ABSTRACT

On The Performance of Convolutional Neural Networks Initialized with Gabor Filters

Mehang Rai, M.S.

Mentor: Pablo Rivas, Ph.D.

Over the years, image recognition has been gaining popularity due to its various possible usages. Convolutional Neural Networks (CNNs) have been the classic approach taken on by many researchers because of their capability to learn through the parameter space given a sufficient amount of representative data. When observing a fully trained CNN, researchers have found that the pattern on the kernel filters (convolution window) of the receptive convolutional layer closely resembles the Gabor filters. Gabor filters have existed for a long time, and researchers have been using them for texture analysis. Given the nature and purpose of the receptive layer of CNN, Gabor filters could act as a suitable replacement strategy for the randomly initialized kernels of the receptive layer in CNN, which could potentially boost the performance without any regard to the nature of the dataset. The findings in this thesis show that when low-level kernel filters are initialized with Gabor filters, there is a boost in accuracy, Area Under ROC (Receiver Operating Characteristic) Curve (AUC), minimum loss, and speed in some cases based on the complexity of the dataset.

On The Performance of Convolutional Neural Networks Initialized with Gabor Filters

by

Mehang Rai, B.S.

A Thesis

Approved by the Department of Computer Science

_____

Erich Baker, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science

Approved by the Thesis Committee

_____

Pablo Rivas, Ph.D., Chairperson

_____

Greg Hamerly, Ph.D.

_____

Liang Dong, Ph.D.

Accepted by the Graduate School
August 2021

_____

J. Larry Lyon, Ph.D., Dean

Page bearing signatures is kept on file in the Graduate School.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my advisor, Dr. Pablo Rivas for mentoring me during the whole research and writing phase. He has blessed me with his wise words, encouragement and support and it has been a wonderful experience to work under him. I would also like to thank my defense committee members, Dr. Greg Hamerly and Dr. Liang Dong. I am extremely grateful towards the Department of Computer Science at Baylor University for providing me with such an incredible opportunity over the span of two years. At last, I want to thank my precious family and friends who where there during my high and low moments. To be honest, it was a huge struggle, especially being far from home, but their moral support always kept my smile and head held high. I will forever be grateful to you all. Thank you everyone.

Dedicated to my family for their sacrifice, unconditional love and support

CHAPTER ONE

Introduction

This thesis explores the impact of Gabor filters on the performance of the Convolutional Neural Network (CNN) when initialized on the receptive layer filters. Image/Object recognition has been a widely popular field for ages now, and with the emerging new technologies, it has begun to pique more interest. This is due to its usage in various fields like Augmented Reality (AR), biometric system, automated driving vehicles, etc. to which a lot of industries have been trying to push forward. When talking about image recognition nowadays it has almost become synonymous with CNN because research has shown that CNN can achieve remarkable performance. Over the years, much research has been conducted, seeking to enhance the performance of CNN concerning various topics, but only a few studies have been done to enhance the fundamental core of CNN. CNN is a supervised learning model, which means that given a training dataset it will begin to correct itself provided that it made a mistake when labelling that image incorrectly. In regard to that, how the CNN is initialized has been seen to be crucial to its performance because there can exist problems like vanishing gradient, local optima, etc. And, although different variants of CNN has evolved over the years to address these issues, only a few studies have been conducted over the improvement in initialization method because most of the time people tend to go for random initialization.

Years before the emerging popularity of CNN, there were different techniques for image processing and Gabor filter has been seen to be one of the popular techniques. Gabor filter is a band pass filter, which can be thought of as Gaussian modulated by a sinusoid. Depending upon the different hyper-parameters like orientation, frequency, etc., Gabor filter when convolved with image is capable of extracting

information regarding texture segmentation. When thinking in CNN, this feature extraction is mainly the job of receptive/first low-level convolutional layer. Since all the deeper convolutional layers build upon the features from preceding layers, it is integral for the receptive layer to perform well. But, as mentioned before in most of the cases these layers are initialized randomly and it takes significant effort for the model to align these layers in correct fashion with back propagation algorithm.

When AlexNet (Krizhevsky, Sutskever, and Hinton 2012) famously won the 2012 ImageNet Large Scale Visual Recognition Challenge (LSVRC)-2012 competition by a large margin, it revolutionized CNN and more importantly, upon close analysis it gave the idea that the kernel filters (convolution window) in the receptive layer of CNN tend to resemble the shape of Gabor filters. While being a significant fact, few studies have been conducted to explore the intersection of these two computer vision tools, and the studies which do have only explored a small sample space or imposed some form of restriction on the structure of the Gabor filters. While imposing restriction on the structure of the Gabor filter may certainly be good in some form of sample space, it definitely adds some complexity. It is to be noted that though CNN is an excellent learner and provides the complexity of general object image recognition task in hand, it is certainly not a good idea to impose such restriction. Since CNN corrects itself when it has made a mistake, it can be hypothesized that when there is a need, CNN will itself correct the Gabor filter as necessary. This particular configuration in CNN can boost the performance of CNN because it could be significantly more efficient than random initialization. Since the Gabor filter has been shown by past studies to work well in different kind of image processing, it can be a good idea to give a thought to Gabor filter as an initialization method for the low-level kernel filters in the receptive convolutional layer to make CNN better in terms of general object recognition.

While previous studies have emphasized on initialization of multiple convolutional layers with Gabor filters, we are focusing on the initialization of the receptive convolutional layer with the Gabor filter because only the receptive filters resemble the nature of Gabor filters, and the succeeding deeper filters tend to become more abstract as shown by previous research (Krizhevsky, Sutskever, and Hinton 2012). Also, any form of restriction on the Gabor filter structure has been removed, because it helps the CNN to unlearn the existing filter on the receptive layer, if necessary. This technique provides the freedom to the CNN to rectify and change the Gabor filter structure as per necessity, which eventually allows to extract complex features from which the succeeding convolutional layer could build upon. This can lead to boost in performance of CNN like in terms of accuracy, AUC, etc. Even if it is a small improvement in accuracy of general object recognition, it can make significant impact. For example, even if the automated ZIP Code recognition is improved by 1%, it could still prevent millions of packages from being delivered to wrong place.

The remainder of the thesis is organized as follows: Chapter Two gives the literature review of the CNN and Gabor filter. Mehang Rai wrote the entire paper under the supervision from Dr. Pablo Rivas. Chapter Three provides the methodology of the research with an explanation of each component within the architecture. Chapter Four presents the result and gives analysis of all the research conducted. Chapter Five summarizes the findings of the research and provides insight for future work.

CHAPTER TWO

Literature Review on CNNs and Gabor Filters

This chapter published as Rai and Rivas (2020). A Review of Convolutional Neural Networks and Gabor Filters in Object Recognition. In 2020 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 1560–1567.

2.1 Gabor Filter

A Gabor filter, derived from Gabor elementary functions (GEF), is a linear filter used for a multitude of image processing applications for texture analysis, edge detection, feature extraction, etc. As a band-pass filter, the Gabor filter enables the extraction of patterns at the specified certain frequency and orientation of the signal. Therefore this resulting property of transforming texture differences into detectable filter-output discontinuities at texture boundary has established itself to mimic the functionality of the visual cortex (Dunn, Higgins, and Wakeley 1994; Dunn and Higgins 1995; Jain, Ratha, and Lakshmanan 1997). While the concept of Gabor elementary function was initially presented by Hungarian-British physicist Dennis Gabor (Gabor 1946), it was later extended to 2-D filters by Daugman (1985).

In basic terms, a Gabor elementary function (GEF) can be thought of as a Gaussian being modulated by a complex sinusoid, where the Cosine and Sine waves generate the real and imaginary component. GEF can be formulated as:

$$g(x, y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right) \tag{2.1}$$

where $(x', y') = (x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta)$ represents rotated spatial-domain rectilinear coordinates; λ represents the wavelength of the sinusoidal factor, θ represents the orientation of the normal to the parallel stripes of a Gabor function, ψ is the offset, σ_x and σ_y characterize the spatial extent and bandwidth of the filter.

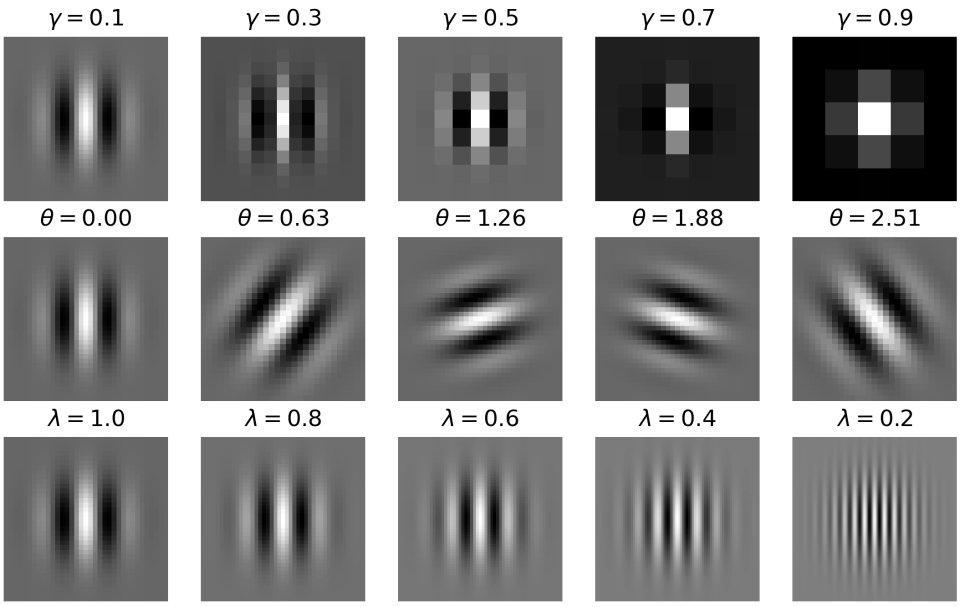


Figure 2.1. Different Gabor filters with different values for λ, θ and γ. Different parameters will change filter properties.

Most of the time a symmetric filter ($\sigma_x = \sigma_y$) will suffice for texture segmentation, but when the texture contains texels not arranged in square lattice, asymmetric filters ($\sigma_x \neq \sigma_y$) could be useful (Dunn, Higgins, and Wakeley 1994). This asymmetric nature is given by $\gamma \neq 1$, where γ is the spatial aspect ratio:

$$\gamma = \frac{\sigma_x}{\sigma_y}. \tag{2.2}$$

Since a single Gabor filter can only be responsible for a certain feature, a multitude of Gabor filters is necessary to yield meaningful features. As shown by Jain, Ratha, and Lakshmanan (1997) a multitude of features computed over different spatial orientations and frequencies was necessary to yield a successful segmentation of an image with complex background. Because of its reputation in texture segmentation, Gabor features have also been popular in the automated defect detection of textured materials (Kumar and Pang 2002; Jing, Fang, and Li 2016; Li, Ma, and Liu 2016). But, since the Gabor filter only gives out texture features, an algorithm that

yields a meaningful result from those features is necessary. Kumar and Sherly (Kumar and Pang 2002), used a multi-channel filtering scheme, while Jing *et al.* (Jing, Fang, and Li 2016) used Kernel Principal Component Analysis upon the extracted features along with the OTSU threshold method to give a high defect detection rate. Correspondingly, Li, Ma, and Liu (2016) used Pulse Coupled Neural Network (PCNN) giving a detection accuracy of around 98.6%

Over the years, the Gabor filter has seen its use in a variety of applications. Li, Ma, and Liu (2016), proposed a method for road detection using Gabor filters. They effectively demonstrated the robustness of Gabor filters by detecting roads in various lighting conditions (night, entering tunnel, and shadowing). Their proposed methods consisted of two steps: locating vanishing-point based on a soft voting scheme upon dominant texture orientations, and then detecting the road lane ahead of the vehicle via edge detection method, while effectively constraining the search of lane marks using the vanishing point.

El-Sayed, Hassaballah, and Abdel-Latif (2016) proposed an authentication mechanism based on the identification of retinal features. They efficiently used the Gabor filter to segment the retinal blood-vessel, and then ran Support Vector Machine (SVM) upon the resulting feature pattern for feature matching. They claim this method to be stable regarding multiple and rotary shifts of digital retina images, and their test result corroborates their claim as they were able to achieve an accuracy of around 96.9%.

In their research, Gornale, Patil, and C. (2016) showed an interesting way of identifying gender based on features gathered from Discrete Wavelet Transform (DWT) and Gabor-based feature. When most of the research was focusing on facial features, this was a pretty interesting method as it was able to achieve 97% accuracy.

Rizvi, Cabodi, Gusmao, and Francini (2016) demonstrated the use of Gabor features for object detection. Aided with Gabor filters, the feedforward Neural Network model was able to achieve an accuracy of 50.71%, which was comparable to that of CNN (52.15%). This was fascinating as it was able to achieve such accuracy in less training time.

Avinash, Manjunath, and Kumar (2016), argued about the failure of previously employed methods in a real-time application for detection of lung cancer in early stages. They propose the usage of the Gabor filter along with a Marker-driven watershed segmentation technique on Computed Tomography (CT) images to overcome the hurdle.

Continuing on Gabor filters, Daamouche, Fares, Maalem, and Zemmouri (2016) proposed an unsupervised method of application of Gabor filters and morphological operators for building detection on remotely sensed images.

Over the years, the Gabor filter has seen its heavy usage in the extraction of facial features. In 2016, Hemalatha and Sumathi (2016) proposed the Median and Gabor filters along with Histogram Equalization as a combined preprocessing method for yielding a better-enhanced image. They argue that their technique will lead to a color-normalized, noise-reduced, edge-enhanced, and contrast-illuminated image.

In the same fashion, Lefkovits, Lefkovits, and Emerich (2017) proposed the use of Gabor filters to aid detection of the eye and its openness. Their methodology primarily consisted of using the Gabor filter to detect the eye which was aided with Viola-Jones face detection (Viola and Jones 2001) to speed up the process and a self-created face classifier based on Haar features to lower false positive of detection rate.

Around the same time, Pumlumchiak and Vittayakorn (2017) presented a novel framework for facial expression recognition. Their method primarily consisted of extraction of Gabor filter responses as facial features, mapped upon feature subspace

using the joint framework of Principal Component Analysis (PCA), Principle Components (PCs) removal, and Linear Discriminant Analysis (LDA). Their experimental result outperformed existing baselines, and thus substantiating that the weighted neighbor as a good approach to classifying facial expressions to four different classes: anger, surprise, happiness, and neutral.

However, Mahmood, Jalal, and Evans (2018) went with a different approach to tackling the facial expression recognition task. Their method comprised of a combined Radon transform and Gabor transform for facial feature extraction, fed towards a fused-classifier approach in the form of Neural Network over Self-Organized Maps (SOM). This particular method achieved accuracy of 84.87% on average over two public datasets on classification of six different expressions - surprise, anger, sadness, disgust, happiness, and fear.

Rather than using Standard Gabor Filter Ensemble (SGFE) of varying scale and orientation, Low, Teoh, and Ng (2016) proposed a Condensed Gabor Filter Ensemble (CGFE) in which the diversified traits of multiple SGFE are condensed into a single one. Their method of self-cross convolving the pre-selected Gabor filters outperformed the state of the art face descriptors Linear Binary Pattern (LBP) variants: Discriminant Face Descriptor (DFD) (Lei, Pietikäinen, and Li 2014) and Compact Binary Face Descriptor (CBFD) (Lu, Liong, Zhou, and Zhou 2015).

Nava, Escalante-Ramirez, and Cristobal (2012) proposed a new filtering scheme, Log-Gabor, designed to eliminate the non-uniform coverage in Fourier domain produced by the Gabor filter, and thus strongly correlating with Human Visual System (HVS). In 2017, Nunes and Pádua (2017), expanded on this filtering scheme and proposed a local descriptor called multi-spectral feature descriptor (MFD), designed specifically to work with images acquired over different frequencies across the electromagnetic spectrum. Upon evaluation, it was found to be computationally efficient

while maintaining the same precision and recall as the extant state-of-the-art algorithms.

The feature point matching method presented by Liu, Lao, and Pang (2019) for infrared and visible image matching also effectively utilizes Log-Gabor for generating the descriptors. This method based upon the Log-Gabor filters and Distinct Wavelength Phase Congruency (DWPC) effectively helps in matching non-linear images with different physical wavelength, and the experiment results corroborate it, as this method outperformed traditional approaches: edge-oriented histogram descriptor (EHD), phase congruency edge-oriented histogram descriptor (PCEHD), and log-Gabor histogram descriptor (LGHD), in infrared and visible images by 50%.

In the case of image segmentation, the Gabor filter has been the standard, and Premana, Wijaya, and Soeleman (2017) demonstrated this by using just simple, yet powerful K-Means clustering algorithm to segment the object from its complex background with the aid of Gabor filter responses. Fan, Zhang, Mei, and Liu (2017) proposed a novel woven fabric recognition method based on a similar concept. They proposed the utilization of Gabor filter to determine the orientation of texture at yarn crossing points segmented with K-means clustering and gradient accumulation. This segmentation capability of the Gabor filter is further demonstrated by Srivastava and Srivastava (2019) as they propose a novel method for salient object detection. Combined with the foreground saliency map formed from backgroundness score via minimum directed backgroundness and segmented images obtained from Gabor filters, this method utilizes an objectness criterion to choose the segment containing the salient object. Although failing in some conditions, this method effectively outperforms state-of-the-art algorithms (evaluated by PR-curve, F-Measure curve, and Mean Absolute Error upon 8 different public datasets).

Recently, Khaleefah, Mostafa, Mustapha, and Nasrudin (2019) proposed an interesting method to combat the deformations in paper images formed by extant

scanners. Their novel Automated Paper Fingerprinting (APF) utilized the combined effort of Gabor filters and Uniform Local Binary Patterns (ULBP) for extracting both local and global information for better texture classification. Their evaluation effectively highlights the need for Gabor filter as the combined approach was able to outperform the standalone ULBP system by 30.68%.

2.2 Convolutional Neural Network

In the field of image processing, many consider Convolutional Neural Network (CNN) to be state-of-the-art. CNN represents a family of statistically learning models which is primarily based upon the convolution operation of images with filters leading to feature-mapping layers. In a similar fashion to any other Neural Network (NN) models, it is biologically inspired by visual neuroscience theory. Hubel and Wiesel (1962), found out that in a cat's visual cortex there are simple cells and complex cells present which fire in response to certain properties of visual sensory inputs. While the simple cell showed a response to low-level spatial features like the orientation of edges, complex cells exhibited more spatial invariance. And, the architecture of CNN is similar to that - a hierarchical multi-layer network where receptive layers are designed to capture some specific peculiarity of the image while the following layers build upon that to create more abstract features.

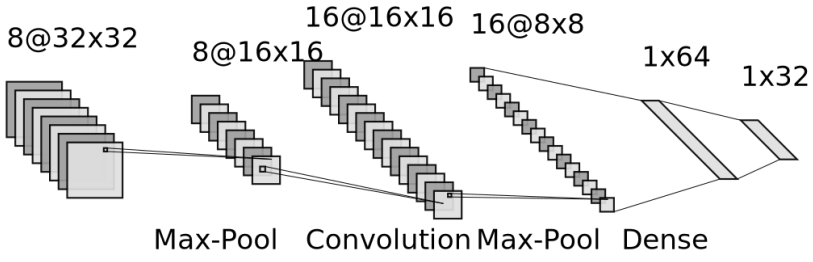


Figure 2.2. An illustration of architecture of CNN with convolutional layers, pooling layers and dense layers.

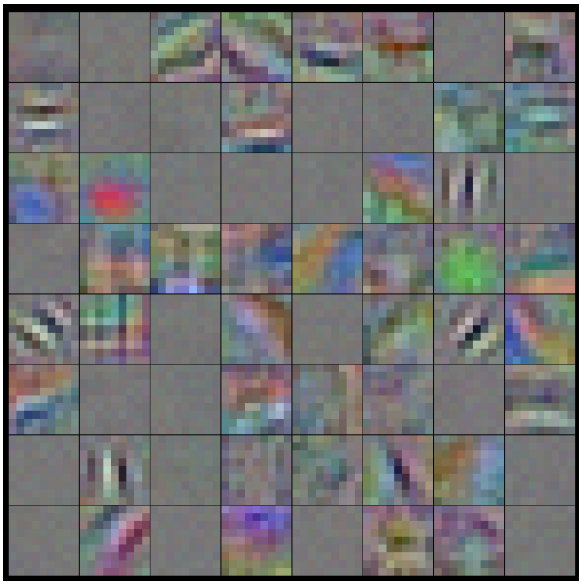


Figure 2.3. Visual representation of 64 convolutional kernels of size 9×9×3 learned by the first convolutional layer on the 32×32×3 input images.

A general CNN consists of some combinations of convolution layers, pooling layer, activation layer, and dense (fully-connected) layer, but modification can be seen according to the application as required like the addition of dropout layer, normalization layer, etc. for issues like overfitting, uniformity, etc. CNN is trained usually via backpropagation (Le Cun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel 1990) in which the weight is updated using variation of gradient descent like Stochastic Gradient Descent (SGD), Mini-batch Gradient Descent, etc. (Ruder 2017).

CNN is not new in the field. It has seen its usage back in the 1990s too. Le Cun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel (1990) effectively utilized the model for recognizing handwritten zip codes to constrain the error rate to 1%, while Lawrence, Giles, Ah Chung Tsoi, and Back (1997) and Rivas and Chacon (Chacon and Rivas 2009) effectively demonstrated the capability of CNN in face recognition by outperforming Karhunen-Loeve (KL) transform and Multi-Layer Perceptron (MLP). But it was the work of Krizhevsky, Sutskever, and Hinton (2012)

that brought CNN back into the limelight. AlexNet made a significant stride in the field of image recognition as it effectively demonstrated that a deeper model is much better than a wider model. With a margin of 10.9% compared to the second-placed model, it won the ILSVRC-2012 competition. Following the success of AlexNet various deeper models like Residual Networks (ResNets) (He, Zhang, Ren, and Sun 2016), GoogleNet (Szegedy, Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke, and Rabinovich 2015), VGGNet (Simonyan and Zisserman 2015), etc. have been developed, and also various studies like object recognition (Ishii, Nakamura, Nakada, Mochizuki, and Ishikawa 2015; Kumar and Sherly 2017; Sudharshan and Raj 2018; Zulkeflie, Fammy, Ibrahim, and Sabri 2019), 3D object detection (Wang, Lu, Chen, and Wu 2015; Schwarz, Schulz, and Behnke 2015), pedestrian detection (Szarvas, Yoshizawa, Yamamoto, and Ogata 2005), learning scene gist (Wu, Wu, and Kreiman 2018), etc. have been conducted.

CNN generally outperforms other supervised learning algorithms when it comes to image processing. Ishii, Nakamura, Nakada, Mochizuki, and Ishikawa (2015) confirmed CNN outperforms Support Vector Machine (SVM) while automating the task of analysis of the image coming from the satellite. Szarvas, Yoshizawa, Yamamoto, and Ogata (2005) showed that CNN is able to reduce the False Positive Rate (FPR) to less than $\frac{1}{5}$ of SVM when trained on pedestrian images with complex background and subject, and they mainly attributed this to the optimization of feature representation by CNN. Likewise, regarding the multi-class object recognition problem, Hayat, Kun, Tengtao, Yu, Tu, and Du (2018) showed that a 5-layered CNN was able to achieve 90.12% accuracy, which completely outperformed different classical bag-of-words (BOW) approaches. Similarly, the research of Zulkeflie, Fammy, Ibrahim, and Sabri (2019) evaluates AlexNet, basic CNN, and Bag of Features (BoF) with Speeded-Up Robust Feature (SURF) and SVM classifier, and found out that AlexNet and basic CNN model outperforms BoF model. Looking at all this research, it is evident that

most of the timen the general CNN model suffices for the task at hand, though sometimes tweaks may be necessary concerning the complexity of task, accuracy, memory requirements, etc.

Kawano and Yanai (2014) integrated conventional hand-crafted image features, namely, Fisher Vectors with Histogram of Oriented Gradients (HOG) and Color patches, with the convolutional features, boosting the accuracy to 72.6% in a 100-class food dataset. It completely outperformed the existing best accuracy rate - which was at 59.6%.

Biologically inspired, Wu, Wu, and Kreiman (2018) proposed integration of scene's gist for object recognition improvement, similar to how humans foveate on an object and incorporate periphery information to aid object recognition. Coined as GistNet, their model consisted of two CNN models - a fovea sub-network for object recognition and a periphery sub-network for contextual modulation. With VGG-16 as a baseline, their approach improved the accuracy by 50% for certain object recognition while increasing the size by only 5%.

Kumar and Sherly (2017) fine-tuned the last two layers of a pre-trained VGG-16 CNN model and trained on their augmented data to avoid overfitting due to lack of training data. This approach led them to an accuracy of 81.6%, which is good considering the scarcity of training data.

In regard to overfitting and data deficiency, as a supervised learning approach, CNN needs a large amount of data in order to boost its performance and generalization. While transfer learning could be done to deviate from the need for an expensive labeling process, Dosovitskiy, Springenberg, Riedmiller, and Brox (2014) proposed a discriminative unsupervised feature learning approach. Training the network to discriminate between surrogate classes, created by applying a variety of transformations to a randomly sampled seed image path, led it to outperform extant state-of-the-art

unsupervised methods. With regard to the context and argument, this novel approach is certainly worth taking.

Over the years, CNN has seen its fair share of use as a feature extractor too (Chen, Lam, Jacobson, and Milford 2014; Kataoka, Iwata, and Satoh 2015; Wang, Lu, Chen, and Wu 2015; Cao, Chen, and Khosla 2015; Fang, Ding, Zhong, Love, and Luo 2018). Upon evaluation of AlexNet and VGGNet, Kataoka, Iwata, and Satoh (2015) showcased that not only the final fully-connected layers but the intermediate layers too, can act as a source of features to enhance recognition performance. Similarly, Chen, Lam, Jacobson, and Milford (2014) also ascertained that different layers in CNN could engender features suitable for the detection of different aspects of place recognition task.

Wang, Lu, Chen, and Wu (2015), proposed the use of CNN along with SVM, where CNN acts as the feature extractor and SVM as the classifier, for 3D object recognition. In their proposed approach, they first converted depth modality into 3 channels, and then fine-tuned two pre-trained Caffe models (Jia, Shelhamer, Donahue, Karayev, Long, Girshick, Guadarrama, and Darrell 2014), in order to extract representative sparse features from color (RGB) and depth (RGB-D) images, finally to be used by SVM classifier. On experimentation, this approach yielded 91.35% accuracy, much better than the state-of-the-art and also CNN model trained solely on RGB images from the RGB-D object dataset.

Similarly, Schwarz, Schulz, and Behnke (2015) proposed the use of a pre-trained CNN model as a feature extractor, in conjunction with SVM classifier for RGB-D object recognition based on its pose estimation. Their approach incorporated depth features by rendering objects from canonical views and coding metric distance from the object center with the color scheme, making it suitable for CNN to extract meaningful features.

Continuing on 3D object recognition, Gao, Wang, Xue, Xu, Zhang, and Wang (2018) proposed pairwise Multi-View CNN (coined as PMV-CNN), designed to explicitly deal with lack of training samples while also maintaining latent complementary information from different views explored via view pooling. Their novel approach used a pair of CNN in order to jointly learn the visual features from multiple views and optimize towards object recognition.

Since Spiking Neural Network (SNN) based architecture is energy efficient when used in conjunction with spike-based neuromorphic hardware,Cao, Chen, and Khosla (2015) proposed a novel approach for converting CNN into an SNN in order to map into spike-based hardware. When training the tailored CNN, this approach gets exposed to the learning capability of CNN, and while transferring the learned weight of the tailored CNN back to SNN, it becomes energy efficient and compatible with spike-based neuromorphic hardware. Regarding real-time object recognition, they found this approach to be more energy efficient than Field Programmable Gate Array (FPGA)-based implementation of CNN by two orders of magnitude.

As shown by all these aforementioned studies, while CNN has been established as the state of the art for object recognition, it can be expanded to recognition in real-time too. Upon implementation of CNN on FPGA, Ahn (2015) was able to achieve 170,000 classifications per second and scale-invariant object recognition from a 720×480 video stream at a speed of 60 fps. Similarly, Radovic, Adarkwa, and Wang (2017) proposed the use of YOLO - a CNN based open-source object detection and classification platform - for classification of the object on real-time video feed obtained form Unmanned Aerial Vehicles (UAV).

Maturana and Scherer (2015) proposed a 3D CNN architecture, coined VoxNet, that integrated a volumetric occupancy grid representation with 3D CNN for real-time object detection. This representation enabled full utilization of information coming from range sensors, ultimately boosting performance to labeling hundreds

of instances per second. Inspired by Maturana and Scherer (2015), Garcia-Garcia, Gomez-Donoso, Garcia-Rodriguez, Orts-Escolano, Cazorla, and Azorin-Lopez (2016) proposed the use of density occupancy grids as the inner representation for input data in a model coined PointNet. When integrated with the 3D CNN model, this approach significantly boosted the performance. Expanding upon Maturana and Scherer (2015), Zhi, Liu, Li, and Guo (2017) proposed LightNet — a lightweight volumetric 3D CNN. Their compact model was more computationally efficient than VoxNet, while a combination of different kinds of auxiliary learning tasks made it less vulnerable to overfitting.

Likewise, Jing Huang and Suya You (2016) introduced a 3D point cloud labeling scheme based on 3D CNN. Representation based on only voxelized data made it straightforward. While complications like exceeding memory usage, biased classification, etc. could exist, they did present solutions for handling such data.

Fang, Ding, Zhong, Love, and Luo (2018) devised a novel approach Improved Faster Regions with CNN Features (IFaster R-CNN) to address the generalization issue while detecting objects on construction sites in real-time. Their approach was also based upon the use of CNN as base feature extractor from images, which then with the use of Region Proposal Network (RPN) to concurrently predict object bounds and objectness scores at a particular position, fed the extracted regional proposals were fed into Fast R-CNN module for detection. With detection speed at real-time at 0.101 s per image and accuracy of about 91% and 95% for worker and excavator respectively, they completely outperformed extant state-of-the-art by an average of 50%.

Du, Muslikhin, Hsieh, and Wang (2020) experimented with a six-degree-of-freedom (6-DOF) robot arm with a gripper, their proposed method successfully yielded an accuracy of 98.44% for the stereo vision-based object recognition and manipulation. Their hybrid algorithm comprised of an adaptive network-based fuzzy

inference system (ANFIS) for the eye-to-hand calibration and R-CNN for object detection.

While accuracy has been the most important aspect researchers prioritized, there has been considerable research done to boost the speed of recognition too (Ciresan, Meier, Masci, Gambardella, and Schmidhuber 2011; Anwar, Hwang, and Sung 2015; Xu, Dehghani, Corrigan, Caulfield, and Moloney 2016). The authors of (Ciresan, Meier, Masci, Gambardella, and Schmidhuber 2011) were able to considerably drop the error rates in fewer epochs when trained using their fast, fully parameterizable GPU based CNN.

Similarly, with regard to speed, Anwar, Hwang, and Sung (2015) proposed fixed-point optimization for reducing the number of parameters. They effectively quantized layers of pre-trained high precision networks using L2 error minimization based on layerwise sensitivity on word-length reduction. Their approach not only significantly reduced memory usage but also generalized the model.

As 3D object detection is computationally demanding, Xu, Dehghani, Corrigan, Caulfield, and Moloney (2016) proposed Volumetric Accelerator (VOLA) for the memory-efficient representation of the 3D volumetric object. With a reduction in memory usage, they claimed their representation model to be better in terms of speed, and their experimental result supported this as their VOLA-based CNN performed 1.5 times faster than the original LeNet.

2.3 Gabor and CNN

Judging from all these studies it can be clearly seen that both the Gabor filter and CNN can act as excellent feature extractors. However, as seen in previous research (Kawano and Yanai 2014; Wu, Wu, and Kreiman 2018), CNN can skip over some of the valid specific information and therefore, can immensely benefit by being complemented with other manual features. Since Gabor has been defined to extract all sorts of features (Pumlumchiak and Vittayakorn 2017; Lefkovits, Lefkovits, and

Emerich 2017; Nunes and Pádua 2017) in a different domain, this makes it a well-suited candidate to complement CNN, and in fact, a lot of studies have shown this to be the case (Yao, Chuyi, Dan, and Weiyu 2016; Molaei, Shiri, Horan, Kahrobaei, Nallamothu, and Najarian 2017; Hosseini, Lee, Kwon, Koo, and Cho 2018; Jiang and Su 2018).

In conjunction with Gabor filter features, Yao, Chuyi, Dan, and Weiyu (2016) found that CNN yielded a 1.26% boost in accuracy when employed for object recognition in the natural scene. With an accuracy of 81.53%, it outperformed standalone CNN marginally and significantly outperformed the Bag-of-Words model with Scale Invariant Feature Transform (SIFT). In the same manner, Hosseini, Lee, Kwon, Koo, and Cho (2018) utilized the Gabor filter responses to boost the accuracy of CNN for classification based on age and gender. Zadeh Taghi Zadeh, Imani, and Majidi (2019), also noted the boost in speed and accuracy when Gabor filter features were incorporated with CNN for fast facial emotion recognition.

The visualization of Gabor filters and first convolutional layers of CNN shows that they are quite alike. This was confirmed by Krizhevsky, Sutskever, and Hinton (2012) as when trained on real images, the first convolutional layers of the deep CNN was found to be similar to Gabor filters. Motivated by this fact, Alekseev and Bobe (2019) modified the architecture where the first layer of CNN was constrained to fit the Gabor function. Upon experimentation with different datasets, it was found to yield the same or even better accuracy with significant improvement in convergence.

Inspired by traditional local Gabor binary patterns, Jiang and Su (2018) proposed Gabor Binary Layer (GBL) as an alternative for the first layer of the CNN model. Composed of a module of predefined Gabor filters with different shape and orientation and a module of fixed randomly generated binary filters, GBL when experimented with different CNN models gave a better performance than the state-of-the-art CNNs.

In a similar fashion, Luan, Chen, Zhang, Han, and Liu (2018) extended the concept to multiple layers CNN. Coined as Gabor Convolutional Networks (GCN), their network comprised of predefined Gabor filters of different scale and orientation in multiple layers. Proposed to enhance the robustness of the model against image transitions, scale changes, and rotations, their model significantly enhanced performance over the baseline model while simultaneously reducing the training complexity too.

Built upon GCN, Liu *et al.* (Liu, Ding, Wang, and Zhang 2018) proposed a new learning model, Hybrid Gabor Convolutional Network (HGCN). While (Luan, Chen, Zhang, Han, and Liu 2018) focused on accuracy, (Liu, Ding, Wang, and Zhang 2018) went for memory efficiency. With hybrid binarized input and Gabor Binarized Filters (GBFs) in an end-to-end framework, HGCN was able to reduce memory usage by a factor of 32 while maintaining accuracy due to usage of GCN.

Molaei, Shiri, Horan, Kahrobaei, Nallamothu, and Najarian (2017) also initialized the first layer of CNN with predefined Gabor filters for effective Left Ventricle segmentation. Due to the robust nature of the Gabor filter, the model increased the performance in terms of specificity and sensitivity. Later on, Molaei and Shiri Ahmad Abadi (2020), expanded the model to maintain the structure of the Gabor filter during the training process. When compared with different initialization methods, it significantly outperformed all, even when dealing with noisy data and a lesser amount of training data.

All these existing approaches have a number of issues that have not been explored yet. First, restricting Gabor filters as the only thing that a CNN can use might be severely limiting the potential of a CNN to alter the structure, even so slightly, of a Gabor filter in order to maximize performance, or even completely destroy the spatial shape of an existing, under-performing filter. Second, current studies combining Gabor filters and CNNs have not shown a conclusive relationship

between the use of Gabor filters and convergence of a CNN, which is crucial to understanding the added computational cost of using Gabor filters as opposed to using randomly generated uniform white noise, which is the traditional approach. Third, while the evidence that CNNs and Gabor filters together are successful in very specific computer vision tasks, there is no sufficient evidence that Gabor filters can provide a significant advantage in general object recognition tasks.

In this thesis, we focus on the impact of Gabor filters on CNN when the receptive layer of CNN is initialized with Gabor filters. We try to obtain an improvement in the performance of CNN like accuracy, loss and convergence, on general object recognition. In the next chapter, we will discuss our proposed approach to experiment on initialization of CNNs with Gabor filters.

CHAPTER THREE

Methodology

This chapter presents the experimental methodology followed in our study. This chapter is organized as follows: Section 3.1 explains the construction of Gabor filter bank. Section 3.2 presents the different natures of dataset used for experimentation. Section 3.3 gives the insight to CNN architecture employed with each dataset. Section 3.4 describes the loss function and other training methodology employed for all CNN models. Section 3.5 highlights the success metrics upon which the experiments are evaluated. Section 3.6 explains the structure of each experiment.

3.1 Gabor Initialization and Control Group

A Gabor filter can be created using Equation 2.1, but in order to extract features from an image, a bank of Gabor filters is necessary. This is because a Gabor filter with certain orientation and frequency can only extract texture features aligning with that specific Gabor filter only. To design a bank of Gabor filters for the experiments, the approach proposed in Meshgini, Aghagolzadeh, and Seyedarabi (2012) has been used. The orientations θ_m and frequencies ω_n of the Gabor filters are obtained by the following equations:

$$\theta_m = \left(\frac{\pi}{8}\right).(m-1), m \in [1,8] \tag{3.1}$$

$$\omega_n = (\frac{\pi}{2}).2^{-\frac{n-1}{2}}, n \in [1,5] \tag{3.2}$$

σ is set as $\sigma \approx \frac{\pi}{\omega}$ and ψ is set by uniform distribution $U(0,\pi)$.

Depending upon the model of CNN, there can be a lot of convolutional layers, but for the sake of simplicity, the experiments have been designed to focus on the

impact of Gabor filters at the first receptive convolutional layer only. The nature of the experiment models can be divided into three major categories:

(1) Random weight initialization (Control group)

(2) Random initialization with a Gabor filter on each channel

(3) Repeated Gabor filter on all channels

3.1.1 Random Weight Initialization

This is the classic/traditional CNN kernel initialization method in which each kernel filter is initialized randomly. To be specific, Glorot uniform initialization method (Glorot and Bengio 2010), also known as Xavier uniform initialization, has been employed for initialization of the filters. In Glorot uniform initialization the samples are drawn from a uniform distribution within $[-limit, limit]$, where $limit = \sqrt{\frac{6}{(fan\_in + fan\_out)}}$, where fan_in is the number of input units and fan_out is the number of output units.

3.1.2 Random Initialization with a Gabor Filter on Each Channel

For this particular approach, a bank of Gabor filters of the necessary filter size is generated using the approach of Meshgini, Aghagolzadeh, and Seyedarabi (2012) described above, and each kernel filter (also called convolution window) of the receptive layer of CNN was initialized with a random Gabor filter from the filter bank, thus giving different filters for the receptive layer. The receptive layer has many sets of kernels, and each set of kernels in the receptive convolutional layer has three different kernel filters corresponding to 3 different channels of the image. Therefore within each set of kernel those 3 filters are initialized with different Gabor filter. During the training period, CNN is allowed to change the structure of those Gabor filters in order to extract features as needed.

