Robust and Efficient Techniques in Deep Learning

with applications in Computer Vision and Language Understanding

A Dissertation

Presented to

The Faculty of the Graduate School of Arts and Sciences

Brandeis University

Department of Computer Science

James A. Storer, Department of Computer Science, Advisor

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

Aaditya Prakash

May, 2019

The signed version of this form is on file in the Graduate School of Arts and Sciences.

This dissertation, directed and approved by Aaditya Prakash's committee, has been accepted and approved by the Graduate Faculty of Brandeis University in partial fulfillment of the requirements for the degree of:

DOCTOR OF PHILOSOPHY

Eric Chasalow, Dean of Arts and Sciences

Dissertation Committee:
James A. Storer, Department of Computer Science, Chair
Antonella DiLillo, Department of Computer Science
Sadid Hasan, Artificial Intelligence Lab, Philips Research

Acknowledgments

I would like to thank my advisor James Storer for the wonderful time I have had during my graduate school years. He provided a perfect amount of supervision while letting me explore various areas of research that interested me. He made sure that I had a good work-life balance and that I was happy and passionate about the research. I could not have asked for a better advisor and a mentor.

I would like to thank Sadid Hasan (Philips Research), Raghuraman Krishnamoorthi (Qualcomm), Dinei Florencio (Microsoft) and Cha Zhang (Microsoft) who were my mentors during my summer internships and from whom I got to learn tremendously.

I am indebted to Nick Moran who not only proofread all the papers that constitute this thesis but also served as a sounding board for all the ideas.

I would also like to thank Solomon Garber, Ryan Marcus and Antonella Di Lillo for various ideas and discussions throughtout my graduate years and for proof-reading all my papers.

I would like to acknowledge generous grants and donations by NVIDIA, Intel and Google, which made possible most of the research.

Last but not least I would like to thank my parents Meena Karna and Vinay K Karna for always encouraging me to strive for the best.

Abstract

Robust and Efficient Techniques in Deep Learning with applications in Computer Vision and Language Understanding

A dissertation presented to the Faculty of
the Graduate School of Arts and Sciences of
Brandeis University, Waltham, Massachusetts

by Aaditya Prakash

Advancements in Deep Neural Networks has made a tremendous impact in various tasks. However, these methods have their limitations. Most of the CNN and RNN based models are computationally expensive and require special hardware to operate at a reasonable rate. More importantly, these methods are easily fooled by small changes in the input. These limitations are a hindrance to the widespread use of these models and prevent use in critical applications. In this dissertation, I discuss these issues in details and propose solutions to overcome both these issues.

Yet another issue with a lot of research in deep learning is their limited use-case beyond typical examples. I explore applications of these novel methods to image compression, paraphrase generation, and disease diagnosis. By changing the existing structure or the training process we are able to apply deep learning methods to these tasks. Our methods yield results that are significantly better than standard methods without paying significantly higher in computational cost.

Preface

This thesis is comprised of various published research. Chapters are divided as independent papers and provides all necessary introductions and related literature. Here is a brief description of the chapters.

Chapter One provides a brief discussion of Neural Networks and Convolutional Neural Networks that is necessary to follow the subsequent chapters. Only necessary elaboration is provided and the reader is encouraged to explore various textbooks in the area for a complete and a thorough guide.

Chapter two presents the research which shows how to use a Convolutional Neural Network to make image compression that is semantically aware. It has long been considered a significant problem to improve the visual quality of lossy image and video compression. Recent advances in computing power together with the availability of large training data sets has increased interest in the application of deep learning CNNs to address image recognition and image processing tasks. Here, we present a powerful CNN tailored to the specific task of semantic image understanding to achieve higher visual quality in lossy compression. A modest increase in complexity is incorporated to the encoder which allows a standard, off-the-shelf jpeg decoder to be used. While jpeg encoding may be optimized for generic images, the process is ultimately unaware of the specific content of the image to be compressed. Our technique makes jpeg content-aware by designing and training a model to identify multiple

semantic regions in a given image. Unlike object detection techniques, our model does not require labeling of object positions and is able to identify objects in a single pass. We present a new CNN architecture directed specifically to image compression, which generates a map that highlights semantically-salient regions so that they can be encoded at higher quality as compared to background regions. By adding a complete set of features for every class, and then taking a threshold over the sum of all feature activations, we generate a map that highlights semantically-salient regions so that they can be encoded at a better quality compared to background regions. Experiments are presented on the Kodak PhotoCD dataset and the MIT Saliency Benchmark dataset, in which our algorithm achieves higher visual quality for the same compressed size. This chapters presents the work published as [Pra+17] –

Prakash, Aaditya, Nick Moran, Solomon Garber, Antonella DiLillo and James A. Storer. ***"Semantic Perceptual Image Compression Using Deep Convolution Networks."*** **2017 Data Compression Conference (DCC - Oral) (2017)**

Chapter three presents the research which improves upon the results from chapter two by making these images robust in the presence of an adversary. As deep neural networks (DNNs) have been integrated into critical systems, several methods to attack these systems have been developed. These adversarial attacks make imperceptible modifications to an image that fool DNN classifiers. We present an adaptive JPEG encoder which defends against many of these attacks. Experimentally, we show that our method produces images with high visual quality while greatly reducing the potency of state-of- the-art attacks. Our algorithm requires only a modest increase in encoding time, produces a compressed image which can be decompressed by an off-the-shelf JPEG decoder, and classified by an unmodified classifier. This chapters presents the work published as [Pra+18b]–

Prakash, Aaditya, Nick Moran, Solomon Garber, Antonella DiLillo and James

A. Storer. *"Protecting JPEG Images Against Adversarial Attacks."* 2018 Data Compression Conference (DCC - Oral) (2018)

Chapter four extends the work of preventing use of adversarial images from being used to fool the deep networks. CNNs are poised to become integral parts of many critical systems. Despite their robustness to natural variations, image pixel values can be manipulated, via small, carefully crafted, imperceptible perturbations, to cause a model to misclassify images. We present an algorithm to process an image so that classification accuracy is significantly preserved in the presence of such adversarial manipulations. Image classifiers tend to be robust to natural noise, and adversarial attacks tend to be agnostic to object location. These observations motivate our strategy, which leverages model robustness to defend against adversarial perturbations by forcing the image to match natural image statistics. Our algorithm locally corrupts the image by redistributing pixel values via a process we term pixel deflection. A subsequent wavelet-based denoising operation softens this corruption, as well as some of the adversarial changes. We demonstrate experimentally that the combination of these techniques enables the effective recovery of the true class, against a variety of robust attacks. Our results compare favorably with current state-of-the-art defenses, without requiring retraining or modifying the CNN. This chapters presents the work published as [Pra+18a]–

Prakash, Aaditya, Nick Moran, Solomon Garber, Antonella DiLillo and James A. Storer. *"Deflecting Adversarial Attacks with Pixel Deflection."* 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR - Spotlight) (2018)

Chapter five finds yet another limitation of deep networks - redundancy in feature repre-

sentation. A well-trained Convolutional Neural Network can easily be pruned without significant loss of performance. This is because of unnecessary overlap in the features captured by the network's filters. Innovations in network architecture such as skip/dense connections and Inception units have mitigated this problem to some extent, but these improvements come with increased computation and memory requirements at run-time. We attempt to address this problem from another angle - not by changing the network structure but by altering the training method. We show that by temporarily pruning and then restoring a subset of the model's filters, and repeating this process cyclically, overlap in the learned features is reduced, producing improved generalization. We show that the existing model-pruning criteria are not optimal for selecting filters to prune in this context and introduce inter-filter orthogonality as the ranking criteria to determine under-expressive filters. Our method is applicable both to vanilla convolutional networks and more complex modern architectures, and improves the performance across a variety of tasks, especially when applied to smaller networks. This chapters presents the work published as [Pra+19]–

Prakash, Aaditya, James A. Storer, Dinei A. F. Florêncio and Cha Zhang. *"RePr: Improved Training of Convolutional Filters."* 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR - Oral) (2019)

Chapter six gives a brief description of how to extend Convolutional Neural Networks to incorporate languages when the task involves multimodal signals. We propose a version of highway network designed for the task of Visual Question Answering. We take inspiration from recent success of Residual Layer Network and Highway Network in learning deep representation of images and fine grained localization of objects. We propose variation in gating mechanism to allow incorporation of word embedding in the information highway. The gate

parameters are influenced by the words in the question, which steers the network towards localized feature learning. This achieves the same effect as soft attention via recurrence but allows for faster training using optimized feed-forward techniques. We are able to obtain state-of-the-art results on VQA dataset for Open Ended and Multiple Choice tasks with current model. This chapters presents the work published as [PB16]–

Prakash, Aaditya and James Storer Brandeis. *"Highway Networks for Visual Question Answering."* IEEE/CVF Conference on Computer Vision and Pattern Recognition (VQA - Workshop - Spotlight) (2016)

Chapter seven goes deeper into languge domains and explores the idea of residual connection in the context of languages. we propose a novel neural approach for paraphrase generation. Conventional paraphrase generation methods either leverage handwritten rules and thesauri-based alignments, or use statistical machine learning principles. To the best of our knowledge, this work is the first to explore deep learning models for paraphrase generation. Our primary contribution is a stacked residual LSTM network, where we add residual connections between LSTM layers. This allows for efficient training of deep LSTMs. We experiment with our model and other state-of-the-art deep learning models on three different datasets: PPDB, WikiAnswers and MSCOCO. Evaluation results demonstrate that our model outperforms sequence to sequence, attention-based and bi-directional LSTM models on BLEU, METEOR, TER and an embedding-based sentence similarity metric. This chapters presents the work published as [Pra+16b]–

Prakash, Aaditya, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu and Oladimeji Farri. *"Neural Paraphrase Generation with Stacked Residual LSTM Networks."* COLING (2016)

Chapter eight explores memory networks and shows its efficacy in diagnosis of diseses. Diagnosis of a clinical condition is a challenging task, which often requires significant medical investigation. Previous work related to diagnostic inferencing problems mostly consider multivariate observational data (e.g. physiological signals , lab tests etc.). In contrast, we explore the problem using free-text medical notes recorded in an electronic health record (EHR). Complex tasks like these can benefit from structured knowledge bases, but those are not scalable. We instead exploit raw text from Wikipedia as a knowledge source. Memory networks have been demonstrated to be effective in tasks which require comprehension of free-form text. They use the final iteration of the learned representation to predict probable classes. We introduce condensed memory neural networks (C-MemNNs), a novel model with iterative condensation of memory representations that preserves the hierarchy of features in the memory. Experiments on the MIMIC-III dataset show that the proposed model outperforms other variants of memory networks to predict the most probable diagnoses given a complex clinical scenario. This chapters presents the work published as [Pra+16a]– **Prakash, Aaditya, Siyuan Zhao, Sadid A. Hasan, Vivek Datla, Kathy Lee, Ashequl Qadir, Joey Liu and Oladimeji Farri. *"Condensed Memory Networks for Clinical Diagnostic Inferencing."* AAAI (2017)**

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Introduction to Neural Networks

Neural Networks are a type of a machine learning model inspired by activities in human brain. A neural network comprises of 'neurons', which are considered as the building blocks. A neuron in a neural network is implemented as weighted sum of its input, which is then passed through a non-linear function. This non-linear function is often denoted as an activation function and most commonly is a sigmoid or a rectified linear function.

Let W be weights and x denote the inputs. Let g be the activation function, then a single neuron is represented as -

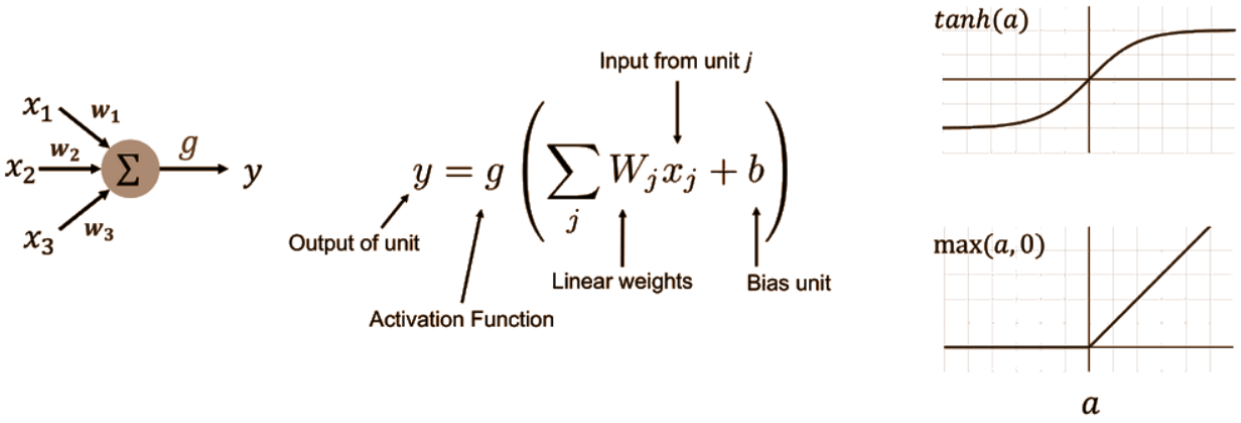


Figure 1.1: Artificial Neuron and common activation functions

When multiple of neurons are connected in a layer to all the inputs and the output of these neurons are used as input to another set of neurons, then the structure is called as Neural Network. This is shown in the figure 1.2. In this network x_1 and x_2 are the inputs to the network. $f_1(e)$, $f_2(e)$ and $f_3(e)$ are the outputs of three neurons in the first layer. They form as the inputs to the second layer. Simillarly $f_4(e)$ and $f_5(e)$ are the outputs of second layer and $f_6(e)$ is the final output which is used to predict the final value - y.

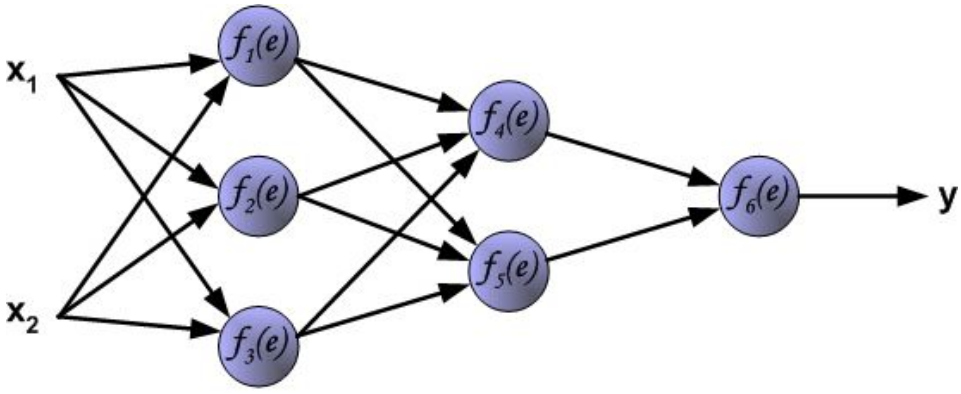


Figure 1.2: Artificial Neuron and common activation functions

1.1.1 Learning

Learning of Neural Networks is done in four steps –

1. Forward pass

   During the forward pass the outputs are computed by multiplying the weights w with the inputs (x). Let $w \times x$ be denoted as e and the activation function be denoted as f, then output of a layer $f(e)$ would be $f_1(w_1 \times x_1 + w_2 \times x_2)$. If we consider the network shown in figure 1.2, then a single forward pass would be as depicted in figure 1.3.

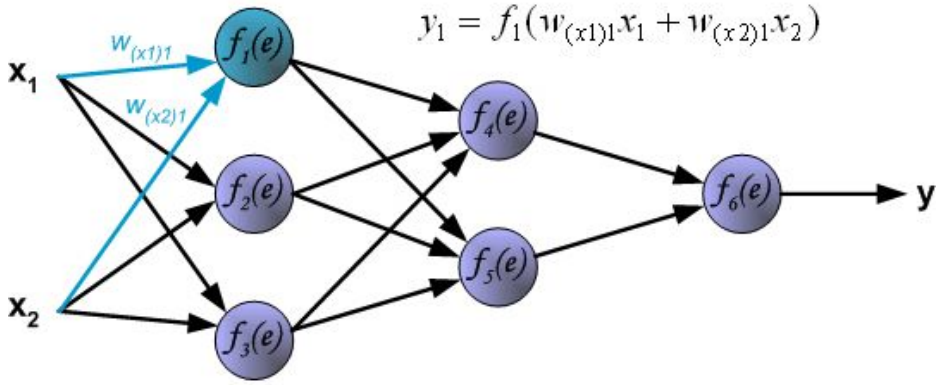


Figure 1.3: Computation of a forward pass in a neural network

2. Compute error

   Error is computed once the model has predicted its output (y). Let z be the true output for the given data points. Error function or the loss function takes in the predicted value (y) and the true value (z) and returns some numeric value. This function is task specific and is different for different tasks. For simplicity let's assume a simple difference, then the error δ is $z - y$, as shown in the figure 1.4.

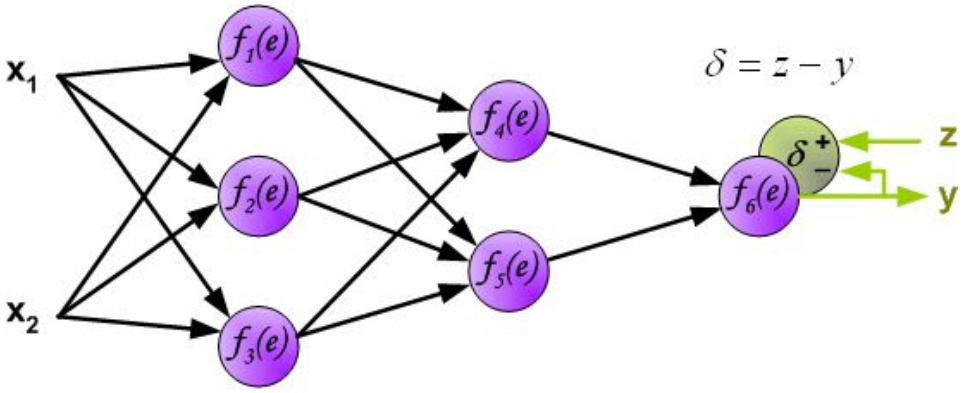


Figure 1.4: Computation of error in a neural network

3. Backward pass

   Backward pass is a phase where error with respect to every neuron is computed. Error at a given layer is the proportion of the error contributed by that layer with respect to to the total error. In order to compute error for layer l we need to know error at layer $l + 1$. Thus, it is computed from layer layer to the first layer and hence called Backward pass. For the networks shown above, error at neuron 2 is weighted sum of errors at neuron 4 and 5. This is shown in the figure 1.5.

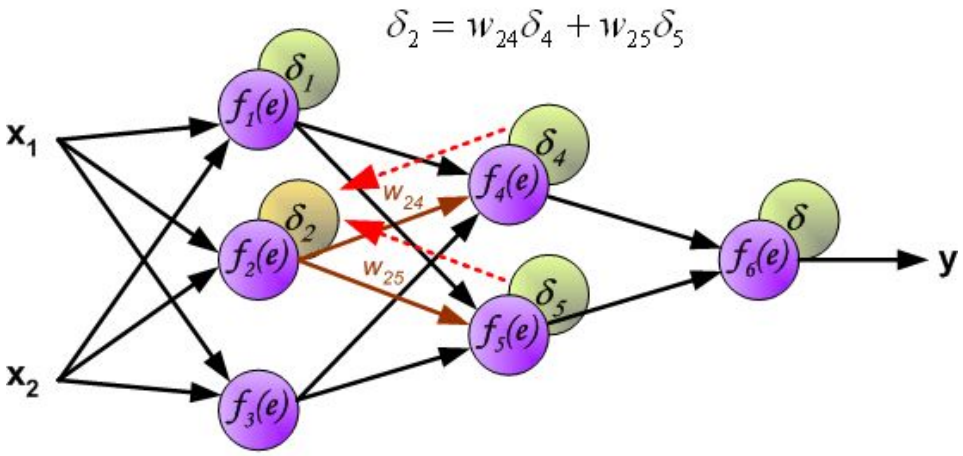


Figure 1.5: Backward pass of the error in a neural network

4. Weight update

Final step is to use the error at each neuron to change the weights. Current weight is changed by product of three values - error at the current neuron (δ), gradient of the output with respect to to the input at that node ($\frac{df(e)}{de}$)and the output y. Generally, this update value is too large and needs to be scaled. This scaling parameter (η) is called as the **learning rate** and is generally a hyper-parameter of the network. This step is shown in the figure 1.6.

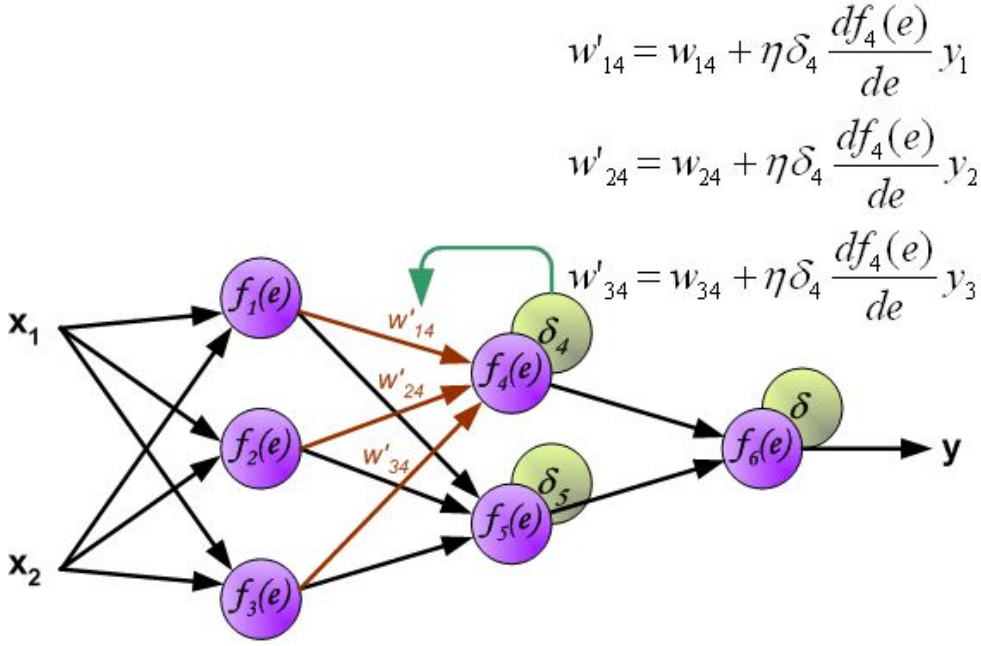


Figure 1.6: Update of weights in a neural network

Learning of the neural networks is summarized in algorithm 1.

Algorithm 1: Training of a Neural Network

Input : $X \in \mathbb{R}^n$

Output: None

Initialize the network weights with random values
FORWARD PASS
for $ex \leftarrow TrainingData$ **do**
    Let z = correct output for ex
    Prediction: y = Output(Network, ex)
    Error: δ = Loss-Function$(y - z)$
end
BACKWARD PASS
    Compute δ for all the weights in all layers
UPDATE WEIGHTS
    Update network weights
IF $\delta < \epsilon$: STOP

1.2 Introduction to Convolutional Neural Networks

Convolutional Neural Network (CNN) are like standard neural networks except that not all neurons are connected to every input. Only local neighbourhood of inputs are shared for some set of neurons as depicted. Figure 1.7 shows the difference between standard Neural Networks (left) and Convolutional Neural Neworks (right).

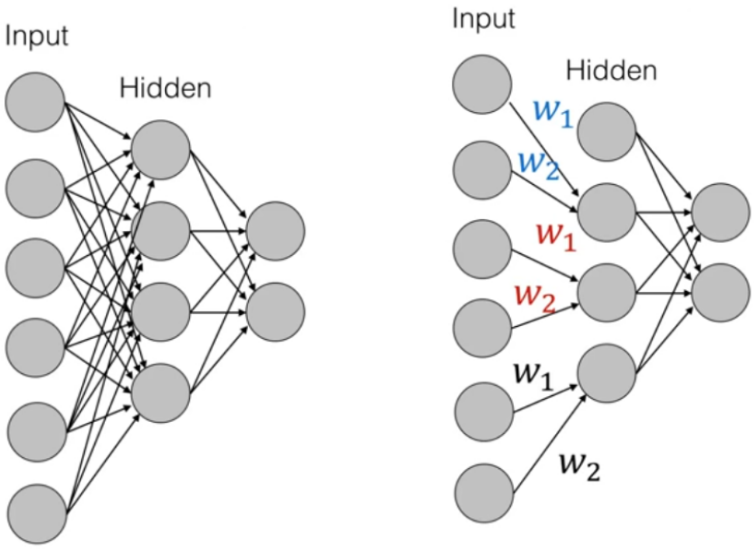


Figure 1.7: Difference between Standard Neural Network (left) and Convolutional Neural Network (Right)

1.2.1 Convolution

While figure 1.7 shows CNN in one-dimensional input, it is most widely used with images which are 2-D. For 2-D data the local inputs shared are a smaller 2-D window in the given image. In order to compute the output a smaller 2-D weights (filter) is convolved across the image. Generally, this leads to 2-D output but with slighly smaller spatial dimensions due to lack of enough values during convolving at the extreme points. This kind of convolution is referred to as a 'VALID' convolution. In order to achieve the same spatial output the input can be padded with zeros. This kind of convolution is commonly referred to as 'SAME' convolution. An example of a VALID 2-D convolution is shown in the figure 1.8. Resulting outputs are called as activation maps. If a layer learns k filters, then there are k activation maps as the output.

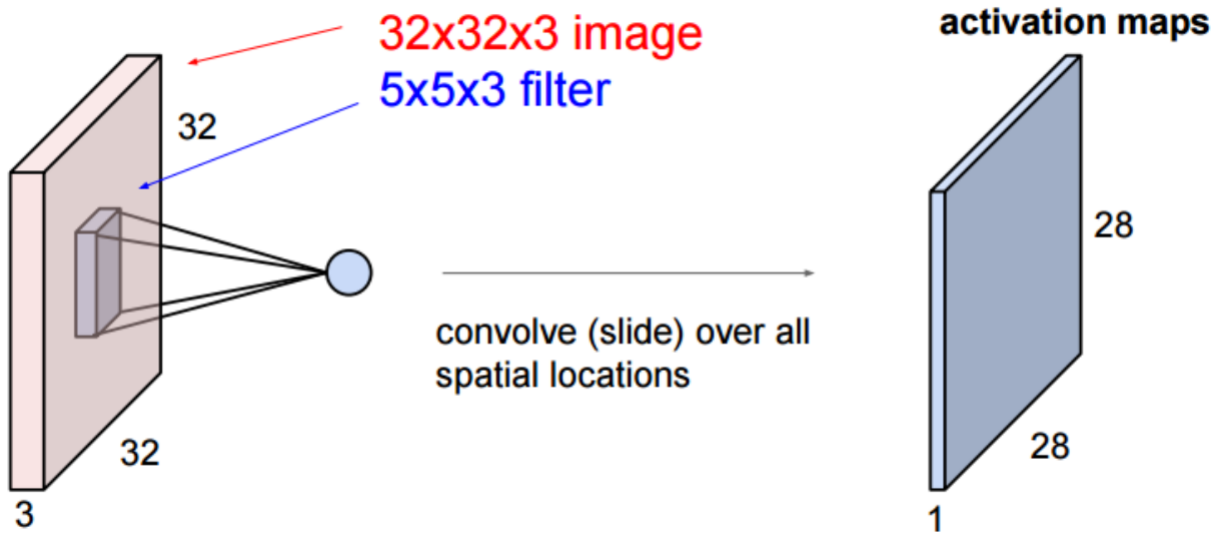


Figure 1.8: VALID 2-D Convolution

1.2.2 Pooling

Convolutional Neural Networks take in images of some size $H \times W$ and generally, output fixed scalar values in $\mathbb{R}^n$, where $n << H, W$. This often necessitates decreasing the dimensions of the activation maps signficantly (much more than the loss of edges due to VALID convolution). One common technique used to decrease the spatial dimensions is to pool the values within a smaller window. For all values in the window of $h \times w$ is replaced by a single value. Quite often the aggregate function used is the **max** operation however some models also use average of the values. Pooling operation is depicted in the figure 1.9. In this example pooling window is of the size 2×2, which results in the activation halfing in spatial dimensions. It is important to note that Pooling does not add any learnable parameters to the network and is often used to reduce the dimensions of the intermediate features.

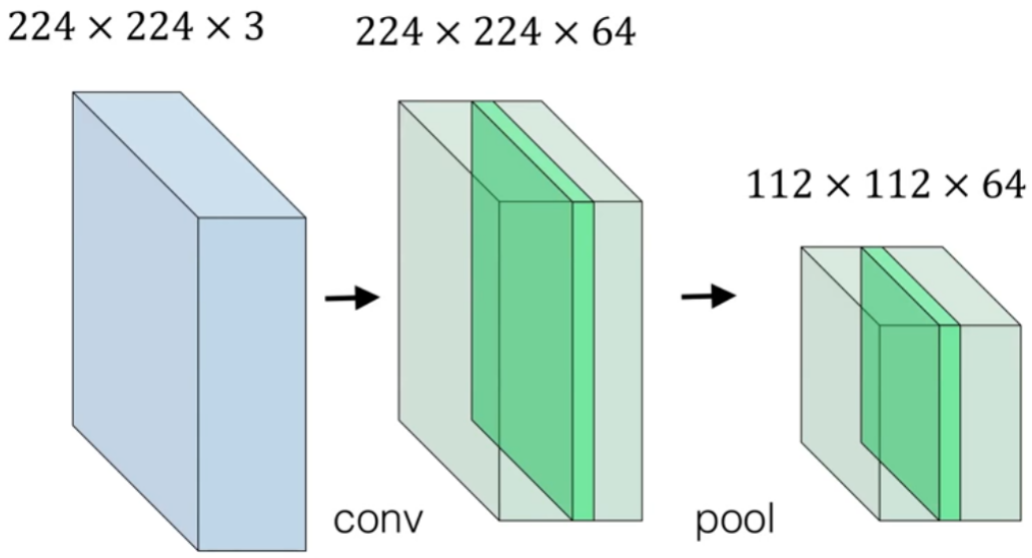


Figure 1.9: Pooling

1.2.3 Normalization

It is common in Convolutional Neural Networks to include a normalization layer. Normalization layer attempts to rescale the activation values to certain parameters. This is often necessitated due to high variance in the natural images. It has also been shown that normalizing activations leads to faster convergence and sometimes improved performance. In recent times Batch Normmalization has become the preferred choice. If x is the input data and let μ be the mean of all the activations of the data and σ be the standard deviation of the data. Batch normalization first scales the value by subtracting the mean (μ) and dividing by standard deviation σ. It then adds two learned parameter β and γ, which are used to shift and scale the activations. Thus,

$$BN(x) = \frac{x - \mu}{\sigma}\gamma + \beta$$

1.2.4 Hierarchical Features

It has been shown that stacking multiplie layers of convolution operation interspread with pooling leads to learning of hierarchical features. These features become more high-level (semantic) as the layer depth increases. This allows model to learn the semantics in the image and use that to accomplish tasks like classification and detection. Figure 1.10 shows example of filters learned at different depths. One can see that the high-level features are generally observed at filters from higher depth (box on the right) and they look to be formed using the filters from earlier layer (box in the middle), which in turn seems to be compiled from first layer features (left). It has also been shown that first layer filters correspond to edge detectors and gabor filters.

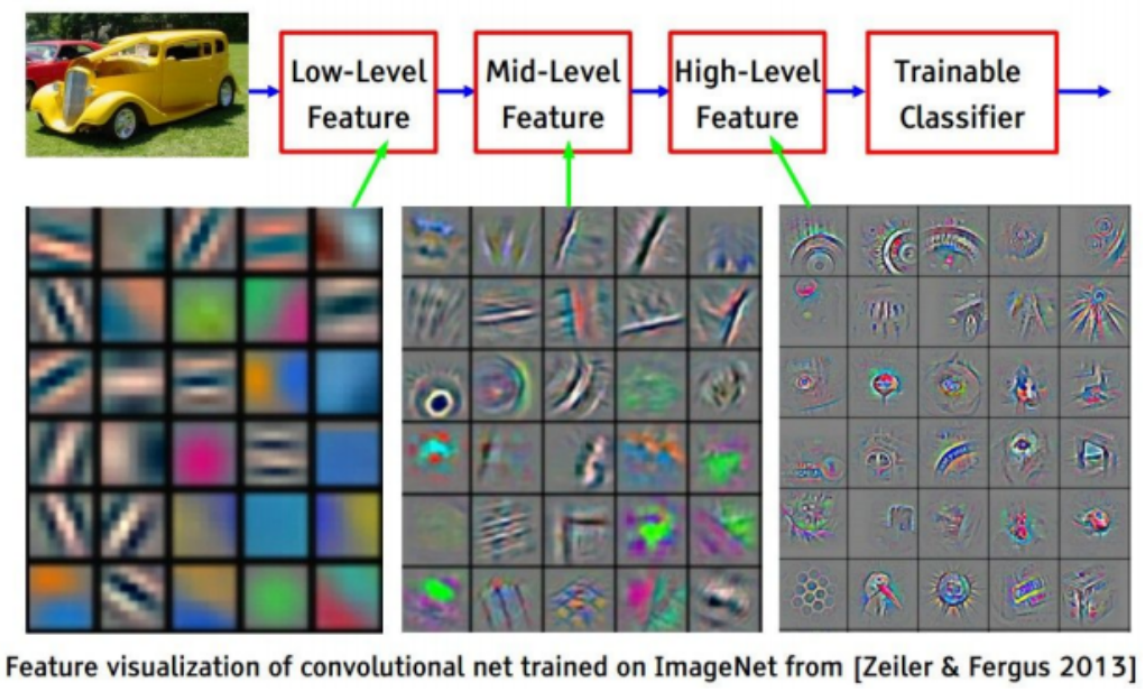


Figure 1.10: Hierarchy of features in CNN

Training of Convolutional Neural Networks is done simillarly to standard Neural Net-

works as shown in the section 1.1.1. However, in practice it has been found that training a CNN with many layers is difficult due to vanishing of gradients (backward pass). Several improvements have been suggested to overcome the problem, augment loss term at intermediate depths (Inception Network) [Sze+16], residual skip connections [He+16a], Highway Networks [SGS15] etc.

Chapter 2

Multi-structure Regions of Interest

2.1 Introduction

Several attempts have been made to improve upon the lossy image compression offered by JPEG [GPG12] [Tod+16]. Despite these efforts, JPEG continues to be the standard image file format on the web. Because of the status of JPEG as the default standard and its wide adoption, it seems unlikely that the new formats will get any traction. We propose an image compression technique to improve the visual quality of standard JPEG by using a higher bitrate to encode image regions flagged by our model as containing an object of interest and lowering the bitrate elsewhere in the image. The compressed output of our method can be decoded by standard JPEG implementations.

The JPEG algorithm uses a scaling factor Q in order to scale the quantization matrix to achieve a variety of compression ratios. However, this ratio is an image level property and all the 8×8 blocks of a given image are compressed using the same scaling factor. Natural images are heterogeneous with respect to frequency, and contain both regions of primarily low frequencies and of primarily high frequencies. While low frequency regions are more tolerant

of higher compression ratios, high frequency regions are not [CE99]. Therefore, variable quantization in JPEG would aim to provide optimum perceptual quality across the image by compressing different blocks at different ratios. Previous approaches to use variable quantization have employed DCT analysis on each block to determine the frequency components [KT98] as well as classification of each block according to a pre-determined look-up table [MT00]. Soon et al [TPN96] proposed classifying blocks as textures, edges or flat regions, and adjusting the quantization matrix for each block accordingly. Adaptive Quantization techniques have also been applied to video coding [Xia+14].

These methods are limited in their ability to improve the perceptual quality of an image because frequency analysis and image metrics do not correlate with human perception [KSC92]. Therefore, we propose variable quantization of JPEG in which the choice of scaling factor is informed by semantic knowledge of the image. Human vision naturally focuses on familiar semantic objects, and is particularly sensitive to distortions of these objects as compared to distortions of background details. Our goal is to develop a technique which improves the visual quality of an image by improving the signal to noise ratio within these semantic objects while keeping the overall visual quality close to that of standard JPEG. Measuring visual quality is an ongoing research area and there is no consensus among researchers on the proper metric. We evaluate our model on a variety of metrics, `SSIM`[Wan+04a], `MS-SSIM`[WSB03], `VIFP`[SB06], `PSNR-HVS`[Egi+06] and `PSNR-HVSM`[Pon+07]; these robust metrics have been shown to correlate better with subjective quality measurement than pixel-level metrics such as `MSE` or `PSNR` [Pon+07]. Yuri et al [KVR15] showed that `PSNR` as an image comparison metric has severe limitations.

Convolutional Neural Networks (CNNs) have been successfully applied to a variety of computer vision tasks [He+16b] [KSH12]. Their feature extraction and transfer learning capabilities are now well known[ZF14] . CNNs, well known for their ability to classify

images by their most prominent object, and have also been used to draw a bounding box around that object [Gir+14]. This method is capable of providing a binary map for the presence of the most salient object. Some success has been obtained in predicting the visual saliency map of a given image [Jia+15b] [KTB14a].

We propose a deep convolution network designed to locate several semantic objects within a single image. Our model differs from traditional object detection models like [Dai+16] [Gir+14] as these models are restricted to detecting a single salient object in an image, typically an instance of a pre-computed class for that image. Previous work has shown that semantic object detection has a variety of advantages over saliency maps [MHG+14] [Zün+13]. Semantic detection models recognize discrete objects and are thus able to generate maps that are more coherent for human perception. Visual saliency models are based on human eye fixations, and thus produce results which do not capture object boundaries. This is particularly evident in the results obtained by Stella et al [SL09], in which image compression is guided by a multi-scale saliency map, and the obtained images show blurred edges and soft focus. Our proposed model captures the structure of the depicted scene and thus maintains the integrity of semantic objects, unlike results produced using human eye fixations [Liu+15].

Our proposed model produces a single class-invariant feature map by learning separate feature maps for each of a set of object classes and then summing over the top features. Our model trades lower confidence of bounding edges for the ability to find all salient regions of an image in a single pass, as opposed to standard object-detection CNNs, which require multiple passes over the image to identify and locate all the objects. In comparison to grid-based features as described by Yuri et al [Rez+13] our features are scale- and transformation-invariant, which allows application of the model to a wider class of images.

We employ a Convolutional Neural Network (CNN) tailored to the specific task of se-

mantic image understanding to achieve higher visual quality in lossy image compression. We focus on the JPEG standard, which remains the dominant image representation on the internet and in consumer electronics. Several attempts have been made to improve upon its lossy image compression, for example WebP [GPG12] and Residual-GRU [Tod+16], but many of these require custom decoders and are not sufficiently content-aware.

We improve the visual quality of standard JPEG by using a higher bit rate to encode image regions flagged by our model as containing content of interest and lowering the bit rate elsewhere in the image. With our enhanced JPEG encoder, the quantization of each region is informed by knowledge of the image content. Human vision naturally focuses on familiar objects, and is particularly sensitive to distortions of these objects as compared to distortions of background details [Jia+15b]. By improving the signal-to-noise ratio within multiple regions of interest, we improve visual quality of those regions, while preserving overall `PSNR` and compression ratio. A second encoding pass produces a final JPEG encoding that may be decoded with any standard off-the-shelf JPEG decoder. Measuring visual quality is an ongoing area of research and there is no consensus among researchers on the proper metric. Yuri et al [KVR15] showed that `PSNR` has severe limitations as an image comparison metric. Richter et al [RK09][Ric11] addressed structural similarity (`SSIM`[Wan+04a] and `MS-SSIM`[WSB03]) for JPEG and JPEG 2000. We evaluate on these metrics, as well as `VIFP`[SB06], `PSNR-HVS`[Egi+06] and `PSNR-HVSM`[Pon+07], which have been shown to correlate with subjective visual quality. Figure 2.1 compares close-up views of a salient object in a standard JPEG and our new content-aware method.

Figure 2.1: Comparison of compression of semantic objects in standard JPEG[left] and our model [right]

CNNs have been successfully applied to a variety of computer vision tasks [KSH12]. The feature extraction and transfer learning capabilities of CNNs are well known [ZF14], as are their ability to classify images by their most prominent object [He+16b], and compute a bounding box [Gir+14]. Some success has been obtained in predicting the visual saliency map of a given image [Jia+15b],[KTB14a]. Previous work has shown that semantic object detection has a variety of advantages over saliency maps [MHG+14], [Zün+13]. Semantic detection recognizes discrete objects and is thus able to generate maps that are more coherent for human perception. Visual saliency models are based on human eye fixations, and thus produce results which do not capture object boundaries [KTB14a]. This is evident in the results obtained by Stella et al [SL09], in which image compression is guided by a multi-scale saliency map, and the obtained images show blurred edges and soft focus.

We present a CNN designed to locate multiple regions of interest (ROI) within a single

image. Our model differs from traditional object detection models like [Dai+16], [Gir+14] as these models are restricted to detecting a single salient object in an image. It captures the structure of the depicted scene and thus maintains the integrity of semantic objects, unlike results produced using human eye fixations [Liu+15]. We produce a single class-invariant feature map by learning separate feature maps for each of a set of object classes and then summing over the top features. Because this task does not require precise identification of object boundaries, our system is able to capture multiple salient regions of an image in a single pass, as opposed to standard object detection CNNs, which require multiple passes over the image to identify and locate multiple objects. Model training need only be done offline, and encoding with our model employs a standard JPEG encoder combined with efficient computation of saliency maps (over 60 images per second for 1920x1080 using a Titan X Maxwell GPU). A key advantage of our approach is that its compressed output can be decoded by any standard off-the-shelf JPEG implementation. It serves to maintain the existing decoding complexity, the primary issue for distribution of electronic media. In contrast to grid based features as described by Yuri et al [Rez+13] our features are scale and translation invariant, which allows application of the model to a wider class of images.

Section 2 reviews CNN techniques used for object localization, semantic segmentation and class activation maps. We also discuss merits of using our technique over these methods. Section 3 presents our new model which can generate a map showing multiple regions of interest. In Section 4 show how we combine this map to make JPEG semantically aware. Sections 5 presents experimental results on a variety of image datasets and metrics. Section 6 concludes with future areas for research.

2.2 Review of localization using CNNs

CNNs are multi-layered feed-forward architectures where the learned features at each level are the weights of the convolution filters to be applied to the output of the previous level. Learning is done via gradient-based optimization [LB95]. CNNs differ from fully connected neural networks in that the dimensions of the learned convolution filters are, in general, much smaller than the dimensions of the input image, so the learned features are forced to be localized in space. Also, the convolution operation uses the same weight kernel at every image location, so feature detection is spatially invariant.

Given an image x, and a convolution filter of size $n \times n$, then a convolutional layer performs the operation shown in equation 2.1, where $\mathbf{W}$ is the learned filter.

$$y_{ij} = \sum_{a=0}^{n} \sum_{b=0}^{n} \mathbf{W}_{ab}\, x_{(i+a)(j+b)} \tag{2.1}$$

In practice, multiple filters are learned in parallel within each layer, and thus the output of a convolution layer is a 3-d feature map, where the depth represents the number of filters. The number of features in a given layer is a design choice, and may differ from layer to layer. CNNs include a max pooling [LB95] step after every or every other layer of convolution, in which the height and width of the feature map (filter response) are reduced by replacing several neighboring activations (coefficients), generally within a square window, with a single activation equal to the maximum within that window. This pooling operation is strided, but the size of the pooling window can be greater than the stride, so windows can overlap. This results in down-sampling of input data, and filters applied to such a map will have a larger receptive field (spatial support in the pixel space) for a given kernel size, thus reducing the number of parameters of the CNN model and allowing the training of much deeper networks.

Max pooling performed with a stride of $k \times k$ will reduce a feature matrix of size $N \times N$ to $\frac{N}{k} \times \frac{N}{k}$. This does not change the depth of the feature map, but only its width and height. In practice, pooling windows are typically of size 2×2 or 4×4, with a stride of two, which reduces the number activations by 75%. CNNs apply some form of non-linear operation such as sigmoid $(1 - e^{-x})^{-1}$ or linear rectifier $max(0, x)$ on the output of each convolution operation.

Region-based CNNs use a moving window to maximize the posterior of the presence of an object [Gir15]. However, this is computationally expensive as it requires looking for all possible classes in all possible regions. Faster RCNNs [Ren+15] have been proposed, but they are still computationally expensive and are limited to determining the presence or absence of a single class of object within the entire image. Moving-window methods are able to produce rectangular bounding boxes, but cannot produce object silhouettes. In contrast, recent deep learning models proposed for semantic segmentation [LSD15], [Gir+14], [Zhe+15] are very good at drawing a close border around the objects. However, these methods do not scale well to more than a small number of object categories (e.g. 20) [Eve+10]. Segmentation methods typically seek to produce a hard boundary for the object, in which every pixel is labeled as either part of the object or part of the background. In contrast, class activation mapping produces a fuzzy boundary and is therefore able to capture pixels which are not part of any particular object, but are still salient to some spatial interaction between objects. Segmentation techniques are also currently limited by the requirement for strongly-labeled data for training. Obtaining training data where the locations of all the objects in the images are tagged is expensive and not scalable [Eve+10]. Our approach only requires image-level labels of object classes, without pixel-level annotation or bounding-box localization.

2.2.1 Class Activation Mapping

Convolution filters learned from natural images are sensitive to the object classes in the training data [Oqu+15b], [LCY14]. In other words, regions of an image which contain relevant objects tend to produce high activations when convolved with learned filters, while regions that do not contain any object class have lower activations, and this is particularly true for the deeper levels of the CNN. In a traditional CNN, there are two fully-connected (non-convolutional) layers as the final layers of the network. The final layer has one neuron for every class in the training data, and the final step in the inference is to normalize the activations of the last layer to sum to one. The second to last layer, however, is fully connected to the last convolution layer, and a non-linearity is applied to its activations. The authors of [Oqu+15b], [Zho+16] modify this second to last layer to allow for class localization. In their architecture, the second to last layer is not learned, but consists of one neuron for each feature map, which has fixed equally-weighted connections to each activation of its corresponding map. They suggest one neuron for each feature map from the last convolution layer, and this neuron has connections of equal weight to every activation of its feature map and no connections to any other feature map. No non-linearity is applied to the outputs of these neurons, so that each activation in this layer represents the global spatial average of one feature map from the previous layer. Since the output of this layer is connected directly to the classification layer, each class will in essence learn a weight for each feature map from the final convolution layer. Thus, given an image and a class, the classification weights for that class can be used to re-weight the layers of activations of the final convolution layer on that image. These activations can be collapsed along the feature axis to create a class activation map, spatially localizing the best evidence for that class within that image. Taking the spatial average of the feature map (global average pooling) at the last convolution layer

reveals possible locations of objects within the image [Oqu+15b], [Zho+16]. This is used to make Class Activation Maps (CAM), which shows the probability distribution of the most likely class.

While standard max pooling combines spatially local activations for each feature independently, 'global max pooling' instead combines the activations of all features for each spatial location. This collapses the feature map to two spatial dimensions by reducing the D features at each location to a single maximum value. If the activation of a convolution layer is of size $M \times N \times D$ where M and N are the spatial dimensions of feature activations and D denotes the number of features in the feature map, then after 'global max pooling', the size of activations becomes $M \times N$. It is important to note that global average pooling is performed in lieu of a final fully-connected layer (or perceptron), which is often used in image classification tasks [SZ14]. Fully-connected layers have dense connections across all neurons and thus do not preserve spatial locality of activations. The substitution of a global max pooling layer retains this locality property in exchange for a fractional loss of classification accuracy[Oqu+15b]. Pooling of feature maps is probabilistic [LRH15], which means the most salient object is not always the one with highest activation across the feature map. If we plot the activation of each feature on the 2-dimensional map and overlay them on each other, we find significant overlap of regions of high activations. This means taking a 'global max pooling' would lead to missing some. Zhou [Zho+16] reported that they obtained better localization by performing 'global average pooling' instead of 'global max pooling'. This is because taking an average across the feature map ensures each detected object has consistently high activations across many features, instead of a single maximal activation of one feature. The application of GAP after the final layer of convolution y_{ij} over **D** features would result in – CAM is obtained by taking output of the last convolution layer $f(x, y)$, and taking a weighted sum across the feature map. Figure 2.2 (c) shows an example

of such a map, the equation of which is given by

$$M_c(x, y) = \sum_{d \,\epsilon\, \mathbf{D}} w_d^c \, f_d(x, y) \tag{2.2}$$

where w_d^c is the learned weight of class c for feature map d. M is referred to as class activation mapping (CAM) 2.2. It is used to generate discriminative localization for the most probable class. Training for CAM minimizes the cross entropy between objects' true probability distribution over classes (all mass given to the true class) and the predicted distribution, which is obtained as

$$P(c) = \frac{\exp(\sum_{xy} M_c(x, y))}{\sum_c \exp(\sum_{xy} M_c(x, y))} \tag{2.3}$$

Since CAMs are trained to maximize posterior probability for the class, they tend to only highlight a single most prominent object. This makes them useful for studying the output of CNNs, but not well suited to more general semantic saliency, as real world images typically contain multiple objects of interest. This has practical applications in understanding and visualizing convolution neural networks but is less applicable to real world images, which typically contain multiple objects of interest. Without using standard resolution, color-space or types of object classes it is hard to use such systems to extract semantic information from the images.

2.3 Multi-Structure Region of Interest

We have developed a variant of CAM which balances the activation for multiple objects and thus does not suffer from the issues of global average pooling. Our method, *Multi-Structure Region of Interest* (MS-ROI), allows us to effectively train on localization tasks independent

of the number of classes. For the purposes of semantic compression, obtaining a tight bound on the objects is not important. However, identifying and approximately locating all the objects is critical. We propose a set of 3D feature maps in which each feature map is learned for an individual class, and is learned independently of the maps for other classes. For $\mathbf{L}$ layers, where each layer l contains d_l features, an image of size $n \times n$, and with $\mathbf{C}$ classes, this results in a total activation size of

$$\sum_{l \,\epsilon\, \mathbf{L}} d_l \times \mathbf{C} \times \frac{n}{k^l} \times \frac{n}{k^l}$$

where k is the max pooling stride size. This is computationally very expensive, and not practical for real world data. CNNs designed for full-scale color images have many filters per layer and are several layers deep. For such networks, learning a model with that many parameters would be unfeasible in terms of computational requirements. We propose two techniques to make this idea feasible for large networks:

i Reduce the number of classes and increase the inter-class variance by combining similar classes

ii Share feature maps across classes to jointly learn lower level features

Most CNN models are built for classification on the Large Scale Visual Recognition Challenge, commonly known as ImageNet. [1]. ImageNet has one thousand classes and many of the classes are fine-grained delineations of various types of animals, flowers and other objects. We significantly reduce the number of classes by collapsing these sets of similar classes to a single, more general class. This is desirable because, for the purpose of selecting a class invariant 'region of interest,' we do not care about the differences between Siberian

[1] http://image-net.org/challenges/LSVRC/

husky and Eskimo dog or between Lace-flower and Tuberose. As long as objects of these combined classes have similar structure and are within the same general category, the map produced will be almost identical. Details of the combined classes used in our model are provided in the Experimental Results section.

It is self-evident that most images contain only a few classes and thus it is computationally inefficient to build a separate feature map for every class. More importantly, many classes have similar lower-level features, even when the number of classes is relatively small. The first few layers of a CNN learn filters which recognize small edges and motifs [ZF14], which are found across a wide variety of object classes. Therefore, we propose parameter sharing across the feature maps for different classes. This reduces the number of parameters and also allows for the joint learning of these shared, low-level features.

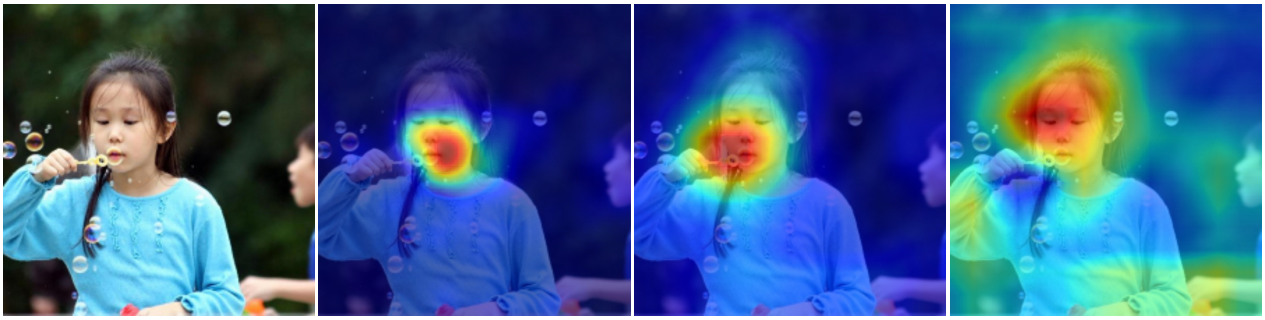


Figure 2.2: Comparison of various methods of detecting objects in an image

Although we do not restrict ourselves to a single most-probable class, it is desirable to eliminate the effects of activations for classes which are not present in the image. In order to accomplish this, we propose a thresholding operation which discards those classes whose learned features do not have a sufficiently large total activation when summed across all features and across the entire image. Let Z_l^c denote the total sum of the activations of layer ℓ for all feature maps for a given class c. Since our feature map is a 4-dimensional tensor,

Z_l^c can be obtained by summation of this tensor over the three non-class dimensions.

$$Z_l^c = \sum_{d \,\epsilon\, \mathbf{D}} \sum_{x,y} f_d^c(x,y) \tag{2.4}$$

Next, we use Z_l^c to filter the classes. Computation of the multi-structure region of interest is shown below.

$$\hat{M}(x,y) = \sum_{c \,\epsilon\, \mathbf{c}} \begin{cases} \sum_d f_d^c(x,y), & \text{if } Z_l^c > T \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

We use the symbol $\hat{M}$ to denote the multi-structure map generated by our proposed model in order to contrast it with the map generated using standard CAM models, M. $\hat{M}$ is a sum over all classes with total activations Z_l^c beyond a threshold value T. T is determined during the training or chosen as a hyper-parameter for learning. In practice, it is sufficient to *argsort* Z_l^c and pick the top five classes and combine them via a sum weighted by their rank. It should be noted that, because $\hat{M}$ is no longer a reflection of the class of the image, we use the term 'region of interest'.

A comparison of our model (MS-ROI) with CAM and human fixation is shown in Figure 2.2. Only our model identifies the face of the boy on the right as well the hands of both children at the bottom. When doing compression, it is important that we do not lower the quality of body extremities or other objects which other models may not identify as critical to the primary object class of the image. If a human annotator were to paint the areas which should be compressed at better quality, we believe the annotated area would be closer to that captured by our model.

To train the model to maximize the detection of all objects, instead of using a softmax function as in equation 2.3, we use sigmoid, which does not marginalize the posterior over

the classes. Thus the likelihood of a class c is given by equation 2.6.

$$P(c) = \frac{1}{1 + \exp(Z_l^c)} \tag{2.6}$$

2.4 Integrating MS-ROI map with JPEG

We obtain from MS-ROI a saliency value for each pixel in the range [0,1], where 1 indicates maximum saliency. Then discretize these saliency values into k levels, where k is a tuneable hyper-parameter. The lowest level contains pixels of saliency $[0, 1/k]$, the second level contains pixels of saliency $(1/k, 2/k]$ and so forth. We next select a range of JPEG quality levels, Q_l to Q_h. Each saliency level will be compressed using a Q value drawn from this range, corresponding to that level. In other words, saliency level n, with saliency range $[n/k, (n+1)/k]$ will be compressed using

$$Q_n = Q_l + \frac{n * (Q_h - Q_l)}{k} \tag{2.7}$$

For each level $l \leq n \leq h$, we obtain a decoded JPEG of the image after encoding at quality level Q_n. For each 8×8 block of our output image, we select the block of color values obtained by the JPEG corresponding to that block's saliency level. This mosiac of blocks is finally compressed using a standard JPEG encoder with the desired output quality to produce a file which can be decoded by any off-the-shelf JPEG decoder.

Details of our choices for k, Q_l and Q_h, as well as target image sizes are provided in the next section. A wider range of Q_l and Q_h will tend to produce stronger results, but at the

expense of very poor quality in non-salient regions.

Algorithm 2: JPEG Encoding with MSROI

Input : Image: I

Output: Image: O, same dimensions as I

Let $Q_{l,h}$ be N fixed values in range l to h

for $b \leftarrow 8 \times 8$ *blocks in* I **do**

    $Q_i = \hat{M}$[b] #Nearest quantized level

    I'[b] $\leftarrow$ JPEG {I[b], Q_i}

end

O $\leftarrow$ JPEG{I', Q_f}

Encoding of image using MSROI on a JPEG standard can be summarized as shown by algorithm 2.

Figure 2.3 shows the steps of taking an image and using the MSROI map to obtain the final encoded image.

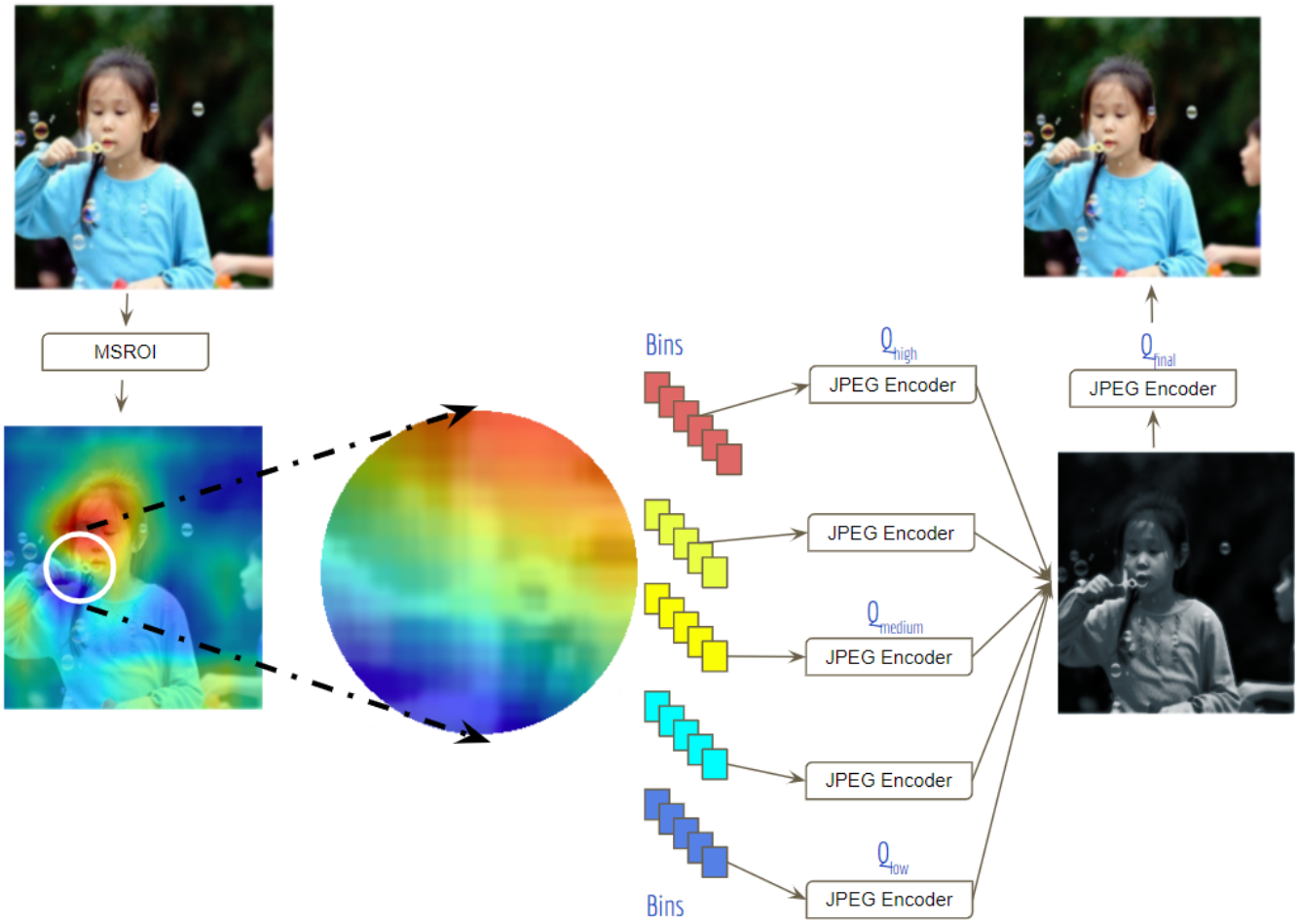


Figure 2.3: Process of combining MSROI maps to obtain final encoded image

2.5 Experimental Results

We trained our model with the Caltech-256 dataset [GHP07], which contains 256 classes of man-made and natural objects, common plants and animals, buildings, etc.

We believe this offers a good balance between covering more classes as compared to CIFAR-100 which contains only 100 classes, and avoiding overly finely-grained classes as in ImageNet with 1000 classes [Den+09a]. For the results reported here, we experimented with

several stacked layers of convolution as shown in the diagram below:

$$\text{IMAGE} \longmapsto \Big[\big[\text{CONV} \to \text{RELU}\big]^2 \to \text{MAXPOOL}\Big]^5 \longmapsto \text{MS-ROI} \longmapsto \text{MAP}$$

MS-ROI refers to the operation shown in the equation 2.5. To obtain the final image we discretize the heat-map into five levels and use JPEG quality levels Q in increments of ten from $Q_l = 30$ to $Q_h = 70$. For all experiments, the file size of the standard JPEG image and the JPEG obtained from our model were kept within $\pm 1\%$ of each other. On average, salient regions were compressed at $Q_f = 65$, and non-salient regions were compressed at $Q = 45$. The average Q for the final image generated using our model was 55, whereas for all standard JPEG samples, Q was chosen to be 50. It should be noted that even though images are at different Q they have same size.

Figure 2.4: Sample of our map for five KODAK images.

We tested on the Kodak PhotoCD set (24 images) and the the MIT dataset (2,000 images). Kodak is a well known dataset consisting primarily of natural outdoor color images. Figure 2.4 shows a sample of five of these images, along with the corresponding heatmaps

generated by our algorithm; the first four show typical results which strongly capture the salient content of the images, while the fifth is a rare case of partial failure, in which the heatmap does not fully capture all salient regions. The MIT set allows us to compare results across twenty categories. In Table 2.1 we only report averaged results across 'Outdoor Man-made' and 'Outdoor Natural' categories (200 images), as these categories are likely to contain multiple semantic objects, and are therefore appropriate for our method. Both datasets contain images of smaller resolutions, but the effectiveness of perceptual compression is more pronounced for larger images. Therefore, we additionally selected a very large image of resolution 8705×8400, which we scale to a range of sizes to demonstrate the effectiveness of our system at a variety of resolutions. See Figure 2.5 for the image sizes used in this experiment. Both Figure 2.5 and Figure 2.6 show the `PSNR-HVS` difference between our model and standard JPEG. Positive values indicate our model has higher performance compared to standard JPEG. In addition to an array of standard quality metrics, we also report a `PSNR` value calculated only for those regions our method has identified as salient, which we term `PSNR-S`. By examining only regions of high semantic saliency, this metric demonstrates that our compression method is indeed able to preserve visual quality in targeted regions, without sacrificing performance on traditional image-level quality metrics or compression ratio. It should be noted that the validity of this metric is dependent on the correctness of the underlying saliency map, and thus should only be interpreted to demonstrate the success of the final image construction in preserving details highlighted by that map.

	PSNR-S	PSNR	PSNR-HVS	PSNR-HVSM	SSIM	MS-SSIM	VIFP
Kodak PhotoCD [24 images]							
Std JPEG	33.91	34.70	34.92	42.19	0.969	0.991	0.626
Our model	39.16	34.82	35.05	42.33	0.969	0.991	0.629
MIT Saliency Benchmark [Outdoor Man-made + Natural, 200 images]							
Std JPEG	36.9	31.84	35.91	45.37	0.893	0.982	0.521
Our model	40.8	32.16	36.32	45.62	0.917	0.990	0.529
Re-sized images of a very large image, see fig: 2.5 [20 images]							
Std JPEG	35.4	27.46	33.12	43.26	0.912	0.988	0.494
Our model	39.6	28.67	34.63	44.89	0.915	0.991	0.522

Table 2.1: Results across datasets

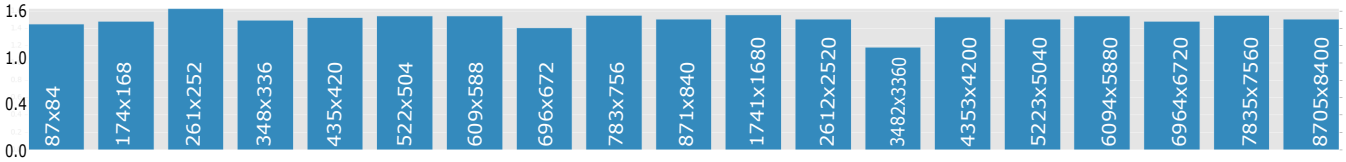


Figure 2.5: PSNR-HVS of our model - JPEG across various image size (higher is better).

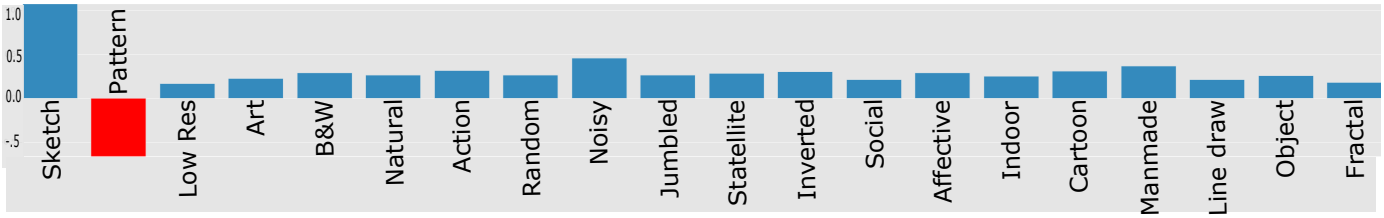


Figure 2.6: PSNR-HVS of our model - JPEG across various categories of MIT Saliency dataset (higher is better).

2.6 Discussion

While the comparison of metrics in Table 2.1 on standard JPEG and variable quality JPEG using our generated map is similar, it should be noted that these metrics still lack ability to judge human perception. It is evident from the results that the salient objects have significantly higher `PSNR-S`, yet maintains overall `PSNR` and the file size. We believe we have effectively created a model to make JPEG semantically aware without sacrificing overall image quality. We know from the JPEG standard that images compressed with higher Q better, and by design our model compresses the salient objects with higher Q; see Figure 2.1 for qualitative evaluation.

The results in Table 2.1 show the success of our method in maintaining or improving performance on traditional image quality metrics. Further, given the efficacy of our method in identifying multiple regions of interest, the `PSNR-S` measurements demonstrate the power of our method to produce superior visual quality in subjectively important regions.

When we look at Figure 2.1 we see the difference in quality of the salient object. Our model consistently outperforms standard JPEG on `SSIM` and `MS-SSIM` metric. These metrics measure the inherent structures in the image and Richter et al [RK09] [Ric11] has shown efficacy of these metric for image compression.

Figure 2.6 shows the performance of our model across all categories of the MIT dataset. Performance was strongest in categories like 'Outdoor Natural', 'Outdoor Man Made', 'Action' and 'Object', while categories like 'Line Drawing', 'Fractal' and 'Low Resolution' showed the least improvement. Not surprisingly, the category 'Pattern', which lacks semantic objects, is the only category where our model did not improve upon standard JPEG. Figure 2.5 shows results on the same image scaled to different sizes. Because our model benefits from

the scale-invariance of CNNs, we are able to preserve performance across a wide range of input sizes. Performance of our model on `PSNR-HVS` is consistent across all tested sizes. CNNs are able to learn scale-invariant features, and therefore the same object at any size should always be classified similarly. Our model preserves this feature even when looking for multiple objects.

2.7 Conclusion

We have presented a model which can learn to detect multiple objects at any scale and generate a map of multiple semantically salient image regions. This provides sufficient information to perform variable-quality image compression, without providing a precise semantic segmentation. Unlike region-based models, our model does not have to iterate over many windows. We sacrifice exact localization for the ability to detect multiple salient objects. Using the output of our model as a map to perform variable-quality compression preserves the overall quality and size of the image, yet improves the visual quality of salient objects. Our variable compression improves upon visual quality without sacrificing compression ratio. Encoding requires a single inference over the pre-trained model, the cost of which is reasonable when performed using a GPU, along with a standard JPEG encoder. The cost of decoding, which employs a standard, off-the-shelf JPEG decoder remains unchanged. We believe it will be possible to incorporate our approach into other lossy compression methods such as JPEG 2000 and vector quantization, a subject of future work. Improvements to the power of our underlying CNN, addressing evolving visual quality metrics, and other applications such as video compression, are also potential areas of future work.

Chapter 3

Robust regions of interest

3.1 Introduction

Deep neural networks (DNNs) have shown tremendous success in image recognition tasks, even surpassing human capability [He+16b]. DNNs have become components of many critical systems, such as self-driving cars [Boj+16], medical image segmentation, surveillance, and malware classification [Pas+15]. However, recent research has shown that DNNs are vulnerable to adversarial attacks, in which minute, carefully-chosen image perturbations can result in misclassification of the image by the neural network [GSS14]. In most cases, this change is imperceptible to humans (the resulting image is visually indistinguishable from the original).

Recently, research has shown that these adversarial attacks are very hard to detect and difficult to protect against. Defense is particularly challenging because the adversarial examples are transferable from one model to another, i.e an image tailored to fool one deep learning model is also likely able to fool other deep learning models. Current adversarial attacks take advantage of the *over-completeness* of the pixel representation of images – that

is, storing images as an array of values results in storing much more information than is required to recognize the content of an image. Traditional image compression techniques, like JPEG, rely on assumptions about the human perception of natural images in order to compress them. Since adversarial attacks are imperceptible to the human eye, they can be viewed as hiding themselves in the over-complete part of the pixel representation of images. Removing these adversarial perturbations is therefore a lossy image compression problem: by effectively encoding natural images, image compression techniques can quantize away imperceptible elements of an image, effectively blocking attacks that rely on adversarial perturbations.

While the research in defending against such adversarial perturbations is still in its infancy, it has been shown that JPEG naturally removes some of these imperceptible perturbations, thereby restoring the original classification of the image [Das+17; DGR16]. More advanced attacks, however, are robust against JPEG compression. In this work, we study the nature of the perturbations generated by adversarial methods, as well as their effect on classification. We present a solution which produces a standard JPEG-decodable image, while allowing significantly improved classification recovery as compared to off-the-shelf JPEG compression. Most of these attacks attempt to change the output of the model directly, using the gradient of the input image with respect to the the model output to iteratively discover a smallest perturbation to the input image that will change the model's classification.

Several techniques have been proposed which attempt to reduce the feasibility of generating adversarial images [Pap+16a; Miy+15]. Most of these methods involve augmenting the DNN training process to create a more robust classifier. However, these defenses often fail against more advanced attacks, and come at the cost of more computationally expensive training. A detailed discussion of proposed defenses is found in section 3.2.8. If a new

attack vector is proposed which falls outside of what defense was trained for, the model has to be trained with the new set of perturbations to make it robust against this new attack. This is an expensive and non-scalable solution. In many cases, it is prohibitively expensive to retrain a classifier to defend against a new attack.

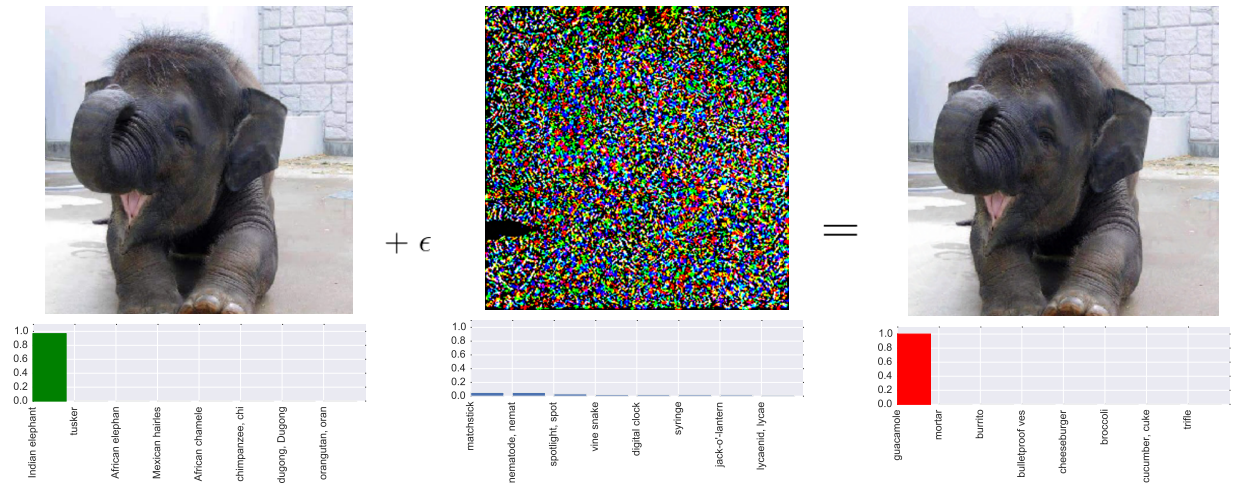


Figure 3.1: Left: Original Image, Center: Perturbed Noise, Right: Adversarial Image

We therefore seek a defense which can be applied to images as a preprocessing step, before they are passed to the classifier. We present an improved JPEG compressor which defends against such attacks without requiring changes to the classification model or its training process. Our algorithm dampens the noise added by the adversary so as to allow the classifier to recover the correct classification. While this kind of defense may not be the most robust solution, it comes at a low computational cost, and the ubiquity of the JPEG decoder makes it easily implementable in practice. It has a low computational cost, and decoding can still be done by any standard, off-the-shelf decoder. The ubiquity of the JPEG decoder makes it easily implementable in practice.

Standard JPEG compression is able to provide some measure of defense against adversarial attacks. Since adversarial perturbations are often high frequency and low magnitude,

the quantization step of JPEG frequently removes them. When the noise added by the adversary is removed or substantially modified, and the resulting image no longer fools the classifier. However, as the severity of quantization increases, important features of the original image are also lost, and the model's ability to correctly classify the resulting image deteriorates regardless of any adversarial perturbation. This trade-off limits the effectiveness of a simple JPEG-based defense, as defending against more sophisticated attacks requires harsh quantization that renders the output image difficult to classify.

Our algorithm employs an adaptive quantization scheme which intelligently quantizes different blocks at different levels. Our analysis of adversarial images generated by various attacks shows that attack methods will tend to perