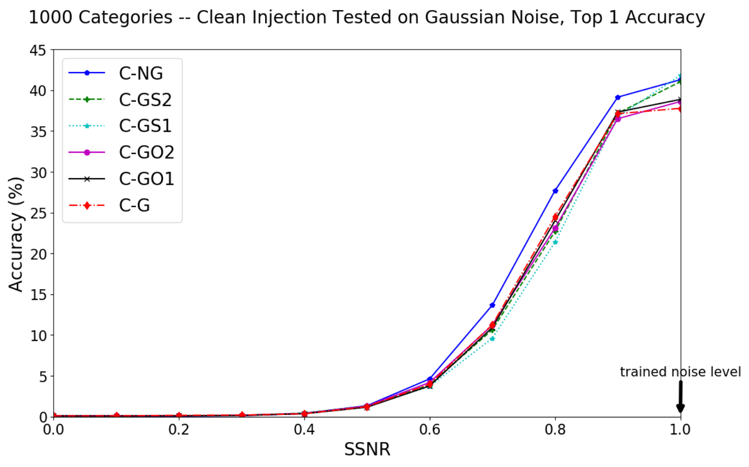


Figure 3.5: Accuracy graph of networks trained on clean ImageNet-1000 dataset, tested on Gaussian noise

Raw values in Appendix Table B.2

3.1.2.2 Uniform injection

We have seen that training the network on a noisy dataset endows the capability to perform on noisy images from the ImageNet-16 case. It does benefit networks in its classification accuracy with respect to noisy images in the ImageNet-1000 case as well. However, unlike the previous contrastive result on networks with Gabor filters performing better on less noisy inputs(SSNR 0.5 - SSNR 1.0), they were only faintly better than the network with freely learned filters. U-NG network has an R-value of 4.8 followed by U-GS1's 4.26, creating a large gap in the noise-robustness.

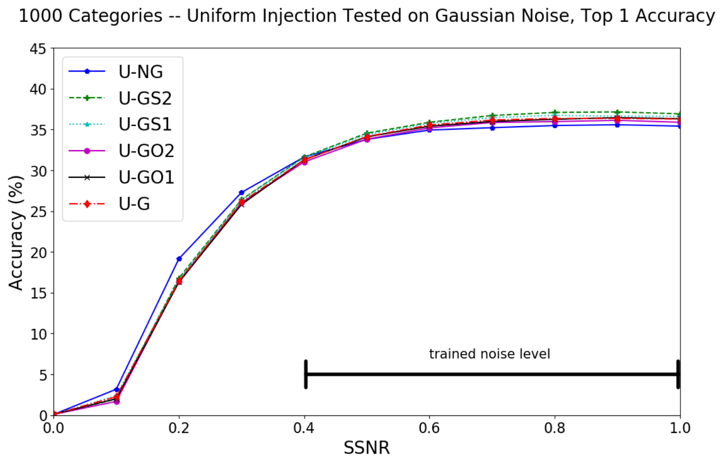


Figure 3.6: Accuracy graph of networks trained on ImageNet-1000 dataset with uniform noise injection, tested on Gaussian noise
Raw values in Appendix Table B.2

3.1.2.3 Binary injection

The network with freely-learned convolutional weights outperformed networks with Gabor filters in the binary injection case as well. Unlike the case with the ImageNet-16 dataset, B-NG network generalizes better than all other networks in unseen levels of noise(0.5 - 0.9). The result is plotted in Figure 3.7. Besides that, an effect that unseen levels of noise have on the network can be observed more clearly with a larger dataset. Compared to ImageNet-16 case where networks did not suffer in the SSNR levels between 0.4 and 1.0, ImageNet-1000 results show that the performance will suffer more as the noise level used during training and the noise level used during testing deviates more. The performance loss is most severe at SSNR 0.7, which is equally distant from two learned noise levels.

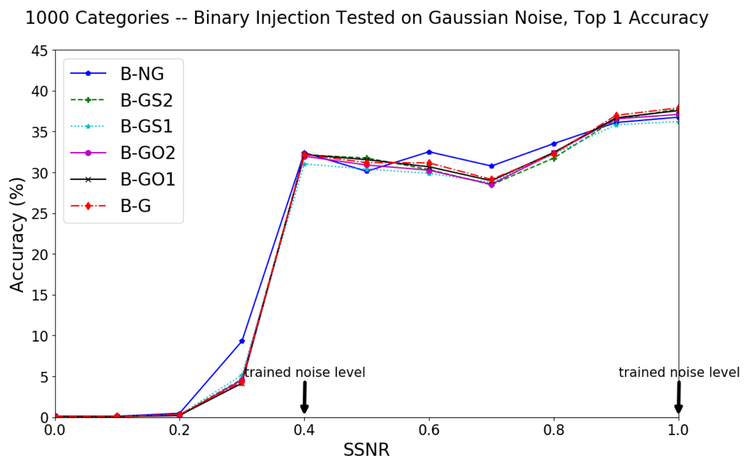


Figure 3.7: Accuracy graph of networks trained on ImageNet-1000 dataset with binary noise injection, tested on Gaussian noise

Raw values in Appendix Table B.2

3.1.2.4 Single-level injection

Like the observation made in ImageNet-16 dataset case, a network without Gabor filters could not converge properly when the dataset does not contain clean or less noisy images. S-NG network ended up classifying about two instances out of 1000 images(0.2% accuracy) at best. This experiment reveals that Gabor filters make it easier for the network to start learning on noisy inputs.

Summarizing from the above results, Gabor filters could not bring about a noticeable increase in the intra-noise generalization capability of CNNs. Also, letting convolutional weights learn freely starting from a random initialization proved to be more powerful as the number of classes in the dataset grows. However, Gabor filters were beneficial at boosting the initial learning of the network, and this benefit was most prevalent in the single-noise injection experiments.

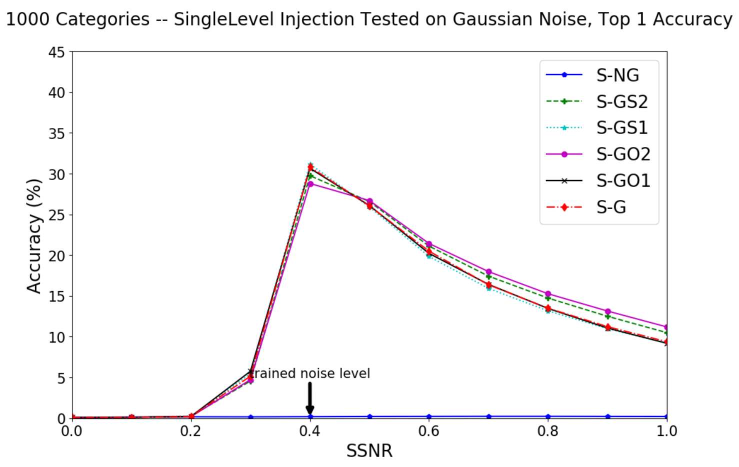


Figure 3.8: Accuracy graph of networks trained on ImageNet-1000 dataset with single-level noise injection, tested on Gaussian noise
Raw values in Appendix Table B.2

3.1.3 Inter-Noise Generalization of All Networks

Networks trained on Gaussian noise with varying methods were tested on Salt-and-Pepper noise to investigate whether they can generalize well to a different types of noise. A recent study showed that Salt-and-Pepper noise is notoriously difficult to generalize to – when the network is trained on a different type of noise [25]. We can observe that same difficulty in this experiment, and it is difficult for all networks. Moreover, on top of being difficult, testing on a different type of noise changes the accuracy by SSNR profiles completely. For example, a response profile for the ImageNet-1000 binary injection case looks like an inverted arch between SSNR 0.4 and SSNR 1.0. This accuracy drop originated from the fact that the network has never seen those noise levels during training. On the contrary, the accuracy does not differ between the unseen noise levels and seen noise levels when the same network is tested on an image mixed with Salt-and-Pepper noise. Rather, the accuracy consistently rises as the noise level decreases. Since there is no meaning in explicitly studying the shape of the accuracy curve apart from the extreme performance loss, results are shown as a heatmap in Figure 3.9.