3.1.3 *Repeated Gabor Filter on All Channels*

Similar to the previous approach, a bank of Gabor filters is generated using same technique, but instead of assigning to each filter in a kernel set randomly, a Gabor filter is chosen and assigned to all the 3 filters in that particular kernel set. So while the Gabor filter is different among different kernel sets, it is same within a set corresponding to the channels of the image during initialization. However, when trained upon given datasets, the structure of the Gabor filters could be changed by the CNN as needed.

3.2 *Datasets*

In order to analyze the effect of Gabor filter on CNNs, diverse multi-class datasets ranging in terms of number of classes, nature of objects, image size and distribution were considered. Depending upon the nature of dataset, the training images were subjected to an optional image pre-processing, followed by rescaling pixels by $\frac{1}{255}$ and one-hot encoding of the labels before being passed to respective CNN architecture for training and validation.

3.2.1 *Cats Vs Dogs Version-1.0*

Cats vs Dogs (Elson, Douceur, Howell, and Saul 2007) is a balanced dataset comprising of images of cats and dogs. There are a total of 25,000 images with equal distribution among the two classes. Since the images are in varying sizes, they have all been resized to 256×256 size and then split up in 80:20 ratio to represent training and test data respectively, giving 10,000 images per class for training set and 2500 images per class for validation set. Samples of the dataset can be seen in Figure 3.1, and its distribution in Figure 3.2.

Figure 3.1. Cats vs Dogs dataset.

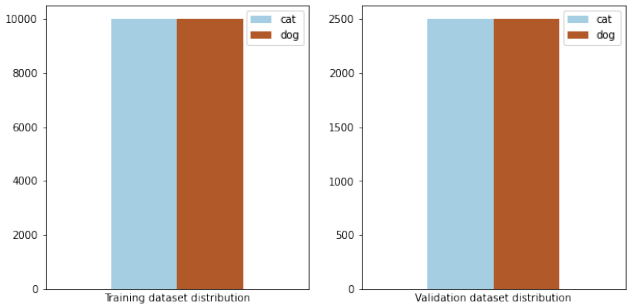


Figure 3.2. Cats vs Dogs dataset distribution.

3.2.2 CIFAR-10 Version-3.0.2

The CIFAR-10 (Krizhevsky 2009) dataset is one of the standard tiny images dataset which was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The dataset consists of 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. Samples of the dataset can be seen in Figure 3.3, and its distribution in Figure 3.4.

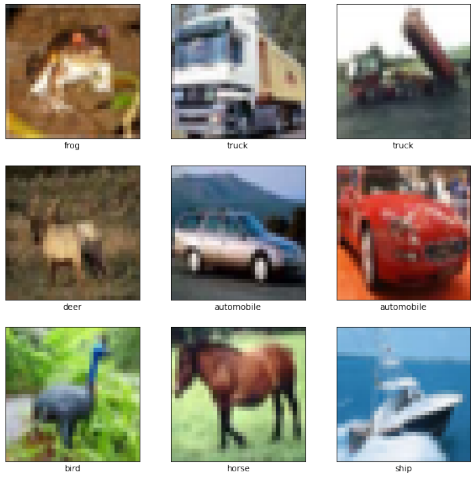


Figure 3.3. CIFAR-10 dataset.

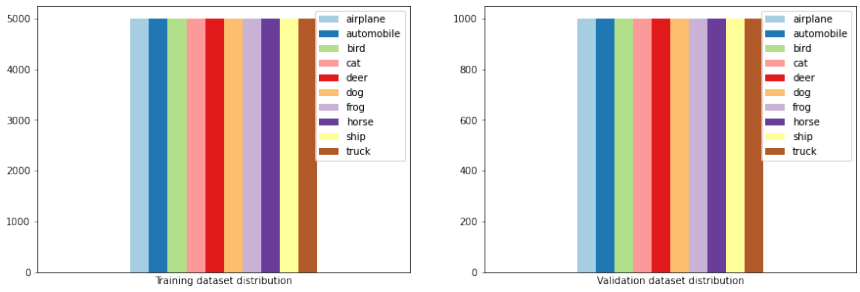


Figure 3.4. CIFAR-10 dataset distribution.

3.2.3 CIFAR-100 Version-3.0.2

The CIFAR-100 dataset is also the contribution of Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. For this experiment, the version 3.0.2 of the dataset has been utilized. The dataset consists of 60,000 32×32 color images in 100 classes, with 600 images per class. There are 50,000 training images and 10,000 test images. Although the dataset is balanced, to address the relatively small number of training dataset per class, data augmentation was performed on the training images. Data augmentation basically means applying set of transformations like rotation, zooming, etc. on the original image, thus giving a new variation of the image. Applying data augmentation does not mean increasing the number of training samples per epoch

of training, but it means that on each epoch the model will be learning from new variations of the original image, thus increasing the total number of unique images in the whole training process from start to finish, but not in terms of epoch. This helps the learned model to become more robust and accurate as it is trained on different variants of the original image. The particular set of transformations applied were:

(1) Rotation - Randomly rotate the image d degree, where $d \in [0, 40]$

(2) Width shift - Randomly shift the image x pixels horizontally, where $x \in [0, w]$ and $w = 0.2*$fraction of total width of the image

(3) Height shift - Randomly shift the image x pixels vertically, where $x \in [0, h]$ and $h = 0.2*$fraction of total height of the image

(4) Shear - Randomly shear the image d degree in counter-clockwise direction, where $d \in [0, 0.2]$

(5) Zoom - Randomly zoom the image in range $[0.8, 1.2]$

(6) Horizontal flip - Randomly flip the image horizontally

Samples of the dataset can be seen in Figure 3.5, and its distribution in Figure 3.6.

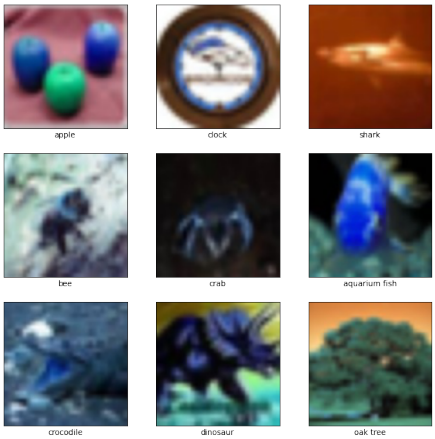


Figure 3.5. CIFAR-100 dataset.

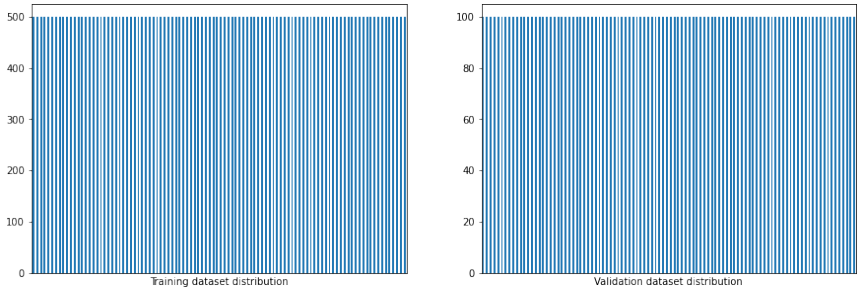


Figure 3.6. CIFAR-100 dataset distribution.

3.2.4 Caltech 256 Version-2.0

Caltech 256 (Griffin, Holub, and Perona 2007) is one of the highly regarded general objects datasets that is considered to be a significant improvement over its predecessor - the Caltech 101 dataset, in terms of larger category sizes, new and larger clutter categories, and overall increased difficulty. This dataset consists of 30,607 images in 257 object categories. Samples of the dataset can be seen in Figure 3.7, and its distribution in Figure 3.8.

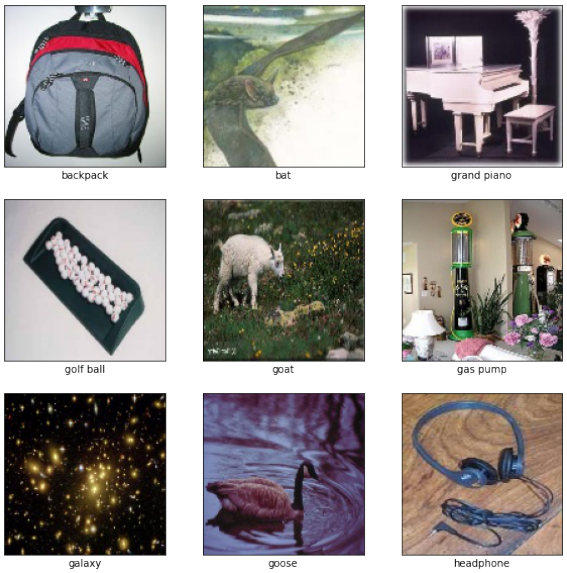


Figure 3.7. Caltech 256 dataset.

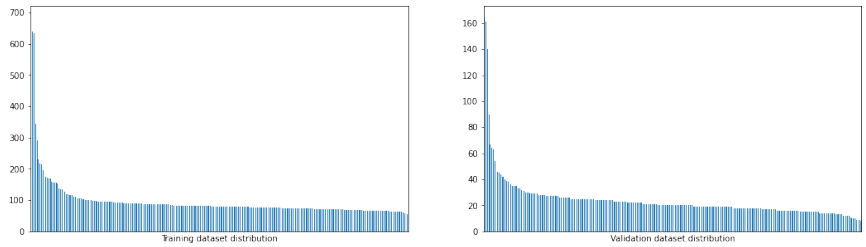


Figure 3.8. Caltech 256 dataset distribution.

The distribution of images per object category are:

(1) Mean: 119.0933

(2) Median: 100

(3) Minimum: 80

(4) Maximum: 827

Since the dataset was not distributed into the training dataset and validation dataset, a split in 80:20 ratio of the whole dataset was made for training and validation. This gave 24,485 images for training and 6,122 images for validation distributed among 257 classes while also maintaining their original distribution ratio among different classes in both training dataset and validation dataset. Since the size of most of the class was small, data augmentation was performed here with a set of transformations, the same as described in Subsection 3.2.3.

3.2.5 Stanford Cars Version-2.0

The Stanford cars dataset (Krause, Stark, Deng, and Fei-Fei 2013) contains large image sets of different car models from different years. The dataset has a total of 16,185 color images of cars in 196 categories. The data has already been split to 8,144 training images and 8,041 testing images, thus giving a balanced 50-50 split for each class. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe. Samples of the dataset can be seen in Figure 3.9, and its distribution in Figure 3.10.

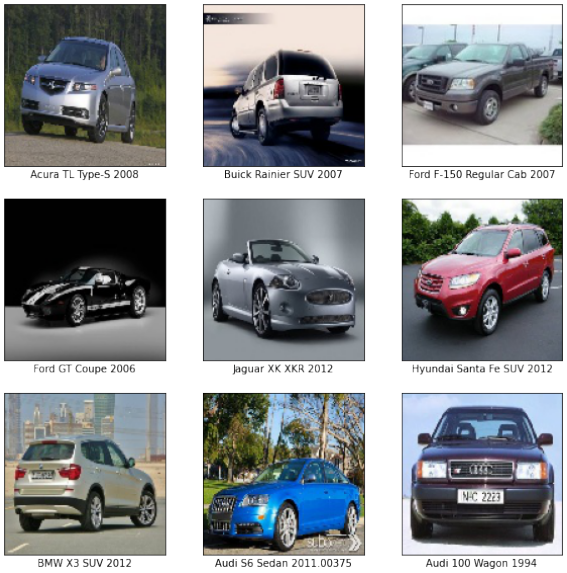


Figure 3.9. Stanford cars dataset.

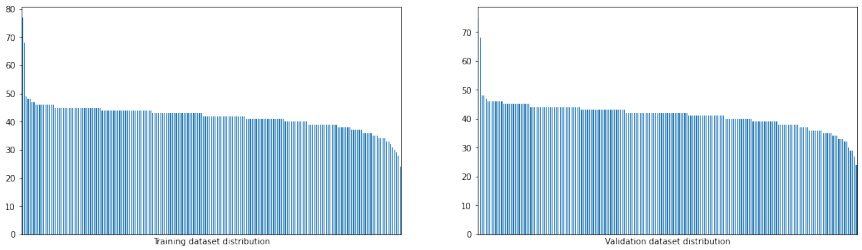


Figure 3.10. Stanford cars dataset distribution.

The distribution of images per object category for training dataset are:

(1) Mean: 41.7641

(2) Median: 42

(3) Minimum: 24

(4) Maximum: 77

The distribution of images per object category for validation dataset are:

(1) Mean: 41.2358

(2) Median: 42

(3) Minimum: 24

(4) Maximum: 75

To tackle the small size of this training dataset and skewness in distribution, data augmentation was employed on this dataset as well. The set of image transformations applied was same as that for Subsection 3.2.3 with one additional transformation, converting the RGB image to grayscale.

3.2.6 Tiny Imagenet

The Tiny Imagenet dataset is a subset of widely popular Imagenet dataset (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, and Fei-Fei 2015). It contains a diverse range of 64×64 object images to be considered as a valid recognition problem. There are 100,000 training images and 10,000 test images distributed over 200 object categories evenly, giving 500 training images and 50 testing images per class. Samples of the dataset can be seen in Figure 3.11, and its distribution in Figure 3.12.

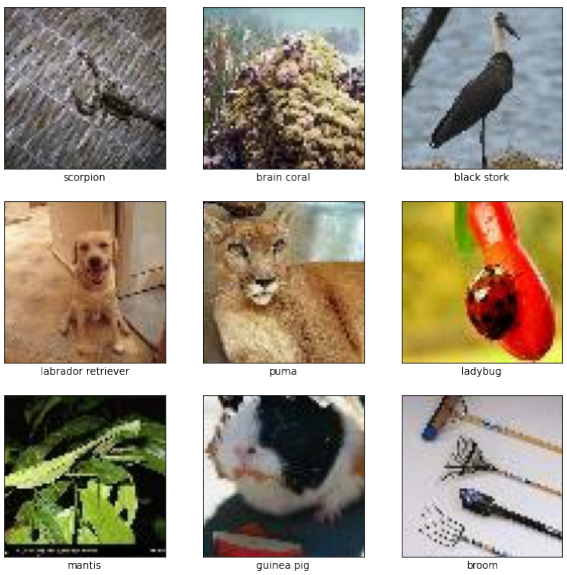


Figure 3.11. Tiny Imagenet dataset.

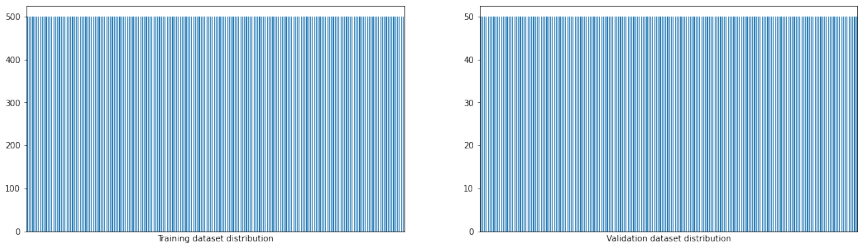


Figure 3.12. Tiny Imagenet dataset distribution.

Since the size of training dataset is relatively small, to make the model more robust data augmentation was applied to the training images. The set of transformations varied slightly compared to the previously employed ones in Subsection 3.2.3.

(1) Horizontal flip - Randomly flip the image horizontally

(2) Width shift - Randomly shift the image x pixels horizontally, where $x \in [0, w]$ and $w = 0.2*$fraction of total width of the image

(3) Height shift - Randomly shift the image x pixels vertically, where $x \in [0, h]$ and $h = 0.2*$fraction of total height of the image

(4) Saturation - Randomly saturate the image with s factor, where $s \in [0.5, 0.2]$

3.3 Architectures

Depending upon the nature of the dataset like number of classes, nature of images, etc. different architectures of CNN were employed. The choice of all CNN architecture was made upon some preliminary experiments and thus it was ensured that our model did not underfit or overfit the data. While these models certainly do not outperform state-of-the-art, they do represent the basic traditional CNN structure which give good performance without the need for any external features or immense change in the basic structure.

3.3.1 Cats vs Dogs

The CNN model designed for this dataset consisted of 3 convolutional layers, where each convolutional layer was followed by batch normalization, activation, max

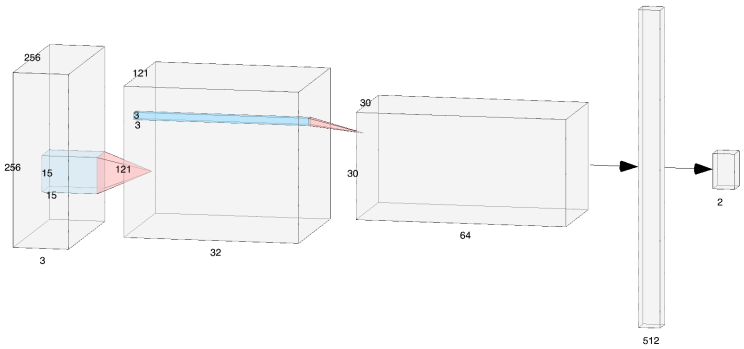


Figure 3.13. CNN architecture for Cats vs Dogs dataset.

pooling and dropout in the same order. The network was expanded with a densely connected neural network followed by batch normalization, its activation and dropout and then the final densely connected neural network. The CNN architecture can be seen in Figure 3.13.

All the convolutional layers were operated with 3×3 kernel filter size, $(2, 2)$ stride and valid padding, except the receptive convolutional layer, which has different kernel size depending upon the experiment and $(1, 1)$ stride. Bias was also each added on all convolutional layers and densely connected neural networks. All biases were initialized to zero, while all kernel filters were initialized with Glorot uniform sampling as mentioned in Subsection 3.1.1, except for the receptive layer where the kernel filters were initialized with either Glorot uniform sampling or a Gabor filter depending upon the experiment. The number of sets of kernel filters in each convolutional layer was in order of 32, 64 and 128, while the number of units in densely connected neural networks was in order of 256 and 2.

Batch normalization helps to normalize the inputs by maintaining the mean close to 0 and standard deviation close to 1. A batch of size 32 was used for the training process. The batch normalization (Ioffe and Szegedy 2015) works in different ways during training and inference (prediction/validation). When training, normalization is done over the given size of a batch by calculating the mean and standard deviation

of the current batch. For each channel the batch input is normalized as:

$$normalization(batch) = \gamma * \frac{batch - \mu_{batch}}{\sqrt{\sigma_{batch} + \epsilon}} + \beta \tag{3.3}$$

where ϵ = small constant (0.001), γ = learned scaling factor (1) and β = learned offset factor (0).

During validation, since there is only batch size 1 involved, the normalization is performed using a moving average of the mean and standard deviation of the batches it has seen during training.

$$normalization\ (batch) = \gamma * \frac{batch - \mu_{moving}}{\sqrt{\sigma_{moving} + \epsilon}} + \beta \tag{3.4}$$

$$\mu_{moving} = \mu_{moving} * momentum + \mu_{batch} * (1 - momentum) \tag{3.5}$$

$$\sigma_{moving} = \sigma_{moving} * momentum + \sigma_{batch} * (1 - momentum) \tag{3.6}$$

μ_{moving} and σ_{moving} is updated during training, thus normalizing the inference inputs after having been trained on data that has similar statistics as that of the inference data.

Except for the last densely connected neural network, the activation layer has utilized ReLU as its activation function. Rectified Linear Unit (ReLU) function is defined as:

$$f(x) = max(0, x) \tag{3.7}$$

Softmax has been used for the activation of the last layer of the classification model because it gives the result as probability distribution. Softmax is defined as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.8}$$

where $\vec{z}$ = input vector to the softmax function, z_i = element of the input vector, K = number of classes in the multi-class classifier.

Max pooling downsamples the input in regard to its spatial dimensions (height and width) by taking the maximum value over the certain input window defined by pool size and shifted along each dimension by specific stride. Both pool size and stride are set at $(2, 2)$ for all max pool, with padding being valid.

All the dropout layer randomly set its input units to 0 with a frequency of specified *rate* during each step in training process. All the inputs not set to 0 are scaled up by $\frac{1}{1-rate}$ so that the sum over all inputs does not change. Since this dropout is only applicable during training process it helps in preventing over-fitting. The rate is fixed at 0.5 for all dropout layers.

3.3.2 CIFAR-10

The CNN model designed for this dataset is similar to that of Cats vs Dogs. All the parameters and structure is the same with only variation being in the hidden densely connected neural network as it is configured to have 512 units and dropout being set to a rate of 0.25 as show in Figure 3.14.

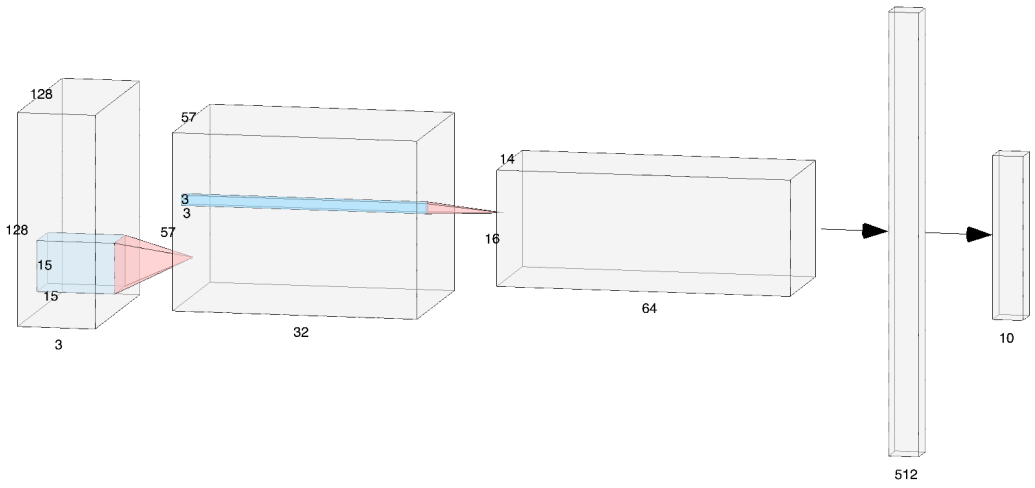


Figure 3.14. CNN architecture for CIFAR-10 dataset.

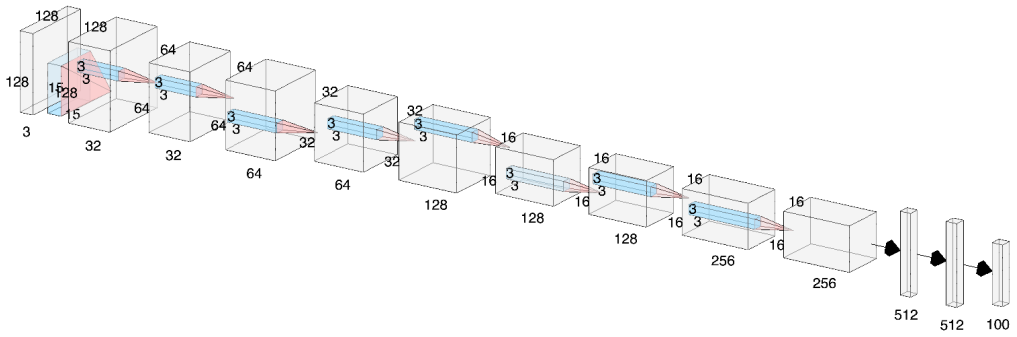


Figure 3.15. CNN architecture for CIFAR-100 dataset.

3.3.3 CIFAR-100

Since the number of classes in CIFAR-100 is really large compared to the previous datasets, a deeper model of CNN was employed. The architecture resembles that of VGG-16, but with fewer filters and units. The architecture follow the sequence of: two convolutional filters with 32 filters, two convolutional filters with 64 filters, three convolutional filters with 128 filters, three convolutional filters with 256 filters, two hidden densely connected neural networks with 512 units with dropout in between them, and finally followed by an output densely connected neural network with 100 units as shown in Figure 3.15.

As before, each of the convolutional layers and hidden densely connected neural networks is followed by batch normalization and activation. Max pooling and dropout is added in between the transition from convolution filters with a certain number of filters to another with a different number of filters. Each of the convolutional layers has a stride of (1,1) and the same padding, and the dropout rate is 0.1 for all dropout layers. Besides these, all the remaining parameters are same as the CNN model for Cats vs Dogs describe in Subsection 3.3.1.

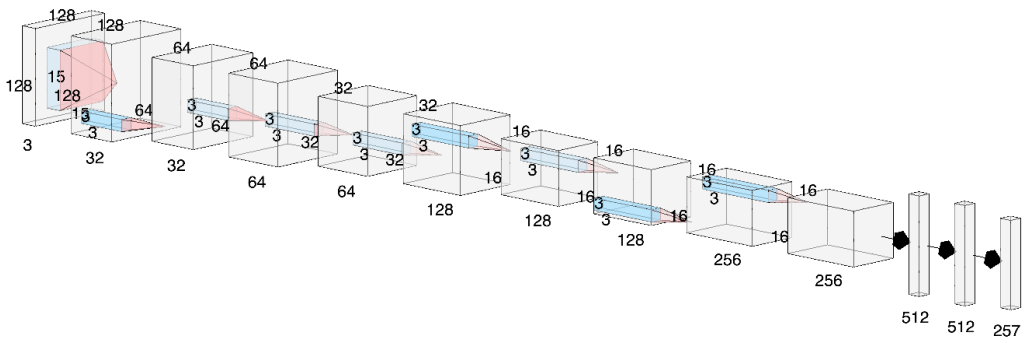


Figure 3.16. CNN architecture for caltech 256 dataset.

3.3.4 Caltech 256

The CNN model for Caltech 256 dataset is same as that for CIFAR-100 with a minor variation being in the number of units in the output layer. It is set to 257 units since there are 257 different classes as shown in Figure 3.16.

3.3.5 Stanford Cars

The CNN model for the Stanford cars dataset is the same as that for CIFAR-100 described in Subsection 3.3.3 with a minor variation being in the number of units in the output layer. It is set to 195 units since there are 195 different classes as shown in Figure 3.17.

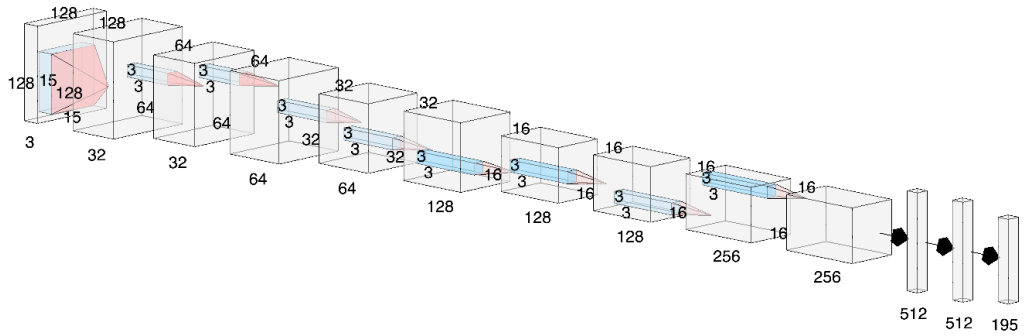


Figure 3.17. CNN architecture for stanford cars dataset.

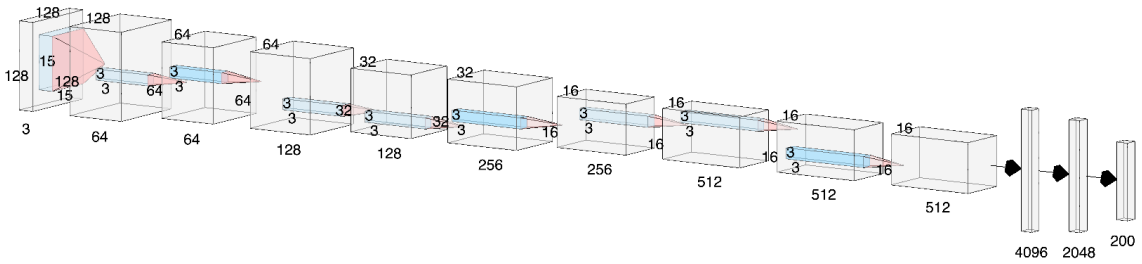


Figure 3.18. CNN architecture for Tiny Imagenet dataset.

3.3.6 Tiny Imagenet

The CNN model for Tiny Imagenet is also similar to that of CIFAR-100 described in Subsection 3.3.3 but with slight variation. The number of sets of kernel filters on each convolutional layer has been doubled and the dropout has been removed from convolutional layers. The number of units in densely connected neural networks has been set to 2048 with dropouts at rate 0.1 in between them. The number of units in output densely connected neural networks is set to 200 units since there are 200 different classes. Besides these, the remaining architecture and its parameters are same. The CNN architecture can be seen in Figure 3.18.

3.4 Loss Functions

Loss function is integral because it defines how much the model is currently off in terms of inference/prediction from the actual one and based on that the model is able to learn to improve upon that certain loss. Since there are many ways to define loss, the training model needs an optimal definition of loss and this can change depending upon the nature of dataset. There are two types of loss:

(1) Training loss - Loss calculated over training dataset

(2) Validation loss - Loss calculated over validation dataset

While minimum loss in both scenario is desired, it is important to focus on validation loss because at the end the learned model will be working on unseen data in a real-world scenario just like the validation data. Because of this need, all the models are being trained with an emphasis on the minimization of the validation loss.

3.4.1 Cats vs Dogs

For calculating the validation loss for this model, categorical cross-entropy function was chosen to be calculated as a sum over the batch of training data. Categorical cross-entropy function is simply cross-entropy function but it expects the class labels to be in one-hot encoding representation. Categorical cross-entropy loss is calculated by computing the following sum:

$$\text{Cross-Entropy loss} = -\sum_{i=1}^{N} y_i.\hat{y}_i \tag{3.9}$$

where $\hat{y}_i$ is the i-th scalar value in the model output, y_i is the corresponding target value and N is the number of scalar values in the model output.

Adaptive Moment Estimation (Kingma and Ba 2017), often referred to as Adam, has been chosen as the optimizer for the model based on the validation loss. Adam optimization is a stochastic gradient descent method that is based on the adaptive estimation of first-order and second-order moments of gradient that adapts the learning rate for each weight of the neural network. Adam is generally preferred because it combines the heuristics of both Momentum and RMSProp. Adam keeps exponentially decaying the average of past gradients m_t, similar to momentum. For each weight parameter w_t, the decaying averages are calculated as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.10}$$

where m_t is the estimate of the first moment (mean) of the gradients along w_j, g_t is the gradient of the loss function on current mini batch at time t and β_1 is the exponential decay rate for the first moment estimates.

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.11}$$

where m_t is the estimate of the second moment (uncentered variance) of the gradients along w_j, g_t is the gradient of the loss function on current mini batch at time t and β_2 is the exponential decay rate for the second moment estimates.

$$w_{t+1} = w_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon} g_t \tag{3.12}$$

where w_{t+1} is the new weight parameter, ϵ is the learning rate and ϵ is very small number to prevent any division by zero.

As suggested by Kingma and Ba (2017), β_1 is kept constant at 0.9, β_2 is kept constant at 0.999 and ϵ at $1e^{-7}$. The learning rate for Adam is set at 0.001 initially. This learning rate is reduced by a factor of 0.5 whenever the validation loss stopped improving for 10 consecutive epochs. The minimum threshold for measuring the new optimum is set at 0.0001. The model is trained until the validation loss stops improving, and this early stopping is executed when the validation loss stops improving at all for 35 consecutive epochs.

3.4.2 CIFAR-10

For CIFAR-10 dataset, same metrics were used as described in Subsection 3.4.1.

3.4.3 CIFAR-100

For CIFAR-100 dataset, same metrics were used as described in Subsection 3.4.1.

3.4.4 Caltech 256

For Caltech 256 dataset, same metrics were used as described in Subsection 3.4.1.

3.4.5 Stanford Cars

For Stanford cars dataset, same metrics were used as described in Subsection 3.4.1.

3.4.6 Tiny Imagenet

For Tiny Imagenet, there is a slight variation in the learning procedure. Although the loss function was the same — categorical cross-entropy — label smoothing was applied to label values. Label smoothing helps to relax the confidence on label values. For example, when a label is smoothed by s, $\frac{s}{\text{number of classes}}$ is assigned to non-target labels and $(1 - s) + \frac{s}{\text{number of classes}}$ for target labels. The value of s is set at 0.1.

Beside this, there is also change in the optimizer. Stochastic Gradient Descent (SGD) with Nesterov momentum (Sutskever, Martens, Dahl, and Hinton 2013) has been used as the optimizer for the learning procedure. The Stochastic Gradient Descent with Nesterov momentum updates the weight as follows:

$$v_{t+1} = \mu * v_t - \eta * g_t \tag{3.13}$$

where v_t is the velocity, g_t is the gradient of the loss function on the current mini batch at time t, μ is the momentum and η is the learning rate

$$w_{t+1} = w_t + \mu * v_{t+1} - \eta * g_t \tag{3.14}$$

where w_{t+1} is the new updated weight.

Momentum accelerates the gradient descent in the relevant direction and dampens oscillations. Velocity and momentum control how fast the velocity can change and how much the local gradient influences long term movement. The momentum is set at 0.9 and the learning rate is set at 0.01. The validation loss is also monitored and if the improvement is not over the threshold of 0.0001 for three consecutive epochs the learning rate is reduced by a factor of 0.2. Similarly, if the validation loss does not improve at all for 15 epochs then early stopping is performed.

3.5 Success Metrics

The experiments were evaluated based on the metrics from the validation. As aforementioned, every dataset has been divided into training datasets and validation datasets. If there was not an existing split in such categories, then an 80-20 split was made while maintaining the distribution of each class. The model was continuously trained and validated on their respective dataset for each epoch until early stopping was executed based on the parameters mentioned above in Subsection 3.4.1. The main success metrics based on validation are:

(1) Accuracy

(2) AUC

(3) Loss

(4) Epoch

which are discussed next.

3.5.1 Accuracy

Accuracy calculates how often our trained model predictions match the target one-hot encoding label. The argmax of logits of classes and probabilites are same, so the logits of the classes can be compared with the one-hot encoding label. The

categorical accuracy is computed as:

$$\text{Categorical accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{3.15}$$

In terms of positives and negatives, it can be also defined as:

$$\text{Categorical accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.16}$$

where TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negative.

True Positive is the total number of outcomes where the model correctly predicts the positive class, and similarly True Negative is the total number of outcomes where the model correctly predicts the negative class. False Positive defines the total number of outcomes where the model incorrectly predicts the positive class, and similarly False Negative defines the total number of outcomes where the model incorrectly predicts the negative class. The higher the accuracy, the better the classification model. It is expected that Gabor initialized CNNs will perform better in terms of accuracy than the classic/traditional CNNs.

3.5.2 AUC

ROC (Receiver operating characteristic) curve is a graph showing the performance of a classification model at all classification thresholds. ROC curve plots True Positive Rate (TPR, recall) vs False Positive Rate (FPR).

$$\text{True Positive Rate} = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{3.17}$$

$$\text{False Positive Rate} = \frac{\text{False Positive}}{\text{False Positive + True Negative}} \tag{3.18}$$

The Area under the curve (AUC) of the ROC curve provides an aggregate measure of performance across all possible classification thresholds. AUC gives the measure of ability of the model to distinguish between classes. Classes approximate AUCs using an approximation of an integral by a finite sum called Riemann sum. During metric accumulation, predictions are accumulated within predefined buckets by value. The AUC is then computed by interpolating per-bucket averages. These buckets define the evaluated operational points. True Positive, True Negative, False Positive and False Negative are then used to compute AUC. Since AUC can be computed at different epochs, the AUC at maximum accuracy epoch has been chosen for analysis. The higher the AUC, the better the classification model. It is expected that Gabor initialized CNNs will give better AUC in comparison to classic/traditional CNNs.

3.5.3 Loss

Loss is calculated between prediction and target labels using categorical cross-entropy and should be as low as as possible. Classification models tend to minimize the validation loss well, but it is expected that Gabor initialized CNNs will give even lower loss in comparison to classic/traditional CNNs.

3.5.4 Epoch

There is no defined number of epoch constraints imposed upon the learning model, so it can learn as long as it reaches the saturation point defined by the early stopping. Due to the lack of this constraint, various models can learn for a different number of epochs till they reach the point of convergence in terms of validation loss. The number of epochs the model took cannot be directly compared in between the Gabor initialized models and the traditional models because it could be true that one of the models was learning more epochs and was able to minimize the loss even more. Since the original hypothesis was to measure the impact of Gabor filters

on traditional CNNs, Gabor initialized CNNs could be constrained to the number of epochs defined by different metrics — particularly maximum validation accuracy and minimum validation loss — of traditional CNNs. Based on the these types of experiments, different metrics including number of epochs could give better insight of how well Gabor initialized models have performed.

3.5.4.1 Gabor initialized CNN constrained to the epoch of maximum accuracy of traditional CNN. The epoch at which the traditional CNN reaches its maximum accuracy can be taken and set as a hard limit for the Gabor initialized CNNs in regard to the maximum number of epochs that model can be trained. It is done by finding the epoch at which the traditional CNN obtained its maximum accuracy, say epoch X. The Gabor initialized CNN is now allowed to train until epoch X and based on success metrics like accuracy, AUC and epoch can be compared. Maximum accuracy and higher AUC are both desirable for the Gabor initialized CNN, as well as the maximum accuracy epoch being lower when compared to its counterpart — traditional CNN.

3.5.4.2 Gabor initialized CNN constrained to the epoch of minimum loss of traditional CNN. The epoch at which the traditional CNN reached its minimum loss can be taken and set as a hard limit for the Gabor initialized CNNs in regard to maximum number of epochs in which the model could be trained. Similar to Section 3.5.4.1 , different success metrics like loss and epoch could be compared. It is desired that both the minimum loss and minimum loss epoch of the Gabor initialized CNN be lower when compared to its counterpart - traditional CNN.

3.6 Experiments

There are different datasets differing in terms of nature, distribution, size, etc. and with regard to this different CNN models have been employed. The CNN models are not necessarily the perfect, ground-breaking models but they do represent

the classic CNN architecture which gave decent performance. A sufficient number of preliminary experiments were performed and the CNN architecture was refined iteratively. In order to have a holistic view of how Gabor filters affected CNN, different types of experiments were conducted.

3.6.1 Multiple Experiments with Same Gabor Size and Same Image Size

In order to be sure of the effect of Gabor filters on CNN, 30 different experiments were performed on the same dataset - 10 experiments for each type of initialization method described in Section 3.1. Gabor size was fixed at 15×15 because initial experiments showed this size to be better. Since there were different sizes of images on the original dataset, they were all resized to a specific size.

(1) Cats vs Dogs - 256×256

(2) CIFAR-10 - 128×128

(3) CIFAR-100 - 128×128

(4) Caltech 256 - 128×128

(5) Stanford cars - 128×128

(6) Tiny Imagenet - 128×128

A larger image size was chosen because CNN tends to perform better with larger images and Gabor filters tends to work better with larger images when compared with a smaller ones. Sizes beyonds 15×15 were not explored since it was computationally expensive. Since there were two different types of Gabor initialization methods, success metrics were calculated with respect to traditional CNNs for each case.

3.6.2 Rigid/Static Gabor Filters vs Trainable Gabor Filters

The Gabor filter itself is a great feature extractor, but CNN could make it better. In order to explore this hypothesis, experiments were designed to see how CNN did when it was allowed to change the Gabor filters compared to when the Gabor filters were statically placed and not allowed to change. For each type of

dataset, 10 different experiments were performed again, but this time the receptive layer of the Gabor initialized CNN was made rigid i.e. not allowing the Gabor filters to change its structure.

3.6.3 Different Gabor Size

Gabor filters could have an effect on the performance of CNN based on the filter's varying sizes. To explore this, Gabor filter of sizes 3, 5, 7, 9, 11, 13 and 15 were chosen for the both types of Gabor initialization method, and experiments were carried out with different datasets, where a success metric being calculated with respect to traditional CNN's random initialization method. Only odd-sized filters were chosen as it maintains the symmetry when encoding the information of the neighborhood during convolution.

This concludes the description of our experimental setup. In the next chapter, we will present and discuss the results of all the experiments.

CHAPTER FOUR

Results and Discussion

4.1 Multiple Experiments With Same Gabor Size and Same Image Size

A total of 10 different experiments were performed on each of the datasets with their respective CNN architecture and receptive convolutional layer kernel configuration, which is comprised of random initialization, Gabor filter randomly assigned to each channel and repeated Gabor filter on the three channels. It was made sure that each experiment used same dataset for training and validation purposes with respect to the kernel configuration.

4.1.1 Fully-trained Models

The presence of Gabor filters could push the Gabor configured models to learn even more. With this in mind, there was no restriction imposed on the number of epochs in which the model could be trained, with the only exception being the early stopping, which could be triggered due to lack of improvement for a certain number of epochs. On each experiment, the traditional (randomly initialized) CNN's maximum accuracy, AUC at maximum accuracy and minimum loss were directly compared with those of the Gabor filter configured ones and evaluated as shown by Tables 4.1, 4.2 and 4.3 respectively.

Table 4.1 shows that on average Gabor configured CNN tended to perform better than the traditional CNN in terms of accuracy. This can be particularly seen in the Cats vs dogs, CIFAR-10 and Stanford cars dataset. The low standard deviation on Cats vs dogs and CIFAR-10 dataset shows that Gabor configured models tend to give better and more consistent performance in terms of accuracy when the dataset is not complex. Additionally, it can be noticed that generally the repeated Gabor configuration performs slightly better than the random configuration, but this is not the

Table 4.1. Improvement in maximum accuracy of Gabor configured CNN with respect to traditional CNN

Dataset	Base maximum accuracy		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.8839	0.004	**+0.0233**	**0.007**	**+0.0263**	**0.006**
CIFAR-10	0.8024	0.004	**+0.0205**	**0.004**	**+0.0213**	**0.005**
CIFAR-100	0.7132	0.003	+0.0065	0.005	+0.0074	0.005
Caltech 256	0.5085	0.007	+0.0147	0.009	+0.0188	0.011
Stanford cars	0.2326	0.070	**+0.1294**	**0.072**	**+0.1625**	**0.072**
Tiny Imagenet	0.5175	0.004	+0.0133	0.003	+0.0003	0.007
Average	0.6097	0.015	**+0.0346**	**0.017**	**+0.0394**	**0.018**

Table 4.2. Improvement in AUC at maximum accuracy of Gabor configured CNN with respect to traditional CNN

Dataset	Base AUC		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.9515	0.003	**+0.0136**	**0.004**	**+0.0169**	**0.004**
CIFAR-10	0.9719	0.001	+0.0031	0.001	+0.0025	0.001
CIFAR-100	0.9621	0.002	+0.0013	0.002	+0.0015	0.002
Caltech 256	0.8885	0.004	+0.0076	0.005	+0.0040	0.005
Stanford cars	0.8077	0.026	**+0.0507**	**0.021**	**+0.0626**	**0.025**
Tiny Imagenet	0.9370	0.003	+0.0024	0.004	-0.0012	0.007
Average	0.9198	0.006	**+0.0131**	**0.006**	**+0.0144**	**0.007**

case when dataset complexity increases. In random Gabor filter configuration, there are multiple random Gabor filters spread out among the kernel set with each kernel filter corresponding to different channels of image. This increases the probability of the presence of a filter that is capable of extracting valuable features from the image. But the probability of such a filter is reduced in case of repeated Gabor filter configuration as a specific Gabor filter is assigned for all kernel filters within that set, i.e. the same Gabor filter will correspond to all three channels of image. Datasets like Cats vs dogs and CIFAR-10 are not that complex when compared to other multi-class datasets, thus few Gabor filters are quite capable of extracting necessary features. On these datasets, the repeated Gabor filter configuration performed better since the same Gabor filter will be used to extract information from all channels and thus giving

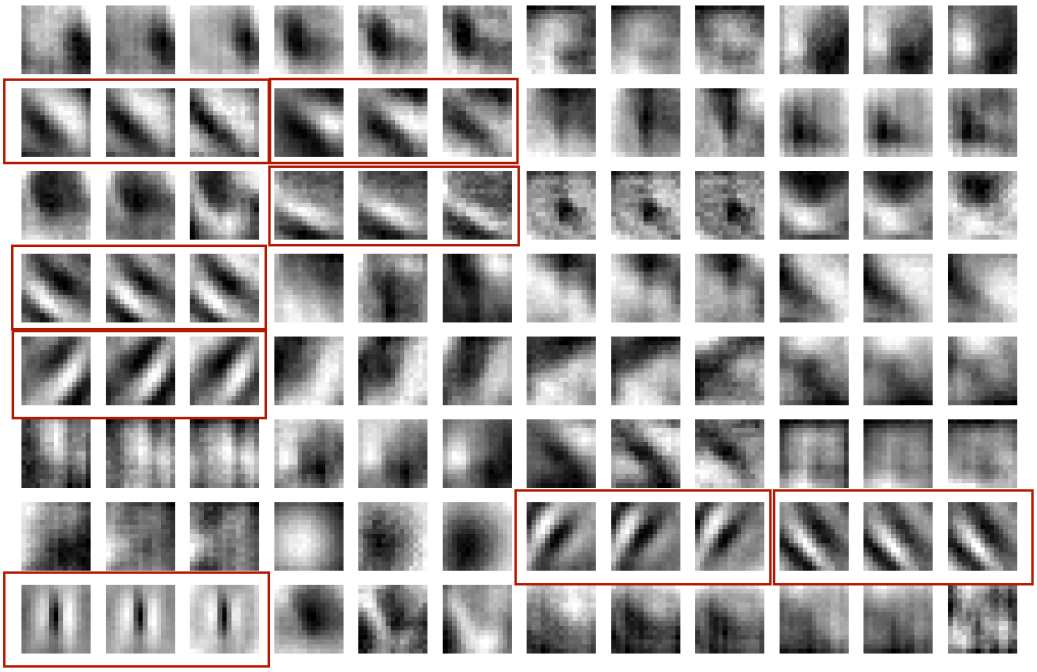
Table 4.3. Improvement in minimum loss of Gabor configured CNN with respect to traditional CNN

Dataset	Base minimum loss		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.2960	0.012	-0.0437	0.017	-0.0562	0.012
CIFAR-10	0.6555	0.013	-0.0544	0.015	-0.0570	0.017
CIFAR-100	1.1823	0.020	-0.0227	0.018	-0.0296	0.020
Caltech 256	2.6428	0.067	**-0.1041**	**0.078**	**-0.1030**	**0.065**
Stanford cars	4.1857	0.356	**-0.7812**	**0.291**	**-1.0398**	**0.360**
Tiny Imagenet	2.7390	0.014	-0.0528	0.024	-0.0037	0.027

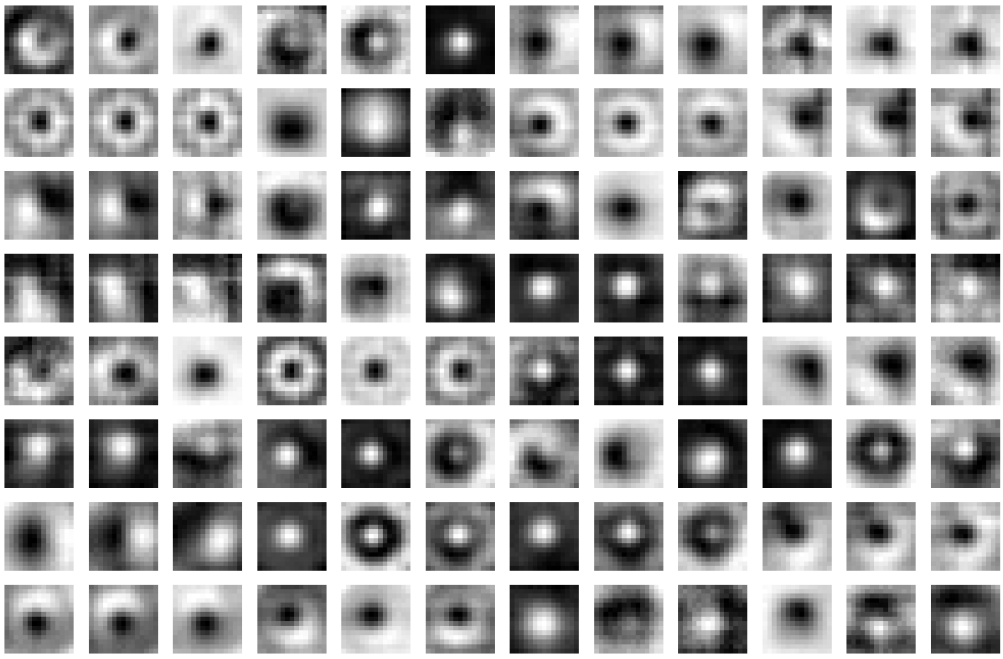
similar texture segmentation analysis for the succeeding layers to build upon. But, in case of random Gabor filter configuration the texture segmentation information varies between the channels, thus requiring some CNN adjustment.

When analyzing the kernel filters in receptive layer of fully trained traditional CNN, it was seen that with simple datasets like Cats vs Dogs the trained kernel filters tended to resemble the Gabor filters as shown by Figure 4.1a, which was not the case with complex datasets as shown by Figure 4.1b. This can explain the higher gain in performance on simpler datasets with a Gabor initialized model. The kernel filters of trained CNNs on complex datasets resemble with each other a lot, and they seem to be developing a form. It can be hypothesized that they were not able to train the kernel filters in the receptive layer to full capacity because the early stopping triggered and did not let the model train for a long enough period of time as desired. Upon closer analysis of the kernel filters in Figure 4.1a, it can be seen that the trained kernel filters corresponding to each channel of the image are similar, as seen in the kernel filters in the bounded red box which belong to the same set of kernel filters. While these kernel filters are not exactly the same, they seem to be extracting similar features from each channel, which explains why the repeated Gabor filter configuration performed slightly better than the random Gabor filter configuration.

Table 4.2 and 4.3 show the analysis of AUC at maximum accuracy and minimum loss and show similar analysis as that of the analysis of maximum accuracy



(a) Kernel filters in receptive layer of fully trained traditional CNN on Cats vs Dogs dataset.



(b) Kernel filters in receptive layer of fully trained traditional CNN on CIFAR-100 dataset.

Figure 4.1. Kernel filters in receptive layer of fully trained traditional CNN, where three consecutive filters belong to same kernel set

to Table 4.1. They both show that on average Gabor configured models tend to have higher AUC and lower minimum loss compared to that of traditional CNN, and that the repeated Gabor filter configuration has slightly better performance than the random Gabor filter configuration provided that the dataset is simple.

4.1.2 Gabor Initialized CNN Constrained to Maximum Accuracy Epoch

The analysis in Subsection 4.1.1 showed that the presence of Gabor filters allowed the CNN models to increase learning within limitations, depending upon the nature of the dataset. While this is desirable, it certainly leads to the question of how will the Gabor configuration models perform when their training periods are constrained to a certain defined number of epochs. With regard to that, the epoch at which the traditional CNN obtained its maximum accuracy during each experiment was taken, and it was defined as the number of epochs the Gabor configured model was allowed to train. On each experiment, the maximum accuracy, AUC at maximum accuracy and epoch at which the model obtained its maximum accuracy metrics of the traditional CNN were compared with that of Gabor configured models as shown by Tables 4.4, 4.5 and 4.6 respectively.

Table 4.4 and 4.5 show considerable improvement in the accuracy and AUC of the model even when the models are restricted to only a certain number of epochs

Table 4.4. Improvement in maximum accuracy of epoch-constrained Gabor initialized CNN with respect to traditional CNN when training period constrained to maximum accuracy epoch of traditional CNN

Dataset	Base maximum accuracy		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.8839	0.004	**+0.0212**	**0.007**	**+0.0253**	**0.006**
CIFAR-10	0.8024	0.004	**+0.0197**	**0.003**	**+0.0212**	**0.005**
CIFAR-100	0.7132	0.003	+0.0054	0.005	+0.0053	0.005
Caltech 256	0.5085	0.007	+0.0131	0.008	+0.0163	0.010
Stanford cars	0.2326	0.070	**+0.1200**	**0.065**	**+0.1576**	**0.068**
Tiny Imagenet	0.5175	0.004	+0.0128	0.003	-0.0008	0.007
Average	0.6097	0.015	**+0.0320**	**0.015**	**+0.0375**	**0.017**

Table 4.5. Improvement in AUC at maximum accuracy of epoch-constrained Gabor initialized CNN with respect to traditional CNN when training period constrained to maximum accuracy epoch of traditional CNN

Dataset	Base AUC		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.9515	0.003	+0.0129	0.004	+0.0164	0.004
CIFAR-10	0.9719	0.001	+0.0033	0.001	+0.0026	0.001
CIFAR-100	0.9621	0.002	+0.0013	0.002	+0.0022	0.002
Caltech 256	0.8885	0.004	+0.0086	0.004	+0.0062	0.005
Stanford cars	0.8077	0.026	**+0.0552**	**0.022**	**+0.0645**	**0.026**
Tiny Imagenet	0.9370	0.003	+0.0023	0.004	-0.0010	0.003
Average	0.9198	0.006	**+0.0134**	**0.006**	**+0.0151**	**0.007**

Table 4.6. Improvement in maximum accuracy epoch of epoch-constrained Gabor initialized CNN with respect to traditional CNN when training period constrained to maximum accuracy epoch of traditional CNN

Dataset	Base epoch		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	88.4	13.8	-5.2	5.4	**-17.6**	**15.2**
CIFAR-10	67.9	5.6	**-8.9**	**6.1**	**-6.6**	**3.6**
CIFAR-100	99.3	6.4	-4.3	2.9	-6.5	6.2
Caltech 256	73.3	5.4	-4.2	4.4	-5.7	4.0
Stanford cars	103.6	13.2	-5.7	5.1	-5.9	5.4
Tiny Imagenet	37.3	6.6	-6.1	5.7	-4.7	5.7

to train. Furthermore, Table 4.6 shows that the Gabor configured model is generally able to achieve that maximum accuracy in fewer epochs compared to traditional CNN. While the standard deviation in this epoch metric is not small, the presence of improvement in all metrics — accuracy, AUC and epoch till maximum accuracy — shows that Gabor configured models performs better and generally faster than the traditional CNN. The analysis in these tables also echoes the same sentiment that the repeated Gabor configuration tends to do better when the dataset is not complex.

4.1.3 Gabor Initialized CNN Constrained to Minimum Loss Epoch

Similar to Subsection 4.1.2, the Gabor configured models were trained only for a defined number of epochs, where this defined epoch was the epoch at which the traditional CNN obtained its minimum loss. On each experiment the metrics — minimum loss and the epoch at which the model obtained its minimum loss — were compared among the traditional CNN and Gabor configured model as shown by Tables 4.7 and 4.8 respectively. Table 4.7 shows that even when constrained to only a fixed number of epochs the Gabor configured model is able to decrease the loss to a greater extent, and the epoch metric in Table 4.8 shows that it is generally able to reach its minimum loss faster than the traditional CNN. Even with the presence of a slightly higher standard deviation in some cases, the Gabor configured model is able

Table 4.7. Improvement in minimum loss of Gabor initialized CNN with respect to traditional CNN when training period constrained to minimum loss epoch of traditional CNN

Dataset	Base minimum loss		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.2960	0.012	**-0.0406**	**0.015**	**-0.0553**	**0.013**
CIFAR-10	0.6555	0.013	**-0.0517**	**0.015**	**-0.0567**	**0.013**
CIFAR-100	1.1823	0.020	-0.0150	0.038	-0.0192	0.029
Caltech 256	2.6428	0.067	-0.0908	0.038	-0.0192	0.029
Stanford cars	4.1857	0.356	**-0.6513**	**0.231**	**-0.8913**	**0.264**
Tiny Imagenet	2.7390	0.014	-0.0522	0.024	-0.0027	0.028

Table 4.8. Improvement in minimum loss epoch of Gabor initialized CNN with respect to traditional CNN when training period constrained to minimum loss epoch of traditional CNN

Dataset	Base epoch		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	70.6	13.5	**-7**	**5.4**	**-14**	**9.8**
CIFAR-10	40.1	5.5	-8.6	8.1	**-10**	**7.4**
CIFAR-100	70.2	6.5	-6.2	3.3	**-8.8**	**7.7**
Caltech 256	42.1	5.1	-3.5	2.7	-5.2	3.5
Stanford cars	74.0	14.9	-5.1	4.4	-6.4	3.9
Tiny Imagenet	32.2	4.6	-5.2	5.6	-5.9	6.1

to perform better. Similar to previous analyses, the metrics in these tables shows that the repeated Gabor configured model is generally better than random Gabor configured model, provided that the dataset is not that complex.

4.2 Rigid/Static Gabor Filters vs Trainable Gabor Filters

While it can be seen from the analysis in Section 4.1 that the presence of Gabor filters definitely helps the CNN to perform better, it begs the question of whether the Gabor alone is sufficient for boosting the performance or if CNN plays a vital role in its correction. To figure out how the Gabor filters alone performed, in each of the CNN architecture the Gabor filters were made rigid/static i.e. in the a sense that the CNN was not allowed to alter the Gabor filters. With a rigid/frozen configuration, all the same type of experiments as in Section 4.1 were performed the same number of times. It is to be noted that while the Gabor filters were frozen the traditional CNN was not frozen in any of the cases. For the same reasoning as in Subsection 4.1.1, the presence of Gabor filters could push the Gabor configured models to learn even more. Therefore there was no restriction imposed upon the number of epochs the model was allowed to train, with early stopping being the only exception. On each experiment, the traditional CNN's maximum accuracy, AUC at maximum accuracy and minimum loss were directly compared with those of the Gabor filter configured ones and evaluated as shown by Tables 4.9, 4.10 and 4.11.

Table 4.9. Improvement in maximum accuracy of Gabor initialized CNN (frozen receptive convolutional layer variant) with respect to traditional CNN

Dataset	Base maximum accuracy		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.8839	0.004	+0.0029	0.009	**+0.0183**	**0.005**
CIFAR-10	0.8024	0.004	+0.0086	0.005	-0.0075	0.007
CIFAR-100	0.7132	0.003	+0.0022	0.004	-0.0559	0.007
Caltech 256	0.5085	0.007	+0.0079	0.011	+0.0012	0.012
Stanford cars	0.2326	0.070	**+0.0924**	**0.096**	**+0.1662**	**0.086**
Tiny Imagenet	0.5175	0.004	+0.0045	0.009	-0.0391	0.004
Average	0.6097	0.015	**+0.0197**	**0.022**	**+0.0139**	**0.020**

Table 4.10. Improvement in AUC of Gabor initialized CNN (frozen receptive convolutional layer variant) with respect to traditional CNN

Dataset	Base AUC		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.9515	0.003	+0.0020	0.006	**+0.0133**	**0.002**
CIFAR-10	0.9719	0.001	+0.0012	0.001	-0.0017	0.002
CIFAR-100	0.9621	0.002	-0.0003	0.003	-0.0095	0.002
Caltech 256	0.8885	0.004	+0.0052	0.007	+0.0048	0.006
Stanford cars	0.8077	0.026	**+0.0408**	**0.035**	**+0.0684**	**0.032**
Tiny Imagenet	0.9370	0.003	+0.0012	0.004	-0.0081	0.003
Average	0.9198	0.006	**+0.0083**	**0.009**	**+0.0112**	**0.008**

Table 4.11. Improvement in minimum loss of Gabor initialized CNN (frozen receptive convolutional layer variant) with respect to traditional CNN

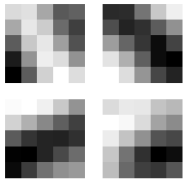
Dataset	Base minimum loss		Random Gabor filter		Repeated Gabor filter	
	Mean	Stdev	Mean	Stdev	Mean	Stdev
Cats vs Dogs	0.2960	0.012	-0.0100	0.018	-0.0475	0.010
CIFAR-10	0.6555	0.013	-0.0352	0.019	+0.0086	0.022
CIFAR-100	1.1823	0.020	-0.0099	0.035	+0.2437	0.037
Caltech 256	2.6428	0.067	-0.0794	0.091	-0.0466	0.068
Stanford cars	4.1857	0.356	**-0.6217**	**0.502**	**-1.0837**	**0.487**
Tiny Imagenet	2.7390	0.014	-0.240	0.027	+0.1628	0.019

As can be seen in Tables 4.9, 4.10 and 4.11, while random Gabor configuration was able to provide some improvement in the performance, the same cannot be said for the repeated Gabor configuration. Repeated Gabor configuration was able to do better in simple datasets like Cats vs Dogs, but as the complexity of dataset increased, the Gabor configuration got worse. This can be attributed to the fact that repeated Gabor configuration will have an even lower probability of having a particular Gabor filter which could extract the desired feature. When compared with the results in Table 4.1, 4.2 and 4.3 respectively, it is evident that both types of Gabor configured models performed fairly worse in terms of maximum accuracy, AUC at maximum accuracy and minimum loss. Thus, it provides some substantial evidence that, while a Gabor filter is certainly great at feature extraction unless carefully designed and

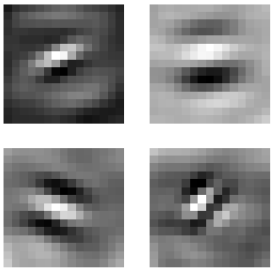
engineered to work with that specific dataset, it is certainly beneficial to let the CNN do the work and correct the filters as necessary.

4.3 Effect of Different Kernel Size and Image Size

Filters can have different sizes, as shown in Figure 4.2. The size of the image and kernel size could definitely impact the performance of the CNN and even the Gabor filter. While the initial experiments showed that comparatively larger Gabor filters performed well on large image sizes, it was needed to be seen on how it fared on different image sizes and different kernel sizes. With this question in mind, each of the CNN architecture was trained in its respective dataset with varying image size and kernel size. The maximum accuracy, AUC at maximum accuracy and minimum loss of traditional CNN was noted and compared with the Gabor configured models. All the models were allowed to train without any restriction on the number of epochs, except the early stopping restriction.



(a) 5×5 Gabor filter.



(b) 15×15 Gabor filter.

Figure 4.2. Gabor filters with different size

4.3.1 Cats vs Dogs

Each experiment with the Cats vs Dogs dataset differed in terms of image size and the kernel size for the receptive convolutional layer. Tables 4.12, 4.13 and 4.14 gives the analysis of improvement in maximum accuracy, AUC at maximum accuracy and minimum loss of Gabor configured models when compared with traditional CNN respectively.

After close inspection of Table 4.12, it can be seen that when the image size is 32×32 traditional CNN performed worse when the kernel size was increased linearly. This makes sense because the image size is already small, and as the kernel size increases, it starts to miss out on the details of some smaller features. But, as the image size was gradually increased, the accuracy started to increase to a considerable extent. The negative effect of linear increase in kernel size was also reduced to a considerable extent.

It can also be noticed that as the image size increased the effect of the Gabor filter came more into effect. With larger Gabor filters, there was more significant improvement when compared to smaller Gabor filters. This can be explained by the fact that on a small window size, the Gabor filter structure is not clear enough, and

Table 4.12. Improvement in maximum accuracy on Cats vs Dogs dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.8261	0.8303	0.8165	0.8143	0.8035	0.8037	0.7869
	Random Gabor (Δ)	-0.0202	-0.0389	-0.0114	+0.0036	**+0.0120**	+0.0044	+0.0094
	Repeated Gabor (Δ)	-0.0258	-0.0174	-0.0170	-0.0120	-0.0174	-0.0020	**+0.0090**
64×64	Traditional CNN (Base)	0.8015	0.8403	0.8381	0.8297	0.8425	0.8315	0.8279
	Random Gabor (Δ)	-0.0168	+0.0038	+0.0100	**+0.0132**	+0.0022	**+0.0128**	+0.0058
	Repeated Gabor (Δ)	+0.0126	-0.0070	**+0.0204**	+0.0162	+0.0044	+0.0116	**+0.0180**
128×128	Traditional CNN (Base)	0.8672	0.9026	0.8948	0.9022	0.8992	0.8804	0.8952
	Random Gabor (Δ)	+0.0062	-0.0138	-0.0022	+0.0114	+0.0150	**+0.0242**	+0.0150
	Repeated Gabor (Δ)	+0.0134	+0.0120	+0.0228	+0.0144	+0.0160	**+0.0341**	**+0.0216**
256×256	Traditional CNN (Base)	0.8932	0.8892	0.8926	0.8862	0.8924	0.8916	0.8870
	Random Gabor (Δ)	-0.0170	-0.0058	-0.0078	+0.0076	+0.0156	**+0.0214**	+0.0142
	Repeated Gabor (Δ)	-0.0214	+0.0120	+0.0142	**+0.0264**	+0.0136	+0.0170	**+0.0240**

Table 4.13. Improvement in AUC at maximum accuracy on Cats vs Dogs dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.9028	0.9092	0.8947	0.8946	0.8789	0.8809	0.8650
	Random Gabor (Δ)	-0.0161	-0.0391	-0.0076	-0.0012	**+0.0137**	+0.0024	**+0.0107**
	Repeated Gabor (Δ)	-0.0208	-0.0149	-0.0110	-0.0081	-0.0110	+0.0008	+0.0095
64×64	Traditional CNN (Base)	0.8900	0.9232	0.9213	0.9077	0.9214	0.9127	0.9097
	Random Gabor (Δ)	-0.0179	+0.0027	+0.0053	**+0.0103**	+0.0022	**+0.0113**	+0.0081
	Repeated Gabor (Δ)	+0.0090	-0.0066	+0.0127	**+0.0146**	+0.0037	+0.0116	**+0.0139**
128×128	Traditional CNN (Base)	0.9461	0.9690	0.9641	0.9670	0.9651	0.9557	0.9638
	Random Gabor (Δ)	+0.0032	-0.0094	-0.0028	+0.0071	+0.0093	**+0.0148**	+0.0094
	Repeated Gabor (Δ)	+0.0068	+0.0044	+0.0118	+0.0089	+0.0097	**+0.0188**	**+0.0103**
256×256	Traditional CNN (Base)	0.9586	0.9565	0.9602	0.9531	0.9570	0.9565	0.9541
	Random Gabor (Δ)	-0.0099	-0.0016	-0.0056	+0.0072	+0.0086	**+0.0139**	+0.0087
	Repeated Gabor (Δ)	-0.0129	+0.0054	+0.0086	+0.0178	+0.0085	+0.0121	**+0.0141**

Table 4.14. Improvement in minimum loss on Cats vs Dogs dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.7039	1.0208	1.3793	0.8991	0.8574	0.9765	1.0263
	Random Gabor (Δ)	-0.0630	-0.3510	**-0.7026**	-0.2184	-0.1522	-0.1801	+0.1596
	Repeated Gabor (Δ)	-0.0696	-0.3772	**-0.6521**	-0.2515	-0.1687	-0.1927	-0.1891
64×64	Traditional CNN (Base)	0.8884	0.9717	0.8448	0.9905	1.2597	1.3066	1.4466
	Random Gabor (Δ)	-0.1744	-0.2768	-0.1251	-0.3048	**-0.5674**	-0.4398	**-0.6611**
	Repeated Gabor (Δ)	-0.2145	-0.2895	-0.1689	-0.3397	**-0.5842**	**-0.6421**	**-0.6685**
128×128	Traditional CNN (Base)	1.0480	0.8813	1.1060	0.7639	0.8840	1.0305	1.3318
	Random Gabor (Δ)	-0.3270	-0.0753	-0.3405	-0.0858	-0.1525	-0.3765	**-0.5583**
	Repeated Gabor (Δ)	-0.4209	-0.2144	-0.4912	-0.2167	-0.2957	-0.3765	**-0.7697**
256×256	Traditional CNN (Base)	1.1261	0.6374	0.7055	0.7233	1.1426	0.8459	0.8025
	Random Gabor (Δ)	**-0.4575**	-0.0646	-0.0882	-0.0916	**-0.5824**	-0.2015	**-0.2225**
	Repeated Gabor (Δ)	**-0.4516**	-0.0561	+0.0252	-0.0221	**-0.5944**	-0.0720	-0.1276

because of this it could not fully extract desired features. Figure 4.2 shows that the Gabor filter is able to obtain full structure with larger window size, thus its better performance with larger sizes. But this does not mean that Gabor filter size is in linear relationship with the performance of CNN or that an increase in the Gabor filter size will always boost CNN's performance. Rather, it can be clearly seen that at some point as the Gabor filter increases in size, the gain in performance stagnates.

The analysis in Table 4.13 shows similar trend. On smaller image sizes, there was significant decrease in the AUC when the kernel size was increased linearly, but as the image size increased this was not the case. Also, larger Gabor filters tended to perform better compared to smaller ones. The analysis of minimum loss in Table 4.14 does not clearly show the trend as expected, but it seems to follow the same trend as seen on the 64×64 image size. As with increase in kernel size, the model does not seem to get better. However, with Gabor filters the CNN tend to learn better because even with higher minimum loss in traditional CNN, the Gabor configured model tends to perform better than its counterpart by decreasing the minimum loss even more.

4.3.2 CIFAR-10

Each experiment with the CIFAR-10 dataset differed in terms of image size and the kernel size for the receptive convolutional layer. Tables 4.15, 4.16, and 4.17 gives the analysis of improvement in maximum accuracy, AUC at maximum accuracy, and minimum loss of Gabor configured models when compared with traditional CNN respectively. When analysing the result corresponding to 32×32 image size in Table 4.15, it can be seen that the CNN gets worse as the kernel size is increased, but on increasing the size of the image, as in the case of 128×128, the kernel size does not

Table 4.15. Improvement in maximum accuracy on CIFAR-10 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.7818	0.7896	0.7929	0.7712	0.7713	0.7744	0.7654
	Random Gabor (Δ)	-0.0049	-0.0090	-0.0122	**+0.0143**	**+0.0124**	+0.0089	**+0.0101**
	Repeated Gabor (Δ)	-0.0037	-0.0028	-0.0087	**+0.0283**	+0.0164	**+0.0234**	+0.0155
64×64	Traditional CNN (Base)	0.7086	0.7257	0.7199	0.7207	0.7115	0.7203	0.7219
	Random Gabor (Δ)	-0.0076	-0.0129	+0.0077	+0.0143	+0.0279	**+0.0393**	**+0.0403**
	Repeated Gabor (Δ)	-0.0098	-0.0107	+0.0206	+0.0348	**+0.0466**	**+0.0416**	+0.0394
128×128	Traditional CNN (Base)	0.7936	0.7988	0.8007	0.7930	0.7989	0.8004	0.8067
	Random Gabor (Δ)	+0.0086	+0.0073	+0.0146	**+0.0258**	**+0.0228**	+0.0271	+0.0177
	Repeated Gabor (Δ)	+0.0093	+0.0113	+0.0134	+0.0273	+0.0281	+0.0199	+0.0142

Table 4.16. Improvement in AUC at maximum accuracy on CIFAR-10 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.9759	0.9773	0.9777	0.9744	0.9734	0.9737	0.9722
	Random Gabor (Δ)	-0.0011	-0.0011	-0.0027	+0.0018	+0.0023	+0.0019	+0.0018
	Repeated Gabor (Δ)	-0.0010	-0.0006	-0.0019	+0.0036	+0.0037	+0.0034	+0.0021
64×64	Traditional CNN (Base)	0.9575	0.9615	0.9606	0.9614	0.9598	0.9621	0.9623
	Random Gabor (Δ)	-0.0026	-0.0019	+0.0020	+0.0032	+0.0069	+0.0073	+0.0081
	Repeated Gabor (Δ)	-0.0018	-0.0006	+0.0050	+0.0080	**+0.0104**	+0.0086	+0.0076
128×128	Traditional CNN (Base)	0.9730	0.9724	0.9734	0.9725	0.9737	0.9733	0.9746
	Random Gabor (Δ)	+0.0008	+0.0017	+0.0011	+0.0023	+0.0023	+0.0047	+0.0033
	Repeated Gabor (Δ)	+0.0005	+0.0029	+0.0021	+0.0044	+0.0031	+0.0023	+0.0015

have great impact on the performance. The result of Gabor configured models shows that it tends to favor larger Gabor filters because as the kernel size increased the performance significantly improved. A similar trend was also observed earlier while presenting our analysis in Table 4.16 and Table 4.17; however, in that example the evidence shown was not as clear as in this case. AUC and loss depends upon the nature of datasets like size, distribution, etc. So, there is no defined correlation of those metrics with accuracy. For example with imbalanced datasets, there could be high accuracy but with low AUC. And, in case of complex datasets, there could be varying loss because of the magnitude of errors made on varying amount of data. If

Table 4.17. Improvement in minimum loss on CIFAR-10 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	1.4764	1.5682	1.9694	1.9144	1.6672	1.5935	2.1591
	Random Gabor (Δ)	-0.1391	-0.2193	**-0.5082**	-0.5611	-0.1535	-0.1015	**-0.6806**
	Repeated Gabor (Δ)	-0.0672	-0.1756	**-0.6604**	**-0.6354**	-0.0718	-0.1837	**-0.8144**
64×64	Traditional CNN (Base)	1.6460	1.9160	2.3001	1.6342	1.6378	1.8921	2.0575
	Random Gabor (Δ)	-0.0266	-0.3585	-0.6622	-0.0978	-0.1412	-0.2156	**-0.4911**
	Repeated Gabor (Δ)	+0.0670	-0.3258	-0.7804	-0.0624	-0.1956	**+0.3132**	**-0.3499**
128×128	Traditional CNN (Base)	1.4920	2.2684	1.2744	1.3457	1.3687	1.7287	1.4233
	Random Gabor (Δ)	-0.2609	**-1.1010**	-0.0477	-0.1184	-0.1824	-0.3236	-0.0045
	Repeated Gabor (Δ)	-0.2781	**-1.0944**	+0.2863	-0.1774	-0.2432	-0.1496	-0.0947

the model made little errors on a few data, there will be low loss, but it made huge errors on a few data, there will be huge loss.

4.3.3 CIFAR-100

Each experiment with the CIFAR-100 dataset differed in terms of image size and the kernel size for the receptive convolutional layer. Tables 4.18, 4.19, and 4.20 gives the analysis of improvement in maximum accuracy, AUC at maximum accuracy and minimum loss of Gabor configured models when compared with traditional CNN respectively. In the case of CIFAR-100 too, the result in Tables 4.18, Table 4.19 and Table 4.20 follows the same trend as discussed earlier. When compared to the Table

Table 4.18. Improvement in maximum accuracy on CIFAR-100 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.5842	0.5740	0.5854	0.5488	0.5605	0.5678	0.5590
	Random Gabor (Δ)	-0.0237	+0.0114	-0.0281	**+0.0192**	**+0.0201**	-0.0139	+0.0081
	Repeated Gabor (Δ)	-0.0189	+0.0003	-0.0021	+0.0023	+0.0004	-0.0086	-0.0029
64×64	Traditional CNN (Base)	0.6803	0.6869	0.6807	0.6866	0.6898	0.6886	0.6867
	Random Gabor (Δ)	+0.0007	+0.0025	-0.0007	-0.0015	-0.0087	-0.0094	-0.0015
	Repeated Gabor (Δ)	+0.0039	+0.0018	+0.0027	-0.0028	-0.0080	-0.0010	-0.0006
128×128	Traditional CNN (Base)	0.7144	0.7065	0.7162	0.7164	0.7138	0.7123	0.7112
	Random Gabor (Δ)	-0.0060	+0.0037	+0.0018	+0.0002	+0.0012	+0.0073	0.0059
	Repeated Gabor (Δ)	+0.0017	+0.0106	-0.0070	-0.0041	+0.0040	+0.0086	**+0.0145**

Table 4.19. Improvement in AUC at maximum accuracy on CIFAR-100 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.9550	0.9503	0.9530	0.9525	0.9511	0.9514	0.9512
	Random Gabor (Δ)	-0.0025	+0.0035	+0.0003	-0.0028	+0.0023	-0.0007	-0.0004
	Repeated Gabor (Δ)	-0.0036	+0.0018	-0.0006	+0.0025	-0.0019	+0.0020	-0.0006
64×64	Traditional CNN (Base)	0.9636	0.9652	0.9659	0.9628	0.9655	0.9643	0.9652
	Random Gabor (Δ)	+0.0008	+0.0002	-0.0009	+0.0030	-0.0006	+0.0025	+0.0007
	Repeated Gabor (Δ)	-0.0004	-0.0004	-0.0007	+0.0027	-0.0012	+0.0021	+0.0006
128×128	Traditional CNN (Base)	0.9694	0.9686	0.9684	0.9690	0.9682	0.9691	0.9682
	Random Gabor (Δ)	+0.0008	+0.0002	+0.0010	+0.0021	+0.0028	+0.0000	+0.0016
	Repeated Gabor (Δ)	+0.0002	+0.0011	+0.0030	+0.0013	+0.0010	+0.0007	+0.0029

Table 4.20. Improvement in minimum loss on CIFAR-100 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	4.3399	4.5608	4.0880	5.9439	4.1416	4.2599	5.1038
	Random Gabor (Δ)	+0.1598	-0.6657	-0.0989	**-1.7213**	+0.1809	-0.2785	+2.1429
	Repeated Gabor (Δ)	+0.4380	-0.4634	+0.7734	-1.5532	+0.5421	+0.2966	-1.1574
64×64	Traditional CNN (Base)	3.6348	3.7467	3.5715	3.8046	3.8158	4.0575	4.0832
	Random Gabor (Δ)	+0.1774	-0.0744	+0.0242	**-0.3521**	+0.2289	+0.1263	+0.2179
	Repeated Gabor (Δ)	+0.5789	+0.3015	+1.7995	-0.0274	+0.1685	+0.8609	+0.1275
128×128	Traditional CNN (Base)	3.4936	4.1385	4.1666	5.1151	3.7694	3.5885	4.0887
	Random Gabor (Δ)	+0.2320	-0.2233	-0.6036	**-1.3428**	+0.9635	+0.0513	-0.0090
	Repeated Gabor (Δ)	+0.4857	-0.2240	-0.1239	-0.6645	+0.0184	-0.0016	+0.1811

4.12 and Table 4.15, it can be seen the the accuracy gain is really subtle, which can be attributed to the increase in the complexity of the dataset, as the CIFAR-100 dataset is very complex compared to the CIFAR-10 and Cats vs Dogs dataset.

4.3.4 Caltech 256

Caltech 256 dataset was experimented with different image sizes with traditional CNN as well as Gabor configured CNN models, each experiment differed in terms of image size and the kernel size for the receptive convolutional layer. Tables 4.21, 4.22 and 4.23 gives the analysis of improvement in maximum accuracy, AUC at maximum accuracy and minimum loss of Gabor configured models when compared

Table 4.21. Improvement in maximum accuracy on caltech 256 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.3086	0.3084	0.3022	0.3096	0.3061	0.3099	0.2978
	Random Gabor (Δ)	-0.0007	+0.0002	**+0.0195**	**+0.0106**	+0.0064	+0.0010	+0.0008
	Repeated Gabor (Δ)	+0.0123	+0.0020	**+0.0146**	**+0.0115**	+0.0056	+0.0008	-0.0005
64×64	Traditional CNN (Base)	0.4296	0.4388	0.4375	0.4404	0.4403	0.4380	0.4313
	Random Gabor (Δ)	-0.0090	-0.0028	+0.0113	+0.0025	-0.0116	+0.0119	**+0.0214**
	Repeated Gabor (Δ)	+0.0039	-0.0054	+0.0082	+0.0072	**+0.0113**	+0.0059	+0.0059
128×128	Traditional CNN (Base)	0.5028	0.5025	0.5350	0.5200	0.5113	0.5195	0.5092
	Random Gabor (Δ)	+0.0151	+0.0208	+0.0008	+0.0026	**+0.0211**	+0.0061	+0.0041
	Repeated Gabor (Δ)	+0.0195	+0.0128	-0.0043	+0.0036	**+0.0188**	+0.0051	**+0.0198**

Table 4.22. Improvement in AUC at maximum accuracy on caltech 256 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.8481	0.8419	0.8513	0.8574	0.8454	0.8474	0.8392
	Random Gabor (Δ)	-0.0021	**+0.0162**	+0.0019	-0.0027	+0.0050	+0.0003	+0.0057
	Repeated Gabor (Δ)	+0.0055	+0.0095	-0.0030	+0.0005	**+0.0131**	+0.0037	+0.0069
64×64	Traditional CNN (Base)	0.8741	0.8853	0.8846	0.8853	0.8837	0.8848	0.8840
	Random Gabor (Δ)	+0.0026	-0.0037	+0.0036	+0.0033	-0.0010	+0.0060	-0.0001
	Repeated Gabor (Δ)	+0.0037	-0.0028	-0.0007	+0.0050	**+0.0114**	+0.0034	+0.0041
128×128	Traditional CNN (Base)	0.9034	0.9097	0.9062	0.9036	0.9046	0.9049	0.9021
	Random Gabor (Δ)	+0.0053	-0.0004	+0.0072	+0.0020	+0.0006	-0.0059	+0.0036
	Repeated Gabor (Δ)	+0.0050	+0.0002	+0.0010	+0.0028	+0.0045	+0.0044	+0.0054

Table 4.23. Improvement in minimum loss on caltech 256 dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	8.1892	6.2550	7.0410	7.8705	8.5046	11.1519	5.8391
	Random Gabor (Δ)	-2.5809	+0.6026	**-1.6581**	**-2.3008**	+0.8493	-0.4026	-0.3982
	Repeated Gabor (Δ)	-1.6821	+1.8756	-1.0074	-1.6438	-0.5016	**-5.2436**	+2.1425
64×64	Traditional CNN (Base)	6.3774	8.1425	6.8721	6.7314	7.0304	8.6386	5.2090
	Random Gabor (Δ)	-0.8742	**-3.0356**	-1.2754	-1.3800	-1.0103	**-2.7627**	+0.8800
	Repeated Gabor (Δ)	-1.1548	**-2.6945**	+3.0506	-0.1708	+5.0498	**-2.1193**	+0.7651
128×128	Traditional CNN (Base)	4.9090	5.1978	13.0624	6.8160	6.3426	7.1236	7.1915
	Random Gabor (Δ)	+0.3547	+0.1951	**-8.0269**	-1.6014	-1.3128	**-1.7678**	**-1.8235**
	Repeated Gabor (Δ)	+0.7467	+0.3598	**-7.7013**	-0.7981	-0.6349	+1.1454	**-1.4903**

with traditional CNN respectively. When looking at the result of the experiment, in general, it was found that as the complexity of dataset increased, traditional CNN itself had a hard time giving a satisfactory performance. But even in such a case, the Gabor configured model was able to bring some performance gain to some extent. Like in our previous experiments, a similar pattern was found in this experiment as well i.e. on larger image size traditional CNN performed better and also Gabor configured tended to improve its performance compared to traditional CNN. This is particularly evident when we look into the improvement in the maximum accuracy,

AUC at maximum accuracy and minimum loss of CNN models on 128 × 128 image size compared to that of CNN models on 32 × 32 and 64 × 64 image sizes.

4.3.5 Stanford Cars

Each experiment with the Stanford Cars dataset differed in terms of image size and the kernel size for the receptive convolutional layer. Tables 4.24, 4.25, and 4.26 gives the analysis of improvement in maximum accuracy, AUC at maximum accuracy and minimum loss of Gabor configured models when compared with traditional CNN respectively.

The performance of traditional CNN is the lowest for the Stanford cars, due to the fact that Stanford cars is a pretty complex dataset. To distinguish cars of different companies produced in different years is a hard job even for human beings, and the fact that the images represent information from only certain perspectives makes it worse. But, the effect of kernel and image size is fairly evident in this case. As can be seen in Tables 4.24 and 4.25, the accuracy and AUC decreased dramatically as the kernel size was increased. Traditional CNN performed better as the image size was increased, and with that the effect of Gabor filter was even more evident. Looking closely at the result of the 128 × 128 image size, it can be seen that as the Gabor

Table 4.24. Improvement in maximum accuracy on stanford cars dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.0493	0.0426	0.0442	0.0425	0.0304	0.0334	0.0300
	Random Gabor (Δ)	-0.0088	+0.0090	-0.0002	+0.0060	**+0.0133**	+0.0076	+0.0009
	Repeated Gabor (Δ)	+0.0051	+0.0024	+0.0004	+0.0059	**+0.0152**	**+0.0115**	**+0.0139**
64×64	Traditional CNN (Base)	0.1774	0.1602	0.1498	0.1350	0.1386	0.0818	0.1143
	Random Gabor (Δ)	-0.0330	+0.0081	+0.0015	+0.0019	-0.0162	**+0.0436**	-0.0009
	Repeated Gabor (Δ)	-0.0339	+0.0117	+0.0326	+0.0281	-0.0092	**+0.0524**	**+0.0410**
128×128	Traditional CNN (Base)	0.4103	0.3879	0.4180	0.3598	0.3010	0.3102	0.3517
	Random Gabor (Δ)	-0.0151	+0.0396	+0.0157	+0.0802	**+0.1398**	**+0.1930**	+0.0600
	Repeated Gabor (Δ)	-0.0818	+0.0274	+0.0005	+0.0029	**+0.1313**	+0.0788	+0.0648

Table 4.25. Improvement in AUC at maximum accuracy on stanford cars dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.7107	0.6970	0.7114	0.6907	0.6290	0.6427	0.6325
	Random Gabor (Δ)	-0.0198	+0.0019	-0.0198	+0.0009	**+0.0526**	**+0.0292**	+0.0041
	Repeated Gabor (Δ)	+0.0111	-0.0038	-0.0106	+0.0173	**+0.0705**	+0.0448	**+0.0568**
64×64	Traditional CNN (Base)	0.8211	0.8046	0.7911	0.7815	0.7713	0.7255	0.7472
	Random Gabor (Δ)	-0.0173	-0.0063	+0.0018	+0.0030	-0.0056	**+0.0358**	+0.0146
	Repeated Gabor (Δ)	-0.0150	-0.0046	+0.0154	+0.0165	+0.0045	**+0.0528**	**+0.0388**
128×128	Traditional CNN (Base)	0.8736	0.8831	0.8723	0.8808	0.8369	0.8344	0.8811
	Random Gabor (Δ)	+0.0020	+0.0020	+0.0180	+0.0033	**+0.0455**	**+0.0458**	+0.0032
	Repeated Gabor (Δ)	+0.0063	+0.0017	+0.0028	+0.0030	**+0.0515**	+0.0278	+0.0035

Table 4.26. Improvement in minimum loss on stanford cars dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	7.3203	47.5599	8.8750	9.2702	22.8231	7.1648	9.1182
	Random Gabor (Δ)	+12.3424	**-37.8715**	+3.6937	+3.4808	**-15.7108**	+0.3171	+0.4810
	Repeated Gabor (Δ)	+7.1468	**-37.3148**	-2.6179	+6.2107	-4.7338	+5.3186	+7.1548
64×64	Traditional CNN (Base)	24.1874	7.3460	14.2081	10.4780	16.0974	17.0614	20.6991
	Random Gabor (Δ)	**-13.0682**	+1.0896	**-6.3144**	+1.3075	-3.0181	-4.1803	-0.5565
	Repeated Gabor (Δ)	**-16.0293**	+13.3526	+13.5375	+2.7611	-1.1857	-0.2307	-3.6890
128×128	Traditional CNN (Base)	18.8136	36.2230	18.1727	6.6971	6.5915	73.8066	6.7081
	Random Gabor (Δ)	-1.3699	**-26.1808**	-9.9487	+9.0980	+4.3474	**-66.8044**	+7.0920
	Repeated Gabor (Δ)	-4.3315	**-23.0504**	-9.2398	+4.6752	+34.7256	**-62.0934**	+4.2712

filter size is increased, it begins to positively impact the CNN even more, exceeding the performance of a traditional CNN.

In terms of minimum loss, traditional CNN was not predictable. However, it can be noticed in Table 4.26 that on average Gabor configured models tended to improve the minimum loss, and they did so even when the traditional model was not performing as well as usual. Although the experiment was a single run, the presence of evidence in multiple cases suggest that with a Gabor filter CNN is able to perform better.

4.3.6 Tiny Imagenet

The Tiny Imagenet dataset is comprised of a diverse range of complex objects, which becomes challenging for a particular set of Gabor filters to capture the features from all the images. This can be seen from the results in Table 4.27 and 4.28. With image sizes of 32×32 and 64×64, most of the time the Gabor configured models did not perform well, especially the repeated Gabor configured model. However, on 128×128 image size, the random Gabor configured model was able to improve performance on average. Looking at all these results, it is evident that smaller image size doesn't work well for the Gabor filter and even traditional CNN. Furthermore, performance of models begins to wind down on smaller image size, when kernel size is increased linearly and the Gabor filter is also not able to impact the CNN. But when the image size is increased, the Gabor filter tends to bring improvement. While this is not true for every kernel size, it does tend to follow when the kernel size is comparatively larger. Performance is not predictable for complex datasets, but such cases with injection of Gabor filters brought a great deal of improvement. The results in Table 4.26 and 4.23 show that when the traditional CNN performed poorly in terms of loss, Gabor filters pushed the model to learn even more, thus bringing some sort of consistency in terms of loss, as the loss decreased significantly under same configuration for Gabor filter configured models.

Table 4.27. Improvement in maximum accuracy on Tiny Imagenet dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.3921	0.3950	0.3832	0.3712	0.3671	0.3649	0.3543
	Random Gabor (Δ)	-0.0077	-0.0029	-0.0419	-0.0223	-0.0294	-0.0340	-0.0083
	Repeated Gabor (Δ)	-0.0050	-0.0465	-0.0462	-0.0453	-0.0401	-0.0612	-0.0410
64×64	Traditional CNN (Base)	0.4806	0.4824	0.4739	0.4699	0.4659	0.4662	0.4562
	Random Gabor (Δ)	+0.0102	-0.0021	-0.0041	-0.0072	-0.0102	+0.0002	**+0.0152**
	Repeated Gabor (Δ)	-0.0037	-0.0390	-0.0186	+0.0004	-0.0244	-0.0229	-0.0021
128×128	Traditional CNN (Base)	0.5199	0.5233	0.5241	0.5216	0.5229	0.5218	0.5104
	Random Gabor (Δ)	+0.0113	+0.0056	+0.0081	-0.0031	-0.0018	+0.0056	**+0.0170**
	Repeated Gabor (Δ)	-0.0066	-0.0153	-0.0411	-0.0126	-0.0142	-0.0099	+0.0060

Table 4.28. Improvement in AUC at maximum accuracy on Tiny Imagenet dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	0.9109	0.9113	0.9062	0.9056	0.9011	0.8991	0.8979
	Random Gabor (Δ)	-0.0002	-0.0054	-0.0118	-0.0109	-0.0079	-0.0098	-0.0006
	Repeated Gabor (Δ)	-0.0038	-0.0181	-0.0082	-0.0163	-0.0103	-0.0172	-0.0175
64×64	Traditional CNN (Base)	0.9361	0.9319	0.9333	0.9312	0.9308	0.9294	0.9262
	Random Gabor (Δ)	-0.0031	+0.0008	-0.0026	+0.0002	-0.0033	-0.0006	+0.0037
	Repeated Gabor (Δ)	-0.0093	-0.0050	-0.0094	-0.0021	-0.0039	-0.0078	-0.0018
128×128	Traditional CNN (Base)	0.9435	0.9418	0.9438	0.9432	0.9439	0.9428	0.9427
	Random Gabor (Δ)	-0.0006	+0.0011	-0.0013	-0.0017	-0.0014	-0.0009	+0.0013
	Repeated Gabor (Δ)	-0.0065	-0.0068	-0.0091	-0.0089	-0.0061	-0.0056	-0.0023

Table 4.29. Improvement in minimum loss on Tiny Imagenet dataset with different kernel size and image size

Image size	Gabor configuration	Kernel size						
		3×3	5×5	7×7	9×9	11×11	13×13	15×15
32×32	Traditional CNN (Base)	5.2912	5.2273	5.2322	5.1488	5.1689	5.2050	5.1729
	Random Gabor (Δ)	**-0.2901**	-0.2618	-0.2595	-0.2248	-0.1753	-0.1808	-0.1660
	Repeated Gabor (Δ)	-0.0636	-0.0202	-0.0505	-0.0025	-0.0429	-0.0107	-0.0384
64×64	Traditional CNN (Base)	5.1014	5.1428	5.1198	5.1246	5.0847	5.1234	5.0616
	Random Gabor (Δ)	-0.2692	**-0.2719**	**-0.2538**	-0.2102	**-0.2546**	-0.1730	-0.1027
	Repeated Gabor (Δ)	+0.1317	+0.0625	-0.0146	-0.0464	+0.0509	-0.0955	-0.0103
128×128	Traditional CNN (Base)	5.0659	5.0616	5.0584	5.0257	5.0092	5.0673	5.2985
	Random Gabor (Δ)	-0.2046	-0.2492	-0.3288	-0.1116	+0.0950	-0.0409	**-0.5205**
	Repeated Gabor (Δ)	+0.1017	+0.0927	+0.0865	+0.0545	+0.0739	-0.0492	**-0.3941**

CHAPTER FIVE

Conclusion

Over the decade, CNN has emerged in popularity, which has led to significant strides in image processing. Being a specialized type of Neural Network, CNN works with the receptive layer by extracting features from the provided image, and builds upon those features to narrow down the classification. This particular process of CNN takes a significant amount of time because it has to learn through backpropagation algorithms. Therefore, the model's accuracy depends on how well it can learn from the training images and the initial state of the CNN's simulation. One popular technique for initializing CNN layers is randomization, where random numbers are generated and assigned as the weights for the neural network. While this is undoubtedly a good technique given the uncertainty in the nature of the image the CNN will be processing, it can take a toll upon the performance of CNN.

In image processing, people have also been increasingly working with a special filter called a Gabor filter for several years. Past studies have shown it to be an excellent feature extractor. Given this nature of a Gabor filter, it can be hypothesized that it could act as a suitable receptive filter for CNN because previous research has found that the receptive filters of CNN tend to resemble Gabor filters. Compared to previous studies (Luan, Chen, Zhang, Han, and Liu 2018; Alekseev and Bobe 2019; Molaei, Shiri, Horan, Kahrobaei, Nallamothu, and Najarian 2017), an extensive range of analysis was done with a wide variety of general object datasets, but with a rather simplistic approach of unrestricted Gabor filter initialization in receptive layer. Tables 4.1, 4.2 and 4.3 show that with the presence of Gabor filters in receptive layers, there is improvement in the performance of CNN as it is able to obtain higher accuracy, higher AUC and lower loss for different datasets. From this, we can conclude that it brings

significant improvement in the case of general object classification. Furthermore, under restriction upon training epoch, it was also found that when CNN is trained in the same number of epochs on the training datasets, the CNN can achieve higher performance with Gabor filters in a shorter amount of time, on average.

Different types of Gabor filters can be generated with varying hyper-parameters like the orientation, wavelength, etc., which corresponds to different features in images. While there is no universal Gabor filter bank capable of extracting all features from the image, building a bank of different Gabor filters ranging over the different hyper-parameters is necessary. The configuration of the Gabor filters in the receptive convolutional layer also impacted the performance. It was found that repeated Gabor filter configuration performed better when the dataset was less complex, because similar filters in all channels were capable of extracting features without any modification from CNN itself. But, as the complexity of the dataset grew, the probability of the presence of such capable filters decreased compared to random Gabor filter configuration. Thus comparatively, the random Gabor filter configuration performed better. To increase the performance, it is ideally recommended to design the Gabor filter manually, which would extract all sorts of features from an image so that the CNN could build upon it. But, since such manual expertise is hard to gain, it is recommended instead to generate random Gabor filters and assign either of the configuration types based upon the complexity of the datasets, and let the CNN train/alter the Gabor filters as needed. It was found that CNN does play a vital role because when restricting the change in the structure of the Gabor filters during training period, the performance decreased comparatively and got even worse than traditional CNN as the complexity of the dataset increased.

The size of Gabor filters can also have a significant impact on the performance of CNNs. CNNs, with or without Gabor filters, perform worse than expected on smaller images because minute details get blurred out. In the case of larger images,

the size of the Gabor filter also needs to be considered. It was discovered that most of the time smaller Gabor filters performed worse than their larger counterparts, irrespective of datasets, because smaller Gabor filters do not have a distinguished shape and structure compared to their larger counterparts. While this does not suggest using very large Gabor filters because it could potentially miss out on smaller details, the size should not be small like 3×3 or 5×5 as well. The size of the Gabor filter should be optimal, which allows to extract necessary features. Since a single experiment was performed for each case, more research could likely help discover the range of the size within which the positive impact of Gabor filters can be seen.

5.1 Future Work

Much research could be extended from this research. The hyper-parameters of Gabor filters can be tuned as per necessary for the low performing CNN to see how much Gabor filters could affect the performance of CNN. Also, while it may be suitable for some specific datasets to tune manually, when considering general object recognition it is certainly desirable to find a range for the hyper-parameters which could produce high-performing Gabor filters. There have not been many explorations of the hyper-parameter space of Gabor filters, which is certainly research to look forward to. Also, for all the experiments only the receptive convolutional layers were initialized with Gabor filters. So, it might be interesting to see how it fares when multiple deeper convolutional layers were initialized in a similar manner.

Similar to Gabor filters, there are other filters like Log-Gabor filters, Gaussian filters, etc. which researchers have been using for image processing. Multiple studies could be conducted to see their impact on the CNN in multiple fashions, such as using those filters solely or combining them together. Finally, in this research, the traditional CNN was kept as a base model, which is a simple CNN without any modification to the convolutional layers structure. With the popularity of CNN, a lot

of variants have emerged. In similar manner to traditional CNN, experiments could be conducted with all the different variants.

As the complexity of datasets grew, the early stopping impacted CNN even more. When analysing the trained filters in traditional CNN, the resemblance between Gabor filters and trained filters decreased significantly when compared to the trained filters in the traditional CNN, trained upon simpler datasets. So, to address this issue, the patience in early stopping could be increased to the degree where trained CNN's filters bear some resemblance to Gabor-like filters, if present, and see how the CNN performed with Gabor filters that were given a long training time. If resources are limited, then if possible parallelization programs, e.g., (Sanjel 2020), could certainly be explored to expedite the process.

APPENDICES

APPENDIX A

Code

A.1 Gabor Filters Generation

```
import math
import numpy as np


def get_gabor_filters(inchannels, outchannels, kernel_size = (3,3)):
    delta = 1e−4
    freqs = (math.pi/2)∗(math.sqrt(2)∗∗(−np.random.randint(0,5, (outchannels, inchannels))))
    thetas = (math.pi/8)∗np.random.randint(0,8, (outchannels, inchannels))
    sigmas = math.pi/(freqs)
    psis = math.pi ∗ np.random.rand(outchannels, inchannels)
    x0, y0 = np.ceil(np.array(kernel_size)/2)


    y, x = np.meshgrid(
            np.linspace(−x0 + 1, x0 + 0, kernel_size[0]),
            np.linspace(−y0 + 1, y0 + 0, kernel_size[1]),
    )
    filterbank = []
    for i in range(outchannels):
        for j in range(inchannels):
            freq = freqs[i][j]
            theta = thetas[i][j]
            sigma = sigmas[i][j]
            psi = psis[i][j]


            rotx = x ∗ np.cos(theta) + y ∗ np.sin(theta)
            roty = −x ∗ np.sin(theta) + y ∗ np.cos(theta)


            g = np.exp(
```

```
                    −0.5 ∗ ((rotx ∗∗ 2 + roty ∗∗ 2) / (sigma + delta) ∗∗ 2)
                )
                g = g ∗ np.cos(freq ∗ rotx + psi)
                filterbank.append(g)
    return filterbank

filterbank = get_gabor_filters(3, NUM_RECEPTIVE_FILTERS, GABOR_SIZE)
```

A.2 Visualization of Gabor Filters

```
import matplotlib.pyplot as plt
filterbank = get_gabor_filters(3, NUM_RECEPTIVE_FILTERS, GABOR_SIZE)


fig = plt.subplots(8, len(filterbank)//8, figsize=(22,22))
for i,gf in enumerate(filterbank):
    plt.subplot(8, len(filterbank)//8, i+1)
    plt.imshow(gf, cmap='gray')
    plt.axis('off')
```

A.3 Random Gabor Filter on All Channels of Receptive Convolutional Layer

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Activation
from tensorflow.keras.layers import Input, Dense, Dropout, BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop, Adam
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau


classifier = Sequential([
    layers.Conv2D(NUM_RECEPTIVE_FILTERS, kernel_size=GABOR_SIZE, strides=(1,1),
        ↪ name="GaborLayer", input_shape=train_generator.image_shape, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
```

```
    layers.MaxPooling2D(pool_size=(2,2)),
    Dropout(0.5),
    layers.Conv2D(64, kernel_size=(3,3), strides=(2,2), padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    Dropout(0.5),
    layers.Conv2D(128, kernel_size=(3,3), strides=(2,2), padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    Dropout(0.5),
    layers.Flatten(),
    layers.Dense(256),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    Dropout(0.5),
    layers.Dense(NUM_CLASSES, activation='softmax')
])


cnnl1 = classifier.layers[GABOR_LAYER_INDEX].name # get the name of the first conv layer
W = classifier.get_layer(name=cnnl1).get_weights()[0] #get the filters
wshape = W.shape #save the original shape
gabor_filters = W
for kernel_index in range(wshape[3]):
    for channel_index in range(3):
        gabor_filters[:,:,channel_index, kernel_index] = filterbank[kernel_index+channel_index]
//Placing the randomly generated Gabor filters randomly in the channels
classifier.get_layer(name=cnnl1).set_weights([gabor_filters, classifier.get_layer(name=cnnl1).
    ↪ get_weights()[1]])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=10,
                              min_delta=1e−4, mode='min', verbose=1)
stop_alg = EarlyStopping(monitor='val_loss', patience=35,
```

```
                                restore_best_weights=True, verbose=1)
callbacks = [stop_alg, reduce_lr]
opt = Adam(learning_rate=0.001)
classifier.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy', 'AUC'])

hist = classifier.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=total_validate//BATCH_SIZE,
    steps_per_epoch=total_train//BATCH_SIZE,
    callbacks=callbacks
)
```

A.4 Repeated Gabor Filter on the 3 Channels of Receptive Convolutional Layer

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Activation
from tensorflow.keras.layers import Input, Dense, Dropout, BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop, Adam
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

classifier = Sequential([
    layers.Conv2D(NUM_RECEPTIVE_FILTERS, kernel_size=GABOR_SIZE, strides=(1,1),
        ↪ name="GaborLayer", input_shape=train_generator.image_shape, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    Dropout(0.5),
    layers.Conv2D(64, kernel_size=(3,3), strides=(2,2), padding='same'),
    layers.BatchNormalization(),
```

```
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    Dropout(0.5),
    layers.Conv2D(128, kernel_size=(3,3), strides=(2,2), padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    Dropout(0.5),
    layers.Flatten(),
    layers.Dense(256),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    Dropout(0.5),
    layers.Dense(NUM_CLASSES, activation='softmax')
])

cnnl1 = classifier.layers[GABOR_LAYER_INDEX].name # get the name of the first conv layer
W = classifier.get_layer(name=cnnl1).get_weights()[0] #get the filters
wshape = W.shape #save the original shape
gabor_filters = W
for kernel_index in range(wshape[3]):
    for channel_index in range(3):
        gabor_filters[:,:,channel_index, kernel_index] = filterbank[kernel_index]
//Placing the same randomly generated Gabor filters on all three channels
classifier.get_layer(name=cnnl1).set_weights([gabor_filters, classifier.get_layer(name=cnnl1).
    ↪ get_weights()[1]])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=10,
                              min_delta=1e−4, mode='min', verbose=1)
stop_alg = EarlyStopping(monitor='val_loss', patience=35,
                         restore_best_weights=True, verbose=1)
callbacks = [stop_alg, reduce_lr]
opt = Adam(learning_rate=0.001)
classifier.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy', 'AUC'])
```

```
hist = classifier.fit(
    train_generator,
    epochs=EPOCHS,
    validation_data=validation_generator,
    validation_steps=total_validate//BATCH_SIZE,
    steps_per_epoch=total_train//BATCH_SIZE,
    callbacks=callbacks
)
```

BIBLIOGRAPHY

Ahn, B. (2015). Real-time video object recognition using convolutional neural network. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7.

Alekseev, A. and A. Bobe (2019). Gabornet: Gabor filters with learnable parameters in deep convolutional neural network. In *2019 International Conference on Engineering and Telecommunication (EnT)*, pp. 1–4.

Anwar, S., K. Hwang, and W. Sung (2015). Fixed point optimization of deep convolutional neural networks for object recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1131–1135.

Avinash, S., K. Manjunath, and S. S. Kumar (2016). An improved image processing analysis for the detection of lung cancer using gabor filters and watershed segmentation technique. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, Volume 3, pp. 1–6.

Cao, Y., Y. Chen, and D. Khosla (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision 113*, 54–66.

Chacon, M. and P. Rivas (2009). Face recognition based on human visual perception theories and unsupervised ann. In *State of the Art in Face Recognition*. IntechOpen.

Chen, Z., O. Lam, A. Jacobson, and M. Milford (2014). Convolutional neural network-based place recognition.

Ciresan, D. C., U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber (2011). Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Daamouche, A., D. Fares, I. Maalem, and K. Zemmouri (2016). Unsupervised method for building detection using gabor filters. In *Special issue of the 2nd International Conference on Computational and Experimental Science and Engineering (ICCESEN 2015)*, Volume 130.

Daugman, J. (1985). Uncertainty relation for resolution in space, spatial frequency and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Amer. A. 2*.

Dosovitskiy, A., J. T. Springenberg, M. Riedmiller, and T. Brox (2014). Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems 27*, pp. 766–774. Curran Associates, Inc.

Du, Y.-C., M. Muslikhin, T.-H. Hsieh, and M.-S. Wang (2020). Stereo vision-based object recognition and manipulation by regions with convolutional neural network. *Electronics 2020 9*.

Dunn, D. and W. E. Higgins (1995). Optimal gabor filters for texture segmentation. *IEEE Transactions on Image Processing 4*(7), 947–964.

Dunn, D., W. E. Higgins, and J. Wakeley (1994). Texture segmentation using 2-d gabor elementary functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*(2), 130–149.

El-Sayed, M. A., M. Hassaballah, and M. A. Abdel-Latif (2016). Identity verification of individuals based on retinal features using gabor filters and svm. *Journal of Signal and Information Processing 7*.

Elson, J., J. J. Douceur, J. Howell, and J. Saul (2007, October). Asirra: A captcha that exploits interest-aligned manual image categorization. In *Proceedings of 14th ACM Conference on Computer and Communications Security (CCS)* (Proceedings of 14th ACM Conference on Computer and Communications Security (CCS) ed.). Association for Computing Machinery, Inc.

Fan, Z., S. Zhang, J. Mei, and M. Liu (2017). Recognition of woven fabric based on image processing and gabor filters. In *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 996–1000.

Fang, W., L. Ding, B. Zhong, P. E. Love, and H. Luo (2018). Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach. *Advanced Engineering Informatics 37*, 139 – 149.

Gabor, D. (1946). Theory of communication. *J. Inst. Elec. Eng. (London) 93*, 429–457.

Gao, Z., D. Wang, Y. Xue, G. Xu, H. Zhang, and Y. Wang (2018). 3d object recognition based on pairwise multi-view convolutional neural networks. *Journal of Visual Communication and Image Representation 56*, 305 – 315.

Garcia-Garcia, A., F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez (2016). Pointnet: A 3d convolutional neural network for real-time object class recognition. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1584.

Glorot, X. and Y. Bengio (2010, 13–15 May). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington (Eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Volume 9 of *Proceedings of Machine Learning Research*, Chia Laguna Resort, Sardinia, Italy, pp. 249–256. PMLR.

Gornale, S., A. Patil, and V. C. (2016). Fingerprint based gender identification using discrete wavelet transform and gabor filters. *International Journal of Computer Applications 152*(4).

Griffin, G., A. Holub, and P. Perona (2007, 03). Caltech-256 object category dataset. *CalTech Report*.

Hayat, S., S. Kun, Z. Tengtao, Y. Yu, T. Tu, and Y. Du (2018). A deep learning framework using convolutional neural network for multi-class object recognition. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 194–198.

He, K., X. Zhang, S. Ren, and J. Sun (2016, June). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hemalatha, G. and C. P. Sumathi (2016). Preprocessing techniques of facial image with median and gabor filters. In *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, pp. 1–6.

Hosseini, S., S. H. Lee, H. J. Kwon, H. I. Koo, and N. I. Cho (2018). Age and gender classification using wide convolutional neural network and gabor filter. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pp. 1–3.

Hubel, D. H. and T. N. Wiesel (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology 160*(1), 106–154.

Ioffe, S. and C. Szegedy (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Ishii, T., R. Nakamura, H. Nakada, Y. Mochizuki, and H. Ishikawa (2015). Surface object recognition with cnn and svm in landsat 8 images. In *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, pp. 341–344.

Jain, A. K., N. K. Ratha, and S. Lakshmanan (1997). Object detection using gabor filters. *Pattern Recognition 30*(2), 295 – 309.

Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, New York, NY, USA, pp. 675–678. Association for Computing Machinery.

Jiang, C. and J. Su (2018). Gabor binary layer in convolutional neural networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 3408–3412.

Jing, J., X. Fang, and P. Li (2016). Automated fabric defect detection based on multiple gabor filters and kpca. *International Journal of Multimedia and Ubiquitous Engineering 11*(6), 93–106.

Jing Huang and Suya You (2016). Point cloud labeling using 3d convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2670–2675.

Kataoka, H., K. Iwata, and Y. Satoh (2015). Feature evaluation of deep convolutional neural networks for object recognition and detection.

Kawano, Y. and K. Yanai (2014). Food image recognition with deep convolutional features. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, New York, NY, USA, pp. 589–593. Association for Computing Machinery.

Khaleefah, S. H., S. A. Mostafa, A. Mustapha, and M. F. Nasrudin (2019). The ideal effect of gabor filters and uniform local binary pattern combinations on deformed scanned paper images. *Journal of King Saud University - Computer and Information Sciences*.

Kingma, D. P. and J. Ba (2017). Adam: A method for stochastic optimization.

Krause, J., M. Stark, J. Deng, and L. Fei-Fei (2013). 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc.

Kumar, A. and G. K. H. Pang (2002). Defect detection in textured materials using gabor filters. *IEEE Transactions on Industry Applications 38*(2), 425–440.

Kumar, A. S. and E. Sherly (2017). A convolutional neural network for visual object recognition in marine sector. In *2017 2nd International Conference for Convergence in Technology (I2CT)*, pp. 304–307.

Lawrence, S., C. L. Giles, Ah Chung Tsoi, and A. D. Back (1997). Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks 8*(1), 98–113.

Le Cun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1990). Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pp. 396–404. Morgan Kaufmann.

Lefkovits, S., L. Lefkovits, and S. Emerich (2017). Detecting the eye and its openness with gabor filters. In *2017 5th International Symposium on Digital Forensic and Security (ISDFS)*, pp. 1–5.

Lei, Z., M. Pietikäinen, and S. Z. Li (2014). Learning discriminant face descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence 36*(2), 289–302.

Li, Z., H. Ma, and Z. Liu (2016). Road lane detection with gabor filters. In *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, pp. 436–440.

Liu, C., W. Ding, X. Wang, and B. Zhang (2018). Hybrid gabor convolutional networks. *Pattern Recognition Letters 116*, 164 – 169.

Liu, X., J. B. Lao, and J. S. Pang (2019). Feature point matching based on distinct wavelength phase congruency and log-gabor filters in infrared and visible images. *Sensors 19*.

Low, C., A. B. Teoh, and C. Ng (2016). Multi-fold gabor filter convolution descriptor for face recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2094–2098.

Lu, J., V. E. Liong, X. Zhou, and J. Zhou (2015). Learning compact binary face descriptor for face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 37*(10), 2041–2056.

Luan, S., C. Chen, B. Zhang, J. Han, and J. Liu (2018). Gabor convolutional networks. *IEEE Transactions on Image Processing 27*(9), 4357–4366.

Mahmood, M., A. Jalal, and H. A. Evans (2018). Facial expression recognition in image sequences using 1d transform and gabor wavelet transform. In *2018 International Conference on Applied and Engineering Mathematics (ICAEM)*, pp. 1–6.

Maturana, D. and S. Scherer (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928.

Meshgini, S., A. Aghagolzadeh, and H. Seyedarabi (2012). Face recognition using gabor filter bank, kernel principle component analysis and support vector machine. *International Journal of Computer Theory and Engineering*, 767–771.

Molaei, S., M. Shiri, K. Horan, D. Kahrobaei, B. Nallamothu, and K. Najarian (2017). Deep convolutional neural networks for left ventricle segmentation. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 668–671.

Molaei, S. and M. E. Shiri Ahmad Abadi (2020). Maintaining filter structure: A gabor-based convolutional neural network for image analysis. *Applied Soft Computing 88*, 105960.

Nava, R., B. Escalante-Ramirez, and G. Cristobal (2012). Texture image retrieval based on log-gabor features. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Appplications*, Volume 7441, pp. 414–421.

Nunes, C. F. G. and F. L. C. Pádua (2017). A local feature descriptor based on log-gabor filters for keypoint matching in multispectral images. *IEEE Geoscience and Remote Sensing Letters 14*(10), 1850–1854.

Premana, A., A. P. Wijaya, and M. A. Soeleman (2017). Image segmentation using gabor filter and k-means clustering method. In *2017 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pp. 95–99.

Pumlumchiak, T. and S. Vittayakorn (2017). Facial expression recognition using local gabor filters and pca plus lda. In *2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 1–6.

Radovic, M., O. Adarkwa, and Q. Wang (2017). Object recognition in aerial images using convolutional neural networks. *Journal of Imaging 3*.

Rai, M. and P. Rivas (2020). A review of convolutional neural networks and gabor filters in object recognition[1]. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 1560–1567.

Rizvi, S. T. H., G. Cabodi, P. Gusmao, and G. Francini (2016). Gabor filter based image representation for object classification. In *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 628–632.

Ruder, S. (2017). An overview of gradient descent optimization algorithms.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*(3), 211–252.

[1]

Sanjel, A. (2020). *Tyro: A First Step Towards Automatically Generating Parallel Programs from Sequential Programs*. Ph. D. thesis. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-05-11.

Schwarz, M., H. Schulz, and S. Behnke (2015). Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1329–1335.

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

Srivastava, G. and R. Srivastava (2019). Salient object detection using background subtraction, gabor filters, objectness and minimum directional backgroundness. *Journal of Visual Communication and Image Representation 62*, 330 – 339.

Sudharshan, D. P. and S. Raj (2018). Object recognition in images using convolutional neural network. In *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, pp. 718–722.

Sutskever, I., J. Martens, G. Dahl, and G. Hinton (2013, 17–19 Jun). On the importance of initialization and momentum in deep learning. In S. Dasgupta and D. McAllester (Eds.), *Proceedings of the 30th International Conference on Machine Learning*, Volume 28 of *Proceedings of Machine Learning Research*, Atlanta, Georgia, USA, pp. 1139–1147. PMLR.

Szarvas, M., A. Yoshizawa, M. Yamamoto, and J. Ogata (2005). Pedestrian detection with convolutional neural networks. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pp. 224–229.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015, June). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Taghi Zadeh, M. M., M. Imani, and B. Majidi (2019). Fast facial emotion recognition using convolutional neural networks and gabor filters. In *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pp. 577–581.

Viola, P. and M. Jones (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Volume 1, pp. I–I.

Wang, J., J. Lu, W. Chen, and X. Wu (2015). Convolutional neural network for 3d object recognition based on rgb-d dataset. In *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 34–39.

Wu, K., E. Wu, and G. Kreiman (2018). Learning scene gist with convolutional neural networks to improve object recognition. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6.

Xu, X., A. Dehghani, D. Corrigan, S. Caulfield, and D. Moloney (2016). Convolutional neural network for 3d object recognition using volumetric representation. In *2016 First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE)*, pp. 1–5.

Yao, H., L. Chuyi, H. Dan, and Y. Weiyu (2016). Gabor feature based convolutional neural network for object recognition in natural scene. In *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, pp. 386–390.

Zhi, S., Y. Liu, X. Li, and Y. Guo (2017). Lightnet: A lightweight 3d convolutional neural network for real-time 3d object recognition. In *Proceedings of the Workshop on 3D Object Retrieval*, 3Dor '17, Goslar, DEU, pp. 9–16. Eurographics Association.

Zulkeflie, S. A., F. A. Fammy, Z. Ibrahim, and N. Sabri (2019). Evaluation of basic convolutional neural network, alexnet and bag of features for indoor object recognition. *International Journal of Machine Learning and Computing 9*.