Modeling Information Flow Through Deep Convolutional Neural Networks

by

Behnaz NASIRI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN ELECTRICAL ENGINEERING
M.A.Sc.

MONTREAL, 9 APRIL 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Dr. Matthew Toews, Thesis Supervisor
Department of Systems Engineering at the École de Technologie Supérieure

Dr. Eric Granger, Co-Supervisor
Department of Systems Engineering at the École de Technologie Supérieure

Dr. Vincent Levesque, President of the Board of Examiners
Department of Software Engineering and IT at the École de Technologie Supérieure

Dr. Vincent Duchaine, External Examiner
Department of Systems Engineering at the École de Technologie Supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON 17 MARCH 2020

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I would like to acknowledge everyone who played a role in my academic accomplishments. The committee members, each of whom has provided patient advice and guidance throughout the research process. I would like to express my gratitude specially to my supervisor Dr. Matthew Toews for providing his invaluable guidance, comments and suggestions throughout the project. Also I would like to thanks Dr. Eric Granger my Co-supervisor and Dr. Marco Pedersoli for their consultations. Moreover I like to show my special gratefulness to Dr. Ahmad Chaddad for his help in evaluating the CENT approach with 3D brain MRI classification.

Furthermore I would like to thank my parents and my friends who supported me with love and understanding. Without you, I could never have reached this current level of success. Thank you all for your unwavering support.

Modélisation du flux d'informations à travers les réseaux de neurones profond

Behnaz NASIRI

RÉSUMÉ

Dans ce travail, nous étudions des méthodes permettant d'optimiser les réseaux de neurones convolutionnels profonds en 1) réduisant la complexité des calculs et 2) en améliorant les performances de classification en utilisant l'apprentissage par transfert. Le CNN est modélisé comme une chaîne de Markov, où la sortie du filtre au niveau d'une couche est conditionnellement indépendante du reste du réseau, à partir d'un ensemble de couches précédentes. La théorie de l'information est ensuite utilisée pour quantifier le flux d'informations d'image à travers le réseau. Les réponses de filtre avec une entropie conditionnelle faible (CENT) se sont révélées très efficaces pour la classification des images, pour le diagnostic assisté par ordinateur de la maladie d'Alzheimer dans les images de résonance magnétique 3D (IRM) du cerveau humain et pour divers objets sur des photographies naturelles.

Mots-clés: Apprentissage Automatique, Réseau de Neurones Convolutionnels, Classification, Apprentissage par Transfert

Modeling Information Flow
Through Deep Convolutional Neural Networks

Behnaz NASIRI

ABSTRACT

In this work we investigate methods for optimizing deep convolutional neural networks (CNN) by 1) reducing the computational complexity and 2) improving classification performance for the task of transfer learning. Based on the work of Chaddad *et al.* (2019, 2017), the CNN is modeled as a Markov chain, where the filter output at a layer is conditionally independent of the rest of the network, given a set of previous layers. Filter banks at each layer are compressed using principal component analysis (PCA), where a reduced set of orthogonal basis filters are used to reduce the number of convolutions required while preserving classification accuracy. Information theory is then used to quantify the flow of image information through the network. Filter responses with low conditional entropy (CENT) are shown to be highly effective in image classification, and can be used as generic features for effective, noise resistant transfer learning. CENT feature analysis is demonstrated in various contexts including computer-assisted diagnosis of Alzheimer's disease (AD) from 3D magnetic resonance images (MRI) of the human brain, and object classification in 2D photographs.

Keywords: Machine Learning, Convolutional Neural Networks, Classification, Transfer Learning, Principal Component Analysis, Information Theory, Conditional Entropy

TABLE OF CONTENTS

Page

INTRODUCTION 1

CHAPTER 1 RELATED WORK 7
1.1 The Neural Networks 7
1.1.1 Deep Learning in Neural Network 11
1.1.1.1 Deep Learning strength and challenges 12
1.1.2 The Convolutional Neural Network 13
1.1.2.1 Pooling 16
1.1.2.2 Data Augmentation and Regularization 17
1.1.2.3 Dropout 18
1.1.2.4 Batch Normalization 19
1.1.3 Several Common CNN Architectures 20
1.1.3.1 ResNet 20
1.1.3.2 DenseNet 23
1.1.3.3 Recurrent Neural Network 24
1.1.3.4 U-net Convolutional Network for Segmentation 25
1.2 Transfer Learning 26
1.3 Methods for Complexity Reduction 29
1.3.1 Low Rank methods 29
1.3.2 Weight Compression 35
1.3.3 Sparse Convolutions 36
1.3.4 Vector Quantization 38
1.3.5 Hashing 39
1.3.6 Pruning 40
1.4 Computation vs. Memory 42
1.5 Information Theory 44
1.5.1 Entropy 45
1.5.2 Joint and Conditional Entropy 46
1.5.3 Mutual Information 48

CHAPTER 2 METHODOLOGY 51
2.1 Bayesian Probabilistic Model of Deep Convolutional Neural Networks 52
2.1.1 Information Theory Analysis in Deep Convolutional Neural Networks 56
2.2 Using Principle Component to Reduce CNN Computation 59
2.2.1 Linear Subspace Model 60

CHAPTER 3 EXPERIMENTS 63
3.1 CENT Analysis 64
3.1.1 Analysis of CENT features efficiency using transfer learning 65

3.1.2 2D Classification of Visual Object Classes 68
3.1.2.1 Classification of 10 categories not used in VGG 69
3.1.2.2 Histogram of information and entropy 73
3.1.2.3 2D classification of two painting category 75
3.1.2.4 2D classification of Alzheimer's Disease vs. Healthy subject 79
3.1.2.5 2D classification of easily fooled classes 82
3.1.3 3D Image Classification: Brain MRIs 88
3.1.3.1 3D CNN Architecture 89
3.1.3.2 Alzheimer's Disease vs. Healthy Brains 91
3.1.3.3 Young vs. Old Brains 95
3.2 Classification performance over CNN reconstructed by principal component 98

CONCLUSION AND RECOMMENDATIONS 105

APPENDIX I METHODOLOGY WAVELET 109

APPENDIX II METHODOLOGY PCA 121

APPENDIX III RESULTS 123

BIBLIOGRAPHY 145

LIST OF TABLES

Page

Table 1.1 Overview of memory the number of parameters in each layer of VGG 43

Table 3.1 Overview of computation saving and changes of error rate for, K = 40 for VGG convolution layer = 1, and K = 150 to 250 for VGG convolution layer = 2, 3, 4, 5 (considering that the original error rate for VGG is 0.3883) 103

LIST OF FIGURES

Page

Figure 0.1 Caltech 101 data-set 2

Figure 0.2 Probability distributions over two different filter responses Y_1 and Y_2 conditioned on a matched filter F_1 (pink flower, left) and (flower) and an unrelated filter (black wheel, right) $F2$. class C a) shows high output response $MaxY_1$ for an informative filter F_1 with low entropy b) shows Low output response $MaxY_2$ for an uninformative filter F_2 with high entropy 4

Figure 1.1 (a) Biological neuron, (b) Mathematical model of neuron 8

Figure 1.2 Feed Forward Neural Network 9

Figure 1.3 Max pooling operation with 2x2 filter and stride 2 17

Figure 1.4 Drop out neural network model 19

Figure 1.5 Building Block of ResNet 21

Figure 1.6 Two kind of blocks in ResNet 22

Figure 1.7 ResNet Structure 22

Figure 1.8 DenseNet Block with the growth rate k = 4 23

Figure 1.9 RNN diagram 24

Figure 1.10 The learning process in transfer learning 27

Figure 1.11 Overview of memory usage in Conv and FC layers of VGG 43

Figure 1.12 Number of parameters in Conv and FC layers of VGG 44

Figure 2.1 The Bayesian probability diagram of VGG which modelizes the non-linear convolution filters F_i as conditional probability $p(Y|C, F_i)$ over the input data Y arising from an object of class C 54

Figure 2.2 The non-linear filter F operation modelization as a conditional probability $p(Y \mid C, F)$ over incoming data Y arising from an object of class C 55

Figure 3.1 21-Layer VGG CNN architecture with convolutional layer (includes ReLu/LRN/Max Pooling) followed by three fully connected layer and at the end one soft-max classification layer a) Illustrates computation of CENT features from feature maps/output for classification b) Illustrates the standard CNN soft-max output used in comparison 67

Figure 3.2 Ten unseen categories in VGG from left to right 1) Riopelle Painting, 2) Pollock Painting, 3) buddha, 4) chandelier, 5) lotus, 6) metronome, 7) minaret, 8) snoopy, 9) stapler, 10) yin yang 70

Figure 3.3 ROC curves transfer learning: classification of 10 natural objects not used in original CNN training. Each image is represented by (a) CENT features computed layer-wise across 5 CNN convolutional layers and (b) the 1000-feature CNN soft-max output 71

Figure 3.4 Histogram : Histogram of VGG CENT features used in RF for the class of metronome. The blue line shows the average of votes in each layer. The red light related to the average of conditional entropy for RF training per each layer 73

Figure 3.5 Scatter plot of entropy vs. the number of votes The least used features in RF have been displayed with a red circle 74

Figure 3.6 Scatter plot of entropy vs. the number of votes The features with less than 0.004 for normalized number of votes has been deleted from the graph 75

Figure 3.7 Sample painting of (a) Riopelle in the left side and (b) pollock painting in the right side 76

Figure 3.8 ROC curves transfer learning: Binary classification for Pollock and Riopelle painting using RF classifier; Each curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output 77

Figure 3.9 Alleged Pollock painting 77

Figure 3.10 Histogram : Histograms of 2 group of painting not used in the original CNN training. Each bin is related to the a filter and shows its importance for classification, The colors indicate each convolution layer, The blue line shows the average of histogram for each layer 79

Figure 3.11 Coronal slices of 3D brain MRI illustrating a case of Alzheimer's disease (AD, left) and a healthy control subject (HC, right) A hallmark of AD is atrophy of the cortical surface (red arrow) surrounding the hippocampus, a neuroanatomical structure intimately linked short-term memory formation 80

Figure 3.12 ROC curves transfer learning: Binary classification for brain images diseased with Alzheimer and the healthy using RF classifier. Each curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output 81

Figure 3.13 Histograms : Histograms of brain images diseased with Alzheimer and the healthy brain not used in the original CNN training. Each bin is related to a filter and shows its importance for classification. The colors indicate each convolution layer. The blue line shows the average of histogram for each layer 82

Figure 3.14 The sample images of two categories missclassified with strawberry, bell pepper with noise in the left side and the strawberry with the noise on the right side 83

Figure 3.15 ROC curves transfer learning: classification of two categories bell pepper and strawberries. The data has been trained with the original images and the test sets are synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output 84

Figure 3.16 ROC curves transfer learning: Binary classification for strawberry and bellpepper with noise (for both test and training set) using RF classifier. Each curve is represented by (Red) CENT features computed layer-wise across 5 CNN convolutional layers and (Blue) the 1000-feature CNN soft-max output 85

Figure 3.17 ROC curves transfer learning: Binary classification for strawberry and bellpepper, only bellpepper images with noise (for both test and training set) using RF classifier. Each curve is represented by (Red) CENT features computed layer-wise across 5 CNN convolutional layers and (Blue) the 1000-feature CNN soft-max output 86

Figure 3.18 Histograms : Histogram of 2 group of bellpepper where one group has been synthesized with noise (VGG classifier recognized it as strawberry) and the other is the original. Each bin is represented

number of times each filter were determinitive for classification across all CNN layers for classification across all CNN layers. The bottom histogram split the bins of each layer by different color 88

Figure 3.19 4-layer CNN architecture with convolutional layers (includes ReLU/subsampling/max pooling) and followed by 1 fully connected layer and finally 1 soft-max classification layer a) Represent the classification results from CENT features computed from feature maps output using RF b) The results from standard CNN soft-max output 90

Figure 3.20 Response distributions $p(Y \mid C, f_i)$ for 10 convolutional features maps in layer 2, note that the vertical axis is displayed in logarithmic units 92

Figure 3.21 (a) ROC curves for AD vs. HC classification using layer-wise CENT features (b) Scatter plot of 3 CENT features showing clear separation between AD and HC classes, each axes denotes to conditional entropy computed at the each CNN layers..... 93

Figure 3.22 ROC curves for AD vs. HC classification using filter-wise CENT features, with 10 filters in layer 1 (blue curves), 10 filters in layer 2 (red curves) and 1 filter in layer 3 (green curve) As in layer-wise CENT classification of Alzheimer's, the most informative features are generally found in layer 2, and highest classification is obtained by combined all features (black curve) 95

Figure 3.23 (a) ROC curve for CENT feature classification of young vs. old subjects (b) 3D plot of per-layer conditional entropy showing clear class separation 97

Figure 3.24 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 1, filters = 64 99

Figure 3.25 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 2, filters = 256.....100

Figure 3.26 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 3, filters = 256.....100

Figure 3.27 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 4, filters = 256.....101

Figure 3.28 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 5, filters = 256.....101

Figure 3.29 Graph of classification error rate vs. number of principal components used to represent filters: VGG Last four convolution layer, for the first Conv layer K = 40 102

Figure 3.30 Graph of computation saving vs. number of principal components used to represent filters: VGG last four layer, for the first Conv layer K = 40 104

LIST OF ABREVIATIONS

CNN	Convolutional Neural Network
MLP	Multi-Layer Perceptron
GPU	Graphics Processing Units
UPS	Unsupervised Pretrained Network
LCN	Local Contrast Normalization
LRN	Local Response Normalization
ReLu	Rectified Linear Unit
BN	Batch Normalization
ResNet	Deep Residual Learning [Conv] Convolutional Layer
DNN	Deep Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
FS	Filter Summary
SVD	Singular Value Decomposition
KFC	Kronocker Factor for Convolutional
LCNN	Lookup-based Convolution Neural Network
SCNN	Sparse Convolutional Neural Network
SSL	Structured Sparsity Learning

SIFT	Scale-Invariant Feature Transform
FreshNet	Frequency-Sensitive Hashed Net
DCT	Discrete Cosine Transform
GAN	Generative Adversarial Network
CENT	Conditional Entropy
AD	Alzheimer's Disease
OASIS	Open Access Series of Imaging Studies
MRI	Magnetic Resonance Imaging
SGD	Stochastic Gradient Descent
ERM	Empirical Error Minimization
MI	Mutual Information
IB	Information Bottleneck
ML	Maximum Likelihood
MAP	Maximum a Posterior
SVD	Singular Value Decomposition
FFT	Fast Fourier Transfrom
DAN	Deep Adaptation Network
DCFNet	Decomposed Convolutional Filters Network
RF	Random Forest
ETS	École De Technologie Supérieure

DWT	Discrete Wavelet Transform
DRAM	dynamic random access memory

INTRODUCTION

The deep convolutional neural network (CNN) architecture has achieved state-of-the-art in image classification Krizhevsky *et al.* (2012). The CNN operates via a layered image filtering structure LeCun *et al.* (1990), where banks of translation-invariant convolution filters are separated by non-linear rectifier units and sub-sampling via max pooling. The multi-layered structure allows the network to represent image patterns as heirarchical combinations of image filter responses, where filters are learned from labeled data samples, typically via the backpropagation algorithm Rumelhart *et al.* (1988).

The CNN can be viewed as generalizing traditional vision systems, where filter banks are specified manually (e.g. Wavelets in Viola-Jones face detection Wang (2014), Jensen (2008), Laplacian-of-Gaussian or orientated gradient filters Lowe (2004b)) or learned from data (e.g. principal component analysis Turk & Pentland (1991a) Turk & Pentland (1991b), independent component analysis Hyvärinen *et al.* (2004)). Traditional vision systems typically employ a shallow network structure, with a single feature extraction phase, followed by classification, (e.g. random forest, support vector machine).

Rapid growth of images over internet and social networks, brings several challenges to manage and classifying them. Measuring similarities between large scale images and train classifiers for new classes with no label is the examples of these challenges Guo (2017). To cope with the classification challenge, computer vision algorithms require to be trained by a large and varied set of training data. Large visual databases has been designed to use in computer vision research and image recognition classification. ImageNet Deng *et al.* (2009) is one example of a large scale data-set for which an annual software contest, ImageNet Large Scale Visual Recognition Challenge (ILSVRC), has been held. Figure 0.1 shows pictures of objects belonging to Caltech 101 data-set used for testing recognition algorithms.

Figure 0.1 Caltech 101 data-set

Although the deep convolutional structure and the ability to learn features make the CNN highly effective for image classification, a major challenge is the computational complexity. In fact, the CNN itself was invented in 1989 LeCun *et al.* (1998) to reduce the computational complexity of the Multi-Layer Perceptron (MLP) Rosenblatt (1958). However was only after the development of high throughput graphics processing units (GPU) that it was possible to train CNN classifiers for large-scale image recognition, e.g. 1000 object categories in the ImageNet database Krizhevsky *et al.* (2012), and demonstrate a major improvement over traditional shallow network classification systems.

A significant focus on the computer vision community is thus investigating methods of 1) reducing the CNN computational complexity and 2) improving classification performance using transfer learning. Typical approaches consist of compressing or removing convolutional filters. This thesis summarizes this work, and experiments with various methods, including filter compression using principal component analysis (PCA) Jolliffe (2011), wavelet methods for filtering Torrence & Compo (1998), and efficient transfer learning using information theory.

The primary contribution of this thesis is a novel information theoretical analysis leading to highly efficient transfer learning using convolutional neural networks. Our approach is to model the flow of image information through the deep convolutional neural network structure as a Markov process, where the filter output is represented as random variable Y defined by a probability distribution $p(Y|C, F\ldots)$ over the filter response Y conditional on the data label C and the filter set F. The information theoretic measure of conditional entropy can then be used to quantify the information content of the output of filters or layers, and can be used as an informative feature to summarize large feature maps and in certain cases to improve classification. This work proposes a novel image feature set based on a principled information theoretic analysis of the convolutional neural network (CNN) based on the work of Chaddad *et al.* (2019, 2017). It shows that conditional entropy (CENT) of filter outputs is a highly compact and class-informative feature in theory and experiments. It shows that using CENT features used to obtain higher classification accuracy than the original CNN itself.

The information theoretic analysis in this thesis follows from Tishby & Zaslavsky (2015), where optimal information theoretic limits of the deep neural network (DNN) are quantified by the mutual information between the layers and input and input and output variables. In this thesis, develop a novel encoding of CNN filter responses use information, using the conditional entropy. Although we investigate this encoding through through the CNN, the theory applies to feed forward DNNs in general.

The theory of Matched FiltersTurin (1960) predicts that for the purpose of image pattern detection, in the case of an additive zero mean noise model, the optimal filter will resemble the content of the pattern to be detected, and produce a convolution maximum in response . Here we demonstrate that the conditional entropy can be used to quantify the degree to which filters are informative or 'matched' to the image content of a given object category. Figure 0.2 a) and b) show a visual examples as to how the filtering response distributions $p(Y|C, F)$ vary in the case

of filters F_1, a matched filter (flower) and F_2, not matched filter (wheel) that are informative and uninformative, respectively, with respect to the object class C.

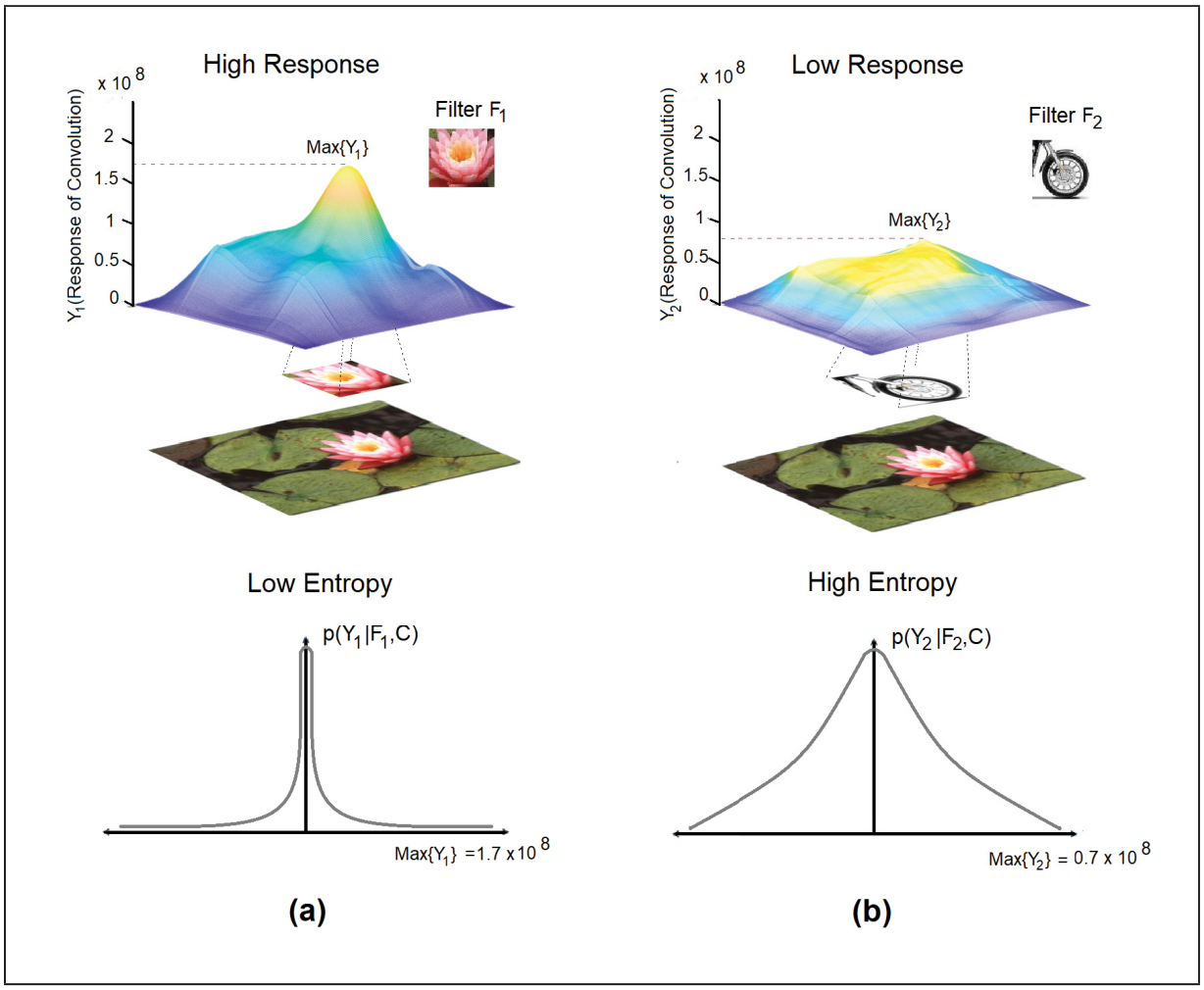


Figure 0.2 Probability distributions over two different filter responses Y_1 and Y_2 conditioned on a matched filter F_1 (pink flower, left) and (flower) and an unrelated filter (black wheel, right) $F2$. class C a) shows high output response $MaxY_1$ for an informative filter F_1 with low entropy b) shows Low output response $MaxY_2$ for an uninformative filter F_2 with high entropy

A secondary contribution of this thesis is to investigate reducing the computational complexity of the CNN while maintaining high classification accuracy. Filters at each layer are approximated as linear combination of reduced set of orthogonal basis filters, obtained via principal component analysis. This allows reducing the number of convolution operations necessary while maintaining approximately the same classification error rate.

The remainder of this thesis is organized as follows. Chapter 2 discusses related literature, particularly pertaining to deep convolutional neural networks. We present brief review on the concept of CNN and its structure, including methods for complexity reduction while maintaining performance. We then review concepts of information theory related to our method.

Chapter 3 describes our two-part methodology. First 1) The main part of this chapter assigned to the CENT feature analysis. We show how we compute the conditional entropy from a CNN model and demonstrate how they could be class-informative codes for classification. Second 2) we present the mathematics behinds our suggested method for compressing CNN filters via PCA.

Chapter 4 includes the experimental part of the survey which is again two part: First) First part validate our proposed method for CENT analysis. Results demonstrate that CENT analysis is an efficient approach for transfer learning in a variety of contexts, including 2D photographs of natural categories and 3D magnetic resonance images of the human brain. CENT analysis achieves comparable accuracy to the original networks, and in some cases offers higher classification in the case of noisy images.

Second) We demonstrate that the proposed model for reducing the computational complexity of the CNN by approximating the filters with fixed number of orthogonal basis filter, can be useful while maintaining high classification accuracy. We can show the error rate by reconstructing the

model for each number of basis filters and performing classification over all number of filters. The experiment indicate not a significant drop in accuracy.

CHAPTER 1

RELATED WORK

At the time of this thesis, the literature pertaining to machine learning and deep neural networks was changing rapidly with perhaps hundreds of relevant works published per year, due to the current popularity of deep neural networks arising from accessible graphics processing units. Nevertheless, many concepts such as multi-layer perceptron and convolutional neural networks are decades old. This chapter focuses on reviewing fundamental aspects of neural network technology most closely related to our work, in addition to relevant mathematical material including information theory.

1.1 The Neural Networks

Artificial neural networks are a brain-inspired system which aims to replicate the human brain learning system. The neural network is currently the state of the art for image-based pattern recognition. Such a system can be trained recognize images via a learning procedure, typically the error backpropagation algorithm, based on a set of labelled training images. It consists of a large number of interconnected processing nodes called neurons which form so-called hidden layers located between the input and output layer. At the input, nodes correspond to input samples, e.g. pixels in an image to processed or classified. Node values in one layer are multiplied by weight parameters and summed to form the values of subsequent nodes in the network up until the output layer, where the node values correspond to the predicted image label. Figure 1.1, shows mathematical model correspond to the biological neuron.

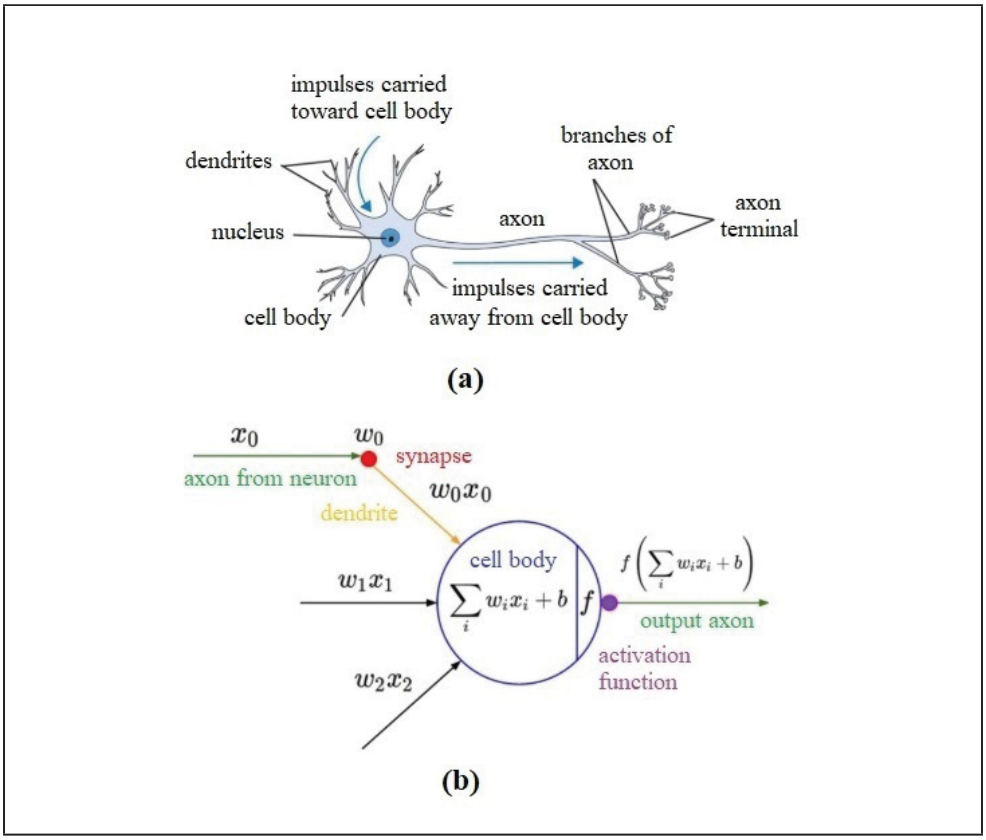


Figure 1.1 (a) Biological neuron, (b) Mathematical model of neuron
Taken from Changhau (2017)

Network learning or training consists of learning weight values that minimize the error of the task at hand, e.g image classification, typically via the iterative backpropagation algorithm, although other methods can be used, i.e. via a single pass of layer-wise estimation Kuo & Chen (2018); Gan *et al.* (2015). Nodes in one layer are generally connected to all nodes in the next layer via trainable weight parameters, i.e. a fully connected neural network, however this is generally computationally intractible for large input data such as images. The widely used convolutional neural network (CNN) solves tractibility by limiting connections, particularly in early layers, to small sets of shared weights which are equivalent to linear filters. Layers nearer to the output are typically fully connected.

Information comes from the input layer and flows to the next layers. Each layer consist of a set of nodes that compute the weighted sum of their inputs which came from the previous layer

and then pass it through a nonlinear function. The out put of each node is get from the applied function to a weighted sum of each node's input.

The inputs of each node multiply by weights of the connection they have to next layer's node, and then adds up all the input it receives. This design is called feedforward network LeCun *et al.* (2015), Hagan *et al.* (1996), Haykin (1994), Schmidhuber (2015). The simplest feed forward neural network is a single-layer perceptron in which there is one series of weights. The weights are updating through training with the learning rule called gradient descent Rosenblatt (1958). Figure1.2 shows the structure of the feed forward pass neural network.

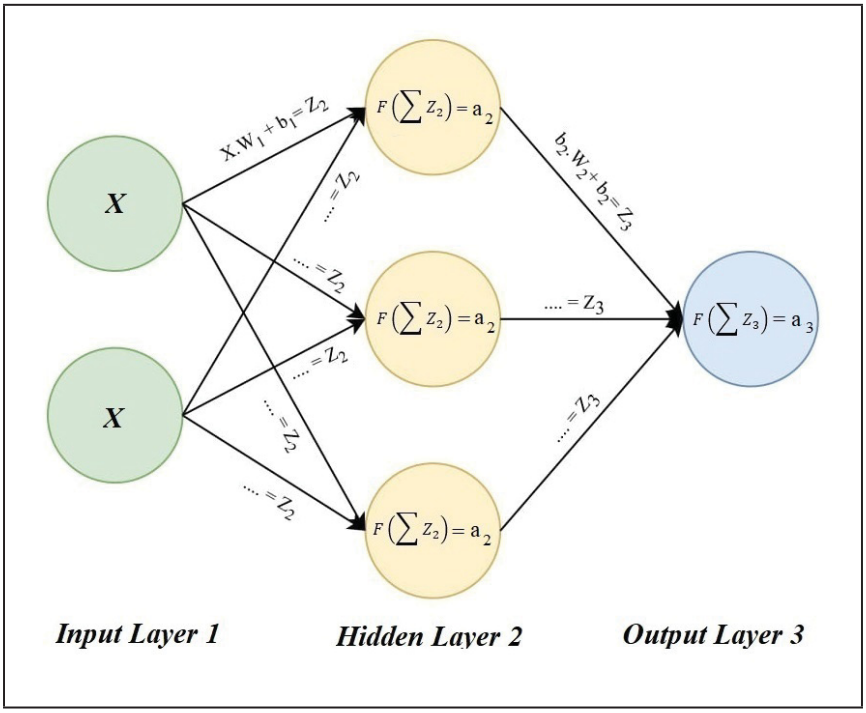


Figure 1.2 Feed Forward Neural Network
Taken from Vink (2017)

In Figure1.2, the input X is multiplied by the weights of connection and adds with the other inputs and also with the bias value b_1. The weighted sum of the inputs pass through the activation function F. The results of each nodes multiply by the weights of next connections to form the output layer. As the weights are drawn from a random distribution it is required to initialize the

weights to keep the neuron from being too big or too small. Accordingly, with each passing layer, the weights are initialized in a way that the variance remains the same Glorot & Bengio (2010); Joshi (2016). This is Glorot uniform initializer also known as Xavier initializer.

Backpropagation algorithm is used to improve the training of multi-layered network efficiently by updating weights iteratively using gradient descent algorithm. Generally, back propagation, calculate the gradient of loss function and calculate the weights updates and pass it back through the network Hecht-Nielsen (1992), Rosenblatt (1961). Considering y as a truth label and $\hat{y}$ as a network prediction, the loss function J is calculated using the squared error loss:

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{1.1}$$

In this process the weights of connection between nodes, are modified backward from the output nodes to input nodes in order to reduce the difference between output produced by the network and the output that meant to be produced Rumelhart *et al.* (1988).

The multilayer neural network is typically trained by stochastic gradient descent (SGD) learning rule. Weights are randomly initialized, then iteratively updated via alternating forward and backward passes of training data through the network. In the forward pass, a training image is sent through the network weight structure in order to generate the output. In the backward pass, the error or loss between the network output and the training label is computed, then backpropagated through the network in order to update the weights of the network such that the output error is reduced. The process iterates through training items until convergence.

Due to differentiable characteristics of multilayer neural network, we can use gradient descent. The calculation is done by the chain rule of derivatives. Partial derivative of a loss function with respect to a particular weight shows the gradients of the curve $\frac{\partial J}{\partial \omega_i}$. Therefore the opposite direction of the gradient minimize the loss function output. The problem can be break into multiplication of derivatives by chain rule of differentiation Vink (2017). For example if we

apply the chain rule upon ω_2 we have:

$$\frac{\partial j}{\partial \omega_2} = \frac{\partial J}{\partial y} . \frac{\partial y}{\partial z_3} . \frac{\partial z_3}{\partial \omega_2} \tag{1.2}$$

The backpropagation algorithm can be derived as a gradient descent process, where the error at the network output represents the derivative or gradient of the objective function with respect to the weight parameters to be updated. The partial derivatives show the direction that increase the loss function, thus we find the learning algorithm by multiplying weights in the opposite direction. The back propagation formula for all layers for updating weights and biases all the layers is:

$$\frac{\partial J}{\partial \omega_{n-1}} = \delta_n . a_{n-1} \tag{1.3}$$

$$\frac{\partial J}{\partial b_{n-1}} = \delta_n \tag{1.4}$$

Modern deep learning was made possible via 1) the CNN which reduces to number of weights in a MLP to a small set of translation-invariant shared filters and 2) the use of GPU processors to efficiently parallelise the backpropagation learning algorithm. By accelerating the computation, multiple layers of nonlinear processing nodes are deployed to extract features. It can pick out the best features in each layer that improve the performance.

1.1.1 Deep Learning in Neural Network

DNN arose from multi-layered perceptron networks, where weight parameters were trained via the backpropagation algorithm LeCun *et al.* (2015), Goodfellow *et al.* (2016), Schmidhuber (2015), Rosenblatt (1961), LeCun *et al.* (1989, 1998). It is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input.

In deep learning, each level learns to transform its input data into a partly more precise and composite representation. In an image recognition application, the input may be a matrix of

pixels; In first layer the pixels are abstracted and the edges are encoded. The second layer may compose and encode arrangements of edges; This procedure continuous until last layer recognized the that the image contains shape. The degree of abstraction is based on the number of layers and the layer sizes Bengio *et al.* (2013), LeCun *et al.* (2015).

1.1.1.1 Deep Learning strength and challenges

Deep learning is largely responsible for the growth in computer vision and artificial intelligent. It gives the computer the ability for image classification and recognizing the sound as good as human.

There are a plenty of advantages behind DNNs. One advantages of DNN over other machine learning algorithm is that there is no need for feature selection. We can feed the DNNs with the raw data. Another is that it gives the best result with unstructured data. The other is its efficiency in delivering a high quality results. The well trained DNN can perform a lot of task with a high level of precision Shchutskaya (2018), Lippi (2017).

Besides the benefits there are some major challenges that we face during working with DNNs. One major problem is the limitation of memory. Memory is one of the biggest challenges in deep neural networks today. To store the high amount of weights and activations we need dynamic random access memory (DRAM) devices with higher capacity. Memory in neural network should be large enough to store the input data, weight parameters and activations Hanlon (2017).

Another challenge is the large amount of data that we need to train the DNN model. The amount of data training in DNN is much higher than the other machine learning algorithm. As the algorithm needs to learn about the domain, it needs to train the model in large amount of data and huge number of parameters to tune. Training data are including data augmentation in order to be robust and usable.

Overfitting is also another problem that we might encounter. When the algorithm model the data very well or in the other word overtrain the data, it happen to learns the detail and noise in

the training data which have impact on the performance of the model. Although there are some ways to avoid overfitting such as dropout, L2 L1 regularization, still modern neural network have a tendency to overfit Cogswell *et al.* (2015).

Convolutional Neural Network (CNN) is an efficient form of deep neural network with shared-weight architecture. In next chapter we will discuss about this architecture.

1.1.2 The Convolutional Neural Network

The Convolutional Neural Network (CNN) has become recognized as the state of the art approach to many computer vision tasks including image-based object recognition. The CNN framework was introduced in the 1980s by LeCun in the context of text and document analysis LeCun *et al.* (1990, 1998), and was developed as an efficient MLP approach for image data, using much smaller sets of shared weights in the form of translation invariant image filters, which greatly reduced the numbers of weights and connections in comparison to fully connected multi-layer perceptron networks, particularly the backpropagation learning algorithm. Nevertheless, general CNNs remained computationally intensive and research focused on highly efficient specialized CNNs with highly efficient, manually specified, e.g. the scale-invariant feature transform (SIFT) Lowe (2004a), were computationally intensive and not widely used until implementation of backpropagation on highly parallelized graphics processing units (GPUs) Garcia & Delakis (2004); Krizhevsky *et al.* (2012), which allowed training on large-scale data sets.

While fundamental aspects of CNN technology keep layers of image filters trained via backpropagation, various algorithmic improvements have been introduced such as dropout Srivastava *et al.* (2014a), batch normalization Ioffe & Szegedy (2015b), improved pooling Graham (2014), different activation nodes Clevert *et al.* (2015) and better typology Huang *et al.* (2016). The general CNN framework has proliferated into a variety of different architectures, which can be described in terms of stagesGu *et al.* (2017).

At first few layers there is convolutional layer and max pooling layer. The results of local weighted from convolutional layer pass through nonlinear non saturated rectifier linear layer (ReLU) function. Most of the network has two fully connected layer at the end Agarap (2018).

The capacity of the network, i.e. the number of objects it is capable of representing, can be controlled by layer depth, where deeper networks can generally achieve higher accuracy but consist of more parameters that must be trained with larger training sets. Recently many CNN utilize another layer called local contrast normalization (LCN). This layer is placed after max pooling layer and aim to subtract the mean and divide the standard derivation of incoming neurons Jarrett *et al.* (2009).

This process causes local competition between adjacent features in a feature map. local response normalization (LRN) aids for a better generalization in the network. Similar to LCN it causes competition in the output however the only difference is that LRN do not subtract mean. In Krizhevsky *et al.* (2012) the author uses LRN to reduce its error rates.

Convolution layer consist of different kernels which activate different parts of the input image. Each feature map acts like a window that moves right and down and captures the features in that window. When input is high dimensional, there would be too many connection for each neuron. Therefore by reducing some connection we can make it easier to train and more efficient. This idea is used to design the local connectivity in many layers. Empty connection would have zero value for weights. So there is no need for gradient computation in empty connections.

The other method that causes further reduction in connection pattern is weight sharing. In this method the weights of some connection confine to have the same value with each other. Due to equality of some weights, there is no need to store all of them, thus the network can be more efficient as the value of other connection can be concluded by stored value.

This processed can be resembled by convolution operation in signal processing, which mostly come with the max pooling layer. The outputs of max pooling layer is invariant to location. For instance, since the weights of two nodes are equal, a motif can appear in any parts of the picture

that share the same weights. Therefore, convolutional layer detect local points and features occurred in same space of previous layer. Also Max pooling layer, calculate the maximum value of convolutional layer's output and combine the similar features in one node.

Convolution is a mathematical operation that multiply two function and create a new function. Each convolution layer consist of several convolution kernels. Each learned kernel convolves with the input layer or output of previous layer and after applying a nonlinear activation function a new feature maps will be created. The equation below shows the convolution output.

$$Z(m,n) = I(m,n) * F(m,n) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} I(i,j).F(m-i,n-j) \tag{1.5}$$

Where $I(m,n)$ and $F(m,n)$ are the input layer and the convolution kernel with m and n dimension respectively. There are different linear and non-linear activation functions such as linear functions, step function, logistic sigmoid function and rectified linear unit (ReLu) function. Activation functions are used to bring nonlinearity to the network and help the network to capture more complex features. They map the results in between 0 to 1 Clevert *et al.* (2015).

Most of the networks use ReLu as an activation function. Sigmoid functions, $\phi(z) = \frac{1}{1+e^{-z}}$, are also used to map the probability to the result since the range of result is between 0 to 1. The other function is Tanh or hyperbolic tangant activation function which map the results between the range of -1 and 1.

The most common activation function is ReLu $R(z) = max(0,z)$. It assigns zero to all the negative values and keeps the positive values equal to themselves. It causes an inappropriate mapping for negative values. To solve this issue, the leaky ReLU has been introduced which increase the range of ReLU.

After ReLU layer, there is a LRN layer which create a contrast over local input regions to capture the large responds in high frequency features. The formula below shows the normalized output

y_{ij} at the position (i, j)

$$y_{ij} = \frac{x_{ij}}{1 + \frac{\alpha}{N} \sum_{l=k-\frac{N}{2}}^{k+\frac{N}{2}} (x_{ij})^2} \tag{1.6}$$

The soft-max function is found at the CCN output. CNN often makes use of the loss function called cross-entropy during training, also known as the log loss function. The derivatives of soft-max function, defined as log loss. The log loss act as a error signal which updates the CNN weights while back propagating the network. Considering N as positive classes of samples, the cross-entropy with soft-max activations would be:

$$H_p = -\frac{1}{N} \sum_{i=1}^{N} \log(p(y_i)) \tag{1.7}$$

Where $p(y)$ is the predicted probability of the label y.

1.1.2.1 Pooling

A pooling layer is used to group neighboring neurons for subsampling. Pooling layer or downsampling layer reduces the spatial dimension of the input and decrease the computation cost by reducing the weights. The max-pooling is the most common option for pooling layer while average-pooling or L2-norm pooling are also used Graham (2014). Figure 1.3 shows an example of max-pooling is shown with the stride of 2.