As expected, all networks suffered from poor performance when tested with Salt-and-

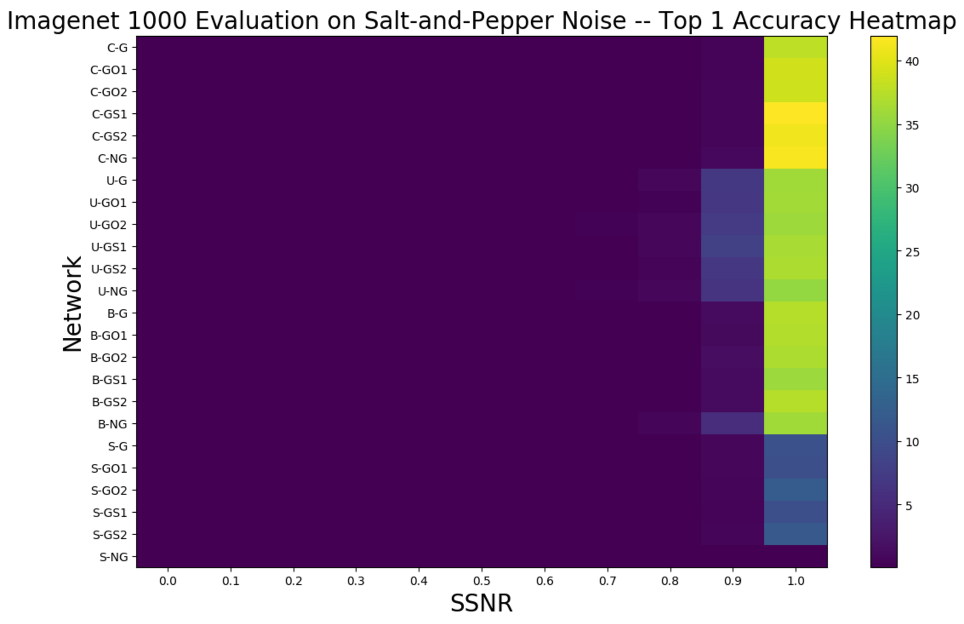


Figure 3.9: Accuracy graph of networks trained on ImageNet-1000 dataset, tested on Salt-and-Pepper noise

Raw values in Appendix Table B.4

Pepper noise. However, the network without Gabor filters achieves better accuracy than others. This indicates that all NG networks, regardless of the noise injection method, has the best inter-noise generalization capability.

3.1.4 An Effect of Orthogonality in Classification Performance

To evaluate the hypothesis that filters that are orthogonal(strokes' orientations are more perpendicular to one another) will do better than the ones that are not, we compared five networks; G, GO1, GO2, GS1, GS2; to see which one had the highest R-values throughout all experiments. To recap, filters G, GO1, and GO2 are orthogonal and filters GS1 and GS2 are skewed. The best performing networks for each dataset are summarized in Table 3.1. Also, their full accuracy results are shown in Table B.4.

On the intra-noise generalization aspect, observations say that orthogonal filters perform better than skewed filters. However, the difference gap is almost negligible in most cases, suggesting that the outcome can be inverted in case of a slight change in the envi-

		Clean	Uniform	Binary	Single-Level
Gaussian Noise	ImageNet-16	GO2	GO2	GO2	G
	ImageNet-1000	G	GS1	G	GO2
Salt-and-Pepper	ImageNet-16	GS2	GS1	GS2	GS2
	ImageNet-1000	GO2	NG	-	GS2

Table 3.1: Gabor filter networks compared to investigate the effect of orthogonality

ronment. The lack of trend also holds for inter-noise generalization case. Thus, we could not find strong evidence of orthogonality benefiting the network's performance. On the other hand, this lack of contrast might be due to a small manipulation of the orientation differences(5 degrees) and needs to be further studied in the future for contrastive evidence before making any conclusion.

3.2 Qualitative Assessment

As opposed to quantitative results, qualitative aspects of different networks might reveal some intuition for future studies on the subject and give rise to new fresh views of the results. Here, we discuss inspection of the learned filters for freely-learned networks(NG networks). Also, features maps(or activation maps) from the initial convolutional layers of different networks will be presented.

3.2.1 Learned Convolutional Filters

A trained AlexNet typically has some filters that resemble Gabor filters. That is, some of these filters are comprised of what can be interpreted as strokes with different frequencies and orientations. Figure 3.10 shows such filters visualized as an image.

Some of the filters in the learned weights in this study also resembled Gabor filters, but they differed with respect to the noisiness of the learned dataset. Although we do not provide a concrete way to calculate their trend, convolutional weights learned from somewhat noisier datasets tend to be more chaotic(more speckles and dot-like filters). In

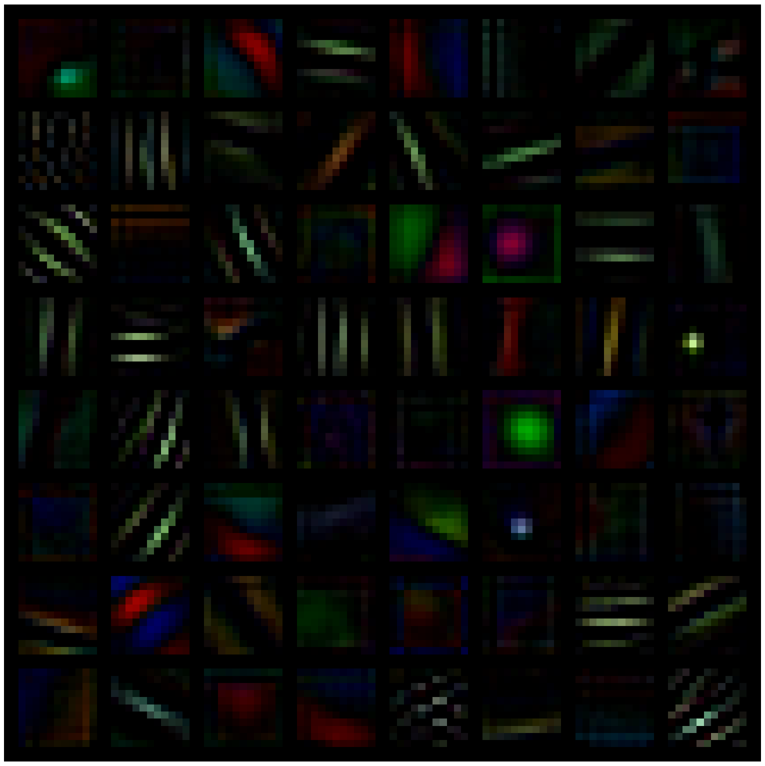


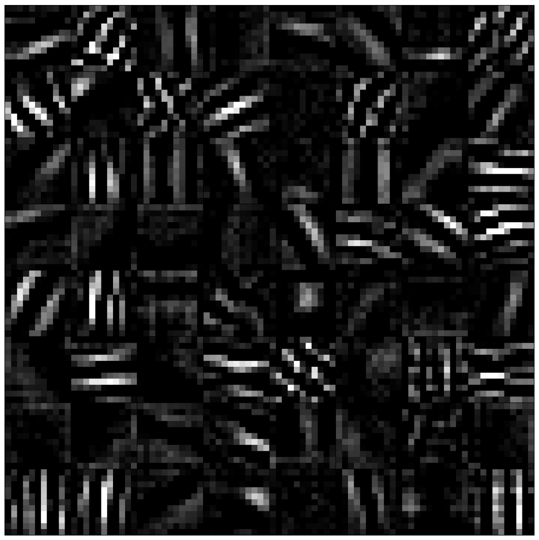
Figure 3.10: Weights of the initial convolutional layer extracted from pytorch pre-trained AlexNet [52].

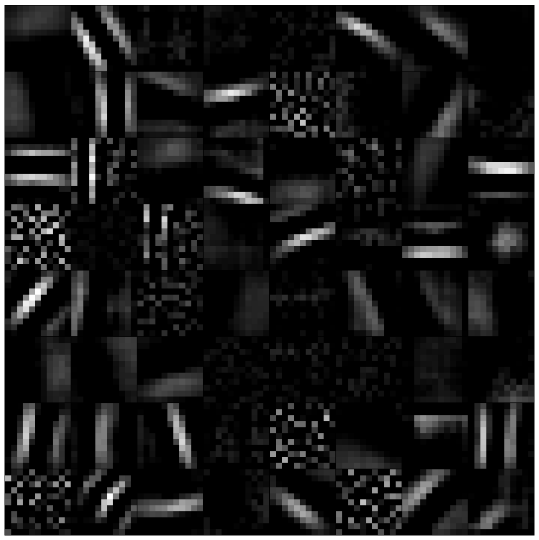
contrast, networks that were trained on less noisy datasets tend to have more Gabor-like filters. The severity of noise increases moving from top left to bottom right in Figure 3.11, and speckles increase along the way.

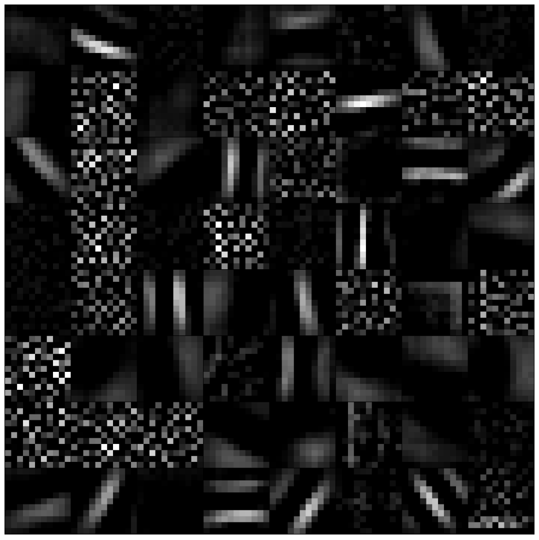
There is no concrete way of addressing this difference in learned convolutional weights, but one suspicion is that the difficulty of the dataset(noisier) makes it hard for the weights to converge to an optimal point. On the other hand, one contrary suspicion could be that these speckles act as a feature extractor for pixel noise.

3.2.2 Feature Maps

Feature maps, or activation maps, are the information passed down to the next layer in the feature extractor. We only demonstrate few feature maps using a single image as it becomes increasingly hard to visualize feature maps in the later layers due to the small width


(a) Trained on a clean dataset


(b) Trained with a binary injection


(c) Trained with a uniform injection

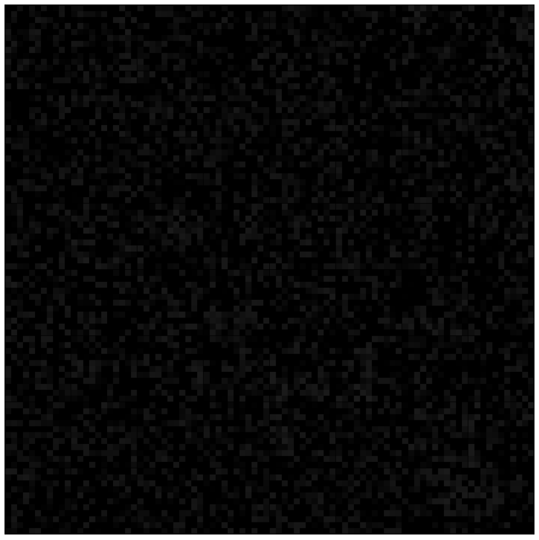
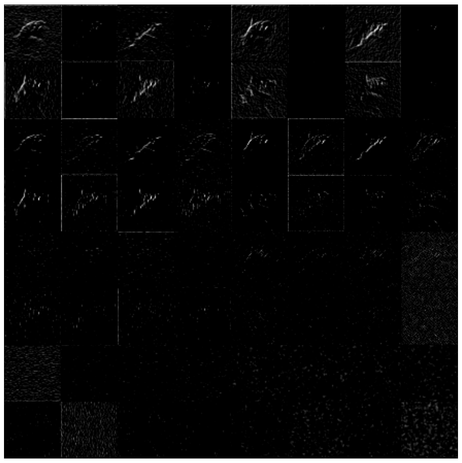

(d) Trained with a single-level injection

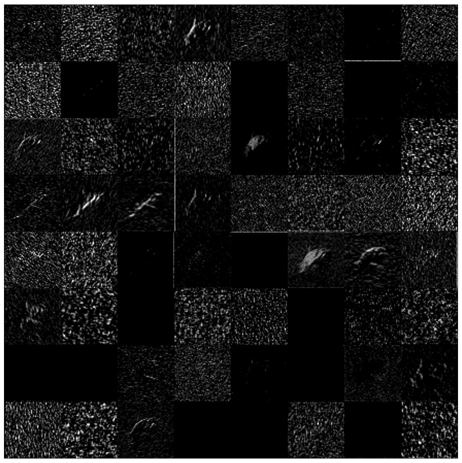
Figure 3.11: Convolutional weights learned from three different noise-injection methods and a clean dataset

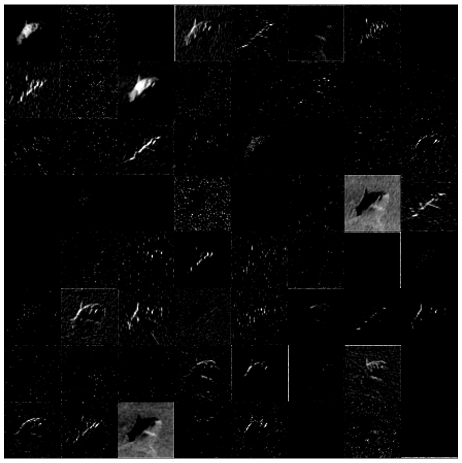
In (d), it is almost impossible to visually identify the filters due to lack of convergence from random initialization and height. Feature maps can vary from images to images and they are highly qualitative, so it is not our attempt to reveal any concrete phenomenon, or a definitive findings from them.

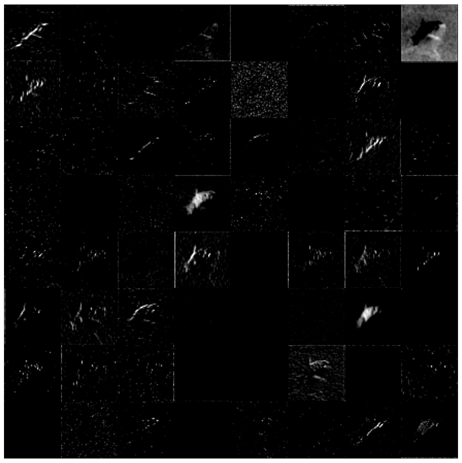
Figure 3.12 shows feature maps obtained from freely learned filters, along with an original image containing a shark. An input image was synthesized with SSNR 0.5 Gaussian noise. It can be seen that networks trained on a noisy dataset (3.12b, 3.12d, 3.12f) output more features such as the outline of the shark. On the other hand, a network that was not trained on noisy images (3.12c) tends to show a static-noise-like response, suggesting that the information that can be used for further feature extraction will be sub-optimal. Finally, a network that did not converge properly (3.12f) passed down features that are almost like the original input with noise. They may be more recognizable for human eyes, but the poor accuracy from the network suggests that they are not the kind of feature that can be used in the future convolutional layers.

(a) Original grayscale image


(b) Feature map from a Gabor network trained with binary injection(B-G)


(c) Feature map from C-NG network


(d) Feature map from U-NG network


(e) Feature map from B-NG network

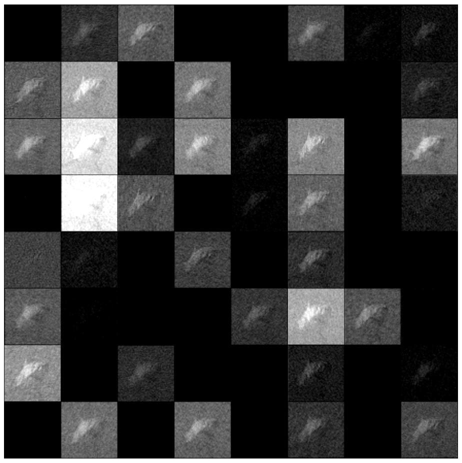

(f) Feature map from S-NG network

Figure 3.12: An original image without noise containing a shark (a).

3.2.3 Learning Profile

Finally, we take a look at the learning profiles of two sampled networks, B-NG and BG. A learning profile does not always reveal how well the network will perform in the test settings, but it does provide us with useful insight into the learning power. For example, a network that converges faster does not necessarily have better performance at the end of training than the network that converges more slowly, but the former network certainly has a benefit over the latter one during the training process, because it can be trained faster. As will be shown shortly, a network with Gabor filters converges faster than a network without them. More precisely, when comparing entropy loss values of two networks(one with Gabor filters and one without) at the same stage of training, a loss value of the network with Gabor filters is lower than that of the network without Gabor filters.

Learning curves for a network with original Gabor filters and a network with freely-learned filters, both trained with a binary noise injection method, are shown in Figure 3.13. It is easily identifiable that B-G network converges faster than B-NG network. A similar trend holds for other networks except for when the networks are trained on clean images where non-Gabor network quickly takes over Gabor filter network's convergence rate in the early stage of the learning process.

The performance result previously shown indicates that B-NG network performs better than B-G network, whereas the training validation plot says otherwise. Moving average is partially responsible for this gap in the plot. Since networks with Gabor filters converge at a faster rate, some earlier accuracy gap is retained in the later part of the plot. One other factor that might be contributing to this difference is due to how noise level is chosen for the validation images during the training process. During the training validation, noise levels are sampled from a uniform distribution ranging from SSNR 0.2 and SSNR 1.0. This causes some noise levels in between discrete SSNR levels(e.g., 0.2, 0.3, 0.4) to be chosen for noise synthesis. Such sampling differs from the noise level sampling for the evaluation. Such observation can lead to another question of whether Gabor filter networks might be

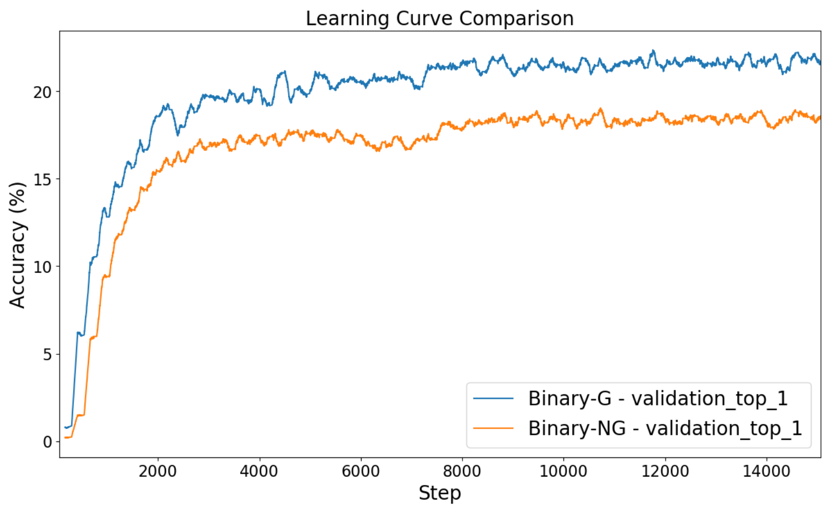


Figure 3.13: Learning curve comparison between B-NG and B-G network
Plotted with data points processed with a 120-window moving average.

generalizing well in the noise levels that were not used in the evaluation process, but the case is not investigated in depth.

Chapter 4

Discussions

Summarizing from the results, Gabor filters could not bring about a noticeable increase in the intra-noise and inter-noise generalization capability of CNNs. More precisely, when we allowed the first layer of the CNN to learn freely, starting from randomly initialized weights, proved to be more powerful as the number of classes in the dataset grows. This could be a weak indicator that flexibility of a feature extractor, i.e., being able to fit the dataset and noise, plays an increasing amount of role in the whole network's aptitude to work robustly on noisy images as the complexity of the task increases. Put differently, freely learned filters are flexible because they are freely-learned, and this freedom leads to a better ability to fit a given dataset.

The experiments partially falsify the hypothesis that Gabor filters, if used in the initial layer, will benefit CNNs in their noise-robustness to grayscale Gaussian and Salt-and-Pepper noise synthesized images. It is partial because Gabor filters actually benefited performance when the number of classes in the dataset were small (ImageNet-16 dataset). Also, we only tested a small subset of Gabor filters and a small subset of CNN architectures and the hyper-parameters, etc. More research would be needed to make a broader claims about the potential utility or lack thereof for using a set of front-end Gabor filters for noise training. Moreover, networks converge faster when with Gabor filters, which opens up a new possibility for their use. In this chapter, we discuss some ideas that could be explored in future research.

4.1 Gabor Filters Benefit Noise-Robustness in a Dataset with Small Number of Categories

In experiments done with ImageNet with 16 categories, networks with Gabor filters showed performance-wise utility in noise-robustness capability. There can be many explanations for this behavior. It could be that performing well on a complex dataset with more categories requires greater flexibility in the set of possible filters. However, in a case where such flexibility is not required, such as hand digit recognition, CNNs with Gabor filters might provide a suitable solution.

4.2 Fixing a Filter Reduces the Network's flexibility

Following the previous point, one interpretation is that using fixed Gabor filters may limit the CNNs ability to flexibly fit the various images in the dataset. It is widely believed that artificial neural networks can be arbitrarily powerful, because they can fit data with an arbitrarily large dimensionality. Despite the fact that the bias term was still learnable, the fixed filter approach is obviously against this adaptability and could be the source of sub-optimal performance.

4.3 Gabor Filters as an Initialization Method

If the flexibility problem is a major setback for the Gabor filters, then it might find its place as an initialization method. During the training process, networks with Gabor filters converged faster than randomly initialized networks. On top of that, the fact that networks with larger filters such as AlexNet or ResNet variants converge to weights with some stripes and blobs raises a suspicion that some of these filters, including the V1 area in our visual pipeline, might be closing in to something similar to Gabor filters. Such observation suggests that if the convolutional weights begin from Gabor filters, it might be easier and quicker for the network to converge to a good set of weights. However, this suggestion is not without any potential problems. The fact that most modern approaches employ smaller kernels would make use of Gabor filters difficult, because effectively maintaining the properties of Gabor filters requires the use of large convolutional kernels. Finally, it is important to note that this approach might be valid only for image-related tasks, excluding its usability in tasks such as Natural Language Processing [53], Genetics [54], or even ultrasonic signals [55]. It is because the original inspiration came from the V1 area that relates to visual processing.

4.4 Towards an Integration of Biological Brain and Artificial System

Finally, our work borrows a meaningful component from the biological brain. Artificial intelligence is an area where not only practical computing but evolution by nature, can contribute to. For instance, a biological concept, predictive coding [56] has the potential to reduce the amount of computing and promote efficient coding for artificial neural networks. The Gabor filters used in our work shares that similar meaning in that it is a product of natural development.

Chapter 5

Conclusion

Evidence found in this study shows that the Gabor filters we used could not always enhance AlexNet's robustness to visual noise. However, training a network on noisy inputs most certainly improved its ability to perform on noisy images when tested. Also, we did not find any compelling evidence that CNNs with Gabor filters that are evenly distributed across orientation space have better performance than irregularly spaced filters, based on our measure of performance accuracy of networks when presented with noisy object images.

Although we found evidence that failed to support our initial hypothesis, that the use of Gabor filters in convolutional neural networks for the image classification task will enhance their noise-robustness, Gabor filters can be used in different ways. Taking advantage of its fast convergence, Gabor filters might be suitable as an alternative to the random initialization of weights in the first convolutional layer. Also, from the observation that Gabor filters could learn from images presented with severe noise, some subsets of filters could be mixed with freely learned filters to ensure that the network can start learning in applications where the training images are extremely noisy. Finally, the rigid nature of the Gabor filters might find its use in applications that use the dataset with smaller complexity but requiring reliable performance.

A collective effort to approximate natural intelligence via observing, dissecting, and analyzing natural intelligence can sometimes take a shortcut that an evolutionary process cannot always come across. Human intelligence makes such shortcuts possible, because we can sometimes look ahead, think and reason about the best way to achieve a goal. On the other hand, evolution takes a slow heuristic approach without predicting far ahead. Likewise, a natural component, when used in an artificial system, can under-perform compared to a component backed up by a rigorous mathematical reasoning. One such example can be the use of Leaky ReLU [57] over ReLU [12], where the former lets small negative activations to leak. The latter technique is closer to how biological neurons would behave, but sometimes give worse performance than the former technique. However, this does not mean that we can ignore biological plausibility, as a biological component can give us a good starting point where we can begin improving from. The work presented in this paper is one such attempt to model the biological brain and integrate it to an artificial system. Although not very successful, we were able to spot some tiny bits of evidence that a new strand of research can stem from.

BIBLIOGRAPHY

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[2] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.

[3] Qizhe Xie et al. “Self-training with Noisy Student improves ImageNet classification”. In: *arXiv preprint arXiv:1911.04252* (2019).

[4] Bruno A Olshausen and David J Field. “Emergence of simple-cell receptive field properties by learning a sparse code for natural images”. In: *Nature* 381.6583 (1996), pp. 607–609.

[5] S Marĉelja. “Mathematical description of the responses of simple cortical cells”. In: *JOSA* 70.11 (1980), pp. 1297–1300.

[6] John G Daugman. “Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters”. In: *JOSA A* 2.7 (1985), pp. 1160–1169.

[7] Judson P Jones and Larry A Palmer. “An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex”. In: *Journal of neurophysiology* 58.6 (1987), pp. 1233–1258.

[8] David H Hubel and Torsten N Wiesel. “Receptive fields of single neurones in the cat’s striate cortex”. In: *The Journal of physiology* 148.3 (1959), pp. 574–591.

[9] Nicolai Petkov and Easwar Subramanian. “Motion detection, noise reduction, texture suppression, and contour enhancement by spatiotemporal Gabor filters with surround inhibition”. In: *Biological Cybernetics* 97.5-6 (2007), pp. 423–439.

[10] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[12] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[13] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[14] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *arXiv preprint arXiv:1607.02533* (2016).

[15] Pedro Tabacof and Eduardo Valle. "Exploring the space of adversarial images". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 426–433.

[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[17] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks". In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 828–841.

[18] Danilo Vasconcellos Vargas and Jiawei Su. "Understanding the one-pixel attack: Propagation maps and locality analysis". In: *arXiv preprint arXiv:1902.02947* (2019).

[19] Volodymyr Mnih and Geoffrey E Hinton. "Learning to label aerial images from noisy data". In: *Proceedings of the 29th International conference on machine learning (ICML-12)*. 2012, pp. 567–574.

[20] Sainbayar Sukhbaatar and Rob Fergus. “Learning from noisy labels with deep neural networks”. In: *arXiv preprint arXiv:1406.2080* 2.3 (2014), p. 4.

[21] Tong Xiao et al. “Learning from massive noisy labeled data for image classification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2691–2699.

[22] Rohit Verma and Jahid Ali. “A comparative study of various types of image noise and efficient noise removal techniques”. In: *International Journal of advanced research in computer science and software engineering* 3.10 (2013).

[23] Kai Zhang et al. “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising”. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.

[24] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[25] Robert Geirhos et al. “Generalisation in humans and deep neural networks”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 7538–7550.

[26] Hojin Jang and Frank Tong. “Deep neural networks can attain human-level robustness at recognizing objects in visual noise”. In: *Poster presented at CVPR*. 2019.

[27] Tiago S Nazaré et al. “Deep convolutional neural networks and noisy images”. In: *Iberoamerican Congress on Pattern Recognition*. Springer. 2017, pp. 416–424.

[28] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[29] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[30] Michał Koziarski and Bogusław Cyganek. "Image recognition with deep neural networks in presence of noise–dealing with and taking advantage of distortions". In: *Integrated Computer-Aided Engineering* 24.4 (2017), pp. 337–349.

[31] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 `[cs.CV]`.

[32] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 `[cs.CV]`.

[33] Gabriel B Paranhos da Costa et al. "An empirical study on the effects of different types of noise in image classification tasks". In: *arXiv preprint arXiv:1609.02781* (2016).

[34] Steven Diamond et al. "Dirty pixels: Optimizing image classification architectures for raw sensor data". In: *arXiv preprint arXiv:1701.06487* (2017).

[35] Sudipta Singha Roy et al. "A robust system for noisy image classification combining denoising autoencoder and convolutional neural network". In: *International Journal of Advanced Computer Science and Applications* 9.1 (2018), pp. 224–235.

[36] Samuel Dodge and Lina Karam. "Understanding how image quality affects deep neural networks". In: *2016 eighth international conference on quality of multimedia experience (QoMEX)*. IEEE. 2016, pp. 1–6.

[37] Yiren Zhou, Sibo Song, and Ngai-Man Cheung. "On classification of distorted images with deep convolutional neural networks". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 1213–1217.

[38] Gaurav Malhotra and Jeffrey Bowers. "What a difference a pixel makes: An empirical examination of features used by CNNs for categorisation". In: (2018).

[39] David H Hubel and Torsten N Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *The Journal of physiology* 160.1 (1962), pp. 106–154.

[40] David Barina. “Gabor wavelets in image processing”. In: *arXiv preprint arXiv:1602.03308* (2016).

[41] Thomas P Weldon, William E Higgins, and Dennis F Dunn. “Efficient Gabor filter design for texture segmentation”. In: *Pattern recognition* 29.12 (1996), pp. 2005–2015.

[42] Rajiv Mehrotra, Kameswara Rao Namuduri, and Nagarajan Ranganathan. “Gabor filter-based edge detection”. In: *Pattern recognition* 25.12 (1992), pp. 1479–1494.

[43] Tai Sing Lee. “Image representation using 2D Gabor wavelets”. In: *IEEE Transactions on pattern analysis and machine intelligence* 18.10 (1996), pp. 959–971.

[44] Volker Krüger and Gerald Sommer. “Gabor wavelet networks for object representation”. In: *Multi-Image Analysis*. Springer, 2001, pp. 115–128.

[45] Haihong Zhang et al. “Gabor wavelet associative memory for face recognition”. In: *IEEE transactions on neural networks* 16.1 (2005), pp. 275–278.

[46] Avinash Kaushal and JPS Raina. “Face detection using neural network & Gabor wavelet transform”. In: *IJCST* 1.1 (2010).

[47] Siddharth Choubey and Deepika Banchhor. “A Literature Survey of various Fingerprint De-noising Techniques to justify the need of a new De-noising model based upon Pixel Component Analysis”. In: *arXiv preprint arXiv:1512.00939* (2015).

[48] Pawan Patidar et al. “Image de-noising by various filters for different noise”. In: *International journal of computer applications* 9.4 (2010), pp. 45–50.

[49] Roman Garnett et al. “A universal noise removal algorithm with an impulse detector”. In: *IEEE Transactions on image processing* 14.11 (2005), pp. 1747–1754.

[50] Yiting Xie and David Richmond. “Pre-training on grayscale imagenet improves medical image classification”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 0–0.

[51] Christopher Kanan and Garrison W Cottrell. “Color-to-grayscale: does the method matter in image recognition?” In: *PloS one* 7.1 (2012).

[52] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[53] Denny Britz. “Understanding convolutional neural networks for NLP”. In: *URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015)* (2015).

[54] Lex Flagel, Yaniv Brandvain, and Daniel R Schrider. “The unreasonable effectiveness of convolutional neural networks in population genetic inference”. In: *Molecular biology and evolution* 36.2 (2019), pp. 220–238.

[55] Zhanwen Chen. “Noise Suppression in Ultrasound Beamforming Using Convolutional Neural Networks”. PhD thesis. Vanderbilt University, 2019.

[56] Mandyam Veerambudi Srinivasan, Simon Barry Laughlin, and Andreas Dubs. “Predictive coding: a fresh view of inhibition in the retina”. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 216.1205 (1982), pp. 427–459.

[57] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

Appendix A

Gabor Filter Full Set

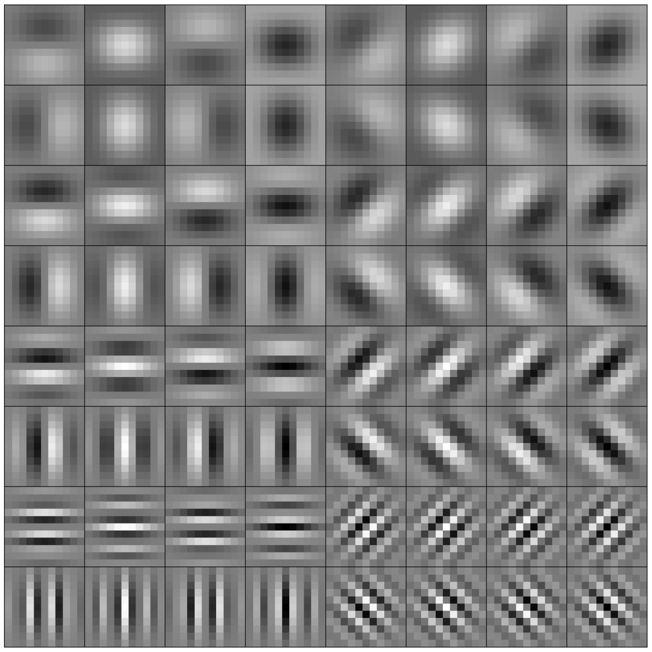


Figure A.1: Original Gabor filter. A set of 64.

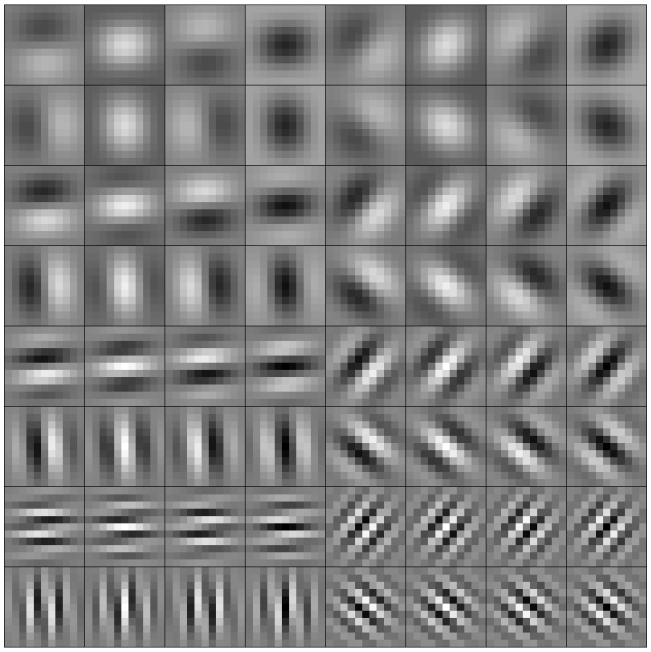


Figure A.2: Orthogonal Gabor filter 1. A set of 64.

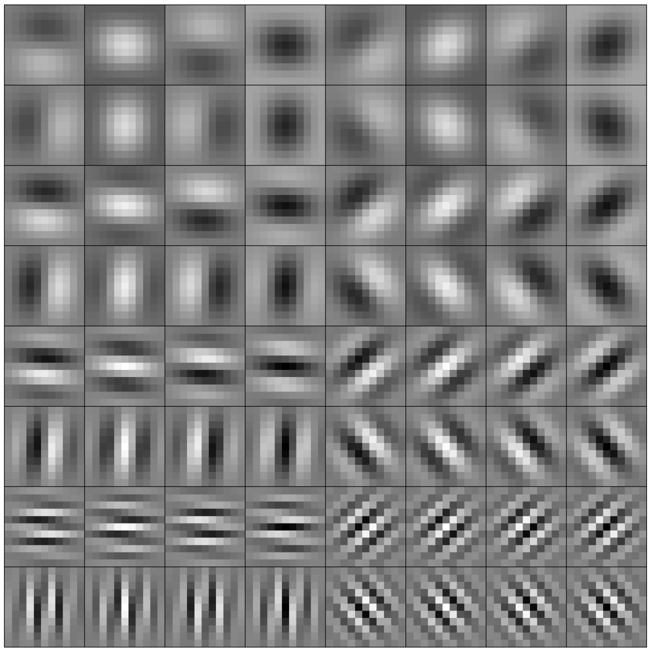


Figure A.3: Orthogonal Gabor filter 2. A set of 64.

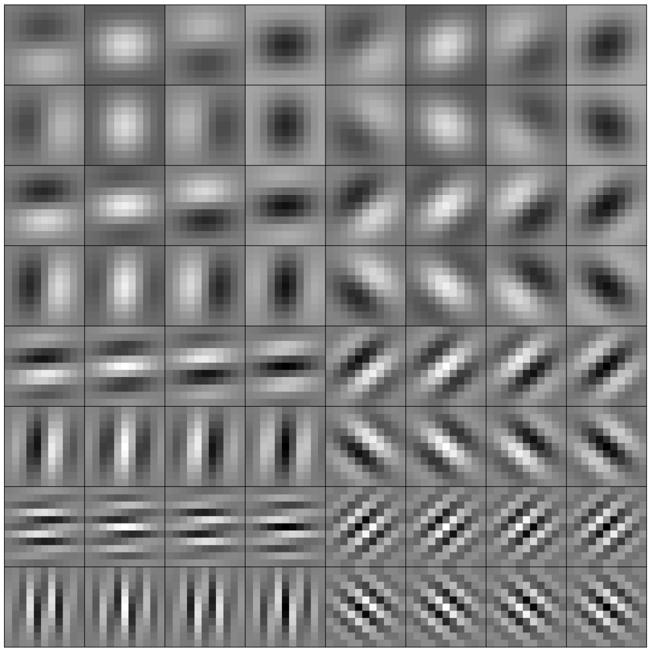


Figure A.4: Skewed Gabor filter 1. A set of 64.

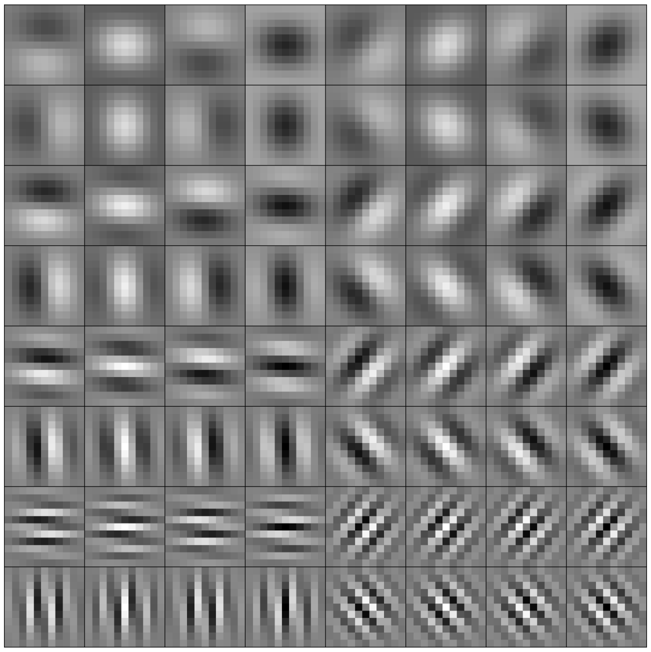


Figure A.5: Skewed Gabor filter 2. A set of 64.

Appendix B

Accuracy Raw Values

SSNR Network	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	R-value
C-NG	6.25	6.25	6.25	7.25	14.25	34.62	53.25	64.62	70.12	72.75	72.25	9.98
C-GS2	6.38	0.60	6.62	8.75	11.62	21.50	39.50	58.25	73.75	79.50	80.62	8.37
C-GS1	6.25	6.25	6.25	6.50	10.50	23.88	41.75	61.38	73.62	78.25	78.88	8.85
C-GO2	6.38	6.38	7.75	7.88	12.88	25.00	44.12	61.75	75.75	80.75	81.25	9.40
C-GO1	6.25	6.25	6.25	6.25	8.25	19.75	35.12	52.75	69.88	78.88	80.00	7.98
C-G	6.25	6.25	6.25	6.25	7.75	17.12	37.00	57.88	73.88	78.50	80.38	8.11
U-NG	6.75	8.12	39.00	59.12	66.00	68.38	69.62	69.50	69.75	68.88	68.75	12.00
U-GS2	4.88	12.25	37.25	65.12	73.12	77.38	77.62	77.75	78.00	77.75	77.75	12.46
U-GS1	6.12	0.90	36.88	65.12	72.88	76.88	78.00	78.88	79.00	79.12	78.62	10.57
U-GO2	7.12	8.50	38.50	65.88	74.38	76.50	77.62	78.25	78.00	78.62	78.62	12.53
U-GO1	0.80	8.62	39.75	64.12	72.50	76.88	77.62	78.38	78.00	77.50	77.62	10.99
U-G	4.75	9.12	38.62	62.75	75.50	79.12	79.25	79.38	79.50	79.75	80.00	11.86
B-NG	6.25	6.38	35.00	58.88	64.88	66.25	65.75	66.50	66.12	67.50	68.38	14.00
B-GS2	6.25	6.38	18.12	57.12	74.88	76.62	76.50	77.88	78.50	79.12	79.12	15.22
B-GS1	6.75	6.38	14.88	55.75	73.00	74.25	73.50	74.38	75.62	76.00	77.00	14.60
B-GO2	6.50	6.38	19.38	60.75	73.75	76.50	75.62	77.62	78.88	79.50	78.88	15.32
B-GO1	6.38	6.38	16.50	56.62	74.75	75.00	76.00	77.12	78.38	78.50	78.50	15.05
B-G	6.25	6.25	13.62	57.62	75.75	75.62	77.50	78.00	78.25	79.12	79.88	15.07
S-NG	6.25	9.50	11.62	11.88	13.12	12.38	12.62	12.62	12.38	12.12	11.88	3.14
S-GS2	6.25	6.25	0.70	39.75	74.50	69.00	66.50	65.25	63.75	63.75	62.88	13.03
S-GS1	6.25	6.25	7.38	39.50	72.38	69.88	67.12	66.12	65.12	65.25	64.12	13.38
S-GO2	6.25	6.25	6.75	42.00	71.12	70.25	66.50	65.62	65.25	64.00	62.75	13.24
S-GO1	6.25	6.25	6.62	38.38	71.50	68.62	66.50	66.00	65.38	63.25	61.88	13.13
S-G	6.25	6.25	7.25	43.12	73.25	71.62	69.88	68.50	67.50	67.12	65.50	13.79

Table B.1: Accuracy table of networks trained on ImageNet-16, tested on Gaussian noise

SSNR Network	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	R-value
C-NG	0.10	0.11	0.07	0.14	0.40	1.35	4.62	13.66	27.70	39.14	41.28	1.52
C-GS2	0.10	0.11	0.12	0.17	0.43	1.33	3.91	10.64	22.70	37.17	41.03	1.30
C-GS1	0.10	0.10	0.09	0.14	0.45	1.26	3.71	9.61	21.38	36.75	41.81	1.23
C-GO2	0.10	0.10	0.10	0.17	0.39	1.30	4.19	11.21	23.11	36.50	38.61	1.33
C-GO1	0.10	0.09	0.11	0.14	0.36	1.16	3.77	10.91	24.00	37.32	38.86	1.32
C-G	0.10	0.10	0.16	0.19	0.36	1.23	3.99	11.25	24.53	37.12	37.77	1.35
U-NG	0.09	3.17	19.15	27.28	31.63	33.78	34.92	35.22	35.48	35.58	35.41	4.80
U-GS2	0.11	1.98	16.78	26.40	31.63	34.55	35.88	36.71	37.08	37.15	36.91	4.22
U-GS1	0.09	2.37	16.56	26.36	31.58	34.37	35.74	36.45	36.73	36.62	36.61	4.26
U-GO2	0.10	1.68	16.40	26.02	31.02	33.81	35.19	35.85	35.96	36.12	35.88	4.09
U-GO1	0.11	2.02	16.28	25.81	31.29	34.09	35.40	35.98	36.24	36.45	36.31	4.13
U-G	0.10	2.26	16.42	26.09	31.28	34.13	35.54	36.15	36.35	36.35	36.28	4.21
B-NG	0.10	0.12	0.47	9.31	32.40	30.12	32.51	30.77	33.51	36.09	36.75	5.48
B-GS2	0.10	0.11	0.26	4.65	32.14	31.74	30.34	28.46	31.74	36.49	37.74	5.16
B-GS1	0.09	0.10	0.28	5.17	31.01	30.39	29.87	28.75	32.24	35.81	36.22	5.15
B-GO2	0.10	0.10	0.19	4.56	31.96	30.89	30.25	28.53	32.37	36.54	37.08	5.16
B-GO1	0.11	0.07	0.22	4.16	32.18	31.54	30.67	28.97	32.42	36.67	37.56	5.21
B-G	0.10	0.10	0.32	4.34	32.18	31.18	31.15	29.12	32.27	36.97	37.89	5.24
S-NG	0.10	0.15	0.18	0.17	0.19	0.21	0.22	0.23	0.23	0.22	0.21	0.05
S-GS2	0.10	0.11	0.24	4.56	29.74	26.57	21.10	17.42	14.73	12.50	10.48	2.83
S-GS1	0.10	0.12	0.26	5.46	31.16	25.91	19.84	15.92	13.16	10.99	9.19	2.57
S-GO2	0.09	0.12	0.19	4.73	28.79	26.69	21.43	17.98	15.26	13.14	11.19	2.94
S-GO1	0.10	0.11	0.21	5.76	30.65	26.07	20.21	16.41	13.46	11.05	9.18	2.61
S-G	0.10	0.09	0.22	5.16	30.81	26.03	20.49	16.37	13.49	11.22	9.38	2.62