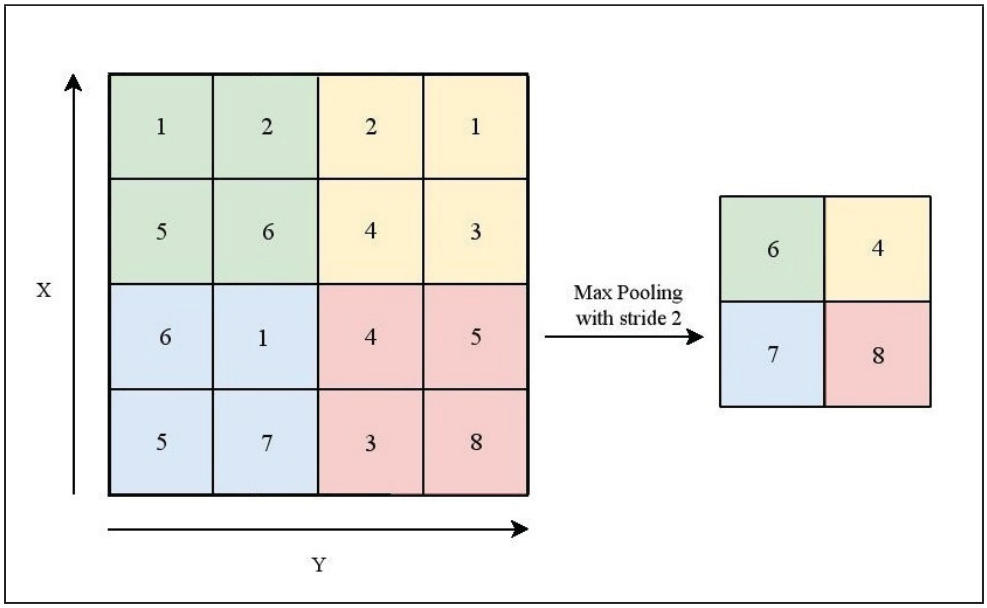


Figure 1.3 Max pooling operation with 2x2 filter and stride 2
Taken from Refianti *et al.* (2019)

It uses a filter with a specific size and output the maximum number in every subregion. Overlapping pooling occurs when the stride is less than the grid of pooling unit spaced. Networks trained with overlapping pooling layer seems slightly more difficult to overfit. As an example in paper Krizhevsky *et al.* (2012) the author reduces the error rate in AlexNet by overlapping pooling.
In Williams & Li (2018) the author proposed a new method for pooling using second-level wavelet decomposition as an alternative to neighbor pooling.

1.1.2.2 Data Augmentation and Regularization

In order to avoid over-fitting and generalizing the network well, the network require a proper regularization. Regularization methods reduce over-fitting by adding penalty to the loss function. Therefore the network does not learn the independent set of features weights.

One common technique to avoid overfitting is data augmentation. Data augmentation is the general strategy of artificially generating additional training samples, in order to avoid overfitting and to improve the network's accuracy. Various methods are applied for data augmentation.

One is generating images by applying several affine transforms such as horizontal and vertical translations, scaling and horizontal reflections in training images or generating new images by alternating the intensities of RGB channel.

DeVries & Taylor (2017) introduce the simple technique for regularization named cutout which enhance the performance of CNN. In this technique relative sections of input image is removed and the data set is augmented with the part of occluded sample. This method is an extension of drop out which has been described in next section.

1.1.2.3 Dropout

The other way to prevent overfitting is a recently-introduced technique called "dropout" Srivastava *et al.* (2014a). Dropout is a stochastic regularization technique which adds noise distribution to the hidden layers and minimize loss function. Dropout prevents overfitting in the network and also combine the neural network exponentially in an efficient way to form an averaging model. By this technique we set the output of each hidden neuron with the certain probability to zero, hence the dropout neurons do not participate in forward pass and backpropagation. It temporarily removes some nodes with all their connection from hidden layer and visible layer. Figure 1.4 below shows a neural network before and after dropout.

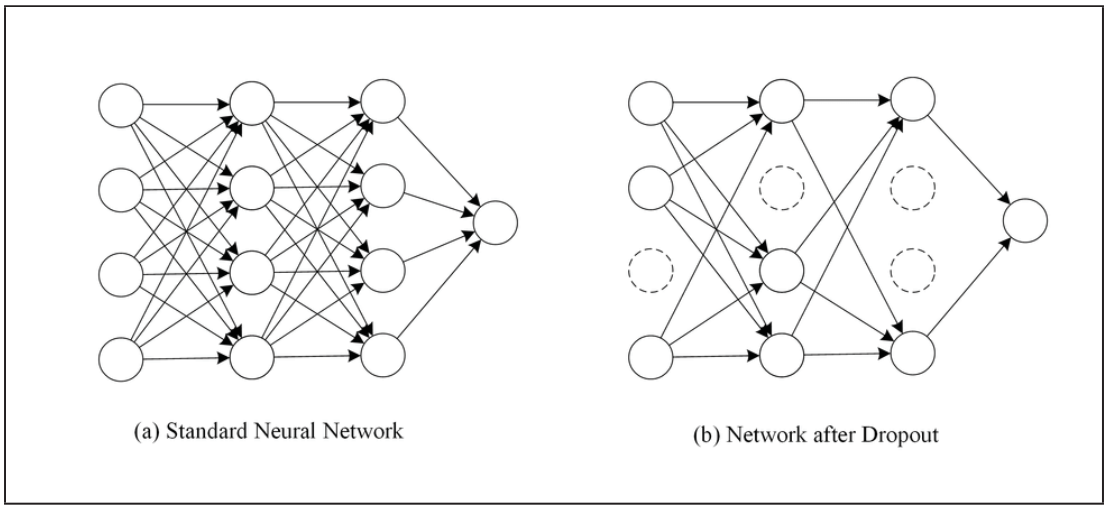


Figure 1.4 Drop out neural network model
Taken from Khalifa & Frigui (2016)

In comparison to the other regularization method dropping out nodes during training time and using averaging method in test time leads to reduce generalization error significantly.
There would be 2^n possible sample network for training by applying dropout. In order to average the prediction from the sample networks in test time, the weights of nodes are multiplied by the probability P of retained scale down trained nodes of single neural net without dropout at the test time. Therefore this method can combine the 2^n network into one single neural net in test time Srivastava *et al.* (2014b).

1.1.2.4 Batch Normalization

Batch normalization (BN) is a technique for normalizing input layers. This technique not only prevents over-fitting but also accelerates the training time due to higher learning rate. It normalizes layers during training by fixing the means variances of input layers Ioffe & Szegedy (2015b).

It added some noise to each layer which resemble the regularization effect. During the training it normalizes the output of each hidden layer by subtracting the batch mean and dividing it by the batch standard deviation of the previous activation layer. The batch mean and variance is

calculated below respectively:

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad mini\ batch\ mean \tag{1.8}$$

$$\sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\beta)^2 \qquad mini\ batch\ variance \tag{1.9}$$

Where m is the number of images per batch and x_i is the activation nodes. The normalize version of activation would be:

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \tag{1.10}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{1.11}$$

The trainable parameters scale γ (Variance) and shift β (mean) are multiplied and added respectively. During the training these two variable denormalize the output by stochastic gradient decent in order to minimize the loss function. Batch Normalization makes the network more stable by only updating two variable mean and variance of each batch instead of changing all the weights during training Ioffe & Szegedy (2015a).

1.1.3 Several Common CNN Architectures

In this section we analyze some commonly used CNN architectures applied in diverse field of computer vision, machine learning, language processing and etc.

1.1.3.1 ResNet

He *et al.* (2016) presents a deeper learning architecture of neural network named deep residual learning (ResNets). The idea of residual learning framework solve the degradation problem which is displayed When the networks depth increases. It causes lesser parameters and tackles vanishing gradient problem. The experiment shows deeper ResNets have the lower training and

test error. This architecture uses a residual function instead of unreferenced function. ResNet uses shortcut connection to skip over one or two layer and flow the information into the deeper network. Figure 1.5 shows this connection.

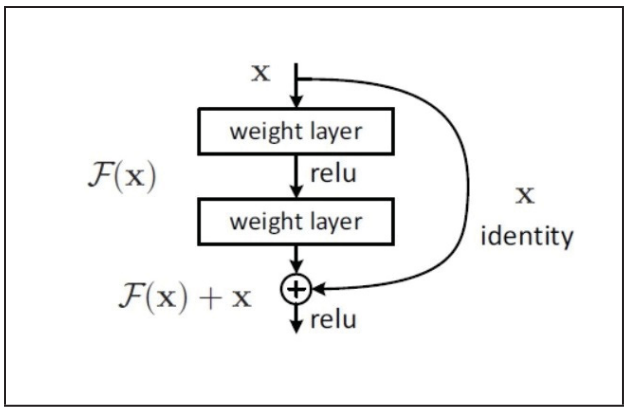


Figure 1.5 Building Block of ResNet
Taken from He *et al.* (2016)

The residual networks consist of residual blocks stacked together. There is two kind of block in ResNet. First one is identity block and second block is Conv block. Figure 1.6 below illustrate the identity block and Conv block.

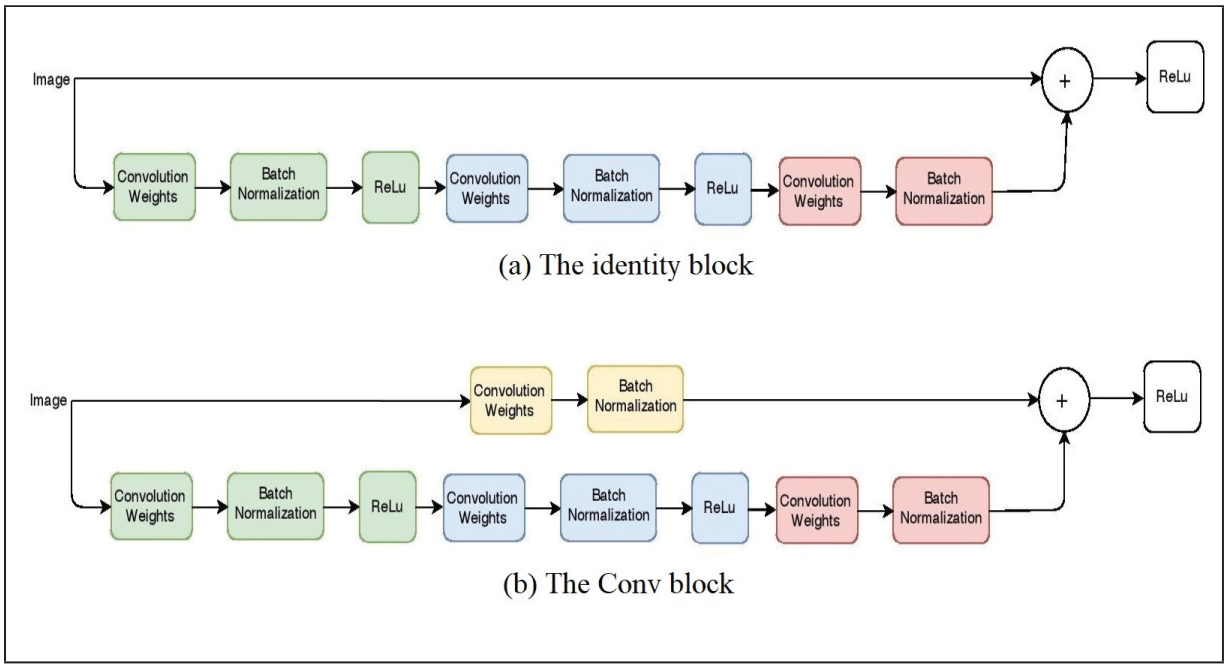


Figure 1.6 Two kind of blocks in ResNet

The Resnet structure consist of the number of Identity block and Conv block. Figure 1.7 shows their combination which forms the ResNet. This structure has been used in the ResNet 3 times. Including the final convolution layer and one dense layer at the end, the whole ResNet consist of 50 layers in total.

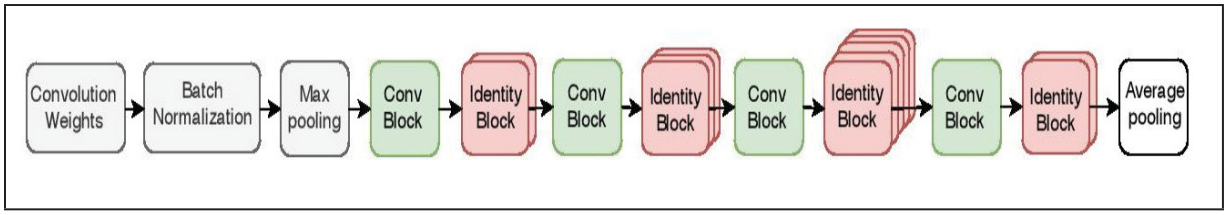


Figure 1.7 ResNet Structure

The ResNet obtain 3.57% error on the ImageNet test set and 28% progress in COCO object detection. Canziani *et al.* (2016) compare state-of-the-art DNN architectures by evaluating top-1 accuracy of all networks with a single central-crop sampling technique. The results demonstrate

the superior accurate performance in Inception and the newest Deep Residual Learning (ResNet) architecture.

1.1.3.2 DenseNet

Huang *et al.* (2017) proposes a more accurate model for convolutional network named (DenseNet) which has the shorter connections. It directly connects each layer to the other subsequent layers. Therefore, the input of each layer is the concatenation of all preceding layers feature maps. Considering that each network has L layer and $H_l(.)$ represent each layer non-linear transformation which is a composite function of BN layer, ReLu function and Conv layer operation. To perform down sampling in the network to avoid changing in the size of feature maps, the network is divided into multiple dense blocks. Figure below 1.8 shows the 5 layer DensNet block.

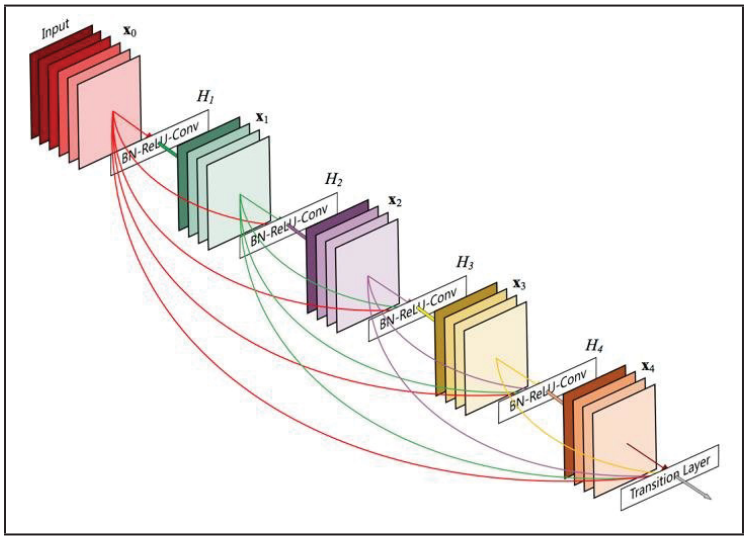


Figure 1.8 DenseNet Block with the growth rate k = 4
Taken from Huang *et al.* (2017)

The layers between each dense block are called transition blocks which contain BN layer, Conv layer follow by average pooling layer. DensNet has a narrower layer in proportion to the other networks which depends on growth rate of the network. Growth rate refers to the number of

feature maps that each function H_l produce. The experiments demonstrate that by growing the number of parameters, DenseNets shows a superior performance. Moreover, we can achieve the state of the art results by the small growth rate which needs fewer parameters and computation.

1.1.3.3 Recurrent Neural Network

For sequential inputs such as language and speech Recurrent Neural Network (RNNs) has been used. RNNs have a state vector in the hidden units that hold information of all the past components. The RNNs consist of a loop in them that keep the previous information and learn to use them. This information is the hidden state that is the representation of the previous inputs. Although RNN have a strong performance, it has difficulty in back propagation due to variations of gradients in each time steps. The new architecture for RNNs let them to train and shows a good performance at predicting characters of a text. Figure 1.1.3.3 shows the structure of RNN loops and how it looks like after unrolling it.

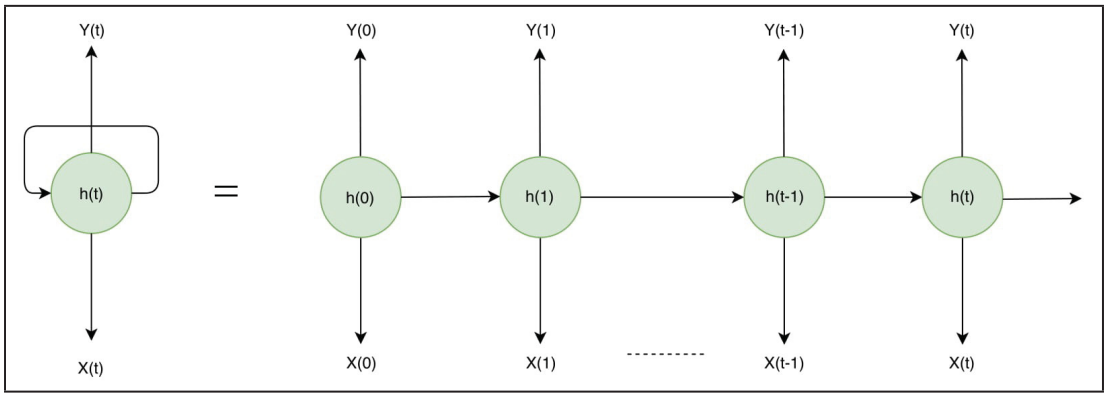


Figure 1.9 RNN diagram

The figure shows the recurrent neural network with the loop at the left side and an unrolled one in the right side. Where the $X(t)$ is the input vector at time step t, $h(t)$ is the new state and the $h(t-1)$ is the old state that connect to the next state and create a sequential architecture. Graves *et al.* (2008) uses RNN for handwritten recognition. This paper design a novel type of RNN that

addresses the issue of data segmentation and bidirectional interdependencies. Paper Graves *et al.* (2013) find the best score in TIMIT by training a LSTM RNN with a proper regularization.

Long Short Term Memory networks (LSTM) is one special kind of a RNN which is able to memorize the long term periods. Paper Chung *et al.* (2014) uses this unit along with the gated recurrent unit (GRU) in a task of sequence modeling and prove the superior performance of LSTRM and GRU over conventional model.

1.1.3.4 U-net Convolutional Network for Segmentation

Ronneberger *et al.* (2015) present a deep convolutional network architecture for training existing annotated biomedical images. The proposed architecture learn to segment output images. It consist of two symmetric path contracting path which detect context and expanding path that localize precisely. In proportion to prior method a sliding-window convolutional network, this method shows a better performance on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks and cell tracking on transmitted light microscopy images. Convolutional network is a classifier which gives a single class label as an output in response to an input image. In biomedical image processing the class labels should be assigned to each pixel.

Previous works like Ciresan *et al.* (2012) uses a sliding window to estimate the class label of each pixel. This method can localize and augment the number of training data by providing a local region (patch) around each pixel. One drawback of using this method is that there is a relation between localization accuracy and using the context. Hence using small patches increase the accuracy of localization, however it causes the network to slow down.

This paper modify the architecture of "fully convolutional network" Long *et al.* (2015) in a way that pooling operators are replaced by up sampling operators. The aim is to yield a more precise segmentation result with very few training images. By up sampling operators in each layer, the resolution of the output increase.

These structures are applied to many machine learning and machine cognitive science task for classification. These structures breaks the big problems to the smaller task that computer can answer. The networks learn input data and decide a specific solution for each related task.

1.2 Transfer Learning

This thesis focuses on the task of transfer learning: using an existing, pre-trained network to classify new or previously unseen classes. In general, a trained DNN can be reused in partly or wholly on related problems. This approach is called as Transfer Learning. Transfer learning is important in a number of contexts, for example when the number of data required for training is insufficient, when a suitable trained network exists. In deep learning the pretrained model is used as an input for the machine learning and computer vision tasks. Transfer learning can be highly effective on enhancing the performance in second task modeling. This method reuse the trained model of neural network on another predictive model task. Supervised learning using a pre-existing CNN a generic feature extractor.

It reuses the weights of layers from a pre-trained network by adapting the weights or fine tuning them or keeping them fixed in another predictive problem. It can accelerate the training of neural network and can act as a feature extracting method.

Therefore transfer learning is a great method to against the problems of insufficient data in machine learning. In this method the information transfer from the source domain to the target domain as the training data does not need to be independently and identically distributed (i.i.d.) to the test data. Rejecting the (i.i.d.) hypothesis results in in-dependency of target domain being trained from scratch and accordingly reducing the need of training data and training time in target domain. Figure 1.2 shows the learning process of transfer learning.

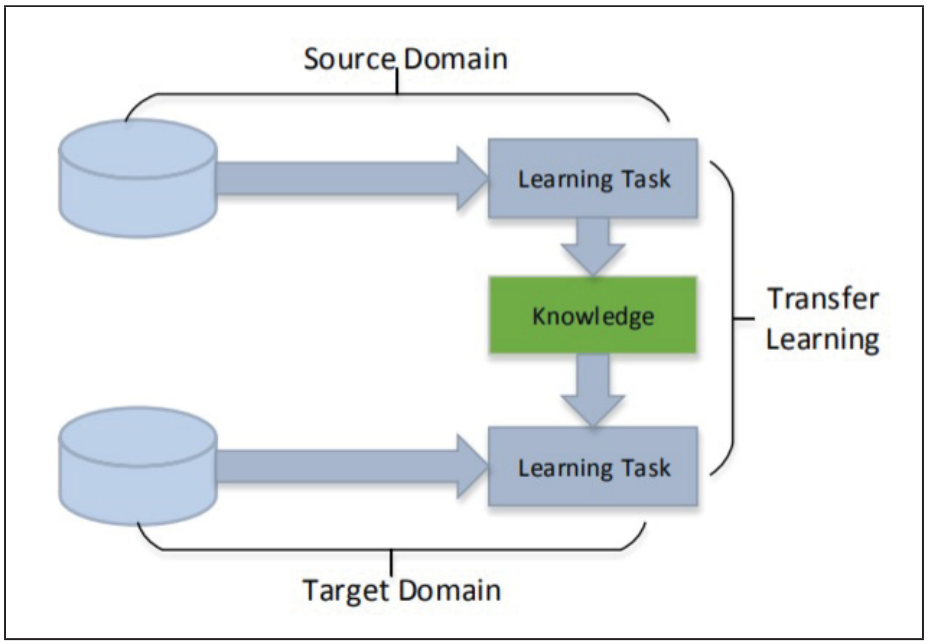


Figure 1.10 The learning process in transfer learning
Taken from Tan *et al.* (2018)

Some classic transfer learning methods has been presented in Pan & Yang (2010), Weiss *et al.* (2016), which categorize transfer learning into three part based on relationship of source domain and target domain. Pan & Yang (2009) discussed about the progress of transfer learning in classification problems. It mentioned there are different settings for transfer learning like inductive transfer learning, transductive transfer learning and unsupervised transfer learning and its relation with other related machine learning technique like domain adaptation.

Tan *et al.* (2018) categorize deep transfer learning into four part. First (1) instances-based deep transfer learning : select examples from source domain and assign appropriate weight values to the selected examples to use it as supplements to the training set. Therefore despite the differences between domains, part of the examples from source domain can be employed by target domain by adjusting appropriate weight value.

Second (2) Mapping-based deep transfer learning : map examples from both source domain and target domain into the new data space. Therefore despite the differences between two domain, they are more similar in new data space.

Third (3) Network-based deep transfer learning : transfer part of the network structure and parameters of a pre-trained network from the source domain to be part of deep neural network of target domain. Therefore the front-layer of networks act as a feature extractor.

Forth (4) Adversarial-based deep transfer learning : is based on the generative adversarial nets (GAN) which extract the representation that is appropriate for both source domain and target domain.

In classification approaches the training is representative of raw data. However, if the test data differ from training data, the model do not perform well. The reason of poor performance is due to changing of domain. In these cases the input data domain is changed while the task domain remain the same. Domain adaptation is a technique that can be used in this situations. Domain adaptation is the a field in machine learning and transfer learning which deals with the situations in which the test data of a model trained in source distribution, is different but related.

In general, domain adaptation is the process of adapting one or more source domains to transfer information and solve new tasks in a target domain. It attempts to change the source domain in a way to make it closer to the target. The adaptation success rely on the level of relatedness between the source and target domains. Therefore for these types of changes, we take domain adaptation and transfer learning into consideration Kouw & Loog (2018).

Oquab *et al.* (2014) enhanced the transfer learning task and deal the different labels and distribution of images in source domain and target domain. It proposed a design that remaps the labels between source domain and target domain. The idea is to use the mid-level image representation, trained with ImageNet and use it in Pascal VOC task of classification.

In this study we demonstrate CENT model in the context of transfer learning and perform the second classification task using the conditional entropy of the response of convolution layer from a pre-trained CNN.

In the next section we will explain about the various methods that can speed up CNN performance by reducing computation complexity.

1.3 Methods for Complexity Reduction

There are several methods that has been applied in the CNN structure in order to improve the efficiency and reduce the computation cost. Here we mention some of the well known methods and technique used to decrease the computation in CNNs Cheng *et al.* (2017), Cheng *et al.* (2018).

We can categorize the studies as follow according to the techniques they used. Low rank methods, weight compression, sparse convolutions, vector quantization, hashing and pruning are the techniques that most studies use to decrease the computation. Our model can fit into the low rank category which approximates the filters at each layer as linear combination of reduced set of orthogonal basis filters, obtained via principal component analysis.

1.3.1 Low Rank methods

Low rank technique is a common used mathematical approximation. The purpose is to minimize the cost function which is the difference between the matrix of data and the approximation data reduced by rank. It employs weight approximation for the compression of neural network parameters. Many authors has proposed methods based on low rank matrix factorization and exploited from linear structure within convolutional network. In this section we discuss some examples of this method used for complexity reduction in CNN structure Markovsky (2008). Low rank technique has been used in Denil *et al.* (2013) to represent the weight matrix as a low rank product of two smaller matrices. The size of parameterization can be controlled by decomposing the weight matrix.

To construct a good factor it uses the smoothness in the structure of the input. The number of free parameters are reduced by representing the matrix of parameter W as the product of two matrices $W = UV$, where W has size $n_v \times n_h$ and U has size $n_v \times n_\alpha$ and V has size $n_\alpha \times n_h$. By reducing the number of n_α less than n_v we can achieve a considerable reduction in parameters. One way to build U is using kernel ridge regression. It uses a linear combination of basic functions as

a simple regression model. The kernel matrix model the covariance matrix $(K_\alpha)_{ij} = k(i, j)$ between locations $i, j \in \alpha$.

The α has been chosen to give high resolution in a local area of pixel space, therefore, a number of different α's are chosen in a way that each has high resolution information for different regions.

In order to build an appropriate dictionary for different layers of the network, the structure of weight space should be considered. When the weights corresponds to the pixels in the image patch and have a topological construction, a kernel-based dictionary is chosen to satisfies the need for smoothness. In the other hand, when we have a non-topological structure, one way is to use a shallow unsupervised feature learning, such as an autoencoder. The results shows that by only a few weight values, we can accurately predict the remaining value. In a best, we can predict more than 95% of the weights of a network without any drop in accuracy.

A new method called Deep Adaptation Networks (DAN) has been introduced by Rosenfeld & Tsotsos (2018) which provide a new learned filters which are linear combination of existing ones. Zhang *et al.* (2015) also design a model for approximating nonlinear units based on minimizing the reconstruction error of nonlinear responses subjected to the low-rank constraint. It evaluates the method on a 7-convolutional-layer model trained on ImageNet and the model speedup by 4× with the 4.7% higher accuracy.

Denton *et al.* (2014) presents a technique for speeding up the CNN by finding an appropriate low-rank approximation and then fine-tuning the upper layer until get a restored prediction. It used several elementary tensor decomposition and filter clustering based on Singular Value Decompositions (SVD).

Sainath *et al.* (2013) exploits low-rank technique to factorize the weight matrix of final layer. It applies this technique on acoustic modeling and language modeling. The experiments shown that we achieve about 28% reduction in the number of parameter of DNN by imposing 128 rank on the final matrix with no loss in accuracy. A new method called one-shot has been

proposed in Kim *et al.* (2015) to simplify the compression of the entire CNN. It perform rank selection by analyzing the principal subspace of matricization of each layer's kernel tensor with global analytic varational Bayesian matrix factorization. Later on, the tucker decomposition, the higher order extension of SVD, is applied on each layer's kernel tensor with the determined rank. Finally, the entire network is fine-tuned with back propagation. various compressed CNNs (AlexNet, VGG-S, GoogLeNet, and VGG-16) are tested on the smartphone to verify the effectiveness of the method.

Although the proposed method results in small loss in accuracy, it provides reduction in model size, run-time, and energy consumption. Also, implementation level issue on 1×1 convolution as a key factor in operation of inception module of GoogLeNet and CNNs compressed by the proposed method will be taken into account.

An interesting approach proposed in Zhou *et al.* (2015) based on low rank approximation for convolutional neural network which exploit the sum of Kronecker product to replace the large weight matrices and weight tensors by multiple products of smaller matrices. However, unlike low rank factorizations such as SVD and CP-decomposition which exploit redundancy along dimension, Kronocker product from vectors can be matrices with arbitrary shape. The author also proposed combination of different Kronocker product to increase the capacity.

The author approximates the fully connected layer weights by Kronocker product and creates the Kronocker fully-connected layer (KFC) layer by this formulation:

$$W \approx A \otimes B \tag{1.12}$$

$$L_{i+1} = f(\,(\sum_{i=1}^{r} A_i \otimes B_i)\;\; L_i + \;\; b_i) \tag{1.13}$$

Where the $m = m_1 m_2$, $n = n_1 n_2$, $A \in R^{(}\, m_1 \times n_1)\,$, $B \in R^{\,m_2 \times n_2}$ and $L_{(}\, i + 1)$ shows a general form of Kronecker Factors for Convolution (KFC) layer which extend the Kronocker product approximation to a sum of Kronocker product approximation. Where L_i is the input $i_t h$ layer and f is the nonlinear activation function. It also expand the work on the convolutional layer.

The results have been tested on three different datasets. Experiments on SVHN dataset gives a 3.4% sequence error, while it only has only 12% of parameters compared to base line. Jaderberg *et al.* (2015) Using this model on CASIA-HWDB, offline Chinese handwriting dataset, achieves 3.37%, which advances the state of the art. The experiment on fully connected layer of Alex net on ImageNet data set and KConv layer on scene text recognition dataset, demonstrate that we can speed up the whole process about 3.3× or 3.6× parameter reduction with less than 1% accuracy loss.

Paper Bagherinezhad *et al.* (2017) proposed a Look-up based convolutional neural network (LCNN). It computes convolutions using lookups to a dictionary that is trained to cover the space of weights in CNNs. The dictionary and small set of linear combination are learned while training LCNN.

The author designs algorithm for CNN by looking up a few vectors from dictionary and linearly combine them to create a weight filter. The author defines a matrix D as a shared dictionary of vectors in each layer of convolution. Along with the dictionary D, there is a tensor for lookup indices I and a tensor for lookup coefficient C for each layer. I is a vector of length which is the indices of the rows of a dictionary that form the linear components of W. We can construct the weight tensor by:

$$W_{[:,r,c]} = \sum_{t=1}^{s} C_{[t,r,c]} . D_{[I_{[t,r,c]},:]} \; \forall \, r, c \tag{1.14}$$

Where the s is the number of components in the linear combinations and considered to be small therefore, instead of storing the weight tensors W, D, I and C would be stored. To apply it on forward pass convolutional layer, the convolution between an $m \times k_w \times k_h$ weight filter and the input X is written as a sum of $k_w k_h$ separate-(1×1) -convolution. We first shift the matrix by $shift_{(r,c)}$ along rows and columns with zero padding. After rewriting the obtained weight by LCNN, we can conclude that the input can be convolved with all of the dictionary vectors and

then compute the output according to I and C.

$$X * W = \sum_{r,c}^{k_h,k_w} shift_{r,c}(X * W_{[:,r,c]}) \tag{1.15}$$

$$X * W = \sum_{r,c}^{k_h,k_w} shift_{r,c}(X * (\sum_{t=1}^{s} C_{[t,r,c]}.D_{[I_{[t,r,c]},:]}) \tag{1.16}$$

$$X * W = \sum_{r,c}^{k_h,k_w} shift_{r,c}(C_{[t,r,c]} * (\sum_{t=1}^{s} X.D_{[I_{[t,r,c]},:]}) \tag{1.17}$$

The efficiency and accuracy of the network has been evaluated in different setting. The experimental results on ImageNet shows that the proposed method can speedup the process by 3.2 and achieving 55.1% top-1 accuracy using AlexNet architecture. The fastest LCNN offer 37.6× speed up over AlexNet while maintaining 44.3% top-1 accuracy. The performance of LCNN also has been evaluated on the task of few-shot learning. The comparison between performance of CNN and LCNN on few-shot learning shows that using LCNN offer 16.2% more accuracy in proportion to CNN at iteration $10K$.

Gan *et al.* (2015) designs a convolutional architecture which composed of feature extraction stage and nonlinearity stages. In this model the filter banks are learned from PCA. The feature extraction stage includes convolutional layer and feature pooling layer. The nonlinearity stage includes binary hashing and histogram statistics. Filters of higher convolution layer formed from the combination of multiple sets of feature maps. The experiments shows even a better performance in classification task than state of the arts approaches.

Similar approach as we propose in section 2.2.1 Jaderberg *et al.* (2014) has introduce to approximate the filter set by a linear combination of a smaller basis set and exploit the redundancy between filters and feature channel. It performs decomposition in a channel dimension as well as cross channel. The filters are low-rank approximation while performed in the spatial filter dimension.

In the meanwhile, paper Denton *et al.* (2014), exploit the linear compression technique to speedup the test time with another approach. The author find the redundancy in the network used SVD eigenvalue method and significantly reduce the number of parameters. At first step it used the SVD and filter clustering method to find the most important tensors and then by finding the proper rank based on the work Denil *et al.* (2013), optimized the over parametrization. It present a method for compressing first layer and two other techniques for hidden layer compression.

The approximation Mahalanobis distance metric has been used to minimize the Frobenius norm of difference between the convolutional tensor W and approximation tensor $\tilde{W}$, $\| W - \tilde{W} \|$ as follow. The approximate Mahalanobis distance metric is:

$$\| W \|_{m\tilde{a}ha} = \sum_{p} \alpha W(p), \quad where \quad \alpha_p = (\sum_{n,l} d_{n,l,s}(p)^2)^{1/2} \tag{1.18}$$

It used L_2 norm to compute the approximate of $\tilde{W}'$ on W'. By computing $W' = \alpha .* W$ the output is achieved as $\tilde{W} = \alpha^{-1} .* \tilde{W}'$. If $\theta = W_1, ..., W_S$ and $U(I_n, \theta)$ shows the set of weights in S layer and output of multivariate soft-max for the image I respectively, for the set of input $(I_1, ..., I_N)$ and labels of $(y_1 ..., y_N)$ in forward propagation, β_n would be the indices of the value of difference between $U(I_n, \theta)$ and y_n so for the layer s we have:

$$d_{n,l,s} = \nabla_{W_s}(U(I_n, \theta) - \sigma(i - l)), \quad n \leq N \ , \ l \in \beta_n \ , \ s \leq S \tag{1.19}$$

Where the $\sigma(i - l)$ is the dirac distribution and the l is the center of that. For matrix decomposing the 2 dimension tensor if $W \in R^{m \times k}$ decompose by SVD in which $W = USV^{\top}$ where the $U \in R^{m \times m}$, $S \in R^{m \times k}$, $V \in R^{k \times k}$. By choosing only t largest entries of S, $\tilde{W}$ can be approximated. Where $\tilde{U} \in R^{m \times t}$, $\tilde{S} \in R^{t \times t}$ and $\tilde{V} \in R^{t \times k}$. As the computation of WI is done in $O(nmk)$, computation with t entries of singular values done in $O(nmt + nt^2 + ntk)$.
In order to compress the tensor further, it applied SVD on $\tilde{V}$. So by repeating K times for

approximating the greedy algorithm $\tilde{W}$ would be:

$$\tilde{W}_S = \sum_{k=1}^{K} \alpha_k \otimes \beta_k \otimes \gamma_k \tag{1.20}$$

Where the $\alpha_k \in R^m$, $\beta_k \in R^n$ and $\gamma_k \in R^k$.

1.3.2 Weight Compression

Weight compression technique is introduced to reduce the complexity and computation. Here we provide some examples of studies that use this technique to reduce computation.
Ullrich *et al.* (2017) used a version of soft weight-sharing introduced in Nowlan & Hinton (1992) which compress the weights to K cluster and applied Gaussians prior model over the weights. The weights concentrate around the number of clusters and the compression achieved by encoding K cluster mean.

A novel method is introduced in Yang *et al.* (2019) for compression of DNN using weight sharing called Filter Summary (FS). In this study the filters are adapted from FS. Therefore FSNet has a 3D tensor FS in each convolutional layer. The 3D tensors FS shown as overlapping 3D blocks. In proportion to conventional CNN, the convolutional layer space with FS is much smaller since the weights are shared across the filters that overlap each other. The results shows the effectiveness of FSNet compression on image classification task without dropping in accuracy. Deep fried Convnet Yang *et al.* (2015), introduced a new adaptive fastfood transform which can change the parameters of fully connected layers and reduce the storage and computational cost. It uses MNIST and ImageNet dataset.

The proposed model replace the fully connected layer with an adaptive Fastfood transform. This network is an end-to-end trainable and use only half number of parameters while get the same results as a conventional CNN on ImageNet. The computation cost for the multiplication matrix in propagating signal from l_{th} layer with d activation to $l_{th} - 1$ with n activation is $O(\, nd)$. After reparameterizing by Fastfood transform, the require storage and computational cost are $O(\, n)$

and $O(n\ log\ d)$ respectively. By using Fast Hadamard transform which is an alternate for Fast Fourier Transform (FFT), the parameter of Adaptive Fastfood transform can be computed in backward pass. This proposed model can be represented as a type of structured projection or an approximation of the feature space of a learned kernel.

A view from structure random projections is based on the random matrix either Gaussian or binary which is projected randomly and could preserve the information. In this method the author uses original Fastfood transform as an alternative to random matrix. In Fastfood transform S, G and B are the diagonal random matrices which respectively implement automatic relevance determination of features, control the bandwidth of the kernel, and represents different kernel type. The other view is related to determination of features associated with kernels. Using a design derive from kernels in order to map it to feature can be desirable since there is a duality between inner product of features and kernels, and working with features are somehow more preferable as it is not as dense as kernel. This method introduce Fastfood transform methods to approximate the kernel. The advantage of this method is that it is end-to-end differentiable and we can achieve reduction during training time.

1.3.3 Sparse Convolutions

Sparse approximation theory is used to find the sparse solution for linear equation. Sparse decomposition is one solution to reduce redundancy in the neural networks and speeding up the computation. As an example reference Liu *et al.* (2015) used the redundancy existed in the convolutional kernels and reduce the amount of computation.
In this model Sparse Convolutional Neural Network (SCNN) consist of sparse convolutional layer formed from multiplication of two sparse matrix. The proposed model sparsifies the kernel weights by decomposing them based on the error obtained from reconstructing the weights. It optimized the error by fine-tunning the weights. To perform decomposition first the convolution is defined as a multiplication of sparse matrixes as follow. Considering I as the Input feature map and the convolution kernel as K, If we consider that the output feature maps of convolution

layer is:

$$O(x, y, j) = \sum_{i=1}^{m} \sum_{u,v=1}^{n} K(u, v, i, j) I(y + u - 1, x + v - 1, i) \tag{1.21}$$

It transfer matrix I to J and kernel K to R as follows:

$$K(\, u, v, i, j) \approx \sum_{k=1}^{m} R(\, u, v, k, j)\, P(\, k, i) \tag{1.22}$$

$$J(\, y, x, i) = \sum_{k=1}^{m} P(\, i, k)\, I(\, y, x, k) \tag{1.23}$$

Where J and R is the tensor transform of I and K respectively. The tesnor R is decomposed into the product of matrix $S_i \in R^{q_i \times n}$ and tensor $Q_i \in R^{s \times s \times q_i}$. where the q_i is the number of bases as follows:

$$R(u, v, i, j) \approx \sum_{k=1}^{q_i} S_i(k, j)\, Q_i(u, v, k) \tag{1.24}$$

$$\tau_i(y, x, i) = \sum_{u,v=1}^{s} Q_i(u, v, k)\, J(y + u - 1, x + v - 1, i) \tag{1.25}$$

$$O(y, x, j) \approx \sum_{i=1}^{m} \sum_{k=1}^{q_i} S_i(k, j)\, \tau(y, x, k) \tag{1.26}$$

Where $O(y, x, j)$ is the convolution result. By using less dimension of tensor τ_i, and combining the first two dimension and concatenate both S_i and T_i along the dimension q_i, $O(y, x, j)$ could be achieved. In the fine-tunning phase, the sparsity is used to impose a constraint over parameters and the loss is measured as a network error. The experimental results shows a significant reduction in the number of parameters by keeping accuracy on ILSVRC2012 dataset.

Wen *et al.* (2016) also propose a Structured Sparsity Learning (SSL) method which regularize the structure of DNN. This method uses a group lasso regularization to adapt the structure of filters, channels and depth of layers. The computation is reduced by learning the compact

structure. SSL obtained on average 5.1× and 3.1× speedups of convolutional layer computation in AlexNet.

1.3.4 Vector Quantization

Vector quantization is a technique that is used for data compression which map each data to a small groups of numbers. It model the probability density functions as a distribution of vectors.

Zhou *et al.* (2012) introduced a method for scalar quantization to compress the CNN parameters which used the fist 10 codes of discriminative bit-vector obtained from quantizing the scale-invariant feature transform (SIFT) feature. Further a theoretical vector quantization methods has been used in Gong *et al.* (2014) which shows by applying K-mean clustering or conducting product quantization we can tackle the storage issue and achieve a good model size. It used a series of information theoretical vector quantization methods instead of traditional matrix factorization methods. Many methods such as binarizing the parameter, using K mean for scalar quantization and product quantization for structured quantization, has been evaluated. The evaluation over different vector quantization methods demonstrated that structured quantization such as product quantization works significantly better than the other one. The basic idea of product quantization is to divide the vector space into many disjoint subspaces, and perform quantization in each subspace. The results in ImageNet datasets show that with only 1% loss of classification accuracy, by proposed method we can achieve 16-24 times compression in the network.

Wu *et al.* (2016) also used a framework named Quantized CNN, which can reduce the storage and memory. In this paper the author minimize the estimation error of each layer's response by quantizing the convolutional layers and weighting matrices in fully connected layer. The training scheme also proposed which prevent error from increasing during quantization of whole model. This framework achieve $4 \sim 6\times$ speed-up and $15 \sim 20\times$ compression for only 1% loss in classification accuracy.

1.3.5 Hashing

A novel network architecture which is named Frequency-Sensitive Hashed Nets (FreshNets) is presented by Chen *et al.* (2015). In this method the redundancy in convolutional layers and fully-connected layers of a deep learning model are exploited to save in memory and storage consumption. Firstly, filter weights are converted to the frequency domain with a discrete cosine transform (DCT) and a low-cost hash function is used to randomly group frequency parameters into hash bucket. Standard back propagation learns a single value that is shared by all parameters assigned the same hash bucket. By allocating fewer hash buckets to high-frequency less important components, the model size is reduced further more. The FreshNet is validated on eight data sets.

The proposed model compressed network by using DCT to convert the filter weights into the frequency domain. It uses hash function for obtained weights from frequency domain in order to group the frequency parameters by random. This method reduces the model size by implementing weight sharing and gradient updates. The difference of proposed model with the typical convolutional neural network is that it use hash function in frequency domain. The convolutional filters are represented in frequency domain by DCT. The filters in frequency domain are represented by:

$$V^{kl} = f_{dct}(\, V^{kl}) \tag{1.27}$$

Where V^{kl} is the weight matrix of $d \times d$ convolutional filter for k^{th} input plan to l^{th} output plan. This method reduce the number of parameter from $m \times n \times d \times d$ to K values by giving of value ω to each filter frequency weight in V randomly. To avoid using significant memory usage, the author uses hashing trick to assign shared parameters by random.

$$V^{kl}_{j_1 j_2} = \xi(\, k, l, j_1, j_2)\ \ \omega_{h\,(\,k,l,j_!,j_2)} \tag{1.28}$$

Where $h(k, l, j_1, j_2) \in 1, \ldots, K$, and $\xi(k, l, j_1, j_2) \in \pm 1$ are the signs factor calculated by hash function. Since the share weights are stored in frequency domain, the gradients form in frequency space. In frequency domain the high frequencies are located in bottom right half of V^{kl} which correspond to small magnitude near zero and the low frequencies is shown in upper left correspond to components with larger magnitude. As each frequency group has different magnitude, the separate hash spaces is assigned to different frequency groups.

The results validate that proposed model can compress model storage affectedly. It can compress the parameters by frequency sensitive hashing and hashing trick for parameter random weight sharing in a way that major model parameters are conserved. In Weinberger *et al.* (2009), author also uses a hashing strategy to reduce the dimensionality. This study offer exponential tail bound for hashed inner product.

1.3.6 Pruning

A broad category of compression methods attempt to prune the unimportant network parameters. Lebedev & Lempitsky (2016) proposed a new approach based on the idea of brain damage process which speed up the convolution layer by shrinking the coefficients kernel to zero. The proposed idea prune the kernel tensor in a group wise way. The idea is to group the inputs of convolutional tensor in a way that all the groups shrinks to zero and therefore makes the matrix multiplication faster. It uses the unrolled convolution which convert the convolution procedure in forward-propagation and backward-propagation to a matrix-matrix product Chellapilla *et al.* (2006). It eliminates the rows and columns of the matrix of both side and speedup the convolution. The author uses group sparsity regularizer for making the group wise brain damage efficiently. It The proposed model results in speeding up the second and third convolution layer of AlexNet by the factor of 8.5×.

Paper Han *et al.* (2015) introduces three stage of compression. It starts with pruning the connections by 9× to 12× and then quantizing the weights and reduces the number of bits from 32 to 5 by applying weight sharing. It then applies Huffman coding used for lossless

data compression and at the last stage the network will be retrained to fine tune the remaining connections. To prune the connections, it first ordinarily train the network and then cutouts the connection with lower weights. The network retrained again to learn the remained weights. The sparse structure would be stored in a compressed sparse row or compressed sparse column format.

In Srinivas & Babu (2015) the author proposes a way to remove the similar neurons and use the redundancy in network parameters. In case of similarity in neurons, it removes one of them and add its related coefficient of next layer to the coefficient of other one which is called surgery step. Based on the Hebbian principle, the presented model wire neurons with the similar weights together. In case of dissimilarity, it defines a saliency for two weight sets i, j as a $s_{i,j} =< a_j^2 \|\epsilon_{i,j}\|_2^2 >$. Where a_j is the coefficient of j^{th} neuron and $\epsilon_{i,j}$ is the difference of the results from non linear function of i^{th} and j^{th} neuron. The author computes the saliency for all pair of weights and pick the minimum amount, removes the j^{th} neuron and updates the coefficient by $a_i \leftarrow a_i + a_j$. The number of neurons to prune is determined by the use of saliency curve which has the low values on the beginning and exponentially high at the end.

Guo *et al.* (2016) proposed a method named dynamic network surgery. It performs two operation pruning and splicing by which it compresses the network and compensates the loss caused by over pruning respectively.

The Yang *et al.* (2017) proposes a layer by layer pruning algorithm that not only reduces the size of the network but also minimize the energy consumption of the system. It targets to minimize the error of the output of each feature map instead of filters. At first it calculates the energy consumption in CNN by calculating energy consumption in computation and data movement and prune the order of layers and then it removes and restores weights and change the value of each weight by locally fine tunning them.

Paper Luo & Wu (2017) uses intermediate activation pruning approach to compress the CNN models. The filters eliminated based on the proposed entropy method. In this method the importance of each filter evaluated by the entropy of activation output of specific filter. Therefor

those filters whose activation function contain less information, eliminated from the network. Further like GoogleNet Szegedy *et al.* (2015) and ResNet-50 the average pooling strategy is adopted to remove fully connected layer LeCun *et al.* (1990).

There are also other techniques that can be used for complexity reduction. Further invention of modern GPUs has a great effect on accelerating the training of neural networks.

1.4 Computation vs. Memory

Neural networks follow the read/write mechanism, in which the information encodes from the network, get stored in specific location and forwarded in time by memeory Gallistel & King (2011).

In the CNN the high-level features of training data are captured by the front layers and then the information needed to make a decision are identified in the subsequent layers. Moving from front layer of CNN to the end layer, the number of computation is increasing and there will be a lower demand for memory to store data. Therefore the speed is improved while the data move forward to the end layer and we can conclude the computation complexity and memory have a reverse relation in CNN. Here we calculate the number of parameters and memory usage of the VGG-19 in table below.

Table 1.1 Overview of memory the number of parameters in each layer of VGG

Layer	Memory	Number of Parameter
INPUT	$224\times224\times3 \approx 150K$	0
Conv3-64, Relu3-64, Norm3-64	$54\times54\times64 \approx 187K$	$(11\times11\times3)\times64$
Pool2	$27\times27\times64 \approx 47K$	0
Conv3-256, Relu3-256, Norm3-256	$27\times27\times64 \approx 47K$	$(5\times5\times64)\times256$
Pool2	$13\times13\times256 \approx 43K$	0
Conv3-256, Relu3-256	$13\times13\times256 \approx 43K$	$(3\times3\times256)\times256$
Conv3-256, Relu3-256	$13\times13\times256 \approx 43K$	$(3\times3\times256)\times256$
Conv3-256, Relu3-256	$13\times13\times256 \approx 43K$	$(3\times3\times256)\times256$
Pool2	$6\times6\times256 \approx 9K$	0
FC, Relu	$1\times1\times4096 \approx 4K$	$(6\times6\times256)\times4096$
FC, Relu	$1\times1\times4096 \approx 4K$	4096×4096
FC	$1\times1\times1000 \approx 1K$	4096×1000

Figure 1.11 and 1.12 show the needed memory and the number of parameters for the layers of VGG-19, respectively.

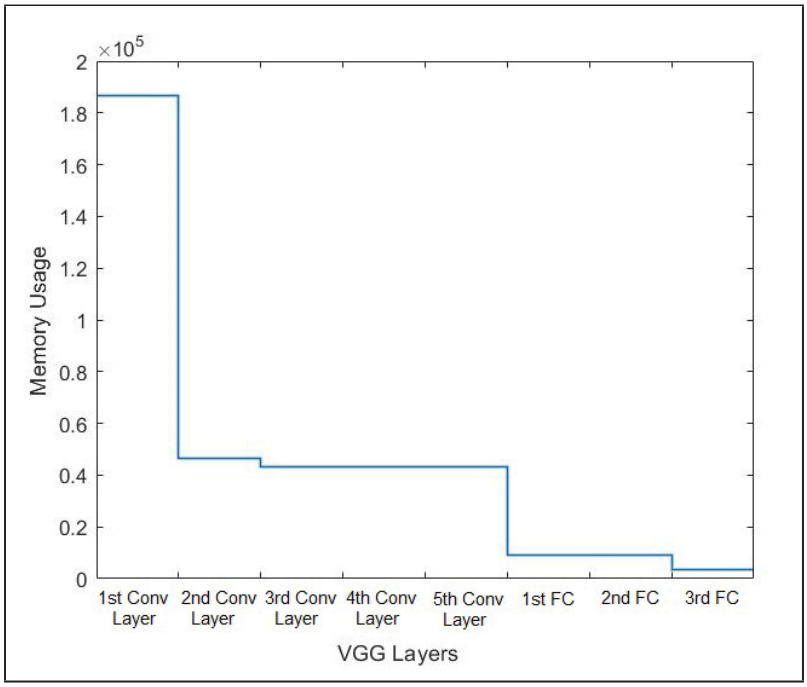


Figure 1.11 Overview of memory usage in Conv and FC layers of VGG

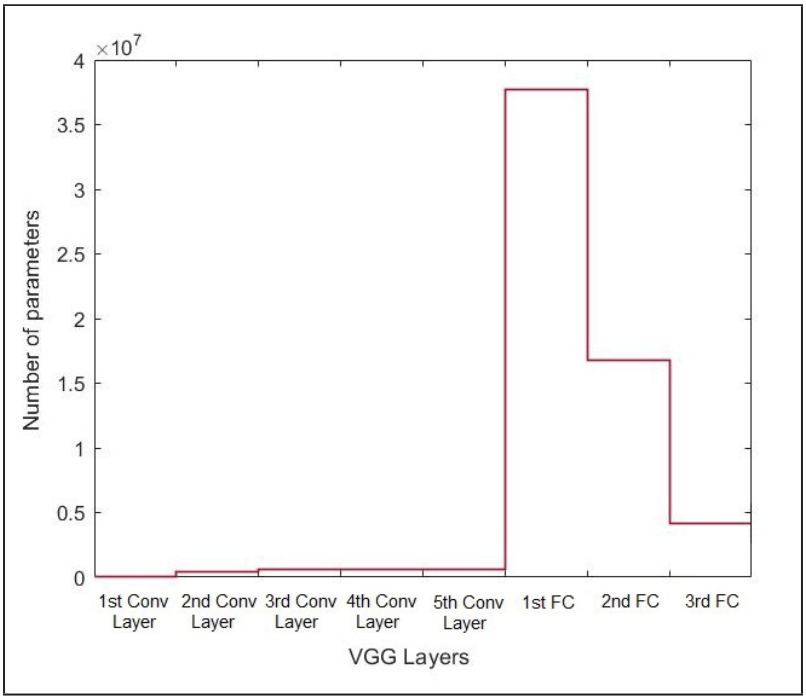


Figure 1.12 Number of parameters in Conv and FC layers of VGG

Therefore, the closer we get to the back layer, less memory is needed for storage and the number of parameter are increased.

1.5 Information Theory

This thesis focuses on quantifying the flow of information in the deep convolutional neural networks in terms of information theory. Information theory is based on probability theory and random variables, and seeks to quantify the information content or uncertainty of a random variable in terms of entropy. Information theory was first proposed by Shannon (1948) in the context of digital data communication, serves as the basis of all modern digital communication systems. A number of information theory references exist, ex.Cover & Thomas (2012). Here we review information theoretical concepts including entropy, conditional entropy and mutual information.

1.5.1 Entropy

Information theory is rooted in the notion of entropy, which quantifies the uncertainty of a random variable. It measures the average information needed to describe that variable. For a random variable $Y = \{y_1, \ldots, y_i, \ldots, y_N\}$ defined over a set of N discrete events y_i which occur with probability $p(Y = y_i)$, or $p(y_i)$, the Shannon entropy $H(Y)$ is defined as:

$$H(Y) \;=\; -\sum_{i=1}^{N} p(y_i)\, \log p(y_i), \tag{1.29}$$

and ranges from $[0, \log N]$ for maximally informative and uninformative distributions $p(Y)$, respectively. For a binary random variable, the entropy is proportional to the expected number of bits required to transmit 1 symbol of information.It attains its highest value when the probability is half. Entropy has been widely used in computer vision task e.g., in classifying image textures Haralick *et al.* (1973), or salient feature selection Kadir & Brady (2001); Toews & Arbel (2003). The analysis of information theory in computer vision and pattern recognition can be found in a textbook Ruiz *et al.* (2009).

Entropy also is used in metrics and optimization. The entropy has been used in perceptual learning. As an example maximum entropy principle ensure that the distribution contain no more information for selecting the matching distribution, in the other hand, it is desirable to select the feature that produce the maximum information (minimum entropy).

In Kadir & Brady (2001) the author measures saliency based on rarity by finding points that maximize the discrimination between the objects using a technique suggested by Schiele (1997). It uses Bayes formula to determine the probability of an object given a vector of local measurements. The high probability of a given point shows the specific descriptor is better to find specific image. To identify local saliency image patches Gilles (1998) suggests to use entropy measures. In Toews & Arbel (2003) the author uses the minimum entropy of likelihood which increases the speed and accuracy of optimization process.

Khadivi *et al.* (2016) used the information theory to interpret information flow in different layers and entropy changes in different layers. It proposed an optimization problem for training based on information bottleneck principal which aim to minimize loss distortion.

In Tishby *et al.* (2000), the author gives a definition about relevant information provides from a signal. It squeezes the information using some code words in a bottleneck format. It uses information theory to generalize the rate disortion and drives consistant equation and algorithms for finding the relevant information. Paper Tishby & Zaslavsky (2015), use the theory of information bottleneck to analyze the DNN. The author explain how we can optimize the generalization bound by quantifying the DNN using mutual information between the layers.

1.5.2 Joint and Conditional Entropy

The joint entropy defined by a pair of random variable; again considering Y as a random variable $Y = \{y_1, ..., y_i, ..., y_N\}$ over a set of N and given a second random variable, e.g., object class $C = \{c_1, \ldots, c_j, \ldots c_M\}$, the joint entropy is defined as follow:

$$H(Y, C) = -\sum_{i}^{N}\sum_{j}^{M} p(y_i, c_j) \log p(y_i, c_j) = E[-\log p(y_i, c_j)], \tag{1.30}$$

Here the $p(y_i, c_j)$ is the joint probability of y_i and c_j.

The conditional entropy defined as the entropy of conditional distribution upon another variable, which is defined as follow:

$$H(Y|C) = -\sum_{i}^{N}\sum_{j}^{M} p(y_i, c_j) \log p(y_i|c_j) = E_{p(y_i,c_j)}[-\log p(y_i|c_j)], \tag{1.31}$$

Therefore we can write it as:

$$H(Y|C) = -\sum_{i}^{N}\sum_{j}^{M} p(y_i|c_j)p(c_j)\log p(y_i|c_j) \tag{1.32}$$

$$= \sum_{j=1}^{M} p(c_j)\, H(Y \mid c_j), \tag{1.33}$$

where $H(Y \mid c_j)$ demonstrates the entropy of Y conditioned on a fixed class c_j, therefore $H(Y \mid C)$ shows the expected conditional entropy across all classes. The main consequence is that entropy is reduced by conditioning $H(Y \mid C) \leq H(Y)$ except in when Y and C are statistically independent, in this case the two expressions are equal.

In this study we treat systems and networks as stochastic machines where the systems produce an output Y for the input of class C based on conditional probability $p(C \mid Y)$ and by stimulating the system, networks produce an output of Y for the class of C based on conditional probability $p(Y \mid C, F)$ where F is the set of weights.

If the input generates the probability of $p(C)$ and an output gives the conditional probability specified by C the desired system defined by $p(C \mid Y)$ or the joint probability $p(C, Y)$ which is the product of input distribution and conditional distribution $q(C)q(C \mid Y)$. If we define $F \in R^m$ a set of m dimensional filters that determines the networks, and the $p(C \mid Y)$ belongs to the the model $p(C \mid Y, F); F \in R^m$, there exist a F for which $p(C \mid Y) = p(Y \mid C, F)$.

In this study we consider the conditional entropy computed across all layers and filters of an existing CNN, and applies generally to a trained CNN with any output loss function such as the cross-entropy, squared or absolute loss, etc, the only obligation is that the resulting filters should be informative regarding the object classes of interest.

1.5.3 Mutual Information

The amount of information shared by Y and C, defined by mutual information (MI), $I(Y, C)$ which is the the difference between the entropy and the conditional entropy.

$$I(Y, C) \;=\; H(Y) - H(Y \mid C) \;=\; H(C) - H(C \mid Y). \tag{1.34}$$

The MI is a fundamental quantity that measures the mutual dependencies between two random variable which is the deduction in uncertainty. It provides an upper bound as to the capacity of a noisy communication channel $Y \rightarrow C$ Cover & Thomas (2012).

Conditional entropy and MI are widely used to measure similarity between images in computer vision. This information-theoretic approach uses mutual information between the images by adjusting positions and orientation till it get maximized Wells *et al.* (1996). For feature selection strategy in machine learning, mutual information maximization score used to rank features to find the useful features are in a classifier Brown *et al.* (2012). MI is known as the Information Gain in decision tree learning Duda *et al.* (2012) and applied to identify optimal data splits in training. In contrast, our analysis involves computing conditional entropy from a pre-trained CNN.

Reference Shwartz-Ziv & Tishby (2017), follow the idea of Tishby & Zaslavsky (2015) by visualizing the mutual information that is preserved in each layer. It unfolds some detail about working of DNN. It shows that each epochs compress the input mostly not fit the training labels. The phases of compression and SGD optimization, drift and deffusion has been explained.

Belghazi *et al.* (2018), proposed an estimator for mutual information called Mutual Information Neural Estimator (MINE). The author exploit the dual optimization of characterization of the mutual information KL-divergence which is formalized in (GAN) Goodfellow *et al.* (2014).

In the following section, based on Chaddad *et al.* (2017), Chaddad *et al.* (2019), we show that for a discriminatively trained filter set F, the conditional information $H(Y \mid C, F)$ is *necessarily*

a highly class-informative code. The output of convolutional filters is modeled as a random variable Y conditioned on the object class C and network filter bank F. A set of conditional entropy (CENT) features derived from a trained CNN is shown in theory and experiments to be a highly informative and compact code for classification, improving upon the output of the original CNN from which the features were originally derived.

CHAPTER 2

METHODOLOGY

This thesis focuses on computational strategies aimed at reducing CNN computational complexity and/or increasing accuracy of the specific task, i.e. image classification.

Our first goal is to increase the accuracy of specific task using efficient transfer learning with the new method for feature selection based on information theory. A number of different strategies in this goal, where the primary methodology focused on analyzing the deep CNN in terms of probability and information theory based on Chaddad *et al.* (2017, 2019), as described in 2.1. The general methodology is to model the neural network as a probabilistic Bayes network, where information at any point in the network is modeled as distribution conditional upon inputs and filtering operations. Conditional entropy is then used identify class-informative features throughout the network for the task of image classification. Specifically output of a network layer is considered as a random variable Y, defined by a distribution $p(Y|C,F)$ conditioned with the object class C and filter F. The conditional entropy $H(Y|C,F)$ introduced as $CENT$ is very comprehensive codes used to achieve higher classification accuracy.

The second goal is to reduce CNN computational complexity by compressing the convolution layar of the CNN. Section 2.2 describes two techniques for representing CNN filtering using alternative linear bases, a research direction which we investigated but ultimately did not pursue due to time constraints. These are included here for completeness. The first technique was to use linear subspaces to compress CNN filter weights using Component Analysis (PCA) (e.g. Jolliffe (2011)), and the second was use of the wavelet transform Graps (1995), Daubechies (1990), both with the goal of reducing computational complexity while maintaining high classification accuracy. The investigation over wavelet transform can be find in the appendix.

2.1 Bayesian Probabilistic Model of Deep Convolutional Neural Networks

In this section we propose a Bayesian probabilistic model for DNN architecture. We show that information can be quantified through network filters in terms of the conditional entropy $H(Y|C, F)$ of neural activation Y, given object class C and filtering operation F. Our mathematical derivation focuses on inputs of a single layer, however the results generalize to multiple layered networks. We develop a class of CENT features that represent a highly compressed code for the network activation maps, and can be used to achieve highly accurate classification Chaddad *et al.* (2017, 2019).

Lets denote Y as a scalar random variable showing the output of neurons in a CNN, e.g., responses in a feature map. heavy-tailed distribution such as a Laplacian can be well approximation of the $p(Y)$, distribution of informative convolution filters, due to the correlation structure of natural images Simoncelli & Olshausen (2001); Simon *et al.* (2007). Though Y is a continuous random variable, it could be approximated as a discrete random variable for the purpose of computing Shannon entropy.

To define the classification with MAP estimation we can consider $C = \{c_j\}$ be a discrete random variable over object classes and Bayesian posterior distribution over possible classes C given filtered image data Y and a set of filtering operations F has this relation

$$p(C \mid Y, F) \propto p(Y \mid C, F)\, p(C \mid F) \tag{2.1}$$

Here $p(Y \mid C, F)$ denotes the conditional likelihood of class C related with data Y and feature set F, and $p(C \mid F)$ is a conditional prior over classes C given filter set F. As the classification normally maximize the Bayesian posterior distribution, we assume distribution for $p(C \mid F)$ is uniform, therefore the connection between data Y and class C is modeled by $p(Y \mid C, F)$. In this case we can equal MAP estimation to maximum likelihood (ML) estimation. Hence the classification find distribution of Y conditioned on class C and filter set F, $p(Y \mid C, F)$, in a way to minimize the uncertainty. As uncertainty described by entropy, in that case the conditional entropy $H(Y \mid C, F)$ is minimized. Better classification accuracy occur when $H(Y \mid C, F)$ tend

to zero. In perfect classification $H(Y \mid C, F)$ is equal to zero through ML estimation, i.e. $C^{\mathrm{ML}} = \mathrm{argmax}_C\, p(Y \mid C, F)$.

In signal processing we can model filtering through probabilistic conditioning, for instance linear Kalman filter Kalman *et al.* (1960), which is applied to an optional filtering operation where convolution and ReLU in CNNs are examples of linear and nonlinear filtering operation respectively.

If we consider that $F = \{f_i\}$ be a discrete random variable over a set of filtering operators f_i applied at neurons, the output of filtering operation could be represented as a conditional distribution $p(Y \mid C, F)$ over neural output Y conditioned on class C and filter F.

Figure 2.1 illustrates the Bayesian probability distribution of a VGG. For the set of neural output $Y = \{y_0, ..., y_i\}$, and set of filters $F = \{f_1, ..., f_i\}$, each convolution filters has been modeled as a conditional distribution $p(y_{i-1} \mid C, f_i)$ over neural output y_{i-1} conditioned on class C and filter f_i.

Figure 2.2 shows how convolution operation is modeled as a conditional distribution $p(Y \mid C, F)$ over neural output Y conditioned on class C and filter F. In the information theory field, the importance of filtering is due to reach the minimal entropy. Thus it reduces the entropy $H(Y \mid C, F)$ by conditioning on a well-advised design for the filter set F. Particularly

$$\begin{aligned} H(Y \mid C) \;&\geq\; H(Y \mid C, F) \\ &=\; \sum_{i,j} p(c_j, f_i)\, H(Y \mid c_j, f_i), \end{aligned} \tag{2.2}$$

The equation represent that the filtering seeks to minimize the conditional entropy as shown in equation, $H(Y \mid C, F)$ which is the conditional entropy of Y given random variables (C, F) is less than the $H(Y \mid C)$. $H(Y \mid c_j, f_i)$ is the entropy of Y conditioned on particular values $(C = c_j, F = f_i)$ and $p(c_j, f_i)$ is the joint probability of a specific class and feature pair

(c_j, f_i). If we presume that class C and filter F variables are statistically independent, we have $p(c_j, f_i) = p(c_j)\, p(f_i)$ in case of image data absence.

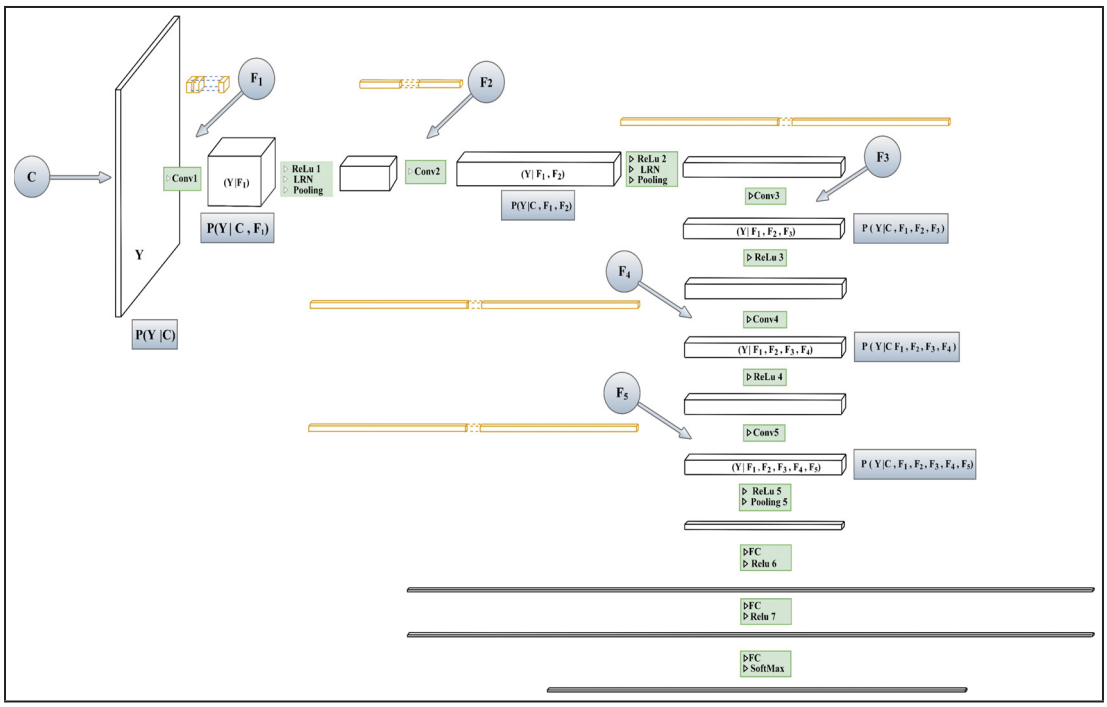


Figure 2.1 The Bayesian probability diagram of VGG which modelizes the non-linear convolution filters F_i as conditional probability $p(Y\,|C, F_i)$ over the input data Y arising from an object of class C

As we know back-propagation algorithm play a highly important role in the success of CNN. It generates the set of highly discriminate filters with respect to a set of training objects C. If we consider the flow of information through the network via the filters F quantified conditional entropy $H(Y \mid C, F)$, the effectiveness of filter discrimination is obtained by tuned filters to image structure characteristic of specific subsets of the objects to be classified. Consequently, filters that capture same structure through all object classes could not be effective in individualizing each object class Chaddad *et al.* (2017, 2019). We can formalize this intuition by partitioning the class set C into two mutually exclusive subset C' and C'' that filter set F is informative and uninformative for them, respectively.

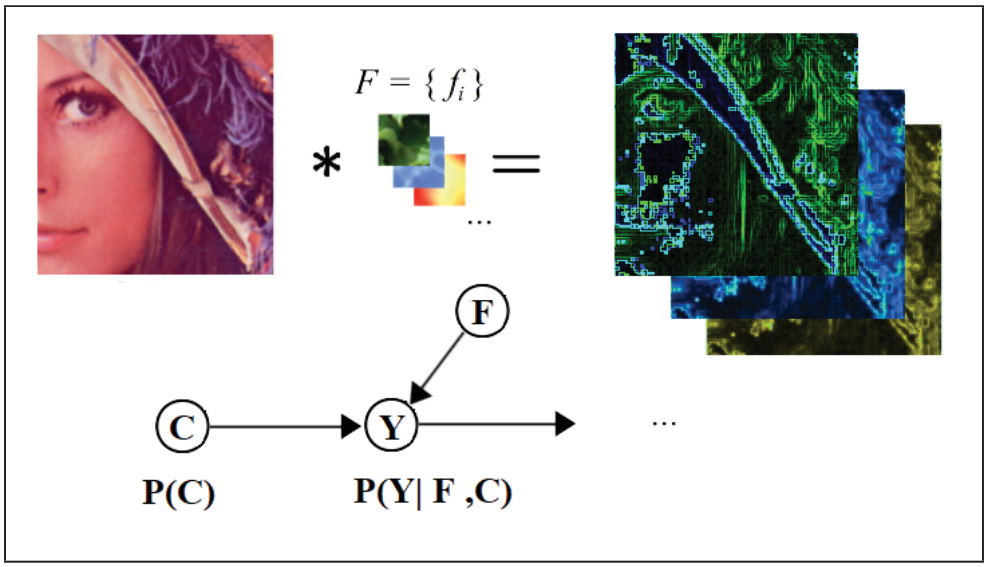


Figure 2.2 The non-linear filter F operation modelization as a conditional probability $p(Y \mid C, F)$ over incoming data Y arising from an object of class C

We can define the conditional entropy as a binary sum over the partition as follow:

$$H(Y \mid C, F) = \tag{2.3}$$

$$p(C')H(Y \mid C', F) + p(C'')\,H(Y \mid C'', F),$$

In accordance with this expression we have:

$$H(Y \mid C', F) < H(Y \mid C'', F). \tag{2.4}$$

For instance, for a specific object class c_j, classifying image from this class needs the neural output Y following a filter set F.

$$H(Y \mid c_j, F) = \begin{cases} \text{low}, & \text{if } c_j \in C' \\ \\ \text{high}, & \text{if } c_j \in C''. \end{cases} \tag{2.5}$$

We can also define the conditional entropy by:

$$H(Y|C, L, F) = \sum_{i=1}^{k} H(Y|C, L, f_i) p(f_i|C, L); \tag{2.6}$$

Where $L = \{l_0, ...l_i\}$ shows the set of convolution layers in a network. $p(f_i|C, L)$ represent the probability of each filter conditioned on class C and layer l_i. We assume the same probability for all the filters of each layer. Therefore the average conditional entropy per layer equals to $H(Y|C, L, F)$ which is lower than the conditional entropy per layer $H(Y|C, L)$ across the response on discriminate layer set L.

This analysis demonstrates that computing the conditional entropy $H(Y \mid C, F)$ across the neural response Y on discriminate filter set F results from CNN training through backpropagation algorithm, leads to exploring an informative feature set that can discriminate between object classes. We refer it as a CENT feature. They can act as a strong codes for capturing the individual filters and filter layers all over the network Chaddad *et al.* (2017, 2019).

2.1.1 Information Theory Analysis in Deep Convolutional Neural Networks

The mathematical framework of information theory gives us a principled means of analyzing deep neural networks, specifically here convolutional neural networks, in order to improve computational performance. Here we analyze the involve information flow through the CNN in terms of information theory.

Input data transformed into recognizable symbols (i.e., object class labels) at the output data while flowing information through a neural network. Along the path neurons and units filter

information before passing it on to other neurons. Our analysis basically related to powerful principles in information theory and CNN technology, in order to derive a set of highly informative features for CNN-based image classification. Here We propose to quantify the flow of information through network filters in terms of the conditional entropy $H(Y \mid C, F)$ of neural activation Y, given object class C and filtering operation F Chaddad *et al.* (2017, 2019).

Our work is most closely links to the so-called IB analysis Shwartz-Ziv & Tishby (2017); Slonim & Tishby (2000), where by modeling layer-wise CNN processing as a Markov chain, rate distortion theory demonstrates how information between input and output network nodes is necessarily limited by intermediate filtering operations.

Shwartz-Ziv & Tishby (2017) extends the work of Tishby *et al.* (2000) and Tishby & Zaslavsky (2015) where they interpret neural networks layers by Markov chain successive internal representations of the input layer X. They analyze the neural network layers in the mutual information plane where they find the mutual information between input and output variable and any other variable. They also declare that an information bottleneck bound gives an optimal representation of input X and optimized DNN. It implies the SGD optimization in two different phases which include the ERM and representation compression where at the end the optimized level lie on optimal BI bound.

Accordingly we provides data processing inequality statement as a final definition. Thus $X \rightarrow Y \rightarrow C$ is a Markov chain, if $I(X,Y) \leq I(X,C)$, i.e. the information shared between endpoints of a network $I(X,C)$ is generally less than that of any intermediate node $I(X,Y)$ Cover & Thomas (2012). In the case of the CNN, this results in a bottleneck for information passing between layers Shwartz-Ziv & Tishby (2017). The reason behind the success of densely connected networks Huang *et al.* (2016) is the ability of this architecture to transmit a greater amount of information to the final classification layer than standard sequential processing, thanks to additional links between all layers and the network output. While the use of information theory to evaluate the suitability of features for classification is acknowledged, our analysis reveals that the conditional entropy itself can be used as highly informative and compact feature.

This intuition has been applied by performing classification based on highly informative CENT features, in which case only small number of data and computation required for dense CNN modeling.

From studies and experiments we can conclude that $H(Y \mid C, F)$ are class-informative codes that can be calculated from the filter outputs through an existing CNN and used to gain higher classification results than the original CNN itself. CENT features identified throughout the CNN can be identified and used directly in classification, thereby bypassing the information bottleneck.

We affirm this assertion in experiment chapter by showing the CENT features performance in multiclass and binary classification tasks.

2.2 Using Principle Component to Reduce CNN Computation

This study is about projecting CNN weight data to linear sub-space which has the lower dimensionality. The aim is to decrease the number of convolution operation in each layer and reduce the computational complexity of linear convolution and saving many computation. As a result we can improve the CNN performance by make it faster.

CNNs has been introduced as a superb classifier in recent year, in machine learning and pattern recognition community. One major drawback of CNNs is the computational complexity and slowness of the network in convolution layer. In proposed model, we use PCA as a tool to reduce the dimension of multidimensional data in filter set to lower dimension while keeping the main information behind the data. In our model the principal components of filter weights in each layer are calculated. In this study we demonstrate by projection of the weights into the space formed by the eigenvalues, we achieve the lower digestion filter that retain the adequate information.

PCA has some advantages over other techniques, first is that it reduces the complexity in images, it has also smaller data base representation and the reduction in noise and redundancy. The main disadvantages of this method is the computational difficulty of covariance matrix calculation Phillips *et al.* (2005) Asadi *et al.* (2010). In proposed model, we aim to reduce the complexity of computation. The experiments shows the reduction in number of filters without a significant drop in accuracy. The compression is obtained using the proposed algorithm. Similar approaches take a place in this regard Garg *et al.* (2019). For instance Dubey *et al.* (2018) display the state of the art compression rate on different CNN models. It exploits the redundancy in the CNN spaces and create a network using quantization and Huffman coding. It achieves the accuracy as the original AlexNet with the 832× smaller memory. Qiu *et al.* (2018) introduces Decomposed Convolutional Filters network (DCFNet) in which the convolutional filters have been decomposed into truncated expansion with pre-fixed bases in the spatial domain. Using functional bases, the number of trainable parameters is reduced to the expansion coefficients.

PCA also is a technique that has been used in area of pattern recognition and image compression, for instance, eigenfaces method in which the features extracted from faces and by linear combination of eignefaces, the faces are reconstructed Duan *et al.* (2008).

2.2.1 Linear Subspace Model

The CNN is parmeterized by layers of filter banks. At a layer, each filter can be represented as a vector $F_i \in R^{W \times H \times D}$, where $W \times H$ is the number of spatial samples and D is the number of input channels. In general, vectors may be projected onto low dimensional subspace, e.g. principal component vectors minimizing the square reconstruction error. The PCA is applied to the set of N filters $\{F_i\}$. Number of rows is assigned based on the number of filters N where each filter is reshaped to a vector and placed on each row.

Considering N filters F_i in the convolution layer, the convolution between image feature map I and the filters defined by:

$$C(F_i, I) = F_i * I \qquad for \quad i = 1, 2, ..., N \tag{2.7}$$

which can be approximated as a weighted sum of K principal components g_j:

$$F_i = \sum_{j}^{K} a_j * G_j \qquad for \quad j = 1, 2, ..., N \tag{2.8}$$

Where K is a threshold on the minimum eigenvalue achieved by PCA and a_j is scalar mixing weights. The size of K is smaller than the number of filters $K \ll N$, however is large enough to hold the main information. Therefore, from equation 2.7, and 2.8, the convolution equation can be written as:

$$C(F_i, I) = F_i * I = (\sum_{j}^{K} a_j G_j) * I \tag{2.9}$$

From distributive property of linear function we have:

$$C(F_i, I) = \sum_{j}^{K} a_j C(G_j * I) \tag{2.10}$$

Equation 2.10 demonstrates that the convolution can be performed between the input I and K number of principal components G_j. Adjusting K can lead to reduction in the number of filters and accordingly reducing computation cost without dropping the accuracy.

Considering I, F and N as the size of input feature map, filters and number of filters respectively. By putting K number of eigenvectors to the convolutional layer weights, we can reduce the computation cost to $(I \times F \times K)$ while the standard convolution cost is $(I \times F \times N)$. Where K is the number of PCA filter.

In the experiment part we will show how we reconstruct the filters using K number of principal components and we will demonstrate that we can compress the number of filters while maintaining accuracy.

CHAPTER 3

EXPERIMENTS

This chapter has been divided in two part. The first and main part 3.1 shows the experiment regarding CENT features performance in multiclass and binary classification tasks. The second part 3.2 devoted to the experiment applying PCA on CNN filters and performing classification with lower number of basis filters.

First section:
In this section CENT features has been demonstrated in different context of CNN classification which consist of two main part. First part analyzes the performance of CENT features derived from an **existed CNN** for classifying different classes and in the second part uses CENT features extracted from the **proposed CNN** architecture in order to classify the subjects based on their age (old vs young) and Healthy control (HC) subjects from subjects diseased with Alzheimer from 3D MRI scans.
At first part 1) In section 3.1.1 the architecture of pretrained CNN (matconvnet-vgg-f) has been illustrated. In the first part we extract the CENT features from existed CNN model for 2D classification of visual object classes in 3.1.2 for unseen categories in existed CNN (matconvnet-vgg-f). In this study we investigate the CENT codes from the aspect of transfer learning. We use 10 categories from Caltech 101 data Fei-Fei *et al.* (2007) which are not existed in ImageNet dataset. Experiments reveals using CENT features leads to improve the classification performance in comparison with soft-max output of the usual CNN.

In another experiment context we use two categories with less significant differences such as painting category Riopelle and Pollock and MR images of healthy subjects and subjects diseased by Alzheimer in 3.1.2.3. In 3.1.2.5 we show the binary classification results for two class that CNN got fooled while recognizing them.

Moreover, we employ Random Forest (RF) algorithm to train the model and classify the considered categories we also aim to visualize the information flowing through the network

using the histogram where each bins represent the importance of each CNN filters. We model this by counting the number of votes from each feature placed in the decision node, selected during training of each decision tree in RF algorithm. As each feature is obtained from one filter, the bins represent the importance of each filter to classify one class across other classes in the CNN. The results and the histogram for each experiment has been demonstrated in related section.

In second part we use volumetric MRI scans of brains to classify the normal subjects from AD and also separating patients by their age. The data is provided from the publicly available OASIS Open Access Series of Imaging Studies (OASIS) data set Marcus *et al.* (2007), Marcus *et al.* (2007). In 3.1.3 we explain about the 3D MRI dataset that we are using for detecting Alzheimer disease vs Healthy ones.

The CNN structure has been described in 3.1.3.1 and section 3.1.3.2 and 3.1.3.3 present the classification results using CENT features for Alzheimer disease vs. Healthy subjects and old and young subjects respectively.

Second Part:
In 3.2 we apply PCA on all layers of VGG model. To find the optimal number of principal component (K) we reconstruct the model with each number of principal components and perform the classification for each model. The results shows the error rate regarding each number of principal component. We will show that we can maintain the accuracy while using lower number of filters in VGG model.

3.1 CENT Analysis

In this section CENT features has been demonstrated in different context of CNN classification. First part analyzes the performance of CENT features derived from an **existed CNN** for classifying different classes.

The second part uses CENT features extracted from the **proposed CNN** architecture in order to classify the subjects based on their age (old vs young) and Healthy control (HC) subjects from subjects diseased with Alzheimer from 3D MRI scans.

The following section we describe the architecture of existed CNN (matconvnet-vgg-f) and the method we use to exploit the conditional entropies across the response of filters at convolution layer. Further we present how we designed the method to illustrate the flow of information and demonstrate the effectiveness of each CNN filter for classification between the group of classes.

3.1.1 Analysis of CENT features efficiency using transfer learning

In recent studies transfer learning emerged as a new learning framework for classification of medical images. It alleviates needs for large dataset as it uses the transfer learning to extract information from images via CNNs originally pretrained for other tasks Huynh *et al.* (2016). Using the knowledge of transfer learning enhance learning efficiency by avoiding expensive data-labeling.

In the work of Huynh *et al.* (2016) the author extract the features of tumor from mammographic images using transfer learning. The author used this approach for the task of discriminating between benign and malignant breast lesions. It compares the features extracted by computers and the ones extracted based on CNN images by support vector machine classifier.

In this study we use 'matconvnet-vgg-f' CNN as a pre-trained model and extract the CENT features from the first five convolution layer output. The VGG CNN has been trained with 1000 object from ImageNet dataset. Chatfield *et al.* (2014),Deng *et al.* (2009)[1] The VGG CNN is composed of 21 layer in total in which there are five convolution layers followed by three fully connected layers. The output of convolution layer passed through the ReLu function and the results are normalized through the LRN layer followed by ReLu layer. The max pooling layer located at the end where pass the results to the next convolution layer. This architecture also include three fully connected layer of convolution type where all are followed by the ReLu

[1] The MatConvNet Matlab software package is used http://www.vlfeat.org/matconvnet/

function and the soft-max layer is located at the end. The input image should have the size $(224 \times 224 \times 3)$. The first convolution layer has 64 filters with the size of $(11 \times 11 \times 3)$. The stride for convolution is 4 and the result pass through the ReLu function and normalization layer and Max Pooling layer. The output would be 265 feature map of size $(54 \times 54 \times 64)$. The second convolution layer has the 265 filters of size $(5 \times 5 \times 64)$. The stride is 1 and again after passing through ReLu function, LRN and soft-max layer there will be 256 feature maps of size $(27 \times 27 \times 256)$. The next convolution layer has the 256 filter map with the size of $(3 \times 3 \times 256)$. The stride is 1 and its output pass through only ReLu function results in a 256 feature map of size $(13 \times 13 \times 256)$ and the last convolution layer has 256 filter maps with the size of $(13 \times 13 \times 256)$ with 1 stride.

In this experiment we use the response of each convolution layer and calculate the conditional entropy by $-sum(p \log_2 p)$. As we know CNN generates 64, 265, 265, 265, 265 feature map from each convolution layer respectively. As each CENT feature extract from one filter map, we will have 1088 (64+265+265+265+265) CENT feature. Figure below 3.1 shows the architecture of vgg.