Table B.2: Accuracy table of networks trained on ImageNet-1000, tested on Gaussian noise

SSNR Network	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	R-value
C-NG	6.25	6.25	6.25	6.25	6.25	6.25	6.50	9.50	17.88	38.25	73.25	3.73
C-GS2	5.88	6.88	6.88	5.88	7.12	7.12	7.38	8.12	10.25	21.00	81.00	3.55
C-GS1	6.38	6.12	6.50	6.50	6.75	6.38	6.38	7.62	1.00	21.38	79.12	3.28
C-GO2	5.75	6.12	5.62	0.60	6.75	6.75	6.50	7.50	9.88	23.88	81.38	2.98
C-GO1	6.25	6.25	6.25	6.25	6.25	6.25	6.38	6.38	8.12	20.12	80.00	3.29
C-G	6.25	6.62	6.12	6.50	6.25	6.38	4.75	5.25	7.88	19.12	80.38	3.23
U-NG	0.60	0.70	6.50	7.38	7.38	12.50	18.12	25.50	41.00	54.62	68.88	1.58
U-GS2	6.25	5.50	6.38	0.60	8.38	8.75	12.12	17.50	27.38	52.62	78.25	3.49
U-GS1	6.12	6.25	5.62	6.62	6.62	10.12	16.12	25.75	36.50	53.62	79.00	3.89
U-GO2	0.60	6.12	6.88	7.50	6.38	7.62	11.00	20.25	34.12	54.12	78.88	2.67
U-GO1	0.60	6.25	6.50	6.75	7.88	11.25	14.50	19.12	27.12	46.38	77.75	2.60
U-G	7.75	7.75	5.38	6.88	8.38	11.00	15.12	22.25	30.25	55.12	80.25	4.57
B-NG	6.50	6.88	7.75	0.80	9.12	14.12	21.50	31.75	45.00	56.38	68.00	6.58
B-GS2	6.50	6.62	6.50	6.38	6.50	7.25	11.25	16.38	25.12	41.62	78.88	4.29
B-GS1	6.25	6.25	6.25	6.25	6.50	6.62	8.25	11.25	20.75	37.50	75.62	3.61
B-GO2	6.12	6.25	6.25	6.50	6.25	8.12	10.75	15.75	25.88	43.25	79.12	4.26
B-GO1	6.25	7.12	6.38	5.88	7.12	8.88	9.88	13.75	20.75	40.25	78.62	3.94
B-G	6.25	6.25	6.25	6.25	6.25	6.38	7.50	13.62	25.50	44.75	79.38	4.01
S-NG	0.60	7.75	9.62	10.88	12.38	13.00	13.00	13.75	13.38	12.62	12.00	2.95
S-GS2	6.25	6.12	6.38	6.25	6.38	6.50	7.62	12.38	22.50	39.25	63.00	7.14
S-GS1	6.25	6.25	6.25	6.25	6.25	6.25	7.62	11.00	17.12	38.00	64.00	6.90
S-GO2	6.25	6.25	6.25	6.25	6.25	6.25	6.50	8.25	18.88	39.62	63.25	6.91
S-GO1	6.25	6.25	6.25	6.25	6.50	6.50	7.38	10.88	21.88	37.38	62.50	6.96
S-G	6.25	6.25	6.25	6.25	6.25	6.38	6.50	7.88	17.00	39.50	65.75	6.96

Table B.3: Accuracy table of networks trained on ImageNet-16, tested on Salt-and-Pepper noise

SSNR Network	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	R-value
C-NG	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.15	0.97	41.55	0.06
C-GS2	0.10	0.10	0.10	0.08	0.11	0.15	0.12	0.15	0.17	0.59	41.10	0.06
C-GS1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.14	0.63	41.97	0.06
C-GO2	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.13	0.62	38.71	0.06
C-GO1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.12	0.42	38.97	0.05
C-G	0.10	0.08	0.11	0.10	0.12	0.10	0.12	0.15	0.15	0.54	37.83	0.06
U-NG	0.10	0.10	0.12	0.14	0.17	0.17	0.21	0.29	0.86	6.44	35.13	0.07
U-GS2	0.12	0.10	0.11	0.10	0.09	0.09	0.09	0.14	0.43	6.64	36.76	0.07
U-GS1	0.09	0.10	0.10	0.11	0.08	0.10	0.12	0.16	0.88	8.20	36.45	0.06
U-GO2	0.10	0.10	0.10	0.11	0.10	0.11	0.18	0.27	0.90	7.35	35.89	0.06
U-GO1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.12	0.40	6.66	36.10	0.06
U-G	0.09	0.09	0.11	0.11	0.11	0.11	0.11	0.22	0.83	6.84	36.06	0.06
B-NG	0.10	0.10	0.10	0.10	0.10	0.10	0.11	0.17	0.62	5.37	35.92	0.15
B-GS2	0.09	0.13	0.12	0.11	0.13	0.15	0.14	0.16	0.23	1.35	37.25	0.07
B-GS1	0.10	0.10	0.10	0.12	0.12	0.13	0.15	0.16	0.19	1.24	35.70	0.06
B-GO2	0.12	0.09	0.10	0.11	0.12	0.12	0.13	0.13	0.16	1.42	36.81	0.06
B-GO1	0.09	0.10	0.10	0.10	0.10	0.12	0.13	0.14	0.22	1.06	37.16	0.06
B-G	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.11	0.16	1.38	37.28	0.06
S-NG	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.11	0.15	0.22	0.04
S-GS2	0.12	0.10	0.10	0.10	0.10	0.10	0.10	0.13	0.17	0.59	11.54	0.68
S-GS1	0.10	0.10	0.10	0.10	0.11	0.10	0.11	0.11	0.14	0.55	10.15	0.60
S-GO2	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.13	0.62	12.13	0.71
S-GO1	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.11	0.15	0.77	10.18	0.61
S-G	0.10	0.10	0.10	0.10	0.10	0.10	0.11	0.14	0.13	0.77	10.40	0.62

Table B.4: Accuracy table of networks trained on ImageNet-1000, tested on Salt-and-Pepper noise

Most of the accuracies are at chance level(0.1), due to lack of inter-noise generalization capability. The capability is higher when trained on clean images, and it could be an indicator that training on a certain type of noise makes the network fit too much to that exact type of noise – depriving of its ability to generalize to different types of noise.