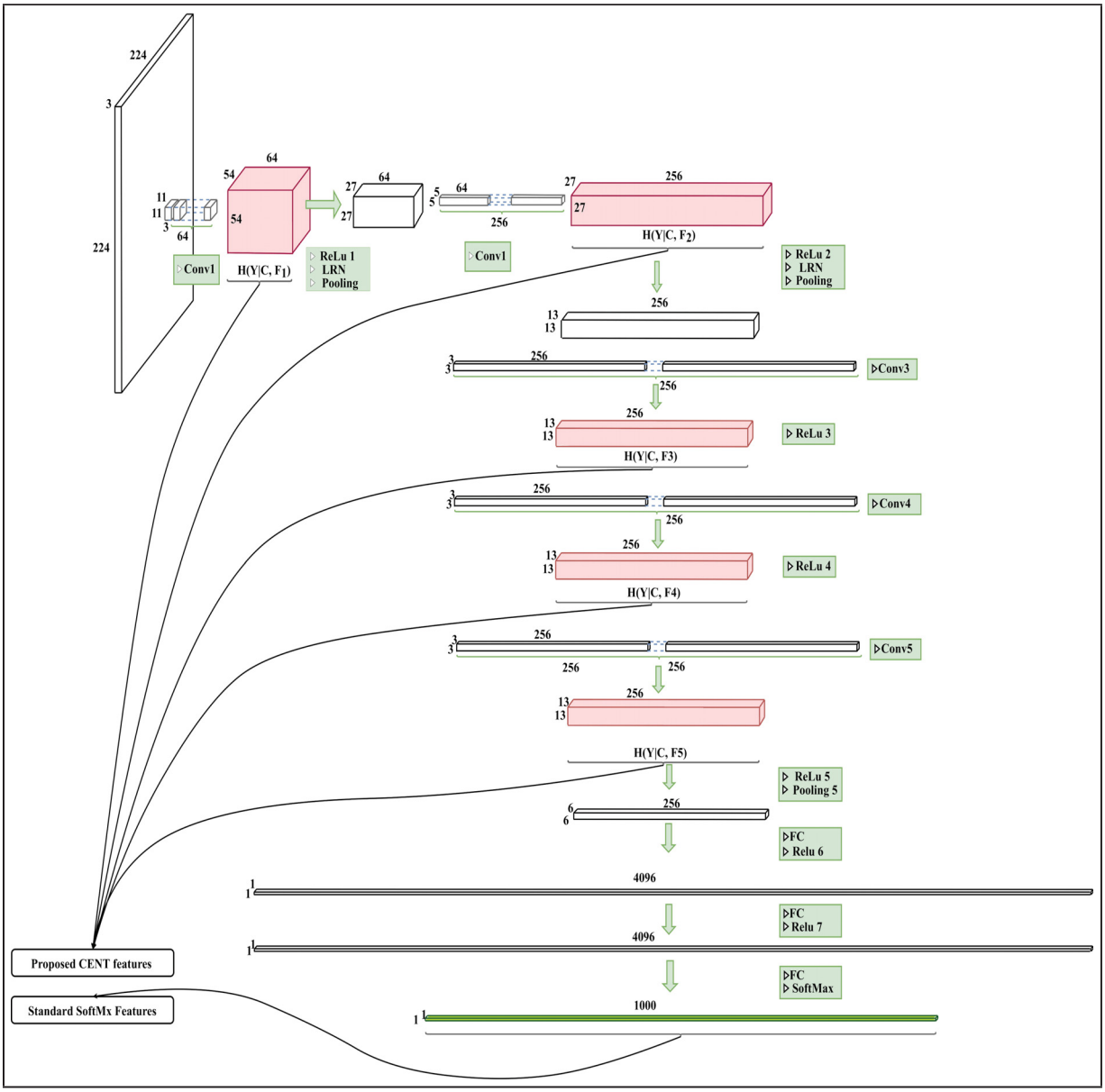


Figure 3.1 21-Layer VGG CNN architecture with convolutional layer (includes ReLu/LRN/Max Pooling) followed by three fully connected layer and at the end one soft-max classification layer a) Illustrates computation of CENT features from feature maps/output for classification b) Illustrates the standard CNN soft-max output used in comparison

The CENT codes obtain from the response of each convolution layer aggregate and create a vector of 1x1088. Therefor for each picture we have 1088 CENT codes. We train these codes with Random Forrest classifier and evaluate the test set with k fold cross validation. We also use

the results from soft-max layer as a 1x1000 codes and train the RF classifier with soft-max codes. In each section a set of classes has been chosen for classification and the Receiver Operating Characteristic (ROC) curves results from CENT features and soft-max has been shown and compared. Using transfer learning in this experiment we understand learned classifier using CENT features could outperforms classification with soft-max codes on a variety of multiclass object classification tasks.

In each section we also display the flow of information through the network for a given class of interest as a histogram of CENT code frequency of RF decision nodes. The semantic information causes the difference between each category in comparison to all other classes participated in training, could be visualized using histogram by counting votes of each presented feature as a significant feature for splitting the categories, generate in RF classifier. The histograms of classes located in the related section to visualize the importance of each specific filters that are responsible for that classification.

We perform classification in a binary fashion. To create a histogram, we go through each tree in a RF and count the number of votes that each nodes generate for both classes. we assign each feature the summation of all the votes it generates. As each feature code links to one filter in the CNN, the effectiveness of each filter contrast by the number of votes. The histograms are normalized by dividing each filters votes by total number of votes.

3.1.2 2D Classification of Visual Object Classes

Here in this section we aim to use the responses of a pretrained CNN to classify an unseen object categories not take a part in objects that existing CNN trained with. We choose the VGG CNN trained with 1000 object from ImageNet dataset. Also any CNN that is trained with the large dataset could be adequate to choose.

3.1.2.1 Classification of 10 categories not used in VGG

In this experiment we aim to test CENT features from the aspect of transfer learning, i.e. classification of object categories not used in network training. For this, we identified 10 object categories that were not used to train the VGG network, i.e. not in the ImageNet data, from the Caltech 101 data Fei-Fei *et al.* (2007). Each categories contain 30 images. The objects are 1) anchor, 2) buddha, 3) chandelier, 4) gramophone, 5) lotus, 6) metronome, 7) minaret, 8) snoopy, 9) stapler, 10) Yin Yang. Images pass through the network and we compute the CENT features across responses of filters at convolutional layer outputs. The first 5 convolution layer are took into account for a total of 64 + 4*256 = 1088 CENT feature. Therefor for each image a vector of 1088 feature is considered as a class informative code. In total we will have a 30 by 1088 CENT codes for each class.

The other step is to extract codes from soft-max to compare the results obtained from transfer learning investigation. The CNN features are taken from soft-max responses of the network train with 1000 categories. Hence the soft-max feature is a 1000-element vector over 1000 training classes. In this experiment soft-max codes should make a proper contribution as an informative code over unseen classes Chaddad *et al.* (2017).

Using RF as a classifier we evaluate the one-vs-all classification with 10-fold cross validation once using CENT features and then by standard CNN soft-max output. The RF is carried out using default setting where the number of splitting variables is equal to the square root of the number of total variables. In this experiment for both CENT and soft-max it is equal to $32 \approx \sqrt{1000}$. We use 400 number of trees for RF in both classification with CENT features and also for soft-max features Chaddad *et al.* (2017). We provide some sample of trees in the appendix.

RF classifier with 5-fold cross validation is used to evaluate one-vs-all classification for a) CENT features and b) standard CNN soft-max output. Default RF settings are used, where the maximum number of splitting variables is equal to the square root of the number of total

variables, here $32 \approx \sqrt{1000}$ for both CENT and soft-max Chaddad *et al.* (2017). The ROC curves results from this experiment have been shown in appendix.

In another experiment we replace two categories in the previous 10 unseen classes. The new categories contain the painting of Jean-Paul Riopelle, a painter and sculptor from Quebec, and Paul Jackson Pollock, an American painter who is known for his technique drip painting, involving pouring liquid paint on to a horizontal surface. Two artist has nearly same artworks, however their technique is different in detail. We substitute these two categories to see if we could obtain a promising results via the CENT features even if two class has significant similarities.

A sample from each ten categories has been shown in 3.2.

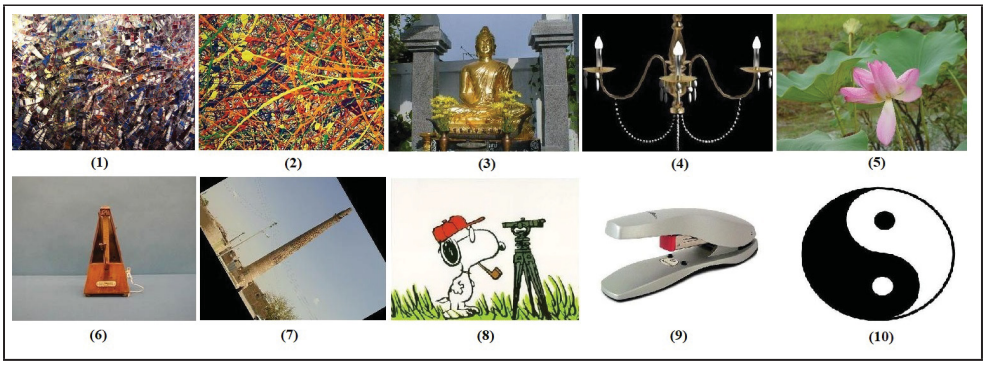
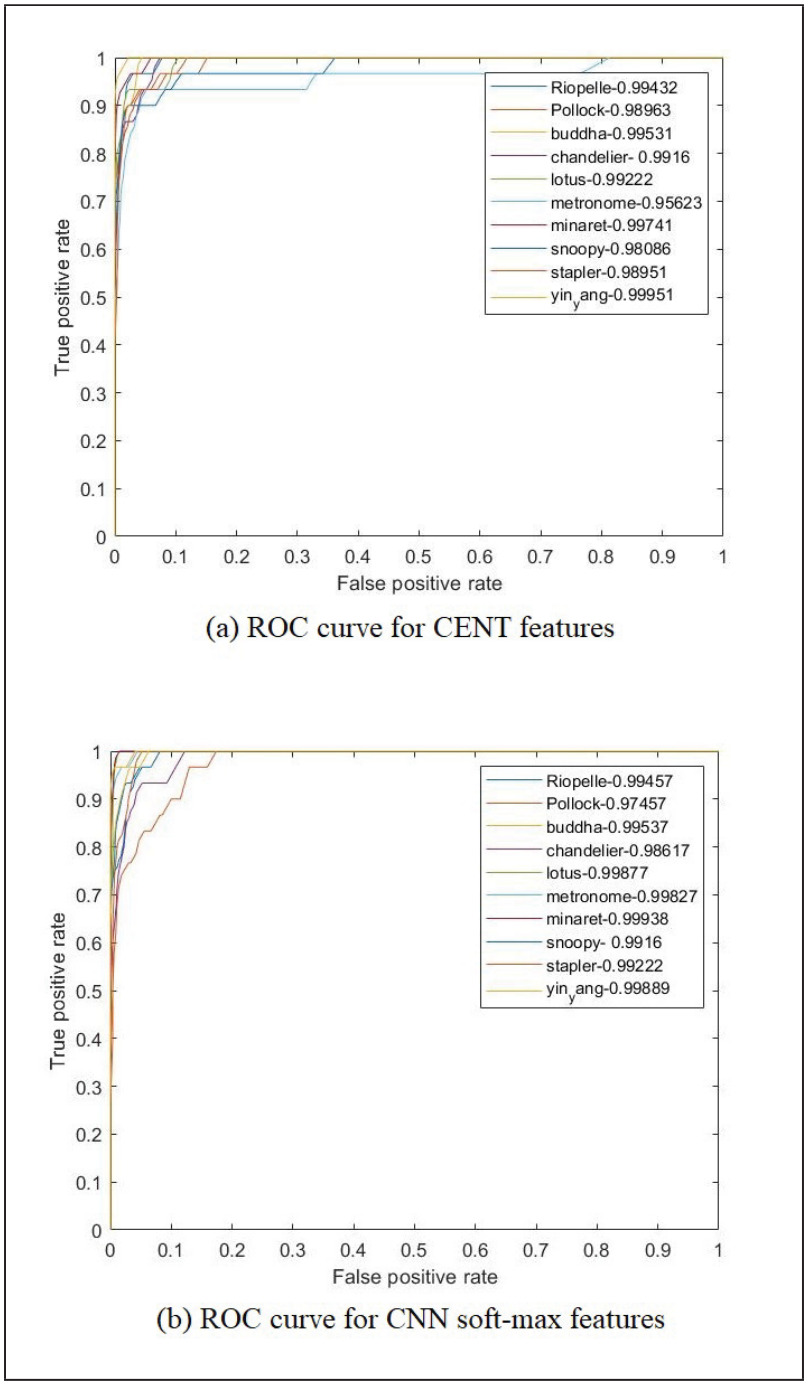


Figure 3.2 Ten unseen categories in VGG from left to right 1) Riopelle Painting, 2) Pollock Painting, 3) buddha, 4) chandelier, 5) lotus, 6) metronome, 7) minaret, 8) snoopy, 9) stapler, 10) yin yang

Figure 3.3 shows the ROC curves for both classification.



(a) ROC curve for CENT features

(b) ROC curve for CNN soft-max features

Figure 3.3 ROC curves transfer learning: classification of 10 natural objects not used in original CNN training. Each image is represented by (a) CENT features computed layer-wise across 5 CNN convolutional layers and (b) the 1000-feature CNN soft-max output

This example shows CENT features could conclude virtually the same AUC using soft-max features. Here we also calculate a T-statistics value for two groups of AUC results gained from the CENT features and soft-max features in order to determine the significance of observed differences between the mean of two groups. Here is the formula used to calculate the paired T Test:

$$t = \frac{\bar{x}_d - \mu_0}{\frac{S_d}{\sqrt{n}}} \tag{3.1}$$

Where $\bar{x}_d$ is the sample mean with the size of n. S_d is the estimate of standard deviation of the sample population and μ_0 is population mean.

The result from paired T test equals $t = 0.9151$ with the degree of freedom equal to $df = 9$ and $standard\ error\ of\ difference = 0.005$. The mean for the AUCs of 10 category usig CENT features is 0.9886600 and the mean for the AUCs of 10 category using soft-max features is 0.9929810. Thus the difference between the mean of two group equals to -0.0043210. The two tail P value equals 0.3840 where by conventional criteria, this difference is considered to be not statistically significant. The results suggests that using conditional entropy of filter outputs throughout the CNN could be an effective method for modeling unseen classes as the experiments demonstrate that there is a high degree of information centered in a CENT features. The most informative CENT features used for RF classifier are related to the CNN filters which contain the most information regarding the class of interest. Therefor for each object category, a unique collection of CNN filters are selected which essentially come from a deeper layers in network.

The histograms are created for all 10 categories which shows the features that are responsible for classifying one category vs others. Related histograms for both 10 category could be find in appendix.

3.1.2.2 Histogram of information and entropy

Here we choose the class of "metronome" from 10 category not used in VGG and show the histogram of VGG CENT feature that are used in RF algorithm. The number of votes in RF for each feature has been displayed in figure 3.4. It shows the importance of information through the whole VGG model. We also display the average of votes and the average of entropy for each convolution layer. As can be seen the average of entropy is getting lower in each layer since the VGG transfer more informative features to the end layers.

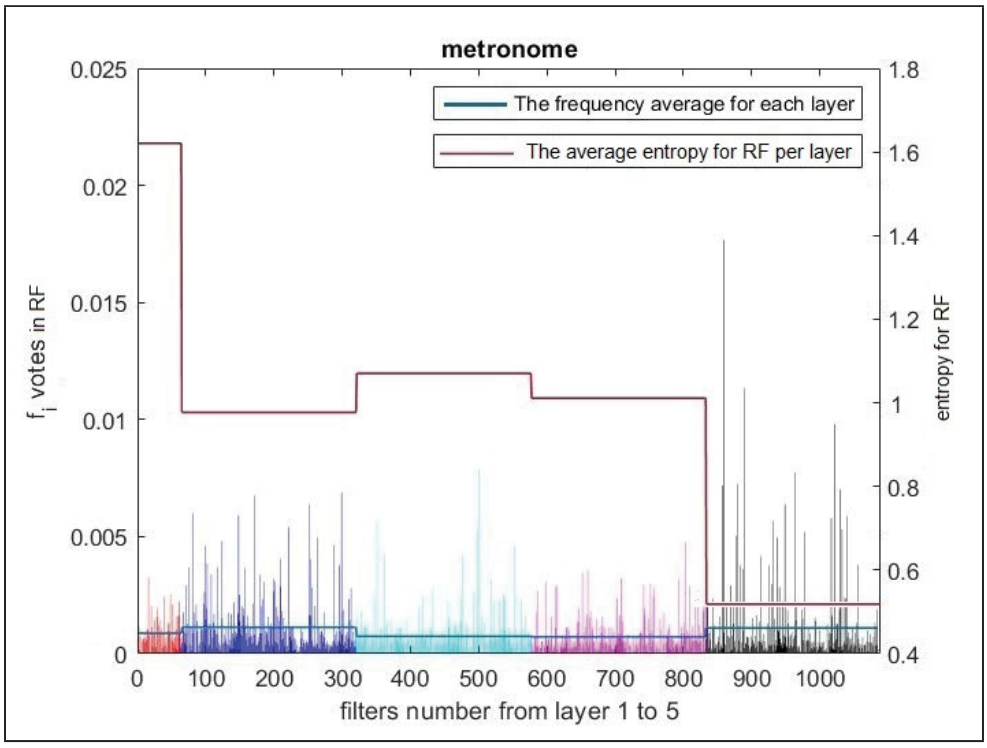


Figure 3.4 Histogram : Histogram of VGG CENT features used in RF for the class of metronome. The blue line shows the average of votes in each layer. The red light related to the average of conditional entropy for RF training per each layer

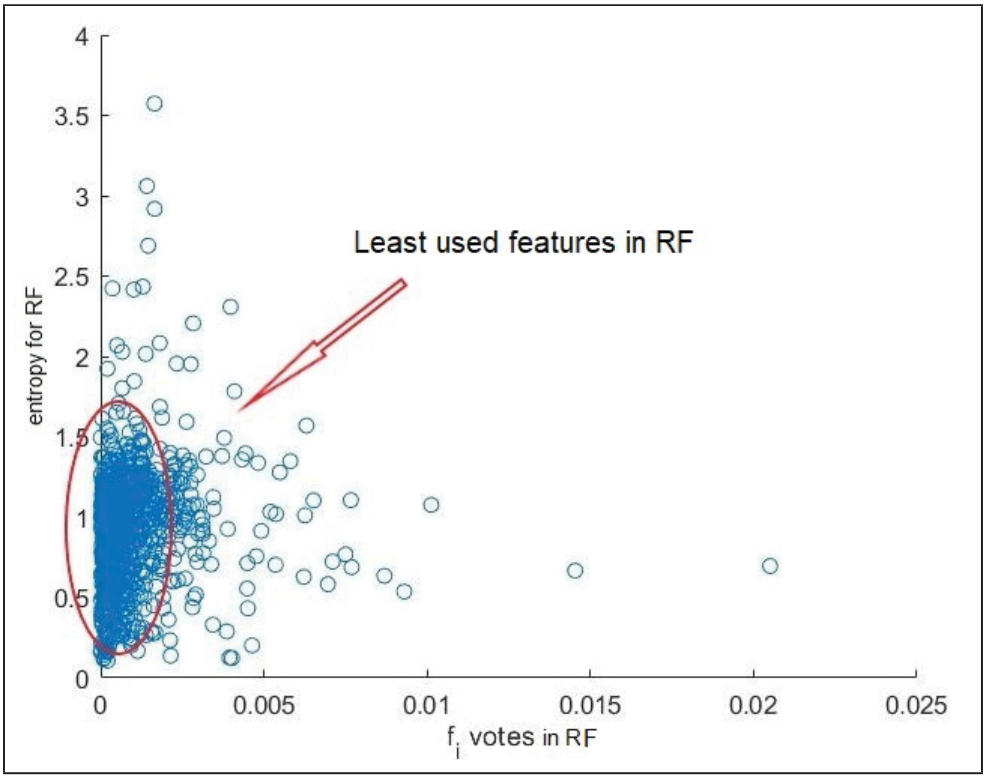


Figure 3.5 Scatter plot of entropy vs. the number of votes The least used features in RF have been displayed with a red circle

Figure 3.5 shows the scattered plot of entropy and the number of votes. It shows the least frequent features in the trees in a red circle. As we can see there is a large number of not used features in RF with low number of votes. In Figure 3.6 we eliminate the least frequent features from the graph.

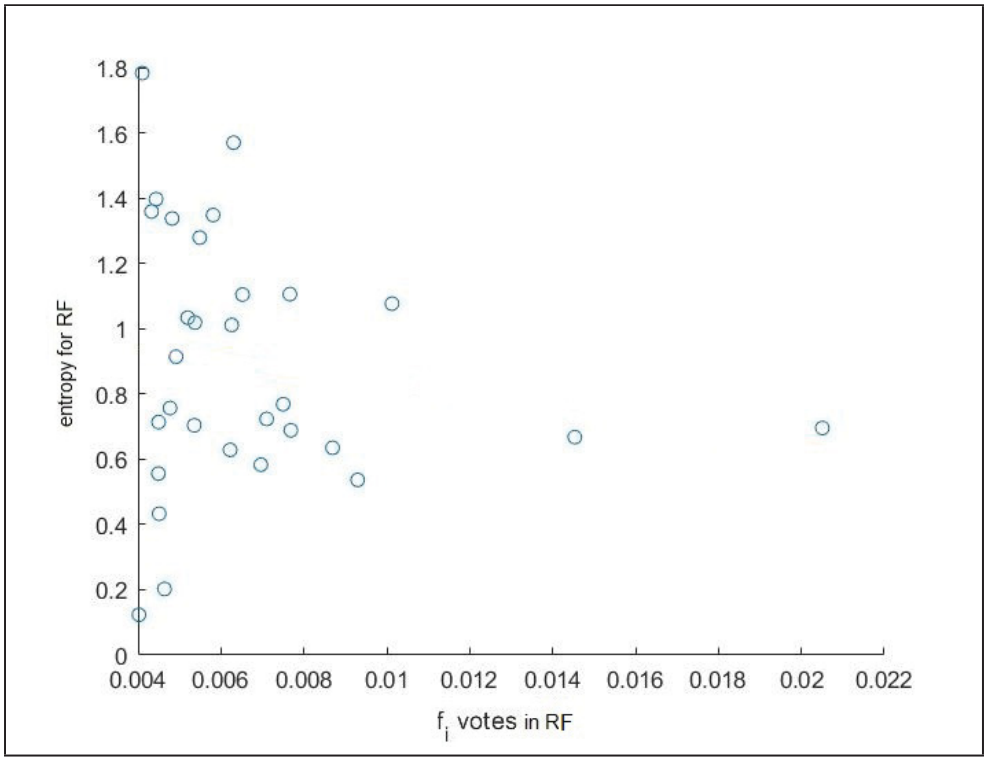


Figure 3.6 Scatter plot of entropy vs. the number of votes The features with less than 0.004 for normalized number of votes has been deleted from the graph

3.1.2.3 2D classification of two painting category

In another trial we use two similar category with insignificant difference, Riopelle vs Pollock painting. It demonstrates the effectiveness of CENT features for binary classification of unseen categories with high degree of similarity. Again we use RF as a classifier and evaluate the binary classification with 10 fold cross validation. We perform the classification via CENT features and soft-max features using 800 number of trees for training. The training set include 50 images from Riopelle paintings and 50 images from pollock dataset.

In figure 3.7 the sample painting of Pollock and Riopelle has been illustrated.

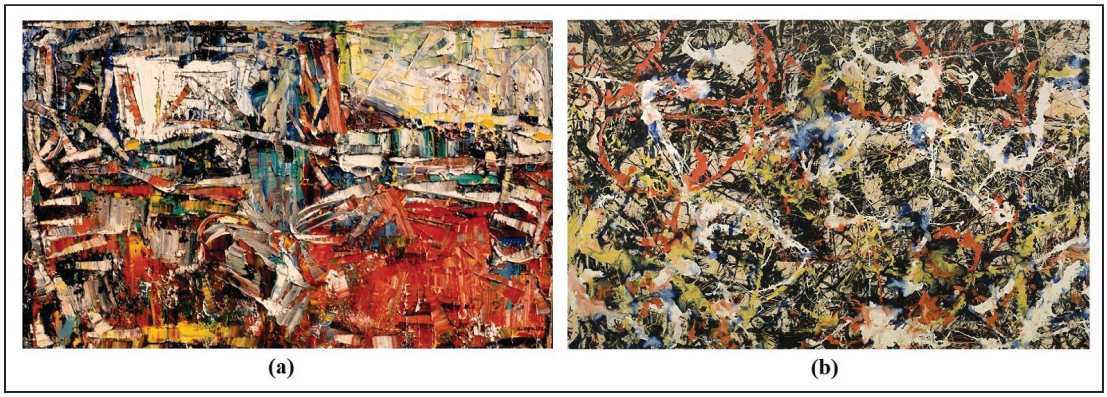


Figure 3.7 Sample painting of (a) Riopelle in the left side and (b) pollock painting in the right side

Figure 3.8 shows the ROC curves for both classification. As can be seen in this figure using CENT features can lead to a supreme classification for Riopelle and Pollock painting. Results obtained by CENT features are slightly better than the one achieved from soft-max. It can demonstrate that informative CENT features in context of transfer learning could lead to even a better classification than soft-max layer.

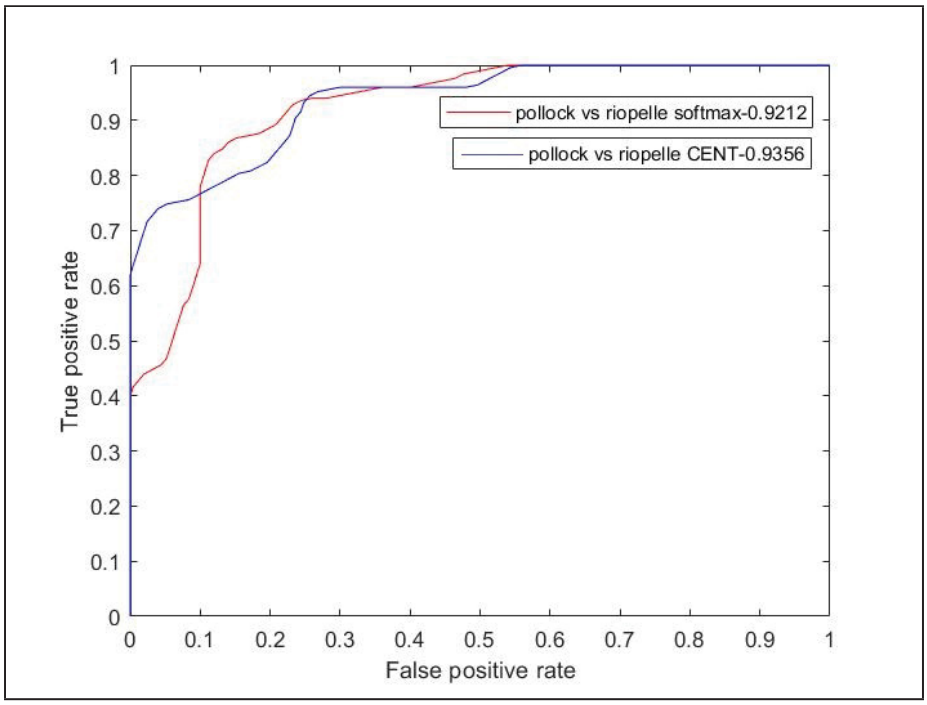


Figure 3.8 ROC curves transfer learning: Binary classification for Pollock and Riopelle painting using RF classifier; Each curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

Figure 3.9 Alleged Pollock painting

We use CENT feature of the painting and send it to the binary trained RF. The classifier recognizes the painting as being 53% of the Pollock, and 47% of the Riopelle. This experiment has been published in Quebec Science Magazine in September 2018 [2].

Histogram of information for Riopelle and Pollock painting

In this section the histogram of flow of information for Riopelle vs Pollock painting has been visualized. We train the model with RF algorithm with 800 trees. We turn on the 'OOBPredictorImportance' feature to store out-of-bag estimates of feature importance in the ensemble.

Fig3.10 displays the histogram for Riopelle and Pollock painting where the information lead to division between them.

[2] The news article which published this study https://www.quebecscience.qc.ca/sciences/art-quand-la-science-mene-enquete/

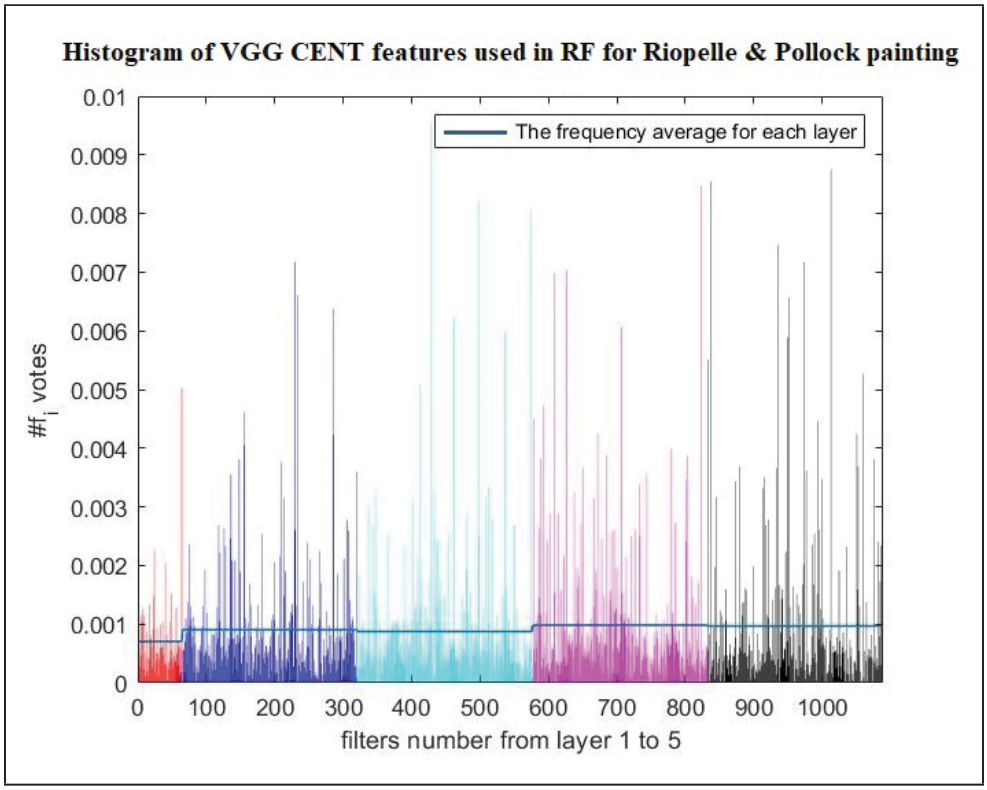


Figure 3.10 Histogram : Histograms of 2 group of painting not used in the original CNN training. Each bin is related to the a filter and shows its importance for classification, The colors indicate each convolution layer, The blue line shows the average of histogram for each layer

3.1.2.4 2D classification of Alzheimer's Disease vs. Healthy subject

In this section we use the 2D MRI scans of subjects diseased with Alzheimer and the healthy subjects for binary classification using transfer learning. Figure 3.11 illustrate the difference between AD subjects and HC subjects. AD subjects brain suffered in some particular parts linked to the areas that form the memory. When the cells of inentorhinal cortex in outer layer of brain dies the connection between the brains cells destroy, which leads to a larger space in brain while the outer layer is shrinking. Hence in subjects diseased with AD are suffered in hippocampuse area and a neuroanatomical structures included in cortex area which is exactly responsible for short term memory. Generally in both AD subjects and naturally old subjects

are recognized by cortical atrophy and enlargement of extra cerebral spaces O'keefe & Nadel (1978). Here we can find the differences in the hippocampus of AD subject vs. HC brain.

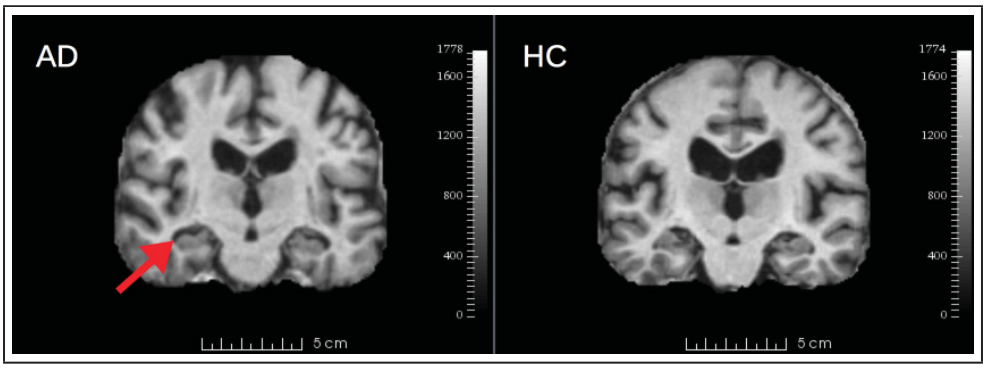


Figure 3.11 Coronal slices of 3D brain MRI illustrating a case of Alzheimer's disease (AD, left) and a healthy control subject (HC, right) A hallmark of AD is atrophy of the cortical surface (red arrow) surrounding the hippocampus, a neuroanatomical structure intimately linked short-term memory formation

For this experiment we use 200 2D slices of brain MRI where half of them are diagnosed with Alzheimer's and others as a healthy one. We use 800 trees to train the data with RF classifier. The ROC curves for Alzheimer and healthy subject can be seen in 3.12. Here we show that the AUC obtained from CENT features and soft-max is approximately similar. It demonstrate that CENT features could be highly determinitive code for classification.

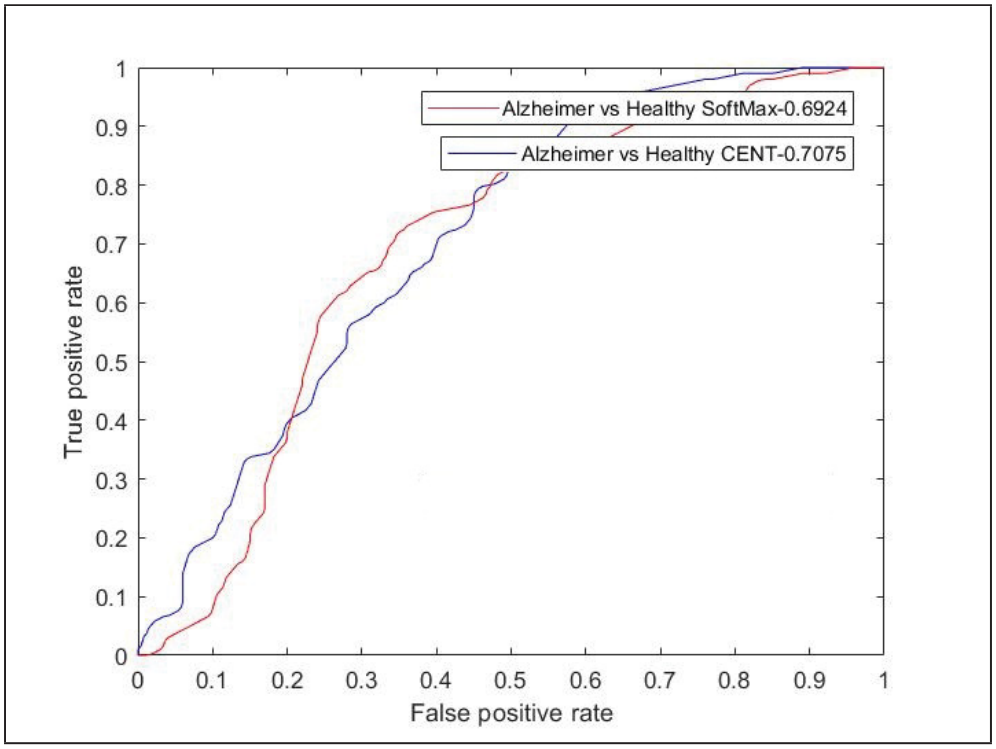


Figure 3.12 ROC curves transfer learning: Binary classification for brain images diseased with Alzheimer and the healthy using RF classifier. Each curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

Histogram of information for Alzhiemer and Healthy

The histogram below shows the importance of each filter for recognizing the Alzheimer in images of brain. Figure 3.13 shows the histogram.

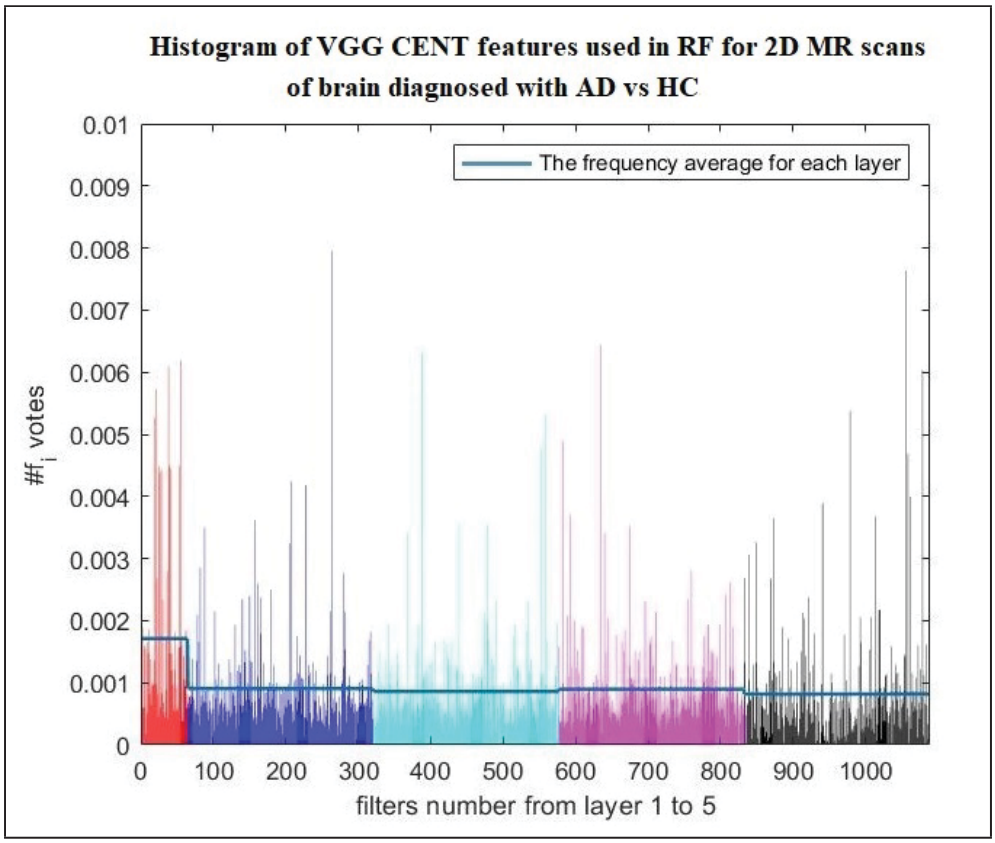


Figure 3.13 Histograms : Histograms of brain images diseased with Alzheimer and the healthy brain not used in the original CNN training. Each bin is related to a filter and shows its importance for classification. The colors indicate each convolution layer. The blue line shows the average of histogram for each layer

3.1.2.5 2D classification of easily fooled classes

Though neural network achieves state of the art in image classification, recent studies reveals the weaknesses of DNNs for classification in some images with imperceptible perturbation. It has been shown that DNN easily get fooled with a little changes Nguyen *et al.* (2015). The new studies in Szegedy *et al.* (2013) mentioned this challenge an intriguing property of DNNs. Liang *et al.* (2018) gives examples of adversarial images that lead to misclassification. The original images are manipulated by introducing them some perturbation.

In this section we select a class of images that are systematically missclassified when noise is introduced. We use a class of bell pepper with 1100 images from ImageNet while all are classified as bell pepper by 'matconvnet-vgg-f' network. We add salt and pepper noise with default noise density of 0.14 to all the images which randomly add black and white pixels and affects approximately 14% of pixels to the original images. The 'matconvnet-vgg-f' Network classify 420 images out of 1100 images as strawberries. We also add noise to a class of strawberries. Figure 3.14 shows the sample images of a bell pepper and a strawberry introduced to the salt and pepper noise which are missclassified by the strawberries via CNN.

Figure 3.14 The sample images of two categories missclassified with strawberry, bell pepper with noise in the left side and the strawberry with the noise on the right side

Using CENT features as a class informative codes, we train RF with the CENT features across responses of convolutional layers filters outputs from the input classes of bell pepper and strawberry without adding any noise as training sets for RF classifier.The classification is also repeated for 1000 element soft-max output. Each category (bell pepper and strawberries with salt and pepper noise) include 420 images which are evaluated by 10 fold cross validation using bell pepper and strawberry categories with salt and pepper noise. The RF is used 800 trees for training images. The ROC curves for both classification has been illustrated in 3.15.

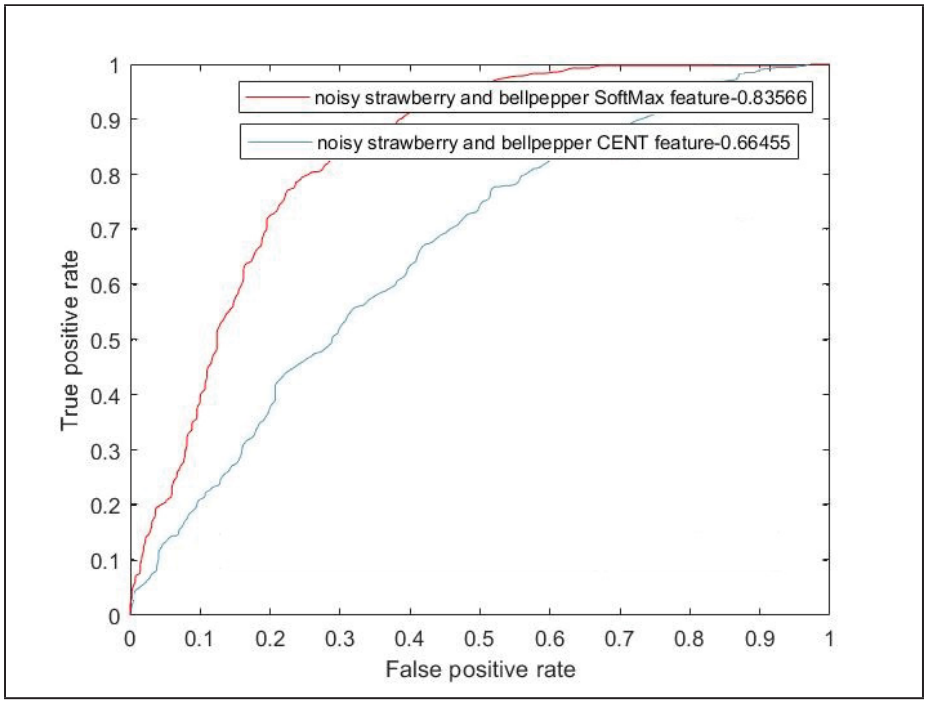


Figure 3.15 ROC curves transfer learning: classification of two categories bell pepper and strawberries. The data has been trained with the original images and the test sets are synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

The results confirm the weakness of CNN in recognition of perturbed images. Here we can conclude that the perturbation has an effect on CENT features extracted from the convolution responses as well which leads to misclassification.

The other experiment is carried out on this dataset using CENT features resulted from the output of convolution layer from the input classes of bell pepper and straw berries with adding noise to all images. The RF classifier with 800 trees evaluates it using 10 fold cross validation. In this experiment both training set and test set are synthesized by noise. The experiment is also repeated for 1000 element soft-max output. The results achieved from soft-max features have a particularly high accuracy and the AUC is slightly better than the results from CENT output. The ROC curves for both classification has been illustrated in 3.16.

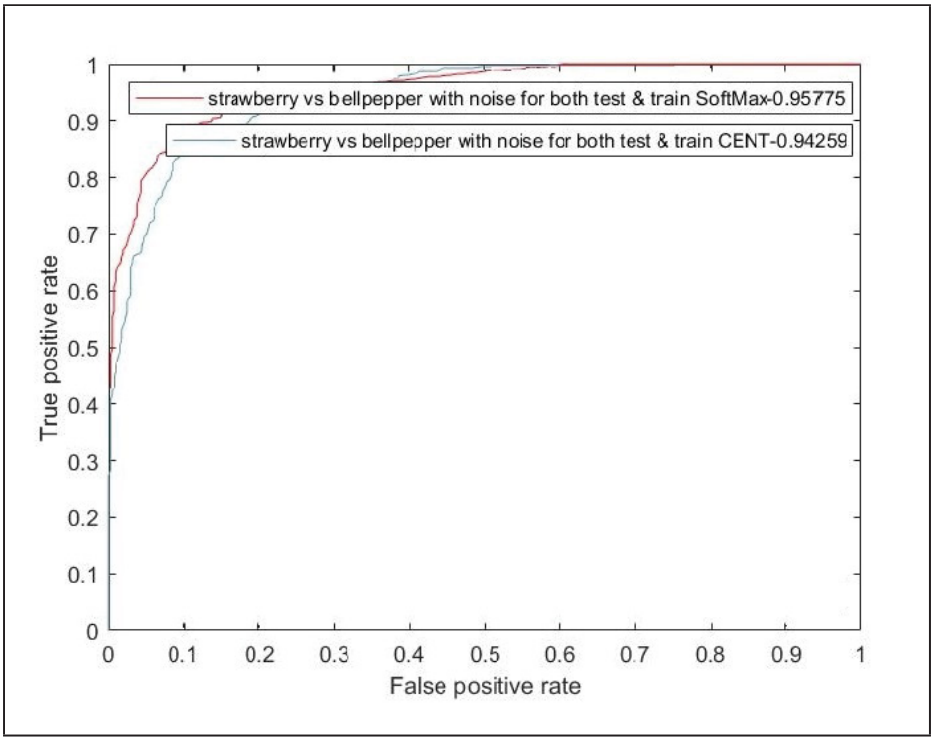


Figure 3.16 ROC curves transfer learning: Binary classification for strawberry and bellpepper with noise (for both test and training set) using RF classifier. Each curve is represented by (Red) CENT features computed layer-wise across 5 CNN convolutional layers and (Blue) the 1000-feature CNN soft-max output

In another trial we synthesize only bell pepper images with noise and keep the original images of strawberries. Similar to other experiments we train RF classifier (800 trees and 10 fold cross validation) with CENT features across responses of convolutional layers filters outputs. We also repeat the classification with soft-max codes. Figure 3.17 shows the ROC curves for classification with CENT features and soft-max features.

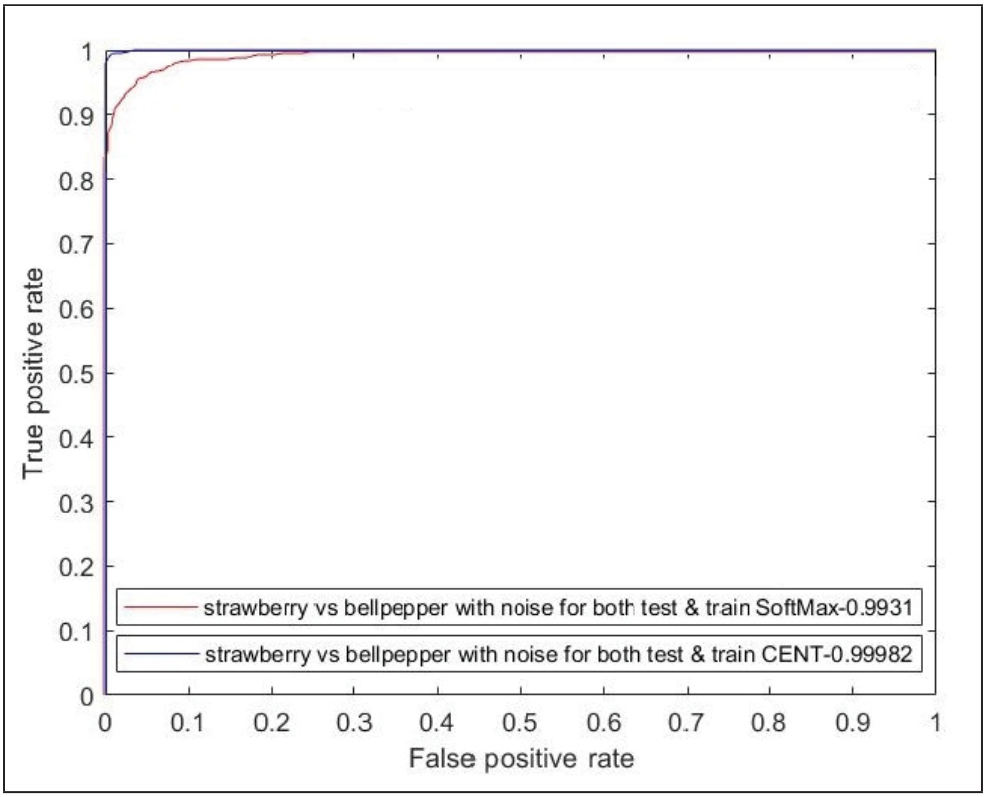


Figure 3.17 ROC curves transfer learning: Binary classification for strawberry and bellpepper, only bellpepper images with noise (for both test and training set) using RF classifier. Each curve is represented by (Red) CENT features computed layer-wise across 5 CNN convolutional layers and (Blue) the 1000-feature CNN soft-max output

The ROC curve for CENT features shows that we could get the best results by retraining the model with noise. As can be seen in the ROC curves using CENT features results in the perfect classification, about 1% better than the classification with Soft-max features. We achieve perfect classification since one class is synthesized with noise and the other is original. As the front layer of convolution layer capture the texture features, CENT features perform better since they include informative codes from frist to fifth convolution layers. In next section we show how front layers of CNN are involved in capturing the noise features. We test this experiment with other noise density pepper and salt noise on the bellpepper images where bell-peppers have been recognized as cucumber and orange. The results has been shown in the appendix.

Visualizing the filters responsible for miss-classification

In this section we choose two categories with 400 images of bellpepper synthesized by noise and the original images of bellpepper. Note that all 400 images of bellpepper are classified as bellpepper by VGG-19 and the images with noise in the other category are classified as strawberries by the VGG-19 output. By training two model using RF with 800 trees, we shows the filters that are responsible for missclassification of bellpepper. As the only difference of two category is the noises added to the images considering that this noise leads to miss-classification of bell peppers as strawberries, the histogram shows the filters responsible for the noises and causes the missclassification of neural network (vgg-matconvenet-f). Figure 3.18 below illustrates the information that lead CNN to missclassification and recognize bellpepper as a strawberry. The histogram obtained from this experience clearly demonstrates the front filters are more concentrated for detecting the noises. Geirhos *et al.* (2018) also demonstrates a high CNN bias to filters at the beginning of the network. We can conclude that the early filters of CNN are more determinative for capturing the textures through the images.

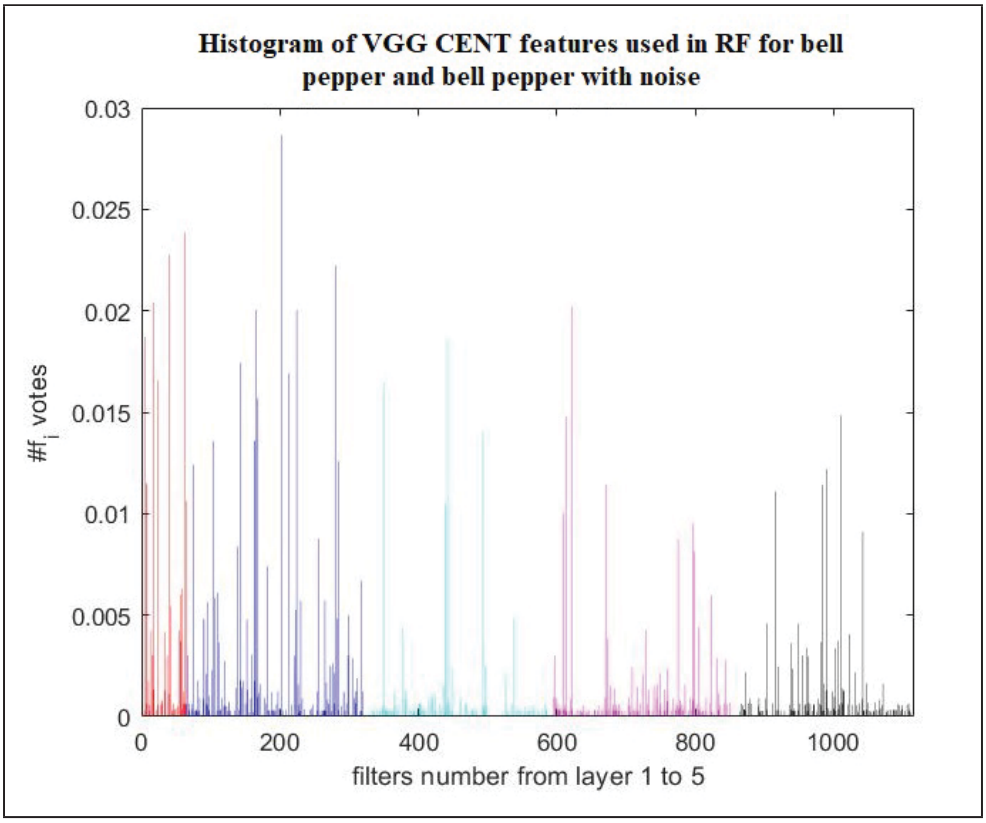


Figure 3.18 Histograms : Histogram of 2 group of bellpepper where one group has been synthesized with noise (VGG classifier recognized it as strawberry) and the other is the original. Each bin is represented number of times each filter were determinitive for classification across all CNN layers for classification across all CNN layers. The bottom histogram split the bins of each layer by different color

3.1.3 3D Image Classification: Brain MRIs

In recent years neurodegeneration analysis is taken into a great consideration due to Alzheimer's disease or natural aging. The need for this analysis is increasing, as the developed countries cope with the growth of population in elderly people. Although new studies affirm that AD subjects are classified with the high precision from brain MRI considering the thickness of cortical Desikan (2009). Nevertheless, grouping MRI subjects using whole-brain MRI data is still a challenging topic Wachinger *et al.* (2016). Currently, the main difficulty is to discriminate

between the healthy older subjects caught with natural atrophy and younger AD subject suffered with minimal atrophy.

Recent studies in Mahmood & Ghimire (2013) presented the highest accuracy nearly 90% in AD diagnosis which uses the mathematical technique for mapping the MRI to one another by reducing the dimensions of MRIs vector space. Comparison to the different models for CNN, Yang *et al.* (2018) used Resnet and accomplished a maximum accuracy AUC = 0.86% based on brain MRI dataset of ADNI Jack Jr *et al.* (2008).

Chaddad *et al.* (2016) has been characterized Alzheimer's disease in brain MRI by employing entropy of derivative filters as a feature to detect the Alzheimer's in brain MRI. It uses local image texture feature filters and applied to the RF classification, however it has never been applied to CNN classification before.

In this study we use the barin MRI of 416 right handed individuals between the age of [18,96] years based on the publicly available OASIS dataset Marcus *et al.* (2007). The number of male and female in the database is approximately equal. Two version of binary classification for first discriminating between AD vs. HC and second old vs. young subjects take a place.

3.1.3.1 3D CNN Architecture

We propose a 4 layer volumetric CNN architecture based on 3D convolution filters based on Chaddad *et al.* (2017, 2019). It consists 2 layer of convolution, 1 fully connected layer and a soft-max at the end. The optimized settings for CNN achieved using the baseline used for similar task for classification Jack Jr *et al.* (2008).

For this model the input image should have the size = $64 \times 64 \times 64$ voxels. First layer has 10 filter with the size = $2 \times 2 \times 2$. The stride for convolution is 2 and the results pass through the Max pooling layer and ReLU function; the output would be 10 feature map of size ($32 \times 32 \times 32$). Second layer consist of 10 filter with the size = $2 \times 2 \times 2$. The stride is 2 and it also consist of Max pooling and ReLU function; the output is 10 feature maps of size ($16 \times 16 \times 16$). The

third layer is a fully conncected layer which the output is the vector size 128. Forth layer is a soft-max with vector size 2. We perform the optimization by stochastic gradient descent with no momentum, default learning rate. The subsampling is occurred through max pooling over 2^3 regions with the default learning rate parameters [3].

As can be seen in this experiment responses from convolution operation generates 10, 10 and 1 individual feature map from each layer respectively which gives us 21 (10+10+1) CENT feature. By aggregating responses at each layer we have 3 (1+1+1) CENT features. The conditional entropy is calculated by $-\sum(p \log_2 p)$ where p includes normalized 256-bin histogram of feature map intensity values.

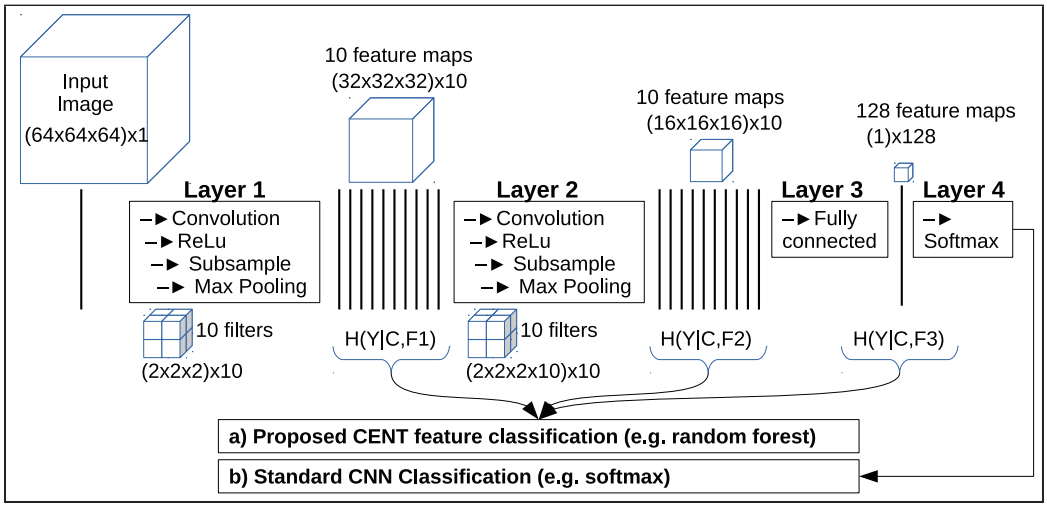


Figure 3.19 4-layer CNN architecture with convolutional layers (includes ReLU/subsampling/max pooling) and followed by 1 fully connected layer and finally 1 soft-max classification layer a) Represent the classification results from CENT features computed from feature maps output using RF b) The results from standard CNN soft-max output

[3] The MDCNN Matlab CNN implementation is used https://www.mathworks.com/matlabcentral/fileexchange/58447-hagaygarty-mdcnn

3.1.3.2 Alzheimer's Disease vs. Healthy Brains

In this experiment we attempt to classify between AD and HC subjects from the brain MRI data using CENT features derived from this CNN architecture Chaddad *et al.* (2017, 2019). We split the data as a subset of a total of 198 (AD=100, HC=98) MRI scans from subjects with age $\geq$ 60 years. The 3D CNN is trained with 50 AD and 50 HC subjects and the rest of images used as a test set in 5-fold cross validation strategy. Using this method, we achieve unbiased evaluation of classification model. In this method we divide training data into 5 subset with equal sized. In each training and testing, we keep one subset aside as a test set and train the data by the remaining 4 subsets.

Two classification model is carried out. First we classify the AD and HC subjects MR images by the CNN explained above. The soft-max classification performed in standard way. The results are compared with the second classification accomplished by RF classifier. RF has been chosen for this task due to its superior performance in general-purpose classification tasks using ensemble of trees. We use RF classifier with 100 tree which is performed efficiently for big dataset with large number of variables Breiman (2001). The ROC curve for both classifiers has been drawn and AUC value is computed by averaging AUC computed across all 5 folds.

Similar to filters that are obtained from natural images in Simoncelli & Olshausen (2001); Simon *et al.* (2007), we also illustrate the distribution of convolution responses. We later use these distribution to compute CENT features. Figure 3.20 shows 10 convolution feature map of second layer. As it can be seen the distribution is centered around zero and reminds the heavy-tail characteristics which denotes to the tails that are heavier than exponential and goes to zero slower.

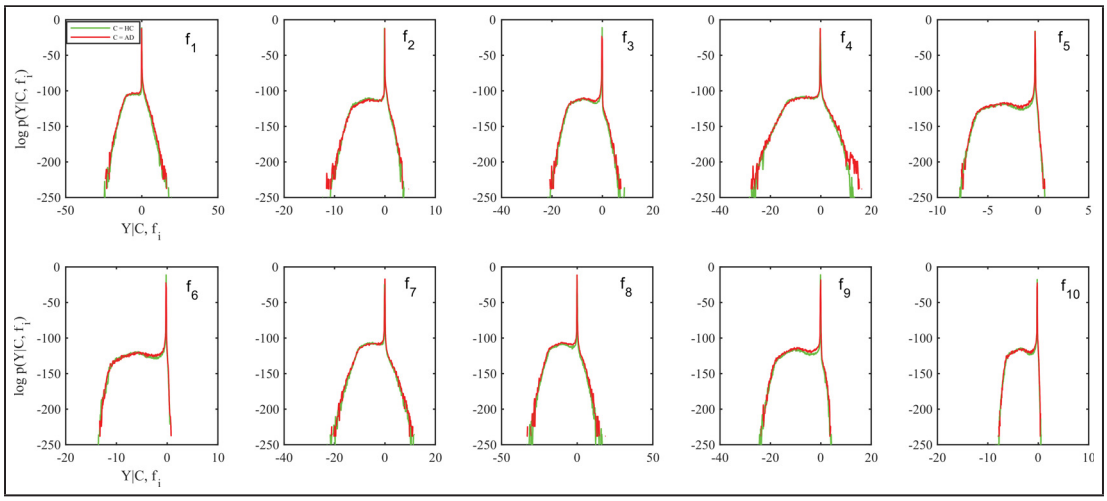


Figure 3.20 Response distributions $p(Y \mid C, f_i)$ for 10 convolutional features maps in layer 2, note that the vertical axis is displayed in logarithmic units

The highest classification results are AUC = 93.6% which is the highest reported classification for AD result achieved from OASIS MRI brain dataset. This results obtained from combination of 3 CENT features computed from all CNN layers. This result is 12% higher than the CNN results from soft-max output. We also compute the AUC for each individual CENT feature for each layer. The ROC curved for each classification has been shown in Figure 3.21. As can be seen figure (a) shows the ROC curve for CENT feature group. The best result achieved from combined 3 CENT feature. It indicate that CENT features of back layer results in a more promising classification and combining all 3 CENT features gives the best result superior than the others. Figure 3.21 (b) is a scatter plot of 3 CENT features of two AD and HC categories which illustrate the class separation clearly Chaddad *et al.* (2017, 2019).

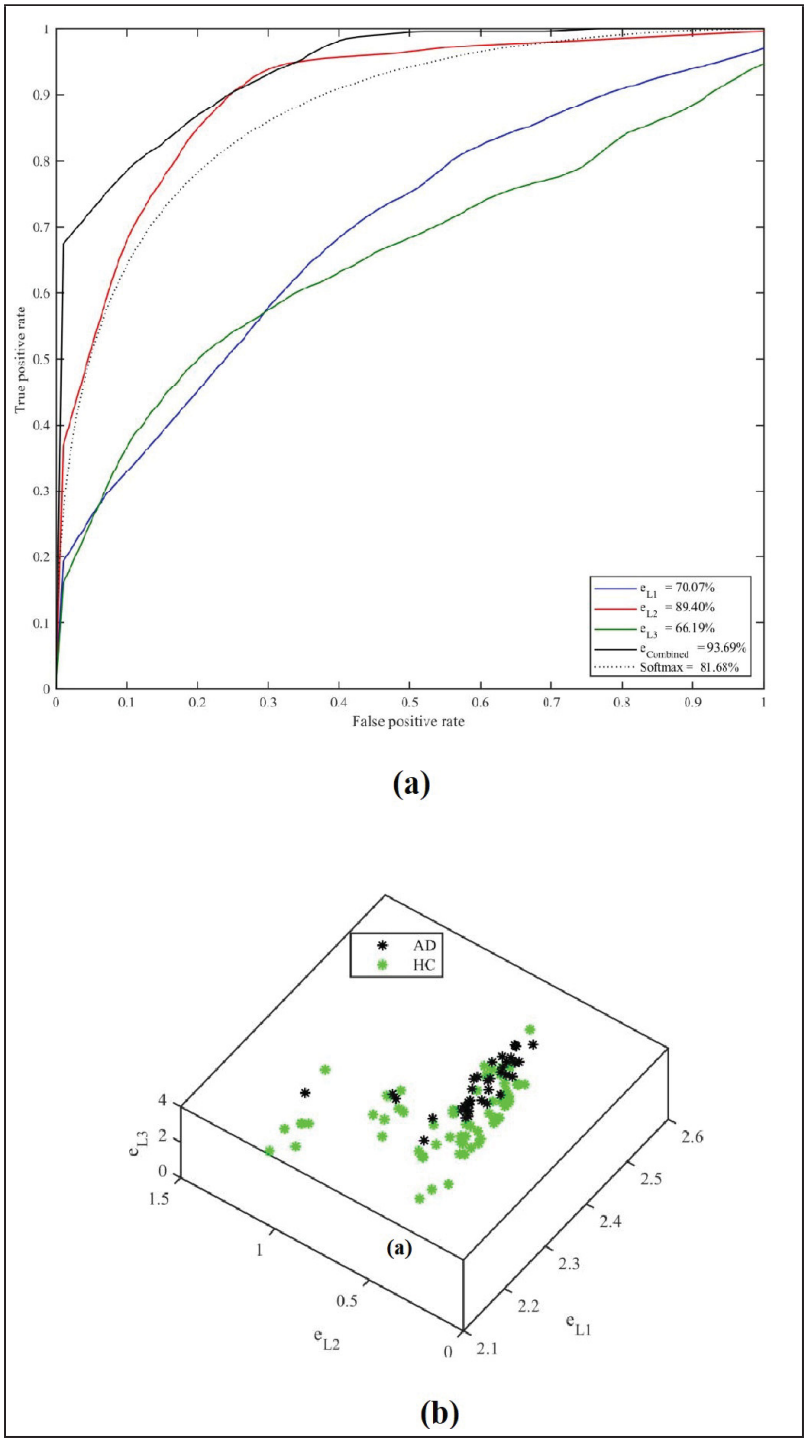


Figure 3.21 (a) ROC curves for AD vs. HC classification using layer-wise CENT features (b) Scatter plot of 3 CENT features showing clear separation between AD and HC classes, each axes denotes to conditional entropy computed at the each CNN layers

Figure below shows ROC curve of classification using each CENT feature computed from each filters individually. So we compute features for 10 filters exist in each layer and carry out the classification. Therefor in figure 3.22 we show the classification using filter-wise features computed from conditional entropy of each filter. Here the blue curves are associated with the 10 filters computed from first layer. The red one is for filters of the second layer which results in a better classification and the green curve is for the single fully connected layer in third layer. Combining all 21 filters and use all the individual filters results in a highest classification with AUC= 93.94%, which is equal to the value achieved by combining layer-wise features Chaddad *et al.* (2017, 2019).

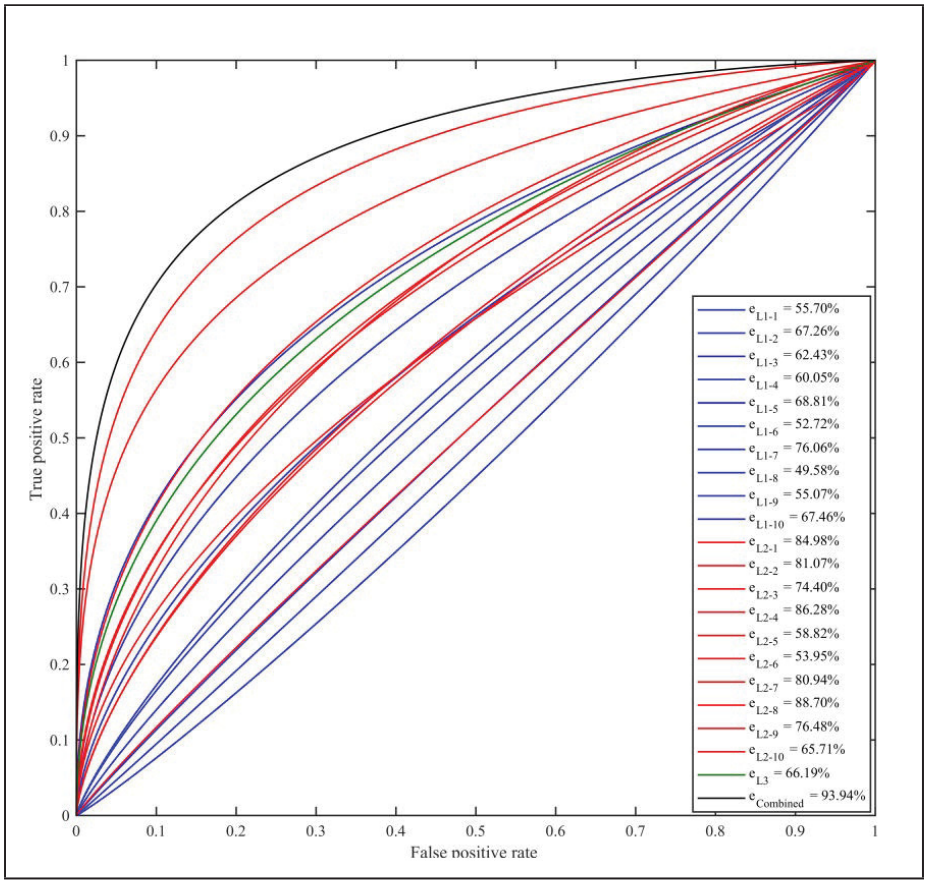


Figure 3.22 ROC curves for AD vs. HC classification using filter-wise CENT features, with 10 filters in layer 1 (blue curves), 10 filters in layer 2 (red curves) and 1 filter in layer 3 (green curve) As in layer-wise CENT classification of Alzheimer's, the most informative features are generally found in layer 2, and highest classification is obtained by combined all features (black curve)

3.1.3.3 Young vs. Old Brains

In this experiment we focused on MRI classification on a variable other than disease. We split the data into old and young age categories. We used a data set of a healthy subjects with 329 MR images. We separate the data using the median age. 200 number of data is used for training (100 for old and 100 for young). The remaining 129 subjects are used as a test set. We evaluate the classification by 5-fold cross validation with the random forest classifier. The same procedure is carried out as we did before for AD and HC classification. The conditional entropy is computed

for all 3 layer. The results indicate that the result obtained across combination of 3 layer-wised feature is about 14% higher than the output of soft-max.

Figure 3.23 shows the results of age classification for 3 layer-wised CENT features. The highest performance achieved from combined 3 features at AUC = 93.34% which has been shown by black line. As can be seen the red line shows the ROC curve for second layer which contain the most informative features. Part (b) of the figure relates to the 3D plot of each layer that shows the clear separation of the young vs. old subjects Chaddad *et al.* (2017, 2019).

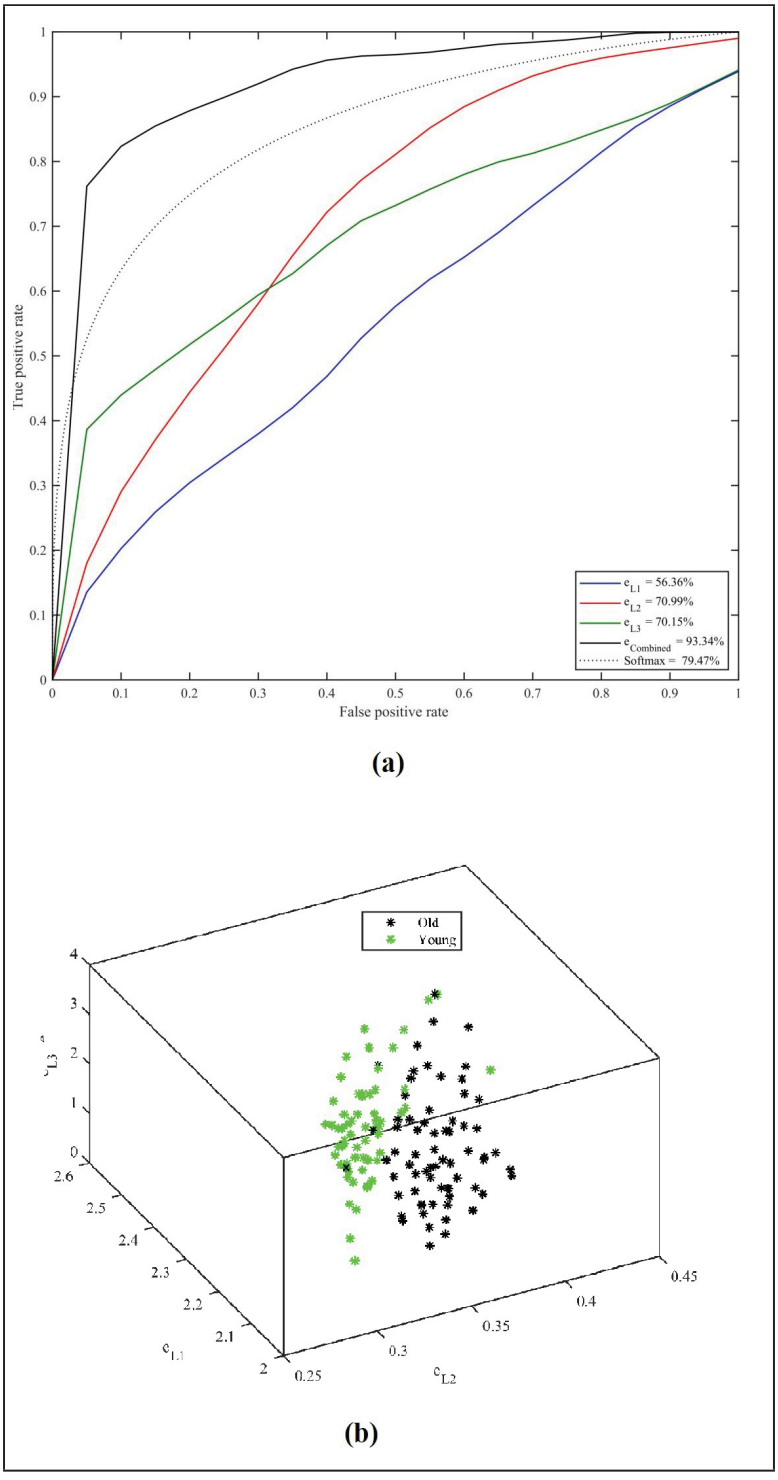


Figure 3.23 (a) ROC curve for CENT feature classification of young vs. old subjects (b) 3D plot of per-layer conditional entropy showing clear class separation

3.2 Classification performance over CNN reconstructed by principal component

In this section we will explain the experiments that we perform to find and optimal number of principal components and to demonstrate that this model can maintain the accuracy with lower number of computation in convolution layers.

We choose (imagenet-vgg-f) pre-trained model Chatfield *et al.* (2014) to run the tests. By calculating the covariance of filters, we will get N number of eigenvalues and eigenvectors. The aim is to reduce the number of filters. Therefore choosing K number of eigenvector can reduce the computation. As an example for the first convolution layer, the size of filter is $11 \times 11 \times 3$ while the number of each filter in our model is 64. Therefore we apply PCA in convolution filters and obtain 64 eigenvalues and eigen vectors. In order to find the best number for principal components without dropping the accuracy, we reconstruct the filters and perform the classification by CNN using new filters. To reconstruct the filters we multiply the scalar mixing weight with eigenvector matrix. In each trial we remove one of the eigenvectors from the matrix and reconstruct the filter. less number of eigenvector results in more lossy compression. The reconstruction formula can be expressed as follow:

$$\hat{F} = \bar{F} + \sum_{j=1}^{M} a_j G_j \tag{3.2}$$

Therefore we perform the classification on 40 class of Caltech dataset where each class includes 30 images. For each convolution layer we reconstruct the filters with different number of eigenvectors. To find the best K we need to find the number of images that are missclassified for each number of eigenvector. We generate the graph which can illustrate the difference between classifications using different number of principle component (K) to reconstruct the convolution filters vs. standard classification with original filters.

The filters are generated for each number of K from 0 to N. We substitute each filter with the original one and perform classification for N times. The number of miss-classified images has been counted in each trial and divided by total number of images to achieved error rate. We

display the error rate for different number of K by graphs shown below. The X and Y axis in the graph shows the number of principal component used for reconstruction K, and error rate respectively.

We apply PCA to all convolution layer filters separately and replace the reconstructed filters by the original one. Figure below illustrates generated graphs related to each of these layer. The number of filters is 64 in first convolution layer and 256 in other convolution layers. The graphs 3.24, 3.25, 3.26, 3.27 and 3.28 show the error rate of CNN vs the number of principal components K needed to keep the main information of the filters in first convolution layer (1st VGG layer), second convolution layer (5th VGG layer), third convolution layer (9th VGG layer), forth convolution layer (11th VGG layer) and fifth convolution layer (13th VGG layer) respectively.

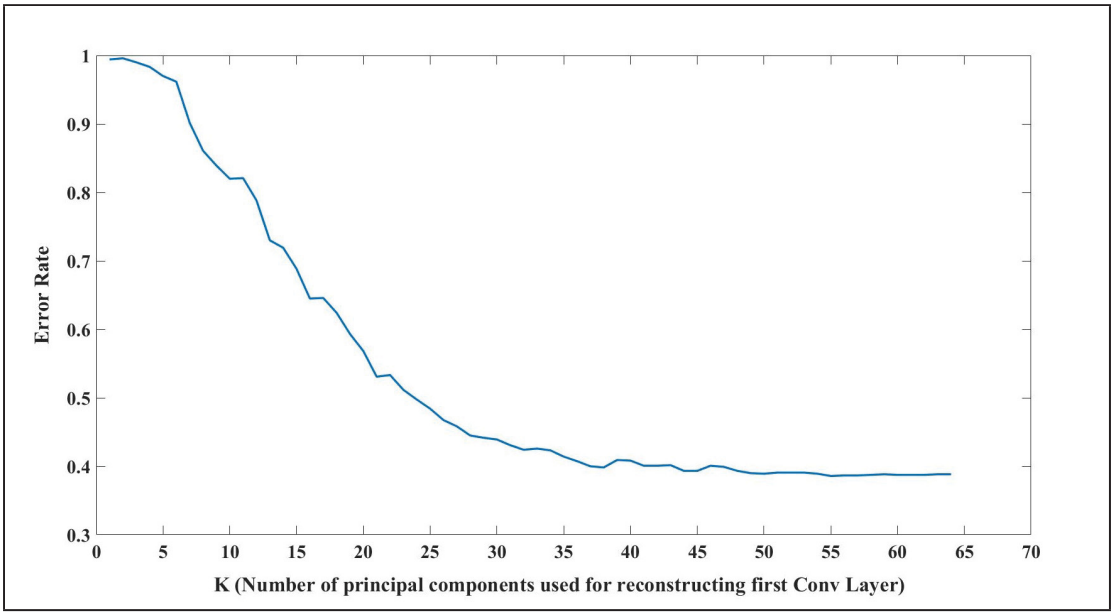


Figure 3.24 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 1, filters = 64

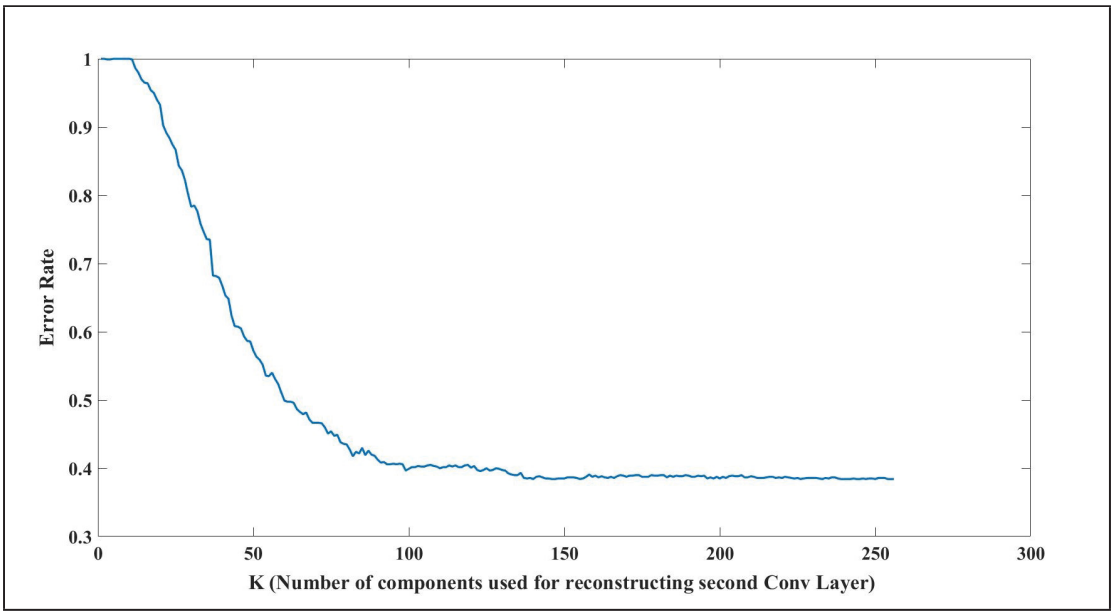


Figure 3.25 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 2, filters = 256

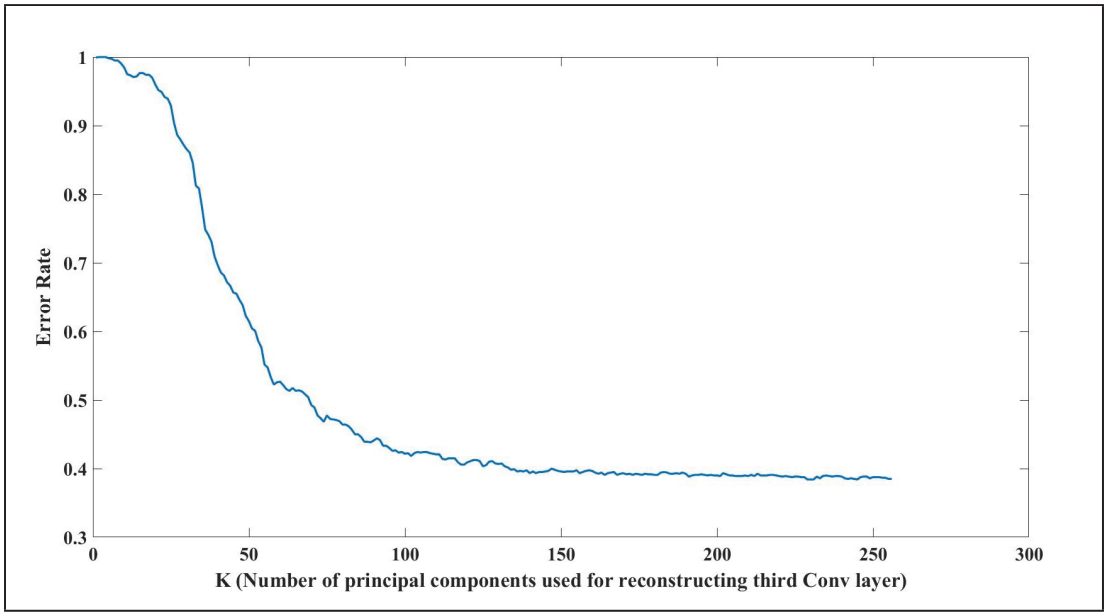


Figure 3.26 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 3, filters = 256

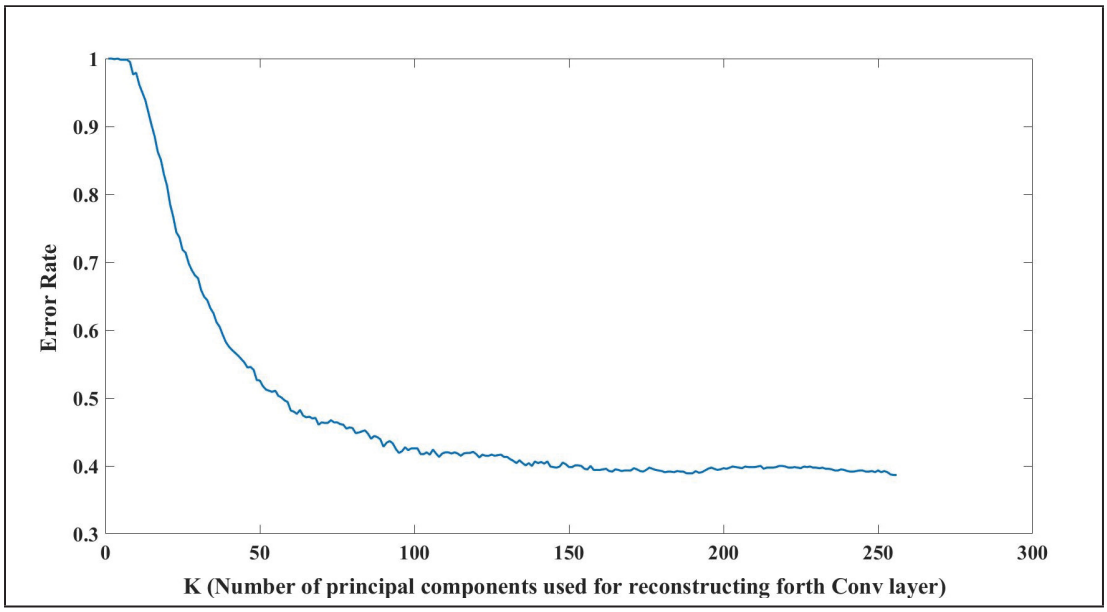


Figure 3.27 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 4, filters = 256

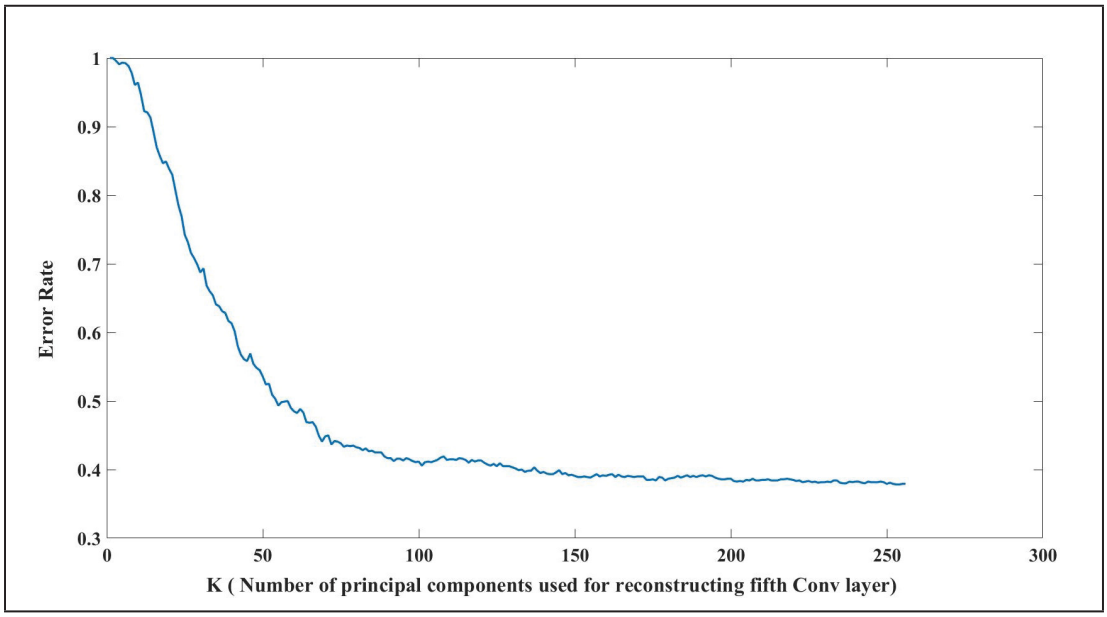


Figure 3.28 Graph of classification error rate vs. number of principal components used to represent filters: VGG convolution layer = 5, filters = 256

We also use only 40 numbers of principal component to reconstruct the first convolution layer and choose different number for principal component from 0 to K for all the other four convolution

layer at a same time. Figure 3.29 illustrate the error rate of CNN while using only 40 number of principal components for the first Conv layer and different number of principal components from 0 to K simultaneously for all the other Conv layer in VGG network. Note that we do not retrain the new model for classification performance. We only reconstruct (compress the filters) and replace them in the old VGG model to check its performance with new compressed filters. We will perform a grid search for the whole number of principal components for all the convolution layer to select a best number of principal components for each layer which can hold the accuracy and also decrease the computation. As a future work we can also train the new model with the an optimal number of principal components.

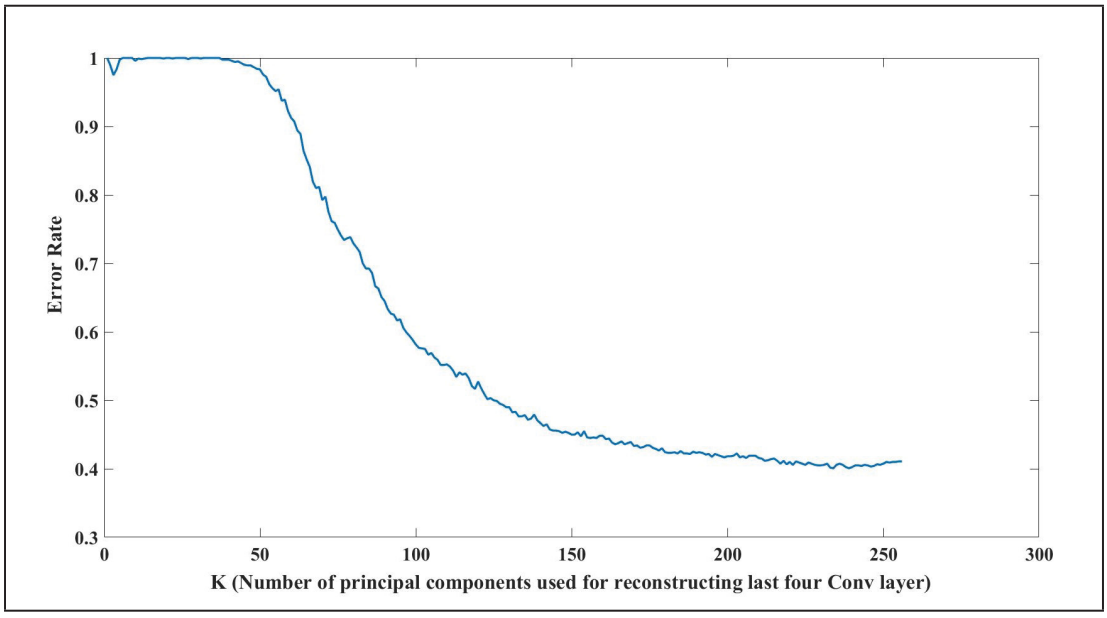


Figure 3.29 Graph of classification error rate vs. number of principal components used to represent filters: VGG Last four convolution layer, for the first Conv layer K = 40

The result indicates that we can compress the filter maps from all the Conv layer and speed up the computation while keeping the main information. As an example, we choose 40 principal components from the first Conv layer and different number from other Conv layer at a same time. The results has been shown in table below.

Table 3.1 Overview of computation saving and changes of error rate for, K = 40 for VGG convolution layer = 1, and K = 150 to 250 for VGG convolution layer = 2, 3, 4, 5 (considering that the original error rate for VGG is 0.3883)

K(Fisrt Conv Layer)	**K(Other Conv Layer)**	**Error rate changes**	**Computation saving**
40	150	+6.17%	52.68%
40	160	+6.00%	48.55%
40	170	+4.50%	44.25%
40	180	+3.58%	39.78%
40	190	+3.50%	35.12%
40	200	+3.00%	30.30%
40	250	+1.92%	3.51%

Figure 3.30 shows computation saving vs. the number of principal components used simultaneously in last four Conv layer.We provide the exact number of operation in appendix.

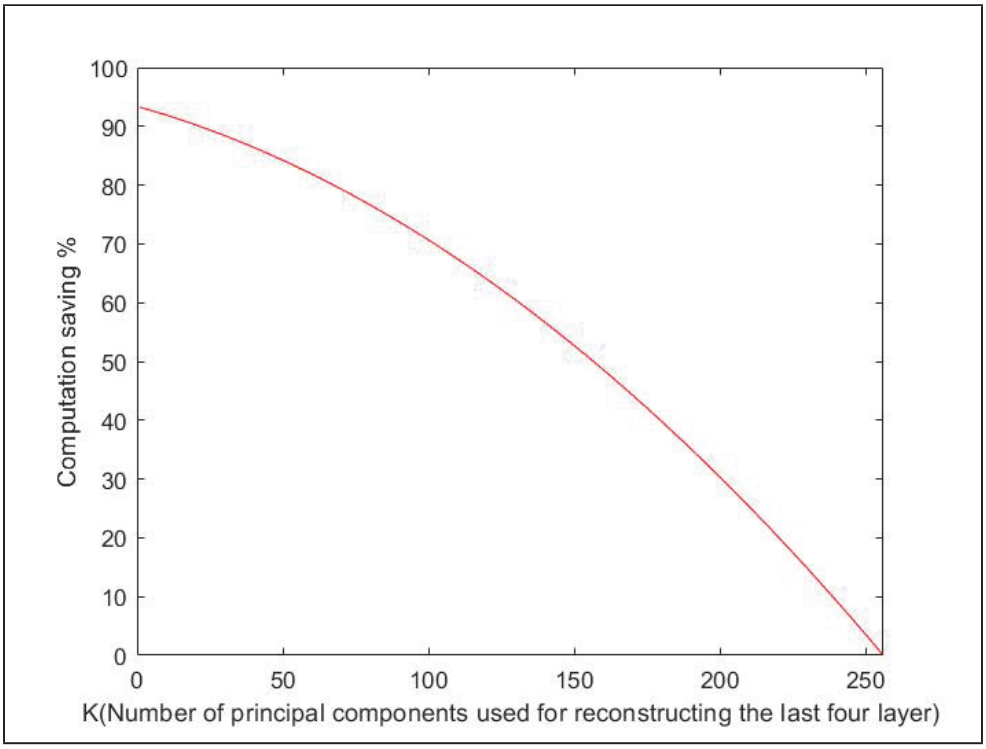


Figure 3.30 Graph of computation saving vs. number of principal components used to represent filters: VGG last four layer, for the first Conv layer K = 40

This approach seems to hold promise to keep the most information while using only 60% of the weights. It guarantees the reduction of computation cost by more than 30% with less than 3% drop the accuracy.

CONCLUSION AND RECOMMENDATIONS

This study analysis the principle of information flow through convolutional neural networks using information theory. We show that a discriminatively trained network filter set F leads to a reduction in conditional entropy $H(Y \mid C, F)$, that is necessarily greater for the subset of object classes for which the filter set is tuned. Both theory and experiments show that the conditional entropy of filter responses, CENT features, result in a highly compact, informative code for image classification.

We also use principal component to reduce the number of filters which leads more than 40% of computation saving with less than 3% drop in accuracy. As a future work we can implement the theoretic model and train the new model.

In experiments using brain MRI data, 3 CENT features computed from each CNN layer result in the highest whole-brain classification rate of Alzheimer's disease reported for the OASIS dataset, with an AUC=93.6%. Most surprisingly, this is 12% higher than the fully connected soft-max output of the the original CNN trained for the task. This success appears to be a consequence of integrating information throughout the network into classification, rather than in a sequential fashion at the output, as predicted by the data processing inequality Cover & Thomas (2012). While this is similar in spirit to the densely connected network approach Huang *et al.* (2016), it is achieved here with only 3 CENT features, rather than thousands of additional dense connections. A similarly high AUC value is achieved for classifying brain MRIs into young and old age categories.

Experiments in transfer learning from 2D photographs show that CENT features computed filters in an existing trained CNN can be used to achieve effective classification for 10 object categories not found in the training data, with ROC AUC values similar to the 1000-element soft-max layer at the output of the original CNN. This performance indicates a promising avenue for low-parameter transfer learning.

It is remarkable that a relatively small number of CENT features computed throughout the network can lead to effective classification. Intuitively, it appears that each CENT feature serves as a bit of information to constrain the identity of the object class, as predicted by a class-discriminative filter set. Thus in the case of ideally discriminative filters capable of partitioning an image unambiguously into one of N categories, a minimum of $\log_2 N$ filters would be required to provide a unique code to each category. This may provide insight into the expected growth of CNN size with respect to the number of object categories, and is an avenue of future investigation.

The idea of analyzing information flow through a neural network using information theory is not new, however to our knowledge it has not been rigorously applied in recent work with deep CNNs. Empirically, one of the reasons why it works so well with modern CNN architectures appears to be that convolution outputs Y are highly normalized/tuned, and may be aggregated across features maps and layers into reliable features. We note that computation of CENT features after vs. before ReLU normalization has a significant improvement on classification. Without ReLU, there is the possibility that bimodal responses Y may be produced for different classes, i.e. highly positive and negative correlations for two different groups of object classes, thus weakening the argument of distinctive class-informative filters F.

Future work will involve further investigation of CENT features in the context of natural object classification from large scale datasets. Conditional entropy computed throughout the network may lead to new optimization techniques in CNN training which seek to maximize information gain of individual filters of filtering layers. Likewise, conditional entropy computed in a spatially localized fashion may prove useful in identifying image regions most informative regarding classification. Alternatively we can use maximum response from each filter as informative codes for classification. In this study we demonstrate the maximum response from output filter of convolution layer contain information since the conditional entropy of maximum responses are

low. We can also use other classifiers like K-Nearest neighbors or other non parametric models to test the hypothesis.Using other pre-trained model of CNN or a combination of them can also be a good way for further study.

APPENDIX I

METHODOLOGY WAVELET

1. Wavelet Analysis

In meanwhile we suggest to perform the convolution in another domain to reduce the computation. We chose wavelet to decompose the input data and convolution filters and perform the convolution in wavelet domain. Due to time constrain we only provide the algorithm that we use for computing convolution in wavelet domain.

Here in this section we present a brief summary on the discrete wavelet transform and review some literature that exploit wavelet approach to reduce the computational complexity. We employ discrete wavelet transform to analyze the convolution of the network in the wavelet domain. We apply Haar wavelet Haar (1910) to the input and convolution layer filters and demonstrate how to convolve them in wavelet domain. The limitations lead to an acknowledgement of the need for further study for this approach.

Based on recent studies wavelet approaches are more practical for physics analysis over traditional Fourier series for the signals contains sharp discontinuities. In sharp signal transitions, large wavelet coefficients are located. Unlike Fourier transform that form from sin and cosine stretched out to infinity and does not localized data in time and space, wavelets form from a short mini wave and can do a good approximation for signals in a finite domain. This feature of localization makes operations in wavelet domain sparse and more practical in data compression, feature detection and other application.

Wang *et al.* (2018) compress the convolution filters by finding common matches in similar filters in frequency domain. A large number of low frequency filters discarded as a less important filters.

Discrete Fourier Transform require $N^2(N \times N)$ multiplication and also $(N-1) \times N$ addition. By exploiting from the alternation property of sine and cosine, the Fourier matrix can be factored into a product of just a few sparse matrices, thus the total operation would be $n \log n$.

Many studies have been made on FFT algorithm, devised by Cooley and Tuckey, to decompose convolution filters such as Highlander & Rodriguez (2016), Nguyen-Thanh *et al.* (2016) and Vasilyev (2015). FFT technique can reduce the number of operation by reducing the step of summation in convolution since inn frequency domain convolution convert to element-wise multiplication.

Wavelets analyze the functions in different scales, therefore both coarse and small features could be captured. As we can define the original signal by a set of coefficients in a linear combination of its functions, it could be a good choice for data compression. Wavelet transform approximates the general signal by the set of different short waves which are characterized by their compact support Graps (1995).

One characteristics of wavelets is that the area underneath of the waves curve is zero. The signal is transformed from the function of time to function of scale and translation by its multiplication with wavelet analyzing function. Translation achieved from shifting the wavelet forward in time. Stretching out the wavelet, lead to lower frequency in higher scaled which is less accurate in time. Conversely by compressing the wave in higher frequency and lower scaled we will have better localization in time. Therefore one of the most important application of the wavelets is compression as the signal can be decomposed to the low frequency part which contain a higher energy and high frequency part which has the least energy. By discarding the high frequency part with less energy, we can obtain the smoother representation of the original signal.

The basis functions of wavelet are a collection of functions that any function can be written with the weighted sum of them. They have the property of linear independence and orthogonality. Hence there are no basis function that can be written as a weighted sum of the other ones and the result of their inner product is zero.

Wavelet is an advantageous way for encoding data as it stores information in layers according to level of details, thus it is easy to approximate the data at any stage and uses less space to store the approximation. With the growth of digital communication the need for storing data and retrieving them is increased. Discrete wavelet transform (DWT) method has been used for compressing the 2D images Talukder & Harada (2010). Here we show the mathematics behind the DWT in 1D and 2D Chun-Lin (2010).

The simplest wavelet basis is Haar basis. It associates with a low pass filter and high pass filters which computes a moving average and a moving difference of the inputs respectively. The low pass filter representation is the approximation coefficient or scaling function $\varphi(t)$, and the high pass filter is called detail coefficient or wavelet function $\psi(t)$. The scaling function $\varphi(t)$ is described by:

$$\varphi(t) = \left\{ \begin{array}{lll} 1 & if & 0 \leqslant t < 1, \\ 0 & otherwise & \end{array} \right\} \tag{A I-1}$$

Where $t \in R$. The Haar wavelet's mother function is represented by

$$\psi(t) = \left\{ \begin{array}{lll} 1 & if & 0 \leqslant t < 1/2, \\ -1 & if & 1/2 \leqslant t < 1, \\ 0 & otherwise & \end{array} \right\} \tag{A I-2}$$

The average of each wavelet function $\psi(t)$ is zero which is defined by:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \qquad j, k \in R \tag{A I-3}$$

Therefore the frame and the basis of scaling and wavelet function could be generated as

$$\varphi(2^j t - k) \tag{A I-4}$$

$$\psi(2^j t - k) \tag{A I-5}$$

Where j shows each "level" that is generated by scaling function. Therefore a linear combination between scaling function defined the wavelets of next level. Hence the solution for the Haar scaling and wavelet function are:

$$\varphi(t) = \varphi(2t) + \varphi(2t - 1) \tag{A I-6}$$

$$\psi(t) = \varphi(2t) - \varphi(2t - 1) \tag{A I-7}$$

As an example if we map the set of input data X with N values to approximation and detail wavelet coefficients

$$\varphi_N = 1/2(X_{2N} + X_{2N-1}) \tag{A I-8}$$

$$\psi_N = 1/2(X_{2N} - X_{2N-1}) \tag{A I-9}$$

Where N is even and $t = 1, 2, ..., \frac{N}{2}$.

1.1 Discrete Wavelets Transform in 2D Images

To apply wavelet transform to the image we first apply the filter bank to the rows of the image and then repeat the procedure to the columns of the results. So the result from scaling function would be an approximation of the image and three high pass channels from wavelet function

corresponding to vertical, horizontal, and diagonal.

$$\varphi(x, y) = \varphi(x)\varphi(y) \tag{A I-10}$$

$$\psi^V(x, y) = \varphi(x)\psi(y) \tag{A I-11}$$

$$\psi^H(x, y) = \psi(x)\varphi(y) \tag{A I-12}$$

$$\psi^D(x, y) = \psi(x)\psi(y) \tag{A I-13}$$

The scaled and translated basis function also can be written as:

$$\varphi_{j,m,n}(x, y) = 2^{j/2}\varphi(2^j x - m, 2^j y - n) \tag{A I-14}$$

$$\psi^k_{j,m,n}(x, y) = 2^{j/2}\psi^k(2^j x - m, 2^j y - n) \tag{A I-15}$$

Where k the three H, V, D part.

In order to show the transformed image, the whole area is partitioned to four sub-sampled spaces. Figure I-1 shows an example of the 2D DWT.

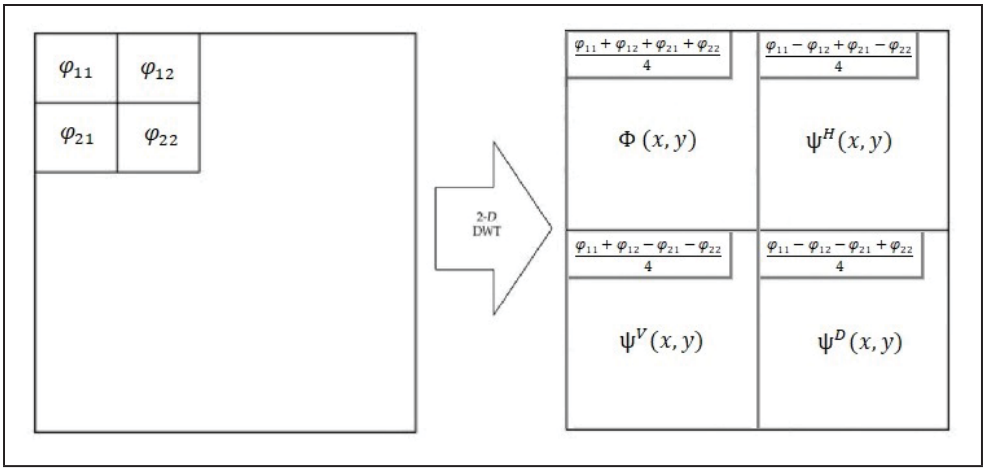


Figure-A I-1 First level 2D DWT using Haar wavelet

As we can see the north west part shows the approximate coefficients $\phi(x, y)$, the northeast shows horizontal detail coefficients the $\psi^H(x, y)$, the south west shows the vertical detail coefficients $\psi^V(x, y)$ and the south east shows the diagonal detail coefficients $\psi^D(x, y)$. In 2D Haar wavelet transform approximate coefficients of each level is four times more compressed than the one in previous level. Therefore each pixel of coefficients at scale j would be:

$$\varphi_j(x,y) = \frac{\varphi_{j-1}(2x-1,2y-1) + \varphi_{j-1}(2x-1,2y) + \varphi_{j-1}(2x,2y-1) + \varphi_{j-1}(2x,2y)}{4} \tag{A I-16}$$

$$\psi_j^H(x,y) = \frac{\varphi_{j-1}(2x-1,2y-1) - \varphi_{j-1}(2x,2y-1) + \varphi_{j-1}(2x-1,2y) - \varphi_{j-1}(2x,2y)}{4} \tag{A I-17}$$

$$\psi_j^V(x,y) = \frac{\varphi_{j-1}(2x-1,2y-1) + \varphi_{j-1}(2x,2y-1) - \varphi_{j-1}(2x-1,2y) - \varphi_{j-1}(2x,2y)}{4} \tag{A I-18}$$

$$\psi_j^D(x,y) = \frac{\varphi_{j-1}(2x-1,2y-1) - \varphi_{j-1}(2x,2y-1) - \varphi_{j-1}(2x-1,2y) + \varphi_{j-1}(2x,2y)}{4} \tag{A I-19}$$

The process of wavelet transform could be iterated and produce detail coefficients and approximate coefficient for the scales of $j-1, j-2, j-3, \ldots$. The low pass filter bank in wavelet transform

describes the low frequency information of the image which has the higher energy. The high pass filter construct the detail components which contain the image's high frequency information.

1.2 Convolution in Wavelet Domain

Considering the input image of I with N by N pixel and the filter set $F(n_1, n_2)$ the convolution formula is:

$$I(x, y) * F(x, y) = \sum_{n_1=0}^{N} \sum_{n_2=0}^{N} I(n_1, n_2).F(x - n_1, y - n_2) \tag{A I-20}$$

In order to convolve them in scale domain we first transform the filter and image to wavelet domain and iterate the procedure till we get the approximate coefficient with only one pixel for each filter. We can start the convolution from the coarsest level by convolving the same coefficients type of both filter and image together.

Considering transforming both filters and image for P times, we must compute the wavelet decomposition of the image after deleting rows and columns in a specific sequence and decompose new version of image. We need all coefficient obtained from new images to be convolved with filter coefficients to recover the convolution result of approximate coefficients of previous level, which should be located between the coefficient pixels obtained from original image. Therefore to recover the convolution we should consider four state for input image.

If we consider stride 1 for the original convolution, and the origin of x and y at the top left of image, we need to calculate:

$$\begin{aligned} 4\times(&\varphi_j^{I(x,y)} * \varphi_j^{F(x,y)} + \psi_j^{VI(x,y)} * \psi_j^{VK(x,y)} \\ &+ \psi_j^{HI(x,y)} * \psi_j^{HF(x,y)} + \psi_j^{DI(x,y)} * \psi_j^{DF(x,y)}), \end{aligned} \tag{A I-21}$$

$$4\times(\varphi_j^{I(x+1,y)} * \varphi_j^{K(x,y)} + \psi_j^{VI(x+1,y)} * \psi_j^{VK(x,y)} + \psi_j^{HI(x+1,y)} * \psi_j^{HK(x,y)} + \psi_j^{DI(x+1,y)} * \psi_j^{DK(x,y)}), \tag{A I-22}$$

$$4\times(\varphi_j^{I(x,y-1)} * \varphi_j^{K(x,y)} + \psi_j^{VI(x,y-1)} * \psi_j^{VK(x,y)} + \psi_j^{HI(x,y-1)} * \psi_j^{HK(x,y)} + \psi_j^{DI(x,y-1)} * \psi_j^{DK(x+1,y)}), \tag{A I-23}$$

$$4\times(\varphi_j^{I(x+1,y-1)} * \varphi_j^{K(x,y)} + \psi_j^{VI(x+1,y-1)} * \psi_j^{VK(x,y)} + \psi_j^{HI(x+1,y-1)} * \psi_j^{HK(x,y)} + \psi_j^{DI(x+1,y-1)} * \psi_j^{DK(x,y)}) \tag{A I-24}$$

Each approximation coefficients at first level can also be constructed from the convolution between the coefficients of next level. To calculate the approximation coefficients of next levels we need to apply wavelet transform after removing some columns and rows in the original image in a way that for each subsequent levels, the number of jumps in the original image is multiplied by two. For other three part the obtained number is added by one, first in x axis, second in y axis and third in both side. This will gives us a tree with four branch where each form from four other branches.

We can also conclude that the convolution in each level is equal to the convolution in original image with different stride based on the considered level. For instance, if the stride is 4 in original image, convolution in the first level would be equal to that if the stride is 2.

Continuing procedure repeatedly results in a tree. Figure I-2 shows the tree of approximate coefficients for two levels (we discard detail coefficients in the tree due to the lack of space):

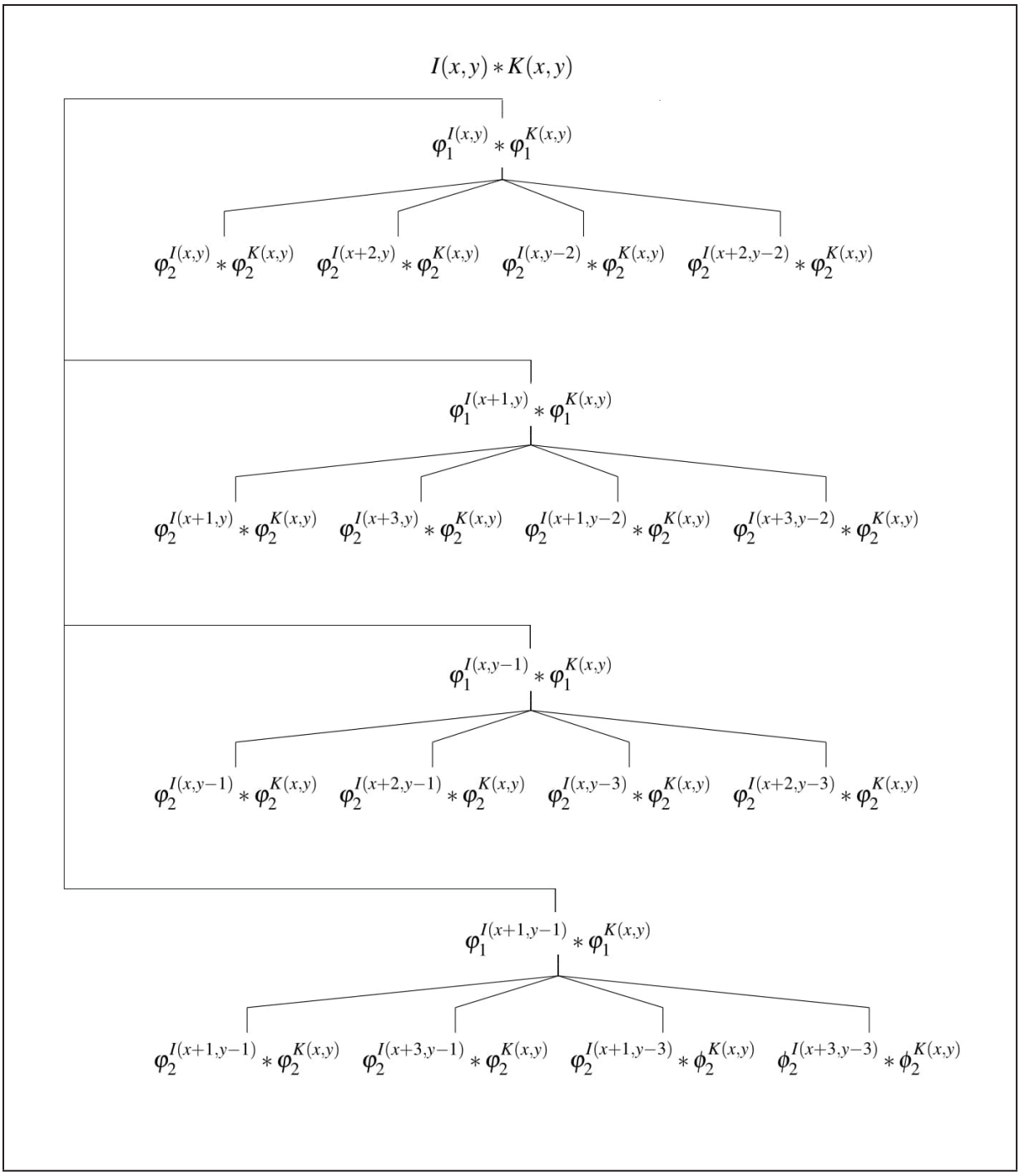


Figure-A I-2 Tree of approximate coefficients for convolution calculation in 2 level

As can be seen from the figure to calculate the convolution for each layer we should consider 4^j state for the original image, where j shows the number of level.

The idea behind this procedure is to make detail coefficient sparse by putting the least significant values to zero. As there is many dot product between the detail coefficients, making them sparse could have an effect on speeding up the convolution performance. Figure I-3 shows the convolution results for different image between a horizontal detail coefficient of accordion image and horizontal detail coefficient of first filter set of Vgg for the second and third level.

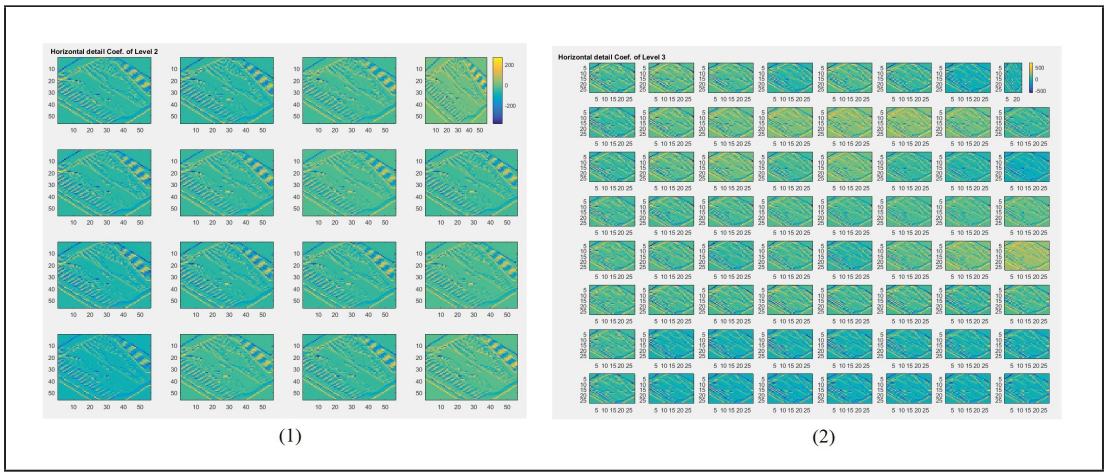


Figure-A I-3 a) convolution result between horizontal detail coefficient of input image and filter set 1) for second level, 2) for third level

As can be seen in figure the convolution procedure in wavelet domain needs a lot of complex computation which can not contribute to computational complexity reduction. The new ideas has been proposed using wavelet approach in Williams & Li (2016). It indicates that by applying wavelet transform to the input data we can use all the sub-bands in different frequencies which lead to a greater accuracy in CNN classification.

The recent study has been taken place in the work of Fujieda *et al.* (2018). The author added the multiresolution alaysis as a components in a CNN structure. It generalizes the form of convolution and average pooling step and equalizes them to part of high pass and low pass filter of multiresolution analysis. It then decomposes the input image and convolve it with two kernel for each level. For the input image of X with the level of j it defines the multiresolution analysis

as:

$$X_{l,j+1} = (X_{l,j} * F_{l,j} \downarrow) \tag{A I-25}$$

$$X_{h,j+1} = (X_{l,j} * F_{h,j} \downarrow) \tag{A I-26}$$

Where F_l and F_h shows two different filters. And as can be seen in the formulation they discard the effect of high pass $X_{h,j}$. It shows that this model can gain a better accuracy with a smaller number of parameters and less needed memory. Since the procedure requires heavy computation, we need a more modification to create an efficient network in terms of computation and also accuracy. As it mentioned Williams & Li (2016) proposed a new idea to use different sub bands of the image and learn the image in different frequency and enhance the accuracy of the classification model, we also need an alternation to find a better solution for this matter.

APPENDIX II

METHODOLOGY PCA

The computation cost of compressed filters is calculated by $(I \times F \times K)$. Considering the $I \in R^{M \times N \times D}$ as input of convolution and $F \in R^{W \times H \times D}$ where $M \times N$ and $W \times H$ is the number of spatial samples of input and filters respectively, and D is the number of channels. The number of operation O for each layer can be calculated by:

$$O = \frac{M + 2P}{S} \times \frac{N + 2P}{S} \times D \times W \times H \times K \tag{A II-1}$$

Where P and S are the number of $paddig$ added to the input and $stride$ respectively. Figure II-1 illustrate the number of computation based on the number of principal components used simultaneously in last four layer. We consider 40 principal components for the first Conv layer.

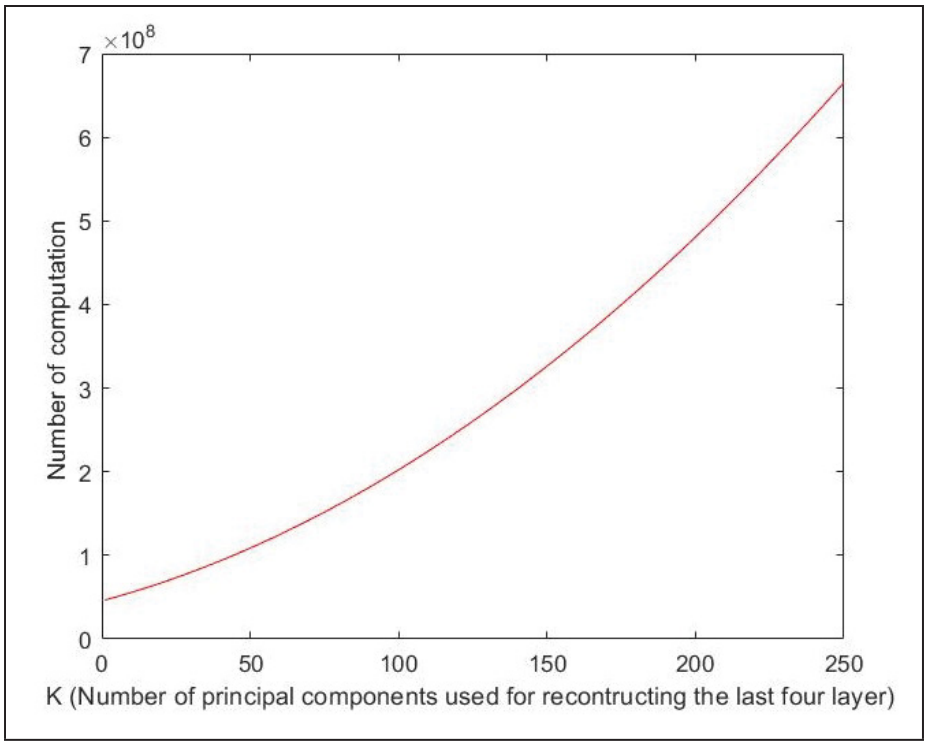


Figure-A II-1 Graph of computation vs. number of principal components used to represent filters: VGG last four layer, for the first Conv layer K = 40

Table II-1 shows the number of computation for seven different number of principal components.

Table-A II-1 Overview of number of computation O changes of for, K = 40 for VGG convolution layer = 1, and K = 150 to 250 for VGG convolution layer = 2, 3, 4, 5. considering that the total number of operation is 6.897×10^8

.

K (First Conv Layer)	K (other Conv Layer)	number of operation 10^8
40	150	3.264
40	160	3.548
40	170	3.845
40	180	4.153
40	190	4.474
40	200	4.807
40	250	6.654

APPENDIX III

RESULTS

1. ROC curves Histograms

Here we show the ROC curves for ten unseen classes. A sample picture from all ten categories has been shown in III-1.

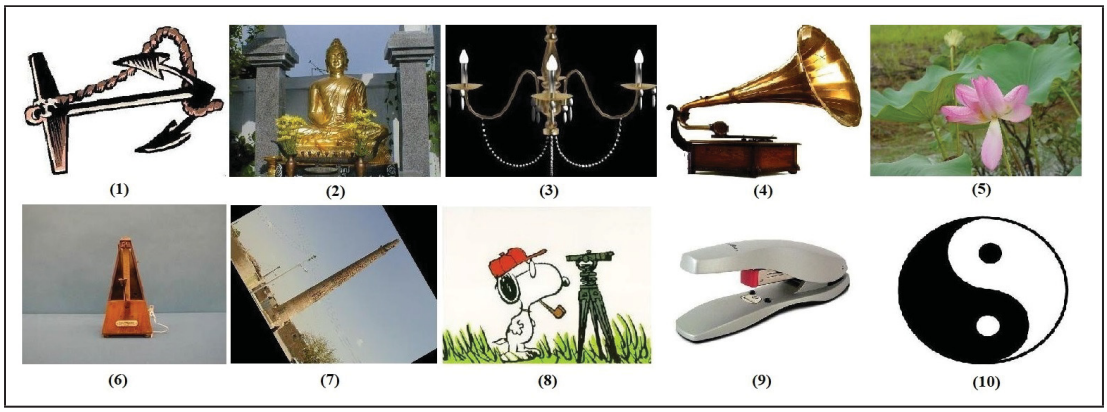


Figure-A III-1 Ten unseen categories from left to right 1) anchor, 2) buddha, 3) chandelier, 4) gramophone, 5) lotus, 6) metronome, 7) minaret, 8) snoopy, 9) stapler, 10) yin yang

The ROC curves obtained by classification is shown in figure III-2. The ROC curves shows an approximately identical curves for both configuration.

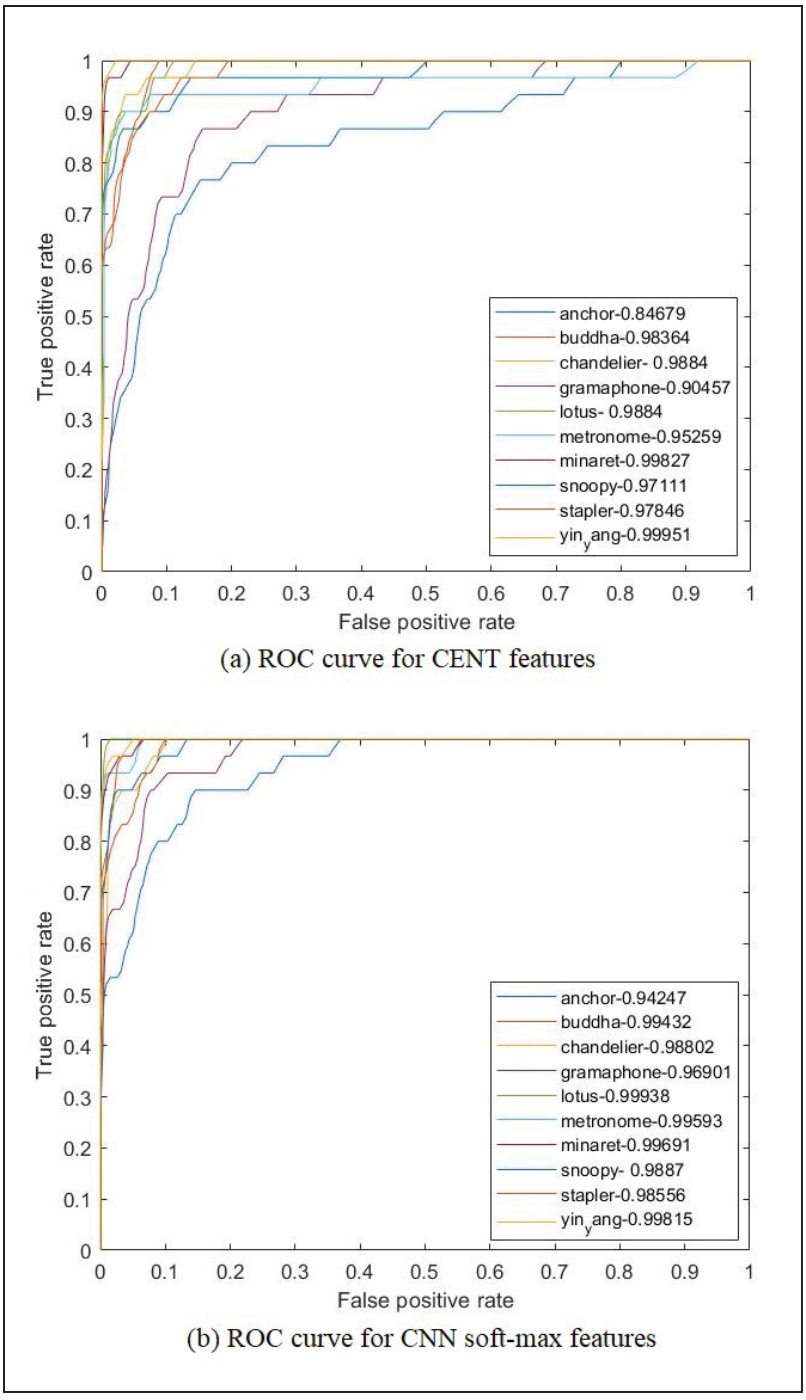
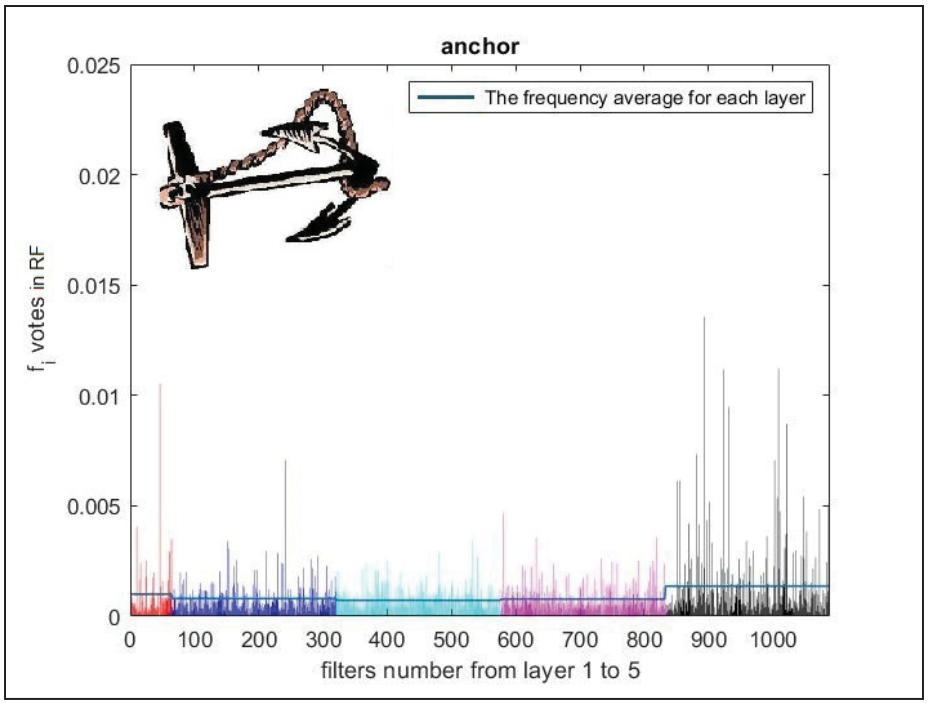


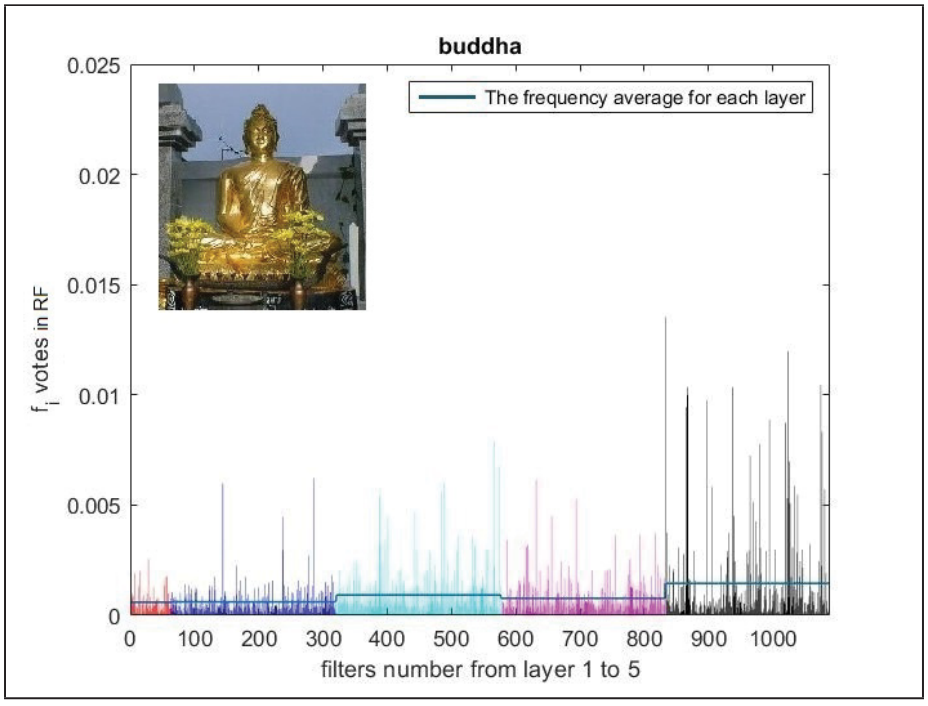
Figure-A III-2 ROC curves transfer learning: classification of 10 natural objects not used in original CNN training. Each image is represented by (a) CENT features computed layer-wise across 5 CNN convolutional layers and (b) the 1000-feature CNN soft-max output

Here we illustrate the flow of information through the CNN for 10 category used for classification where each class contain 30 images. We have used 400 tree to train the data in binary fashion. In this case we label targeted group as 1 and the remain groups as 0.

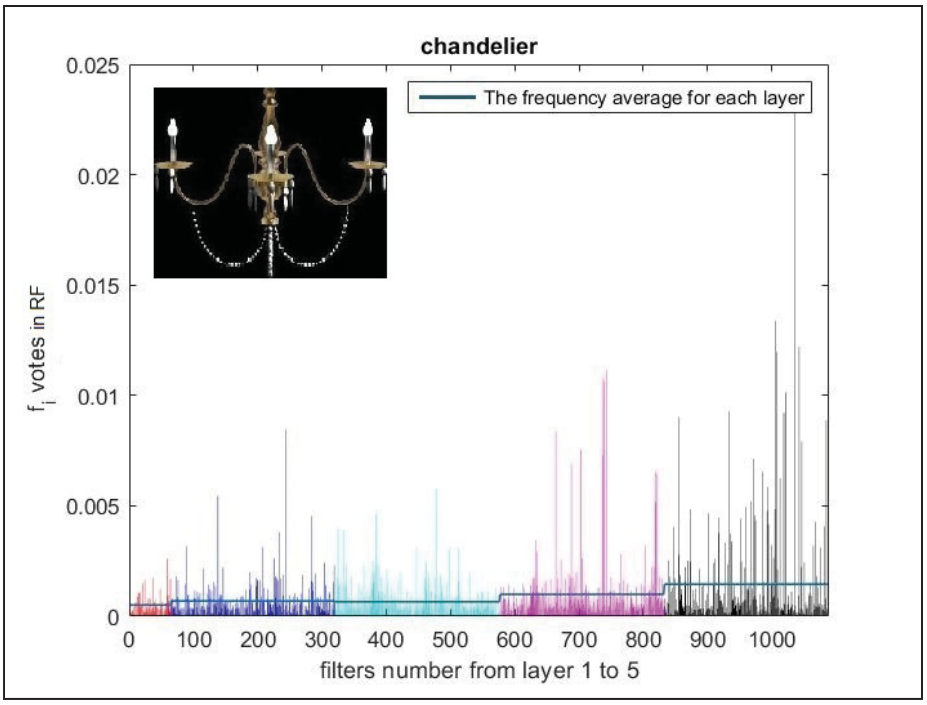
Figure III-3 display the histograms for all this ten object.



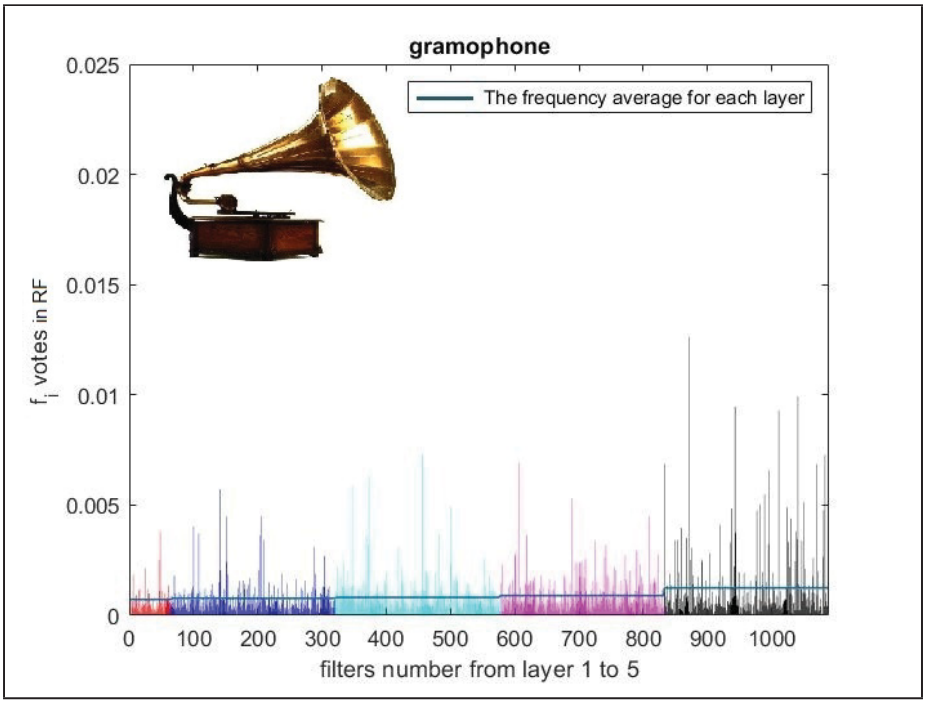
(a) Histogram of VGG CENT features used in RF for the class of anchor



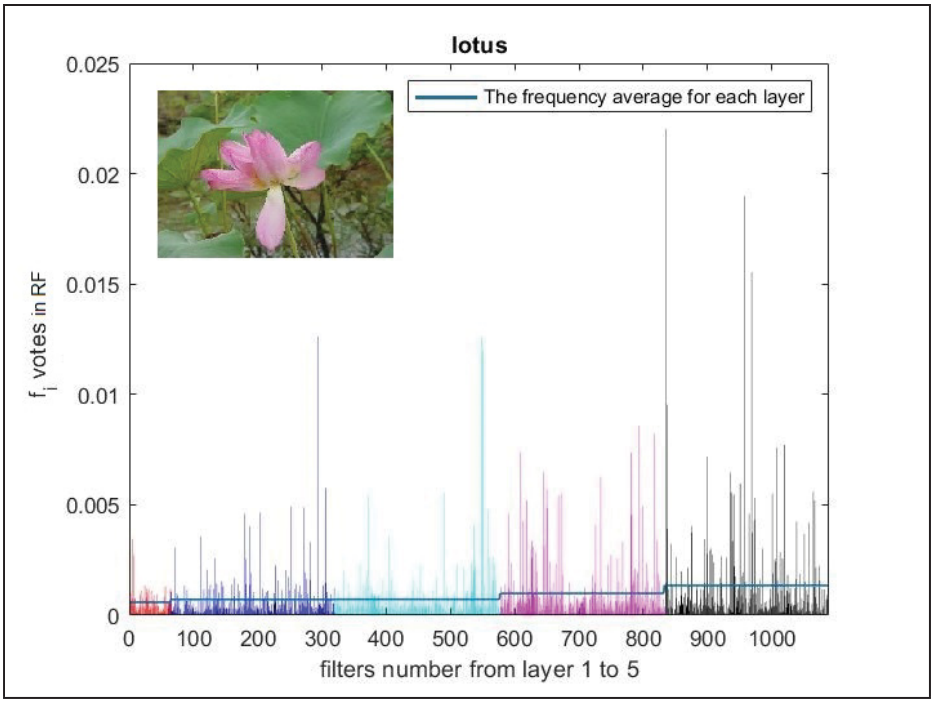
(b) Histogram of VGG CENT features used in RF for the class of buddha



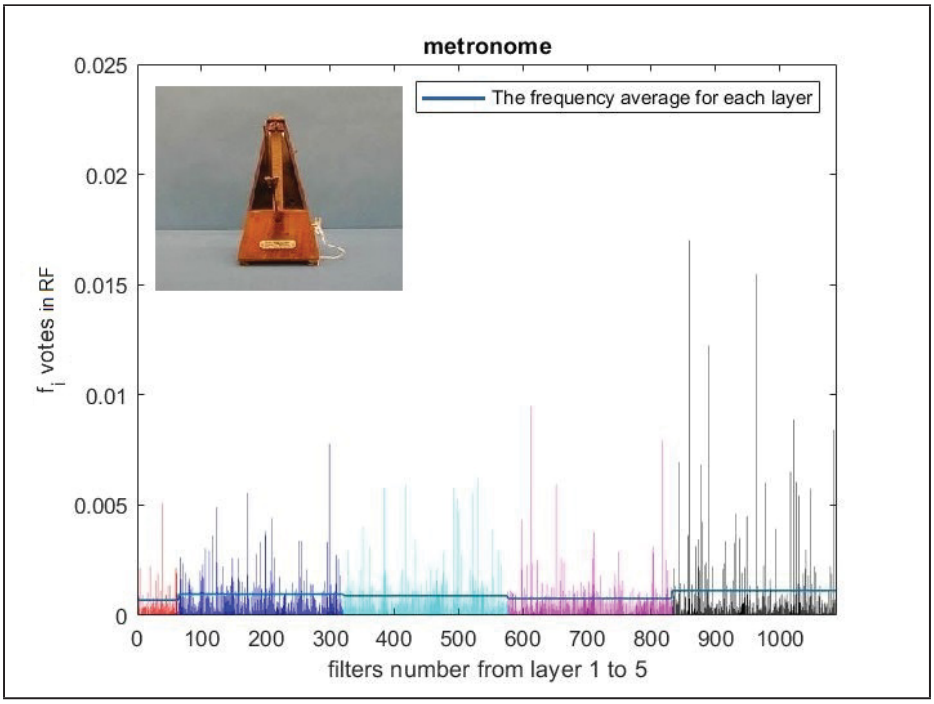
(c) Histogram of VGG CENT features used in RF for the class of chandelier



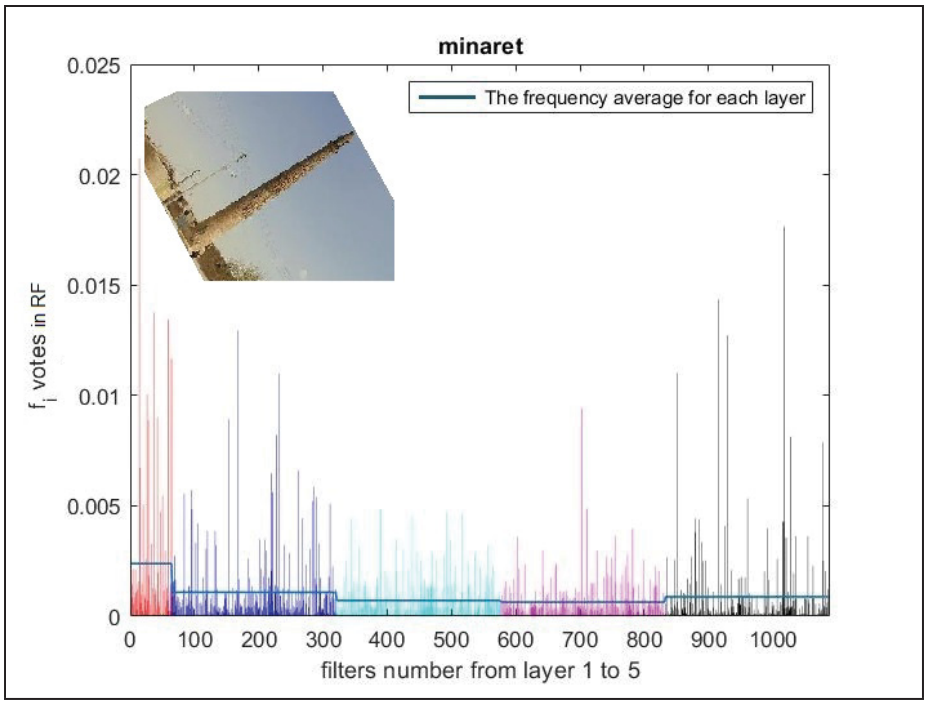
(d) Histogram of VGG CENT features used in RF for the class of gramophone



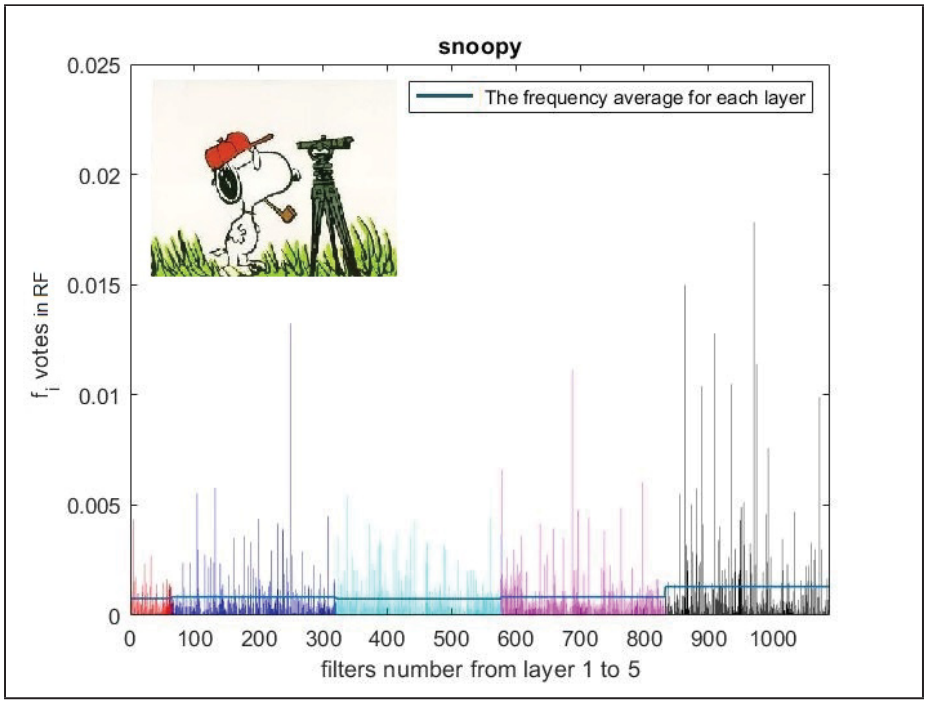
(e) Histogram of VGG CENT features used in RF for the class of lotus



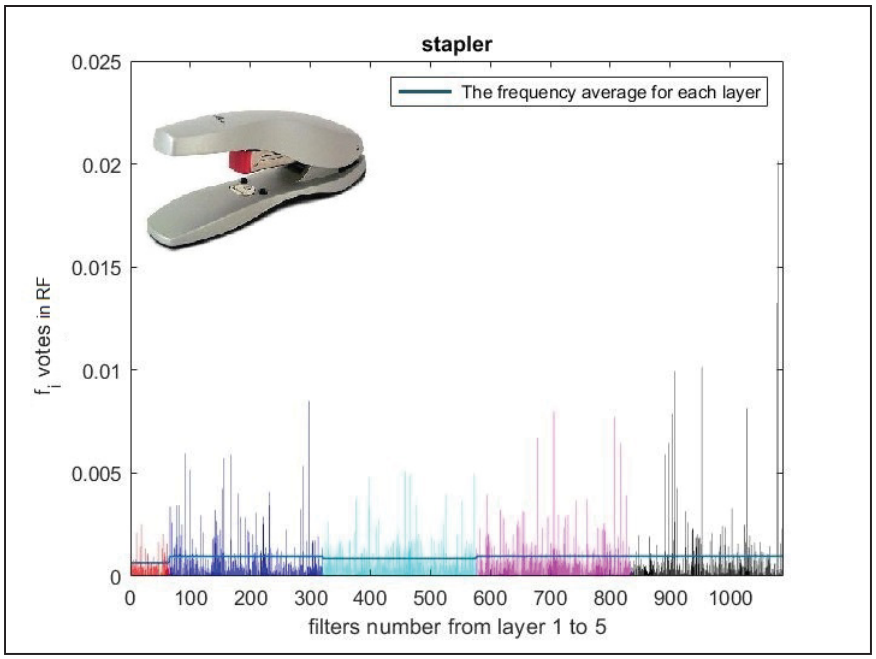
(f) Histogram of VGG CENT features used in RF for the class of metronome



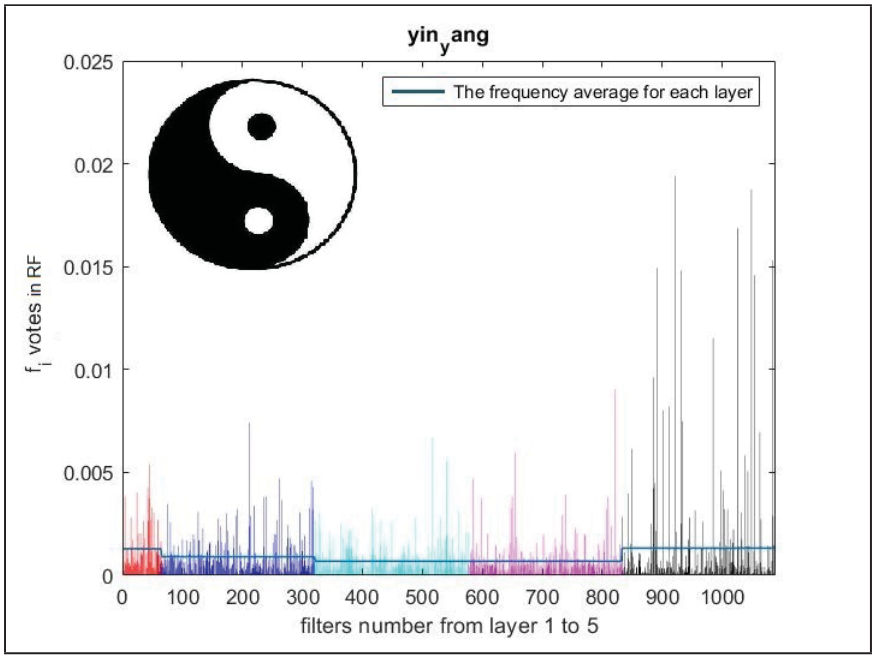
(g) Histogram of VGG CENT features used in RF for the class of minaret



(h) Histogram of VGG CENT features used in RF for the class of snoopy



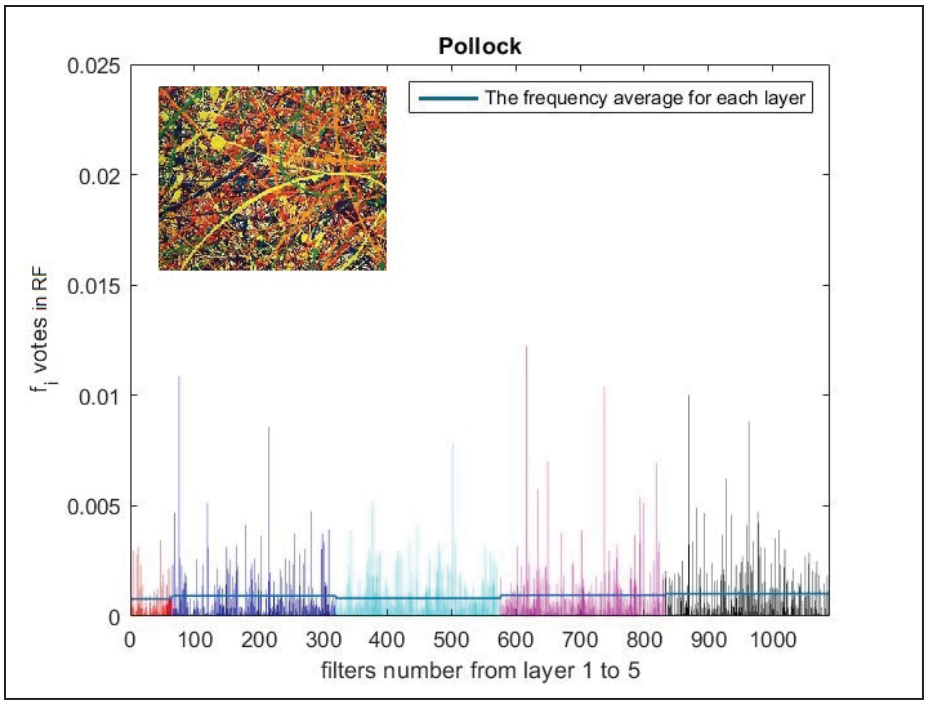
(i) Histogram of VGG CENT features used in RF for the class of stapler



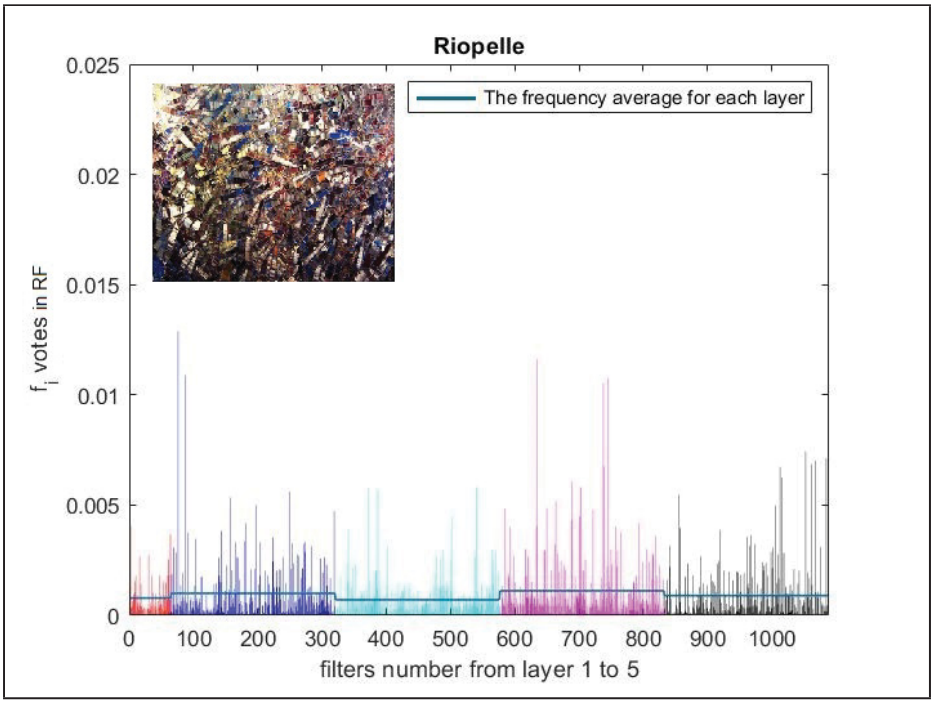
(j) Histogram of VGG CENT features used in RF for the class of yin-yang

Figure-A III-3 Histogram : Histograms of 10 subjects not used in original CNN training. Each image is represented filters used for classification for each category across all CNN layers. The bottom histogram split the bins of each layer by different color

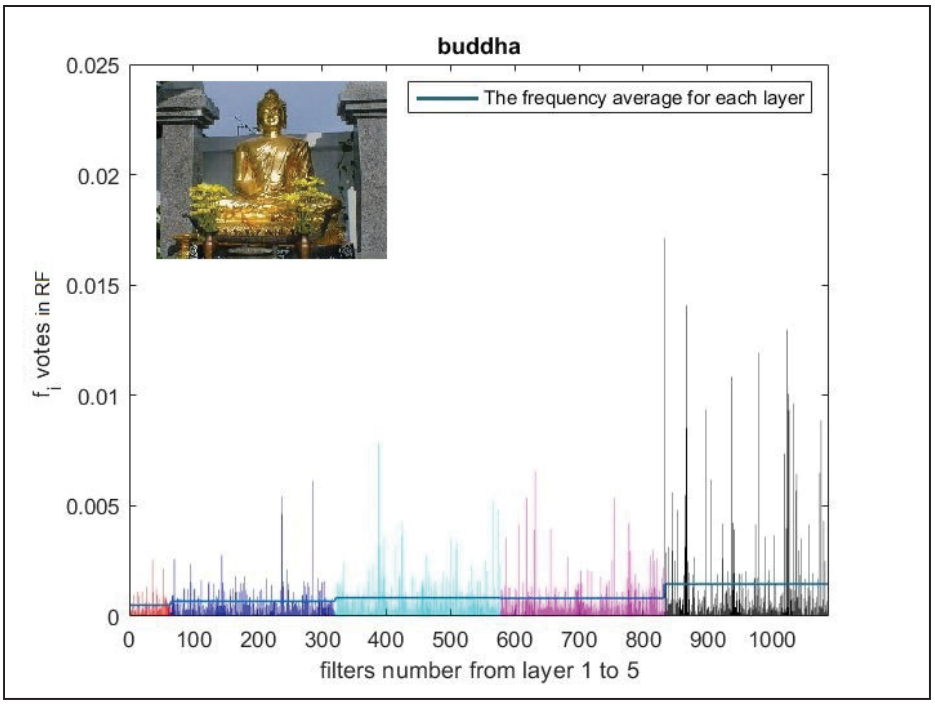
Since two categories "anchor" and "gramophone" do not show a good performance in classification we replace them with Riopelle and Pollock painting categories. Histogram of ten subject after changing two group with Riopelle and Pollock is also provided in figure III-4.



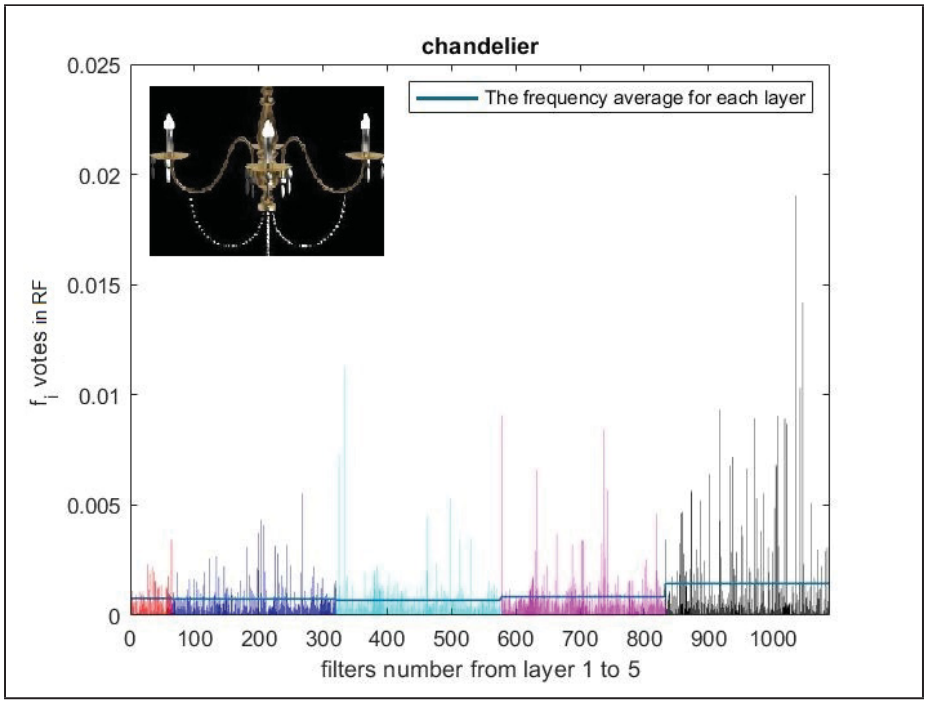
(a) Histogram of VGG CENT features used in RF for the class of Riopelle



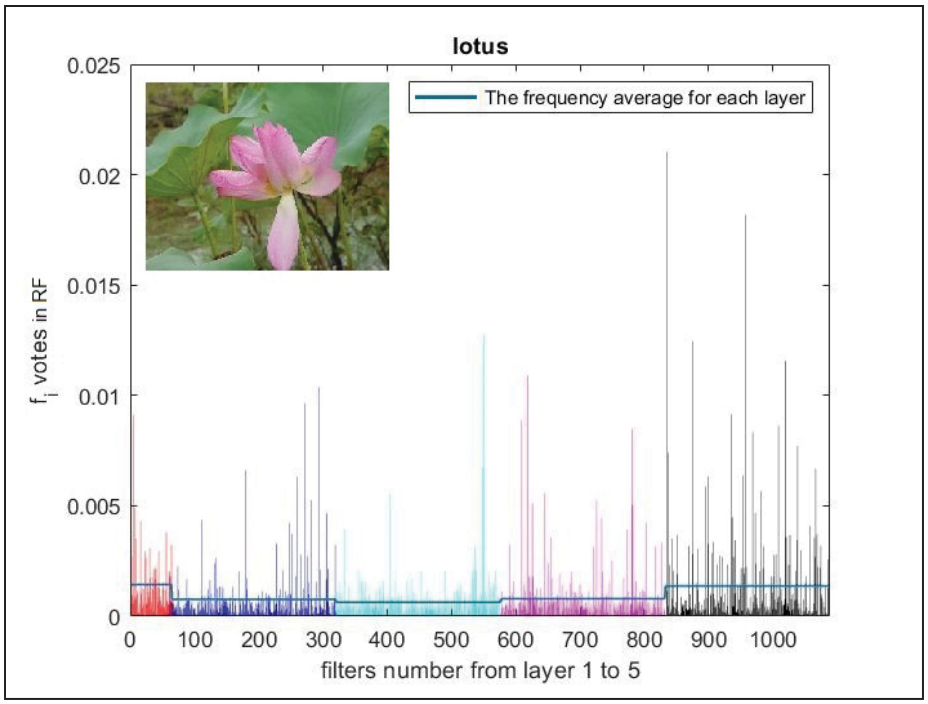
(b) Histogram of VGG CENT features used in RF for the class of Pollock



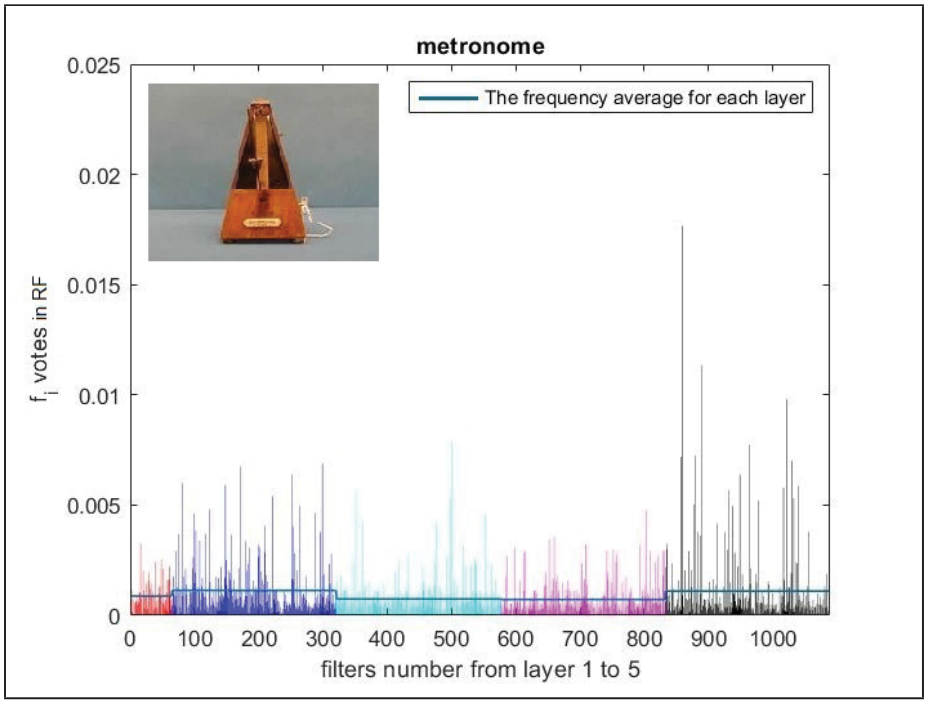
(c) Histogram of VGG CENT features used in RF for the class of buddha



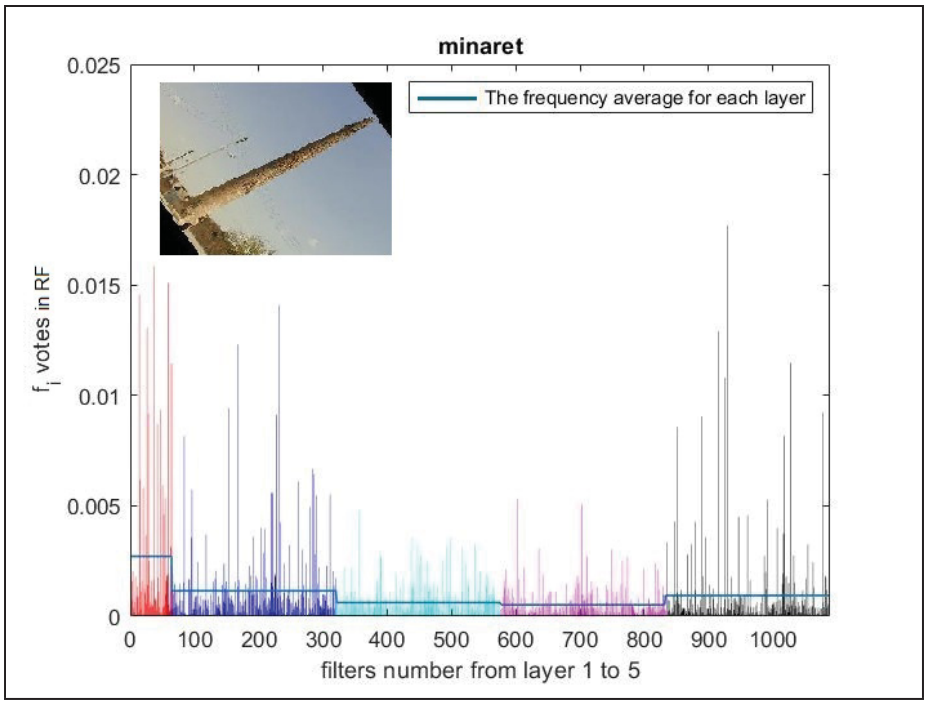
(d) Histogram of VGG CENT features used in RF for the class of chandelier



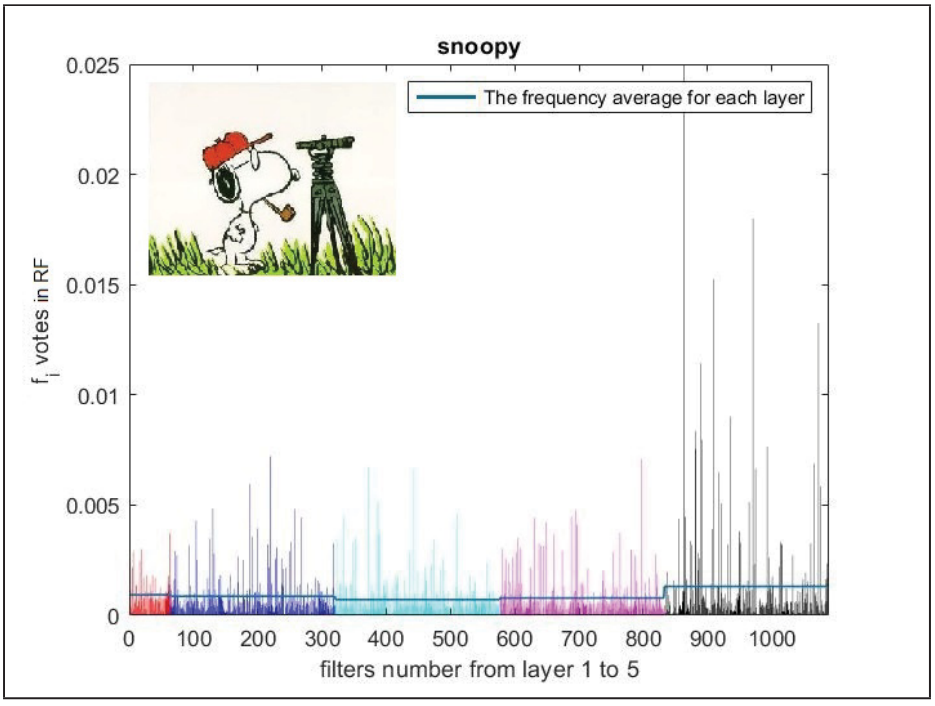
(e) Histogram of VGG CENT features used in RF for the class of lotus



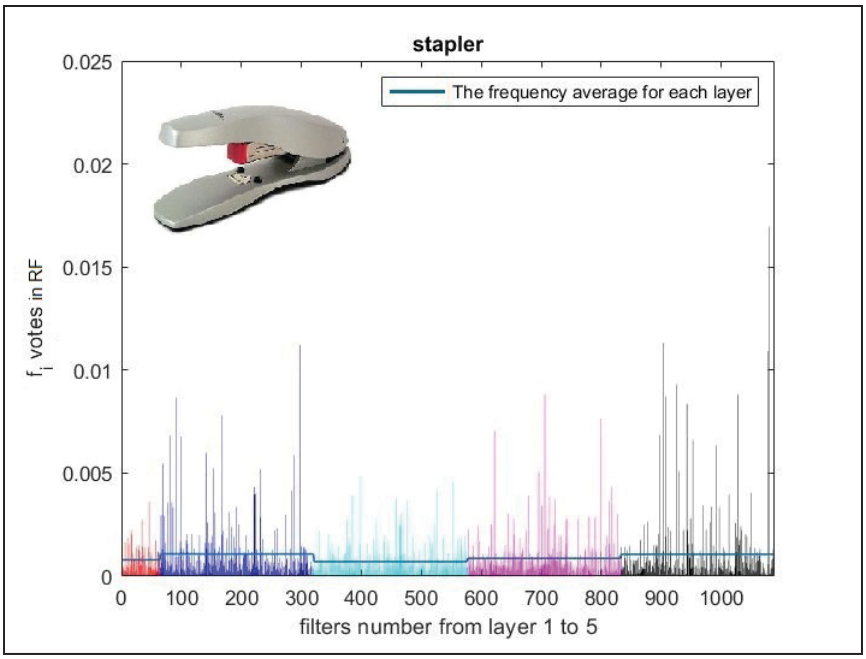
(f) Histogram of VGG CENT features used in RF for the class of metronome



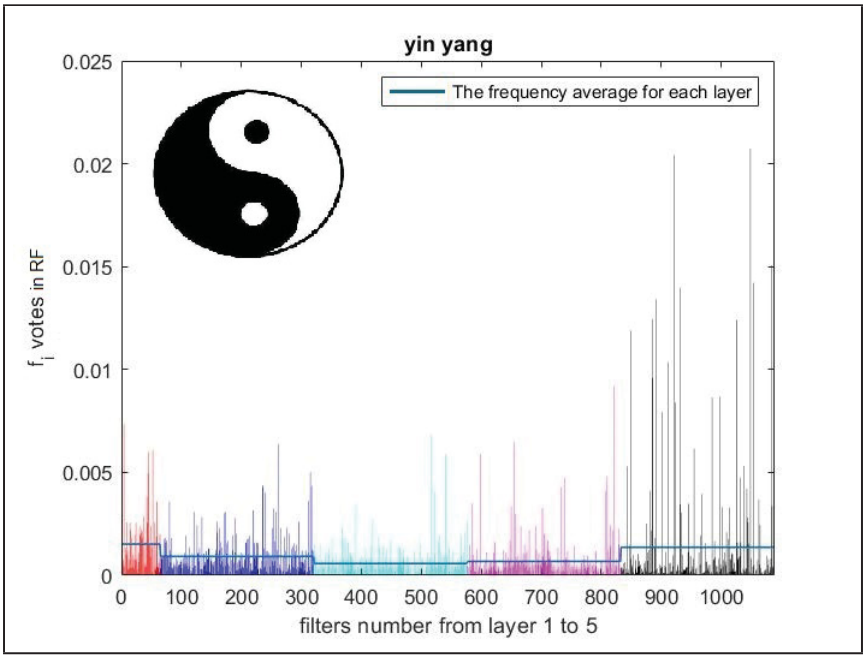
(g) Histogram of VGG CENT features used in RF for the class of minaret



(h) Histogram of VGG CENT features used in RF for the class of snoopy



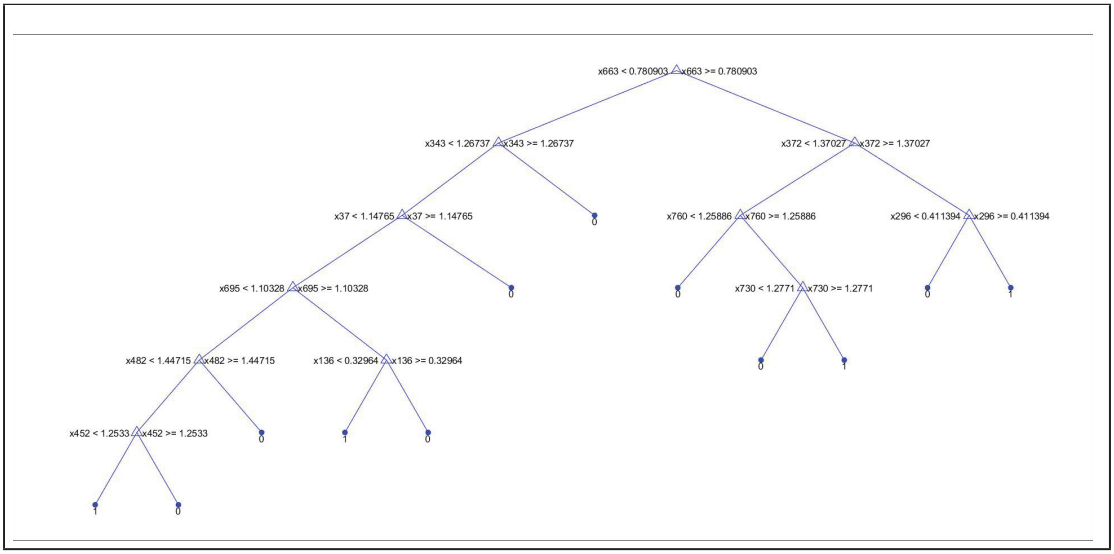
(i) Histogram of VGG CENT features used in RF for the class of stapler



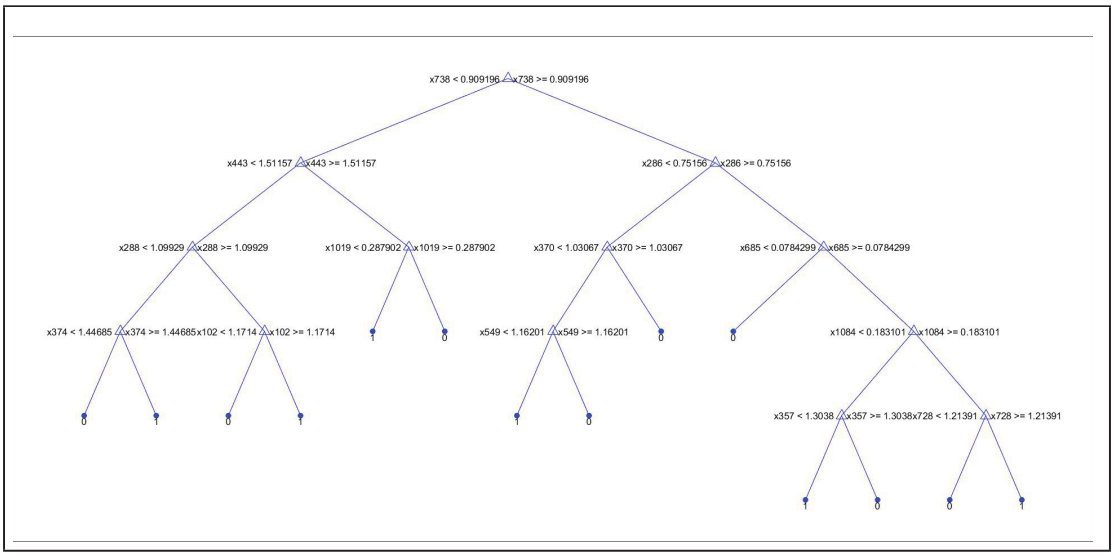
(j) Histogram of VGG CENT features used in RF for the class of yin-yang

Figure-A III-4 Histogram : Histograms of 10 subjects not used in original CNN training. Each image is represented filters used for classification for each category across all CNN layers. The bottom histogram split the bins of each layer by different color

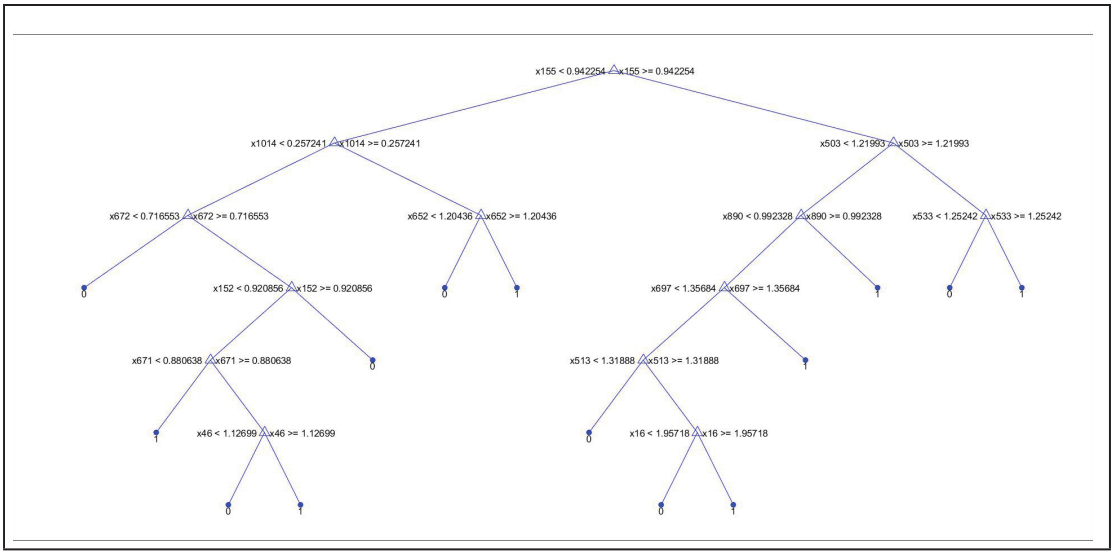
We also provide pictures of some trees used for training these ten categories not used in VGG-19 training in figure III-5. As the classification is binary but we have 10 different category, the ratio of the desired class from the lower amount of a certain entropy for each load is 1 to 9.



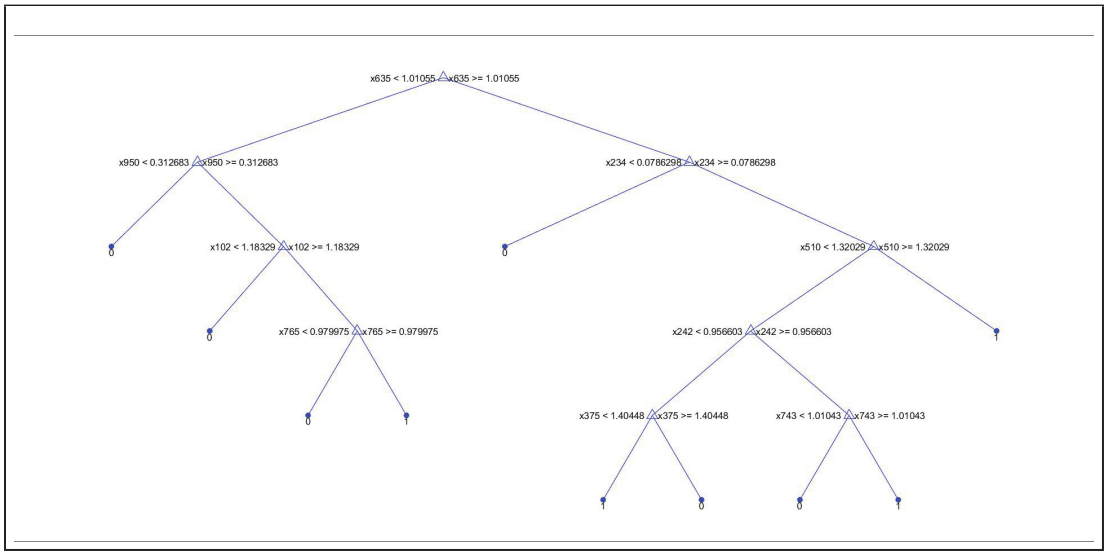
(a) First tree



(b) 27th tree



(d) 258th tree



(e) 300th tree

Figure-A III-5 Sample of trees in RF

2. ROC curves for easily fooled classes

In this section we provide the results for another experience of adding noise on bellpepper. Again we add salt and pepper noise with noise density of 0.437 into 1100 images of bellpepper images where 100 has been recognized as cucumber. We train RF with CENT codes of both

classes and also compare the results from the classification with soft-max output. Note that all images are recognized as cucumber with or without noise by VGG. The number of trees in RF classifier is 800 and it evaluate the results by 10 fold cross validation.

Figure III-6 shows the ROC curves for two classes of bellpepper and cucumber. Both categories are synthesized with noise. We use original images for training and noisy images for the test set.

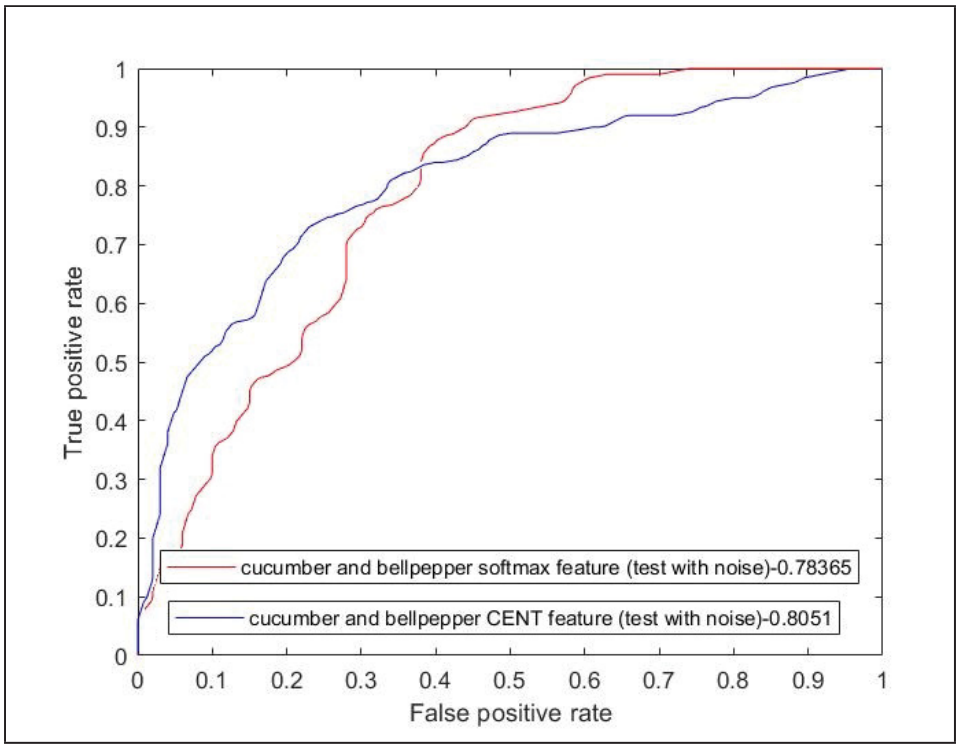


Figure-A III-6 ROC curves transfer learning: classification of two categories bell pepper and cucumber. The data has been trained with the original images and the test sets are synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

The ROC curve shows a better result from CENT features.

FigureIII-7 shows the ROC curves for two classes of bellpepper and cucumber both with noise. We use noisy images for both training set and test set.

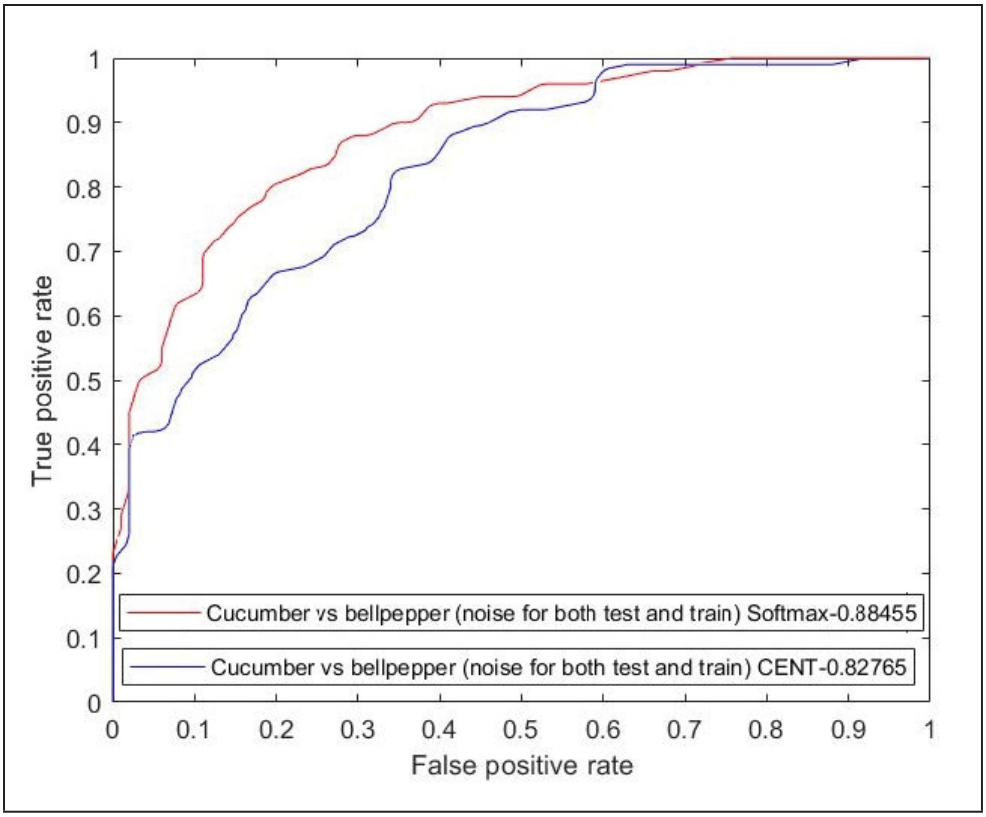


Figure-A III-7 ROC curves transfer learning: classification of two categories bell pepper and cucumber. The data has been trained and test with the images synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

Figure III-8 shows the ROC curves for the classification of two classes of bellpepper and cucumber. In this experiment we only add noise to the class of bellpepper. We use the noisy images of bellpepper for both training and test set.

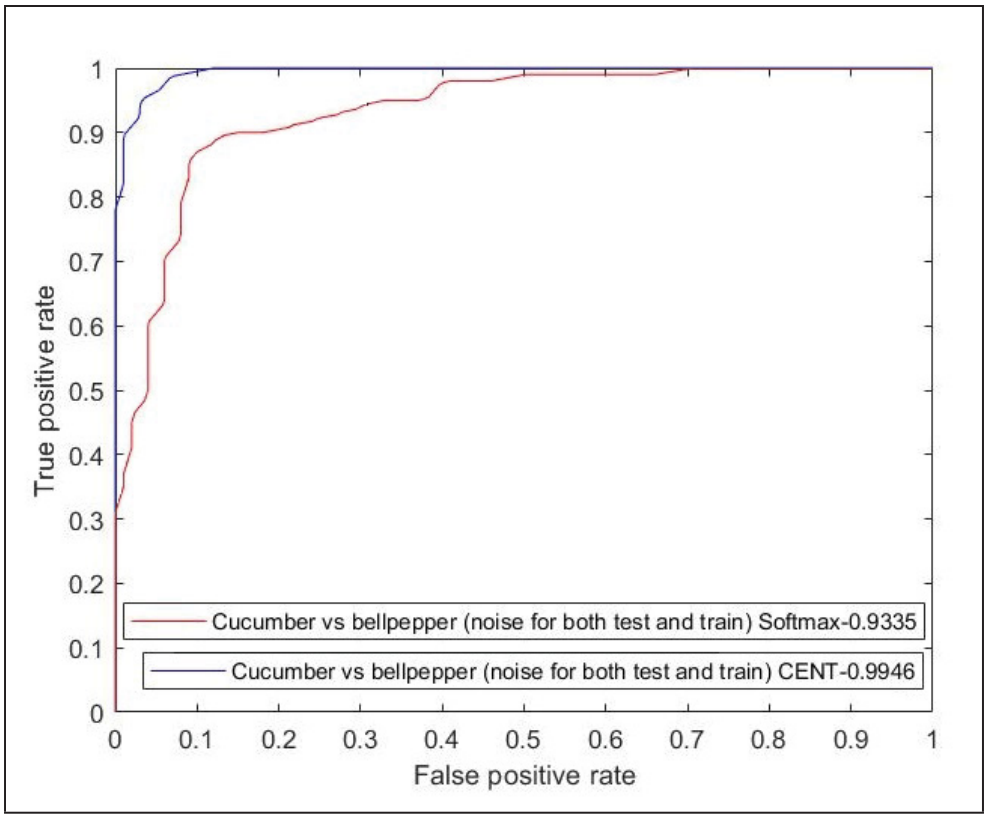


Figure-A III-8 ROC curves transfer learning: classification of two categories bell pepper and cucumber. We add noise only on the class of bellpepper. The data has been trained test with the images synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

The experiment is repeated for the class of bellpepper by synthesising the images with the salt and pepper noise with the noise density of 0.4. In this test 100 images of total 1100 images of bellpepper recognized as orange. RF classifier use 800 trees and evaluate the images with 10 fold cross validation. Figure III-9 shows the results from the CENT codes and soft-max output. We add noise to both categories. The model is trained with original images and the set is synthesised with noise.

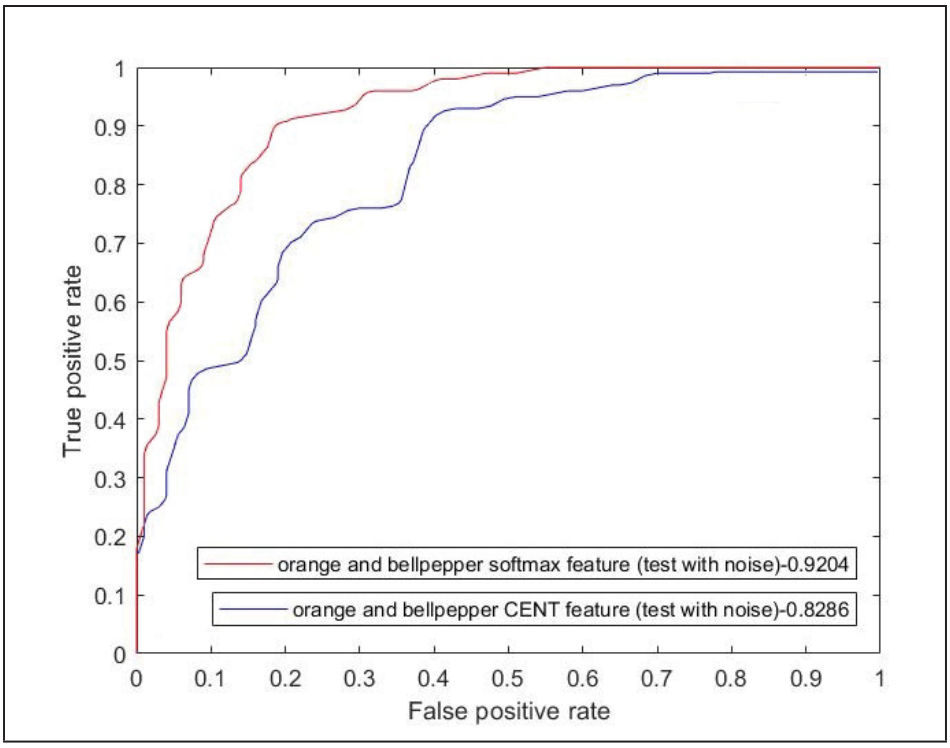


Figure-A III-9 ROC curves transfer learning: classification of two categories bell pepper and orange. The data has been trained with the original images and the test sets are synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

Figure III-10 shows the ROC curves for the class of bellpepper and orange. We add noise for both class. Both test set and train set includes the noisy images.

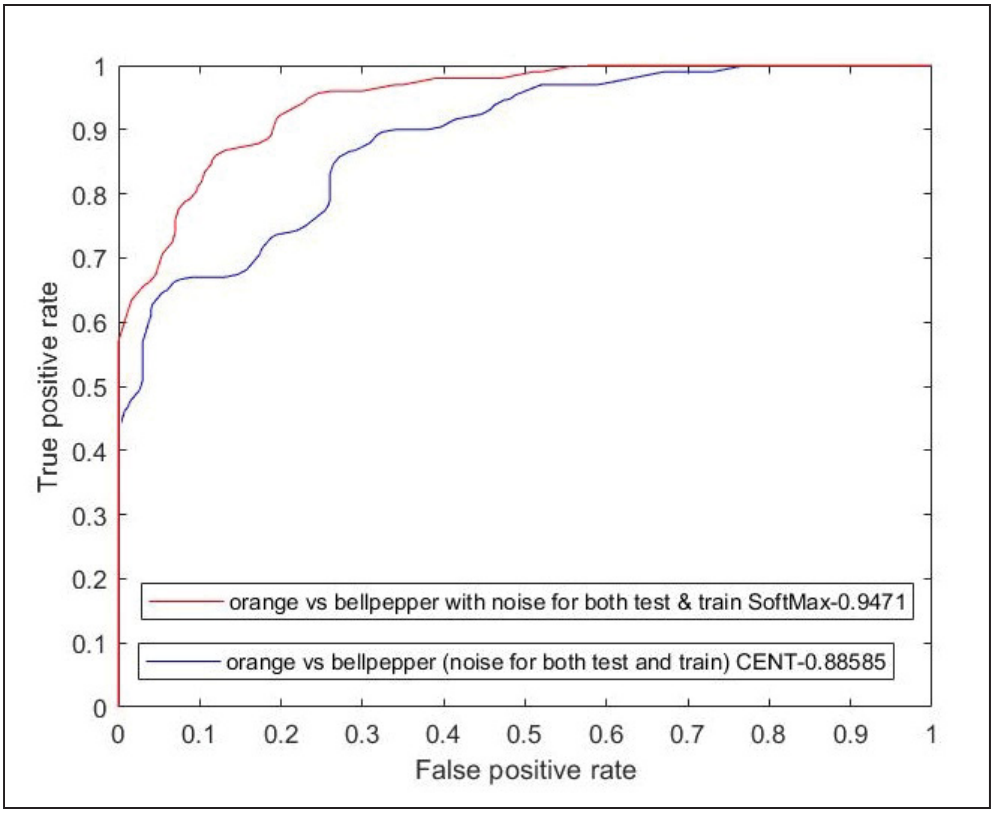


Figure-A III-10 ROC curves transfer learning: classification of two categories bell pepper and orange. Both training and test set images are synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

Figure III-11 shows the result of bellpepper and orange. We add noise only on the class of bellpepper so all the images classified as orange. Both training set and test set is include the images of bellpepper with noise.

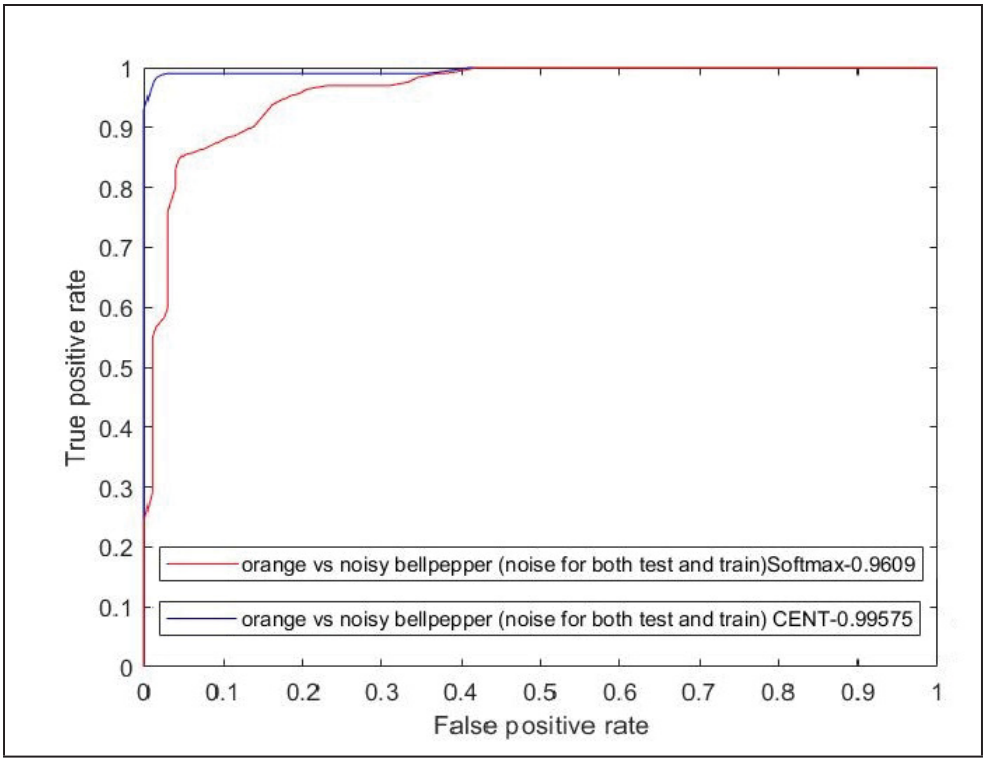


Figure-A III-11 ROC curves transfer learning: classification of two categories bell pepper and orange. We only add noise on the class of bellpepper. Both training and test set images are synthesized by noise. Each ROC curve is represented by (Blue) CENT features computed layer-wise across 5 CNN convolutional layers and (Red) the 1000-feature CNN soft-max output

REFERENCES

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Asadi, S., Rao, C. D. S. & Saikrishna, V. (2010). A comparative study of face recognition with principal component analysis and cross-correlation technique. *International Journal of Computer Applications*, 10(8), 17–21.

Bagherinezhad, H., Rastegari, M. & Farhadi, A. (2017). Lcnn: Lookup-based convolutional neural network. *Proc. IEEE CVPR*.

Belghazi, I., Rajeswar, S., Baratin, A., Hjelm, R. D. & Courville, A. (2018). MINE: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*.

Bengio, Y., Courville, A. & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798–1828.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5–32.

Brown, G., Pocock, A., Zhao, M.-J. & Luján, M. (2012). Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *Journal of machine learning research*, 13(Jan), 27–66.

Canziani, A., Paszke, A. & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

Chaddad, A., Desrosiers, C. & Toews, M. (2016). Local discriminative characterization of MRI for Alzheimer's disease. *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*, pp. 1–5.

Chaddad, A., Naisiri, B., Pedersoli, M., Granger, E., Desrosiers, C. & Toews, M. (2017). Modeling Information Flow Through Deep Neural Networks. *arXiv preprint arXiv:1712.00003*.

Chaddad, A., Toews, M., Desrosiers, C. & Niazi, T. (2019). Deep radiomic analysis based on modeling information flow in convolutional neural networks. *IEEE Access*, 7, 97242–97252.

Changhau, I. (2017). Activation Functions in Neural Networks.

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

Chellapilla, K., Puri, S. & Simard, P. (2006). High performance convolutional neural networks for document processing. *Tenth International Workshop on Frontiers in Handwriting Recognition*.

Chen, W., Wilson, J. T., Tyree, S., Weinberger, K. Q. & Chen, Y. (2015). Compressing convolutional neural networks. *arXiv preprint arXiv:1506.04449*.

Cheng, Y., Wang, D., Zhou, P. & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

Cheng, Y., Wang, D., Zhou, P. & Zhang, T. (2018). Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1), 126–136.

Chun-Lin, L. (2010). A tutorial of the wavelet transform. *NTUEE, Taiwan*.

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ciresan, D., Giusti, A., Gambardella, L. M. & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems*, pp. 2843–2851.

Clevert, D.-A., Unterthiner, T. & Hochreiter, S. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *CoRR*, abs/1511.07289.

Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L. & Batra, D. (2015). Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*.

Cover, T. M. & Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE transactions on information theory*, 36(5), 961–1005.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255.

Denil, M., Shakibi, B., Dinh, L., De Freitas, N. et al. (2013). Predicting parameters in deep learning. *Advances in neural information processing systems*, pp. 2148–2156.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y. & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, pp. 1269–1277.

Desikan, R. S. e. a. (2009). Automated MRI measures identify individuals with mild cognitive impairment and Alzheimer's disease. *Brain*, 132(8), 2048–2057.

DeVries, T. & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Duan, H., Yan, R. & Lin, K. (2008). Research on face recognition based on PCA. *2008 International Seminar on Future Information Technology and Management Engineering*, pp. 29–32.

Dubey, A., Chatterjee, M. & Ahuja, N. (2018). Coreset-based neural network compression. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 454–470.

Duda, R. O., Hart, P. E. & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

Fei-Fei, L., Fergus, R. & Perona, P. (2007). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1), 59–70.

Fujieda, S., Takayama, K. & Hachisuka, T. (2018). Wavelet Convolutional Neural Networks. *CoRR*, abs/1805.08620. Consulted at http://arxiv.org/abs/1805.08620.

Gallistel, C. R. & King, A. P. (2011). *Memory and the computational brain: Why cognitive science will transform neuroscience*. John Wiley & Sons.

Gan, Y., Liu, J., Dong, J. & Zhong, G. (2015). A PCA-Based Convolutional Network. *ArXiv*, abs/1505.03703.

Garcia, C. & Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(11), 1408–1423.

Garg, I., Panda, P. & Roy, K. (2019). A low effort approach to structured cnn design using pca. *IEEE Access*.

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

Gilles, S. (1998). Robust description and matching of images. *Ph. D. thesis, Dept. Eng. Sci., Univ. Oxford*.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Gong, Y., Liu, L., Yang, M. & Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, pp. 2672–2680.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT press.

Graham, B. (2014). Fractional Max-Pooling. *CoRR*, abs/1412.6071. Consulted at http://arxiv.org/abs/1412.6071.

Graps, A. (1995). An introduction to wavelets. *IEEE computational science and engineering*, 2(2), 50–61.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H. & Schmidhuber, J. (2008). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5), 855–868.

Graves, A., Mohamed, A.-r. & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. et al. (2017). Recent advances in convolutional neural networks. *Pattern Recognition*.

Guo, Y., Yao, A. & Chen, Y. (2016). Dynamic network surgery for efficient dnns. *Advances In Neural Information Processing Systems*, pp. 1379–1387.

Guo, Y. (2017). Problems in large-scale image classification. *Thirty-First AAAI Conference on Artificial Intelligence*.

Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3), 331–371.

Hagan, M. T., Demuth, H. B., Beale, M. H. & De Jesús, O. (1996). *Neural network design*. Pws Pub. Boston.

Han, S., Mao, H. & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Hanlon, J. (2017). How to Solve the Memory Challenges of Deep Neural Network.

Haralick, R. M., Shanmugam, K. et al. (1973). Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6), 610–621.

Haykin, S. (1994). *Neural networks*. Prentice hall New York.

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

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65–93). Elsevier.

Highlander, T. & Rodriguez, A. (2016). Very efficient training of convolutional neural networks using fast fourier transform and overlap-and-add. *arXiv preprint arXiv:1601.06815*.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6), 417.

Huang, G., Liu, Z., Weinberger, K. Q. & van der Maaten, L. (2016). Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*.

Huang, G., Liu, Z., Weinberger, K. Q. & van der Maaten, L. (2017). Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1(2), 3.

Huynh, B. Q., Li, H. & Giger, M. L. (2016). Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. *Journal of Medical Imaging*, 3(3), 034501.

Hyvärinen, A., Karhunen, J. & Oja, E. (2004). *Independent component analysis*. John Wiley & Sons.

Ioffe, S. & Szegedy, C. (2015a). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ioffe, S. & Szegedy, C. (2015b). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 448-456. Consulted at http://jmlr.org/proceedings/papers/v37/ioffe15.pdf.

Jack Jr, C. R., Bernstein, M. A., Fox, N. C., Thompson, P., Alexander, G., Harvey, D., Borowski, B., Britson, P. J., L. Whitwell, J., Ward, C. et al. (2008). The Alzheimer's disease neuroimaging initiative (ADNI): MRI methods. *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 27(4), 685–691.

Jaderberg, M., Vedaldi, A. & Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

Jaderberg, M., Simonyan, K., Zisserman, A. et al. (2015). Spatial transformer networks. *Advances in neural information processing systems*, pp. 2017–2025.

Jarrett, K., Kavukcuoglu, K., LeCun, Y. et al. (2009). What is the best multi-stage architecture for object recognition? *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153.

Jensen, O. H. (2008). *Implementing the Viola-Jones face detection algorithm*. (Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark).

Jolliffe, I. (2011). *Principal component analysis*. Springer.

Joshi, P. (2016). Understanding Xavier Initialization In Deep Neural Networks.

Kadir, T. & Brady, M. (2001). Saliency, scale and image description. *International Journal of Computer Vision*, 45(2), 83–105.

Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1), 35–45.

Khadivi, P., Tandon, R. & Ramakrishnan, N. (2016). Flow of information in feed-forward deep neural networks. *arXiv preprint arXiv:1603.06220*.

Khalifa, A. B. & Frigui, H. (2016). Multiple Instance Fuzzy Inference Neural Networks. *arXiv preprint arXiv:1610.04973*.

Kibunja, P. (2018). advanced convolutional neural networks.

Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L. & Shin, D. (2015). Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

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

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp. 1097–1105.

Kuo, C.-C. J. & Chen, Y. (2018). On data-driven saak transform. *Journal of Visual Communication and Image Representation*, 50, 237–246.

Lebedev, V. & Lempitsky, V. (2016). Fast convnets using group-wise brain damage. *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pp. 2554–2564.

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

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, pp. 396–404.

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

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444.

Liang, B., Li, H., Su, M., Li, X., Shi, W. & Wang, X. (2018). Detecting Adversarial Image Examples in Deep Neural Networks with Adaptive Noise Reduction. *IEEE Transactions on Dependable and Secure Computing*.

Lippi, M. (2017). Reasoning with deep learning: an open challenge. *2016 AI* IA Workshop on Deep Understanding and Reasoning: A Challenge for Next-Generation Intelligent Agents, URANIA 2016*, 1802, 38–43.

Liu, B., Wang, M., Foroosh, H., Tappen, M. & Pensky, M. (2015). Sparse convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 806–814.

Long, J., Shelhamer, E. & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Lowe, D. G. (2004a). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91–110.

Lowe, G. (2004b). SIFT-The Scale Invariant Feature Transform. *J. Comput. Vision*, 60, 91–110.

Luo, J.-H. & Wu, J. (2017). An entropy-based pruning method for CNN compression. *arXiv preprint arXiv:1706.05791*.

Mahmood, R. & Ghimire, B. (2013). Automatic detection and classification of Alzheimer's Disease from MRI scans using principal component analysis and artificial neural networks. *Systems, Signals and Image Processing (IWSSIP), 2013 20th International Conference on*, pp. 133–137.

Marcus, D. S., Wang, T. H., Parker, J., Csernansky, J. G., Morris, J. C. & Buckner, R. L. (2007). Open Access Series of Imaging Studies (OASIS): cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. *Journal of cognitive neuroscience*,

19(9), 1498–1507.

Markovsky, I. (2008). Structured low-rank approximation and its applications. *Automatica*, 44(4), 891–909.

Nguyen, A., Yosinski, J. & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436.

Nguyen-Thanh, N., Le-Duc, H., Ta, D.-T. & Nguyen, V.-T. (2016). Energy efficient techniques using fft for deep convolutional neural networks. *Advanced Technologies for Communications (ATC)*.

Nowlan, S. J. & Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4), 473–493.

O'keefe, J. & Nadel, L. (1978). *The hippocampus as a cognitive map*. Oxford: Clarendon Press.

Oquab, M., Bottou, L., Laptev, I. & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724.

Pan, S. J. & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10), 1345–1359.

Pan, S. & Yang, Q. (2010). A survey on transfer learning. IEEE Transaction on Knowledge Discovery and Data Engineering, 22 (10). IEEE press.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), 559–572.

Phillips, P. J., Flynn, P. J., Scruggs, T., Bowyer, K. W., Chang, J., Hoffman, K., Marques, J., Min, J. & Worek, W. (2005). Overview of the face recognition grand challenge. *Computer vision and pattern recognition, 2005. CVPR 2005. IEEE computer society conference on*, 1, 947–954.

Qiu, Q., Cheng, X., Calderbank, R. & Sapiro, G. (2018). Dcfnet: Deep neural network with decomposed convolutional filters. *arXiv preprint arXiv:1802.04145*.

Refianti, R., Mutiara, A. B. & Priyandini, R. P. (2019). Classification of Melanoma Skin Cancer Using Convolutional Neural Network. *IJACSA*, 10(3), 409–417.

Ronneberger, O., Fischer, P. & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.

Rosenblatt, F. (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*.

Rosenfeld, A. & Tsotsos, J. K. (2018). Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*.

Ruiz, F. E., Pérez, P. S. & Bonev, B. I. (2009). *Information theory in computer vision and pattern recognition*. Springer Science & Business Media.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3), 1.

Sainath, T. N., Kingsbury, B., Sindhwani, V., Arisoy, E. & Ramabhadran, B. (2013). Low-rank matrix factorization for deep neural network training with high-dimensional output targets. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6655–6659.

Schiele, B. (1997). *Recognition of Objects based on Multidimensional Receptive Field Histograms*. (Ph.D. thesis, PhD thesis, INP Grenoble).

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61, 85–117.

SHANNON, C. (1948). A Mathematical Theory of Communication-The Bell System Technical Journal, vol. 27, pag. 379-423, 623-656. Jul-Out.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27, 379–423.

Shchutskaya, V. (2018). Deep Learning: Strengths and Challenges.

Shi, Q., Petterson, J., Dror, G., Langford, J., Smola, A. & Vishwanathan, S. (2009). Hash kernels for structured data. *Journal of Machine Learning Research*, 10(Nov), 2615–2637.

Shwartz-Ziv, R. & Tishby, N. (2017). Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Simon, I., Snavely, N. & Seitz, S. M. (2007). Scene summarization for online image collections. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8.

Simoncelli, E. P. & Olshausen, B. A. (2001). Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1), 1193–1216.

Slonim, N. & Tishby, N. (2000). Agglomerative information bottleneck. *Advances in neural information processing systems*, pp. 617–623.

Srinivas, S. & Babu, R. V. (2015). Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014a). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929-1958. Consulted at http://jmlr.org/papers/v15/srivastava14a.html.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014b). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.

Stanković, R. S. & Falkowski, B. J. (2003). The Haar wavelet transform: its status and achievements. *Computers & Electrical Engineering*, 29(1), 25–44.

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

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Talukder, K. H. & Harada, K. (2010). Haar wavelet based approach for image compression and quality assessment of compressed image. *arXiv preprint arXiv:1010.4084*.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. & Liu, C. (2018). A survey on deep transfer learning. *International Conference on Artificial Neural Networks*, pp. 270–279.

Tishby, N. & Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. *Information Theory Workshop (ITW), 2015 IEEE*, pp. 1–5.

Tishby, N., Pereira, F. C. & Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

Toews, M. & Arbel, T. (2003). *Entropy-of-likelihood feature selection for image correspondence*. IEEE.

Toews, M., Wells, W., Collins, D. L. & Arbel, T. (2010). Feature-based morphometry: Discovering group-related anatomical patterns. *NeuroImage*, 49(3), 2318–2327.

Torrence, C. & Compo, G. P. (1998). A practical guide to wavelet analysis. *Bulletin of the American Meteorological society*, 79(1), 61–78.

Turin, G. (1960). An introduction to matched filters. *IRE transactions on Information theory*, 6(3), 311–329.

Turk, M. & Pentland, A. (1991a). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1), 71–86.

Turk, M. A. & Pentland, A. P. (1991b). Face recognition using eigenfaces. *Proceedings 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 586–591.

Ullrich, K., Meeds, E. & Welling, M. (2017). Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*.

Vasilyev, A. (2015). CNN optimizations for embedded systems and FFT. Stanford, Thesis.

Vink, R. (2017). Programming a neural network from scratch.

Wachinger, C., Reuter, M., Initiative, A. D. N. et al. (2016). Domain adaptation for Alzheimer's disease diagnostics. *Neuroimage*, 139, 470–479.

Wang, Y.-Q. (2014). An analysis of the Viola-Jones face detection algorithm. *Image Processing On Line*, 4, 128–148.

Wang, Y., Xu, C., Xu, C. & Tao, D. (2018). Packing convolutional neural networks in the frequency domain. *IEEE transactions on pattern analysis and machine intelligence*.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A. & Attenberg, J. (2009). Feature hashing for large scale multitask learning. *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1113–1120.

Weiss, K., Khoshgoftaar, T. M. & Wang, D. (2016). Transfer learning techniques. In *Big Data Technologies and Applications* (pp. 53–99). Springer.

Wells, W. M., Viola, P., Atsumi, H., Nakajima, S. & Kikinis, R. (1996). Multi-modal volume registration by maximization of mutual information. *Medical image analysis*, 1(1), 35–51.

Wen, W., Wu, C., Wang, Y., Chen, Y. & Li, H. (2016). Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems*, pp. 2074–2082.

Williams, T. & Li, R. (2016). Advanced image classification using wavelets and convolutional neural networks. *Machine Learning and Applications (ICMLA), 2016 15th IEEE International Conference on*, pp. 233–239.

Williams, T. & Li, R. (2018). Wavelet Pooling for Convolutional Neural Networks.

Wu, J., Leng, C., Wang, Y., Hu, Q. & Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4820–4828.

Yang, C., Rangarajan, A. & Ranka, S. (2018). Visual Explanations From Deep 3D Convolutional Neural Networks for Alzheimer's Disease Classification. *arXiv preprint arXiv:1803.02544*.

Yang, T.-J., Chen, Y.-H. & Sze, V. (2017). Designing energy-efficient convolutional neural networks using energy-aware pruning. *arXiv preprint*.

Yang, Y., Jojic, N. & Huan, J. (2019). FSNet: Compression of Deep Convolutional Neural Networks by Filter Summary. *arXiv preprint arXiv:1902.03264*.

Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L. & Wang, Z. (2015). Deep fried convnets. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1476–1483.

Zhang, X., Zou, J., Ming, X., He, K. & Sun, J. (2015). Efficient and accurate approximations of nonlinear convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1984–1992.

Zhou, S., Wu, J.-N., Wu, Y. & Zhou, X. (2015). Exploiting local structures with the kronecker layer in convolutional networks. *arXiv preprint arXiv:1512.09194*.

Zhou, W., Lu, Y., Li, H. & Tian, Q. (2012). Scalar quantization for large scale image search. *Proceedings of the 20th ACM international conference on Multimedia*, pp. 169–178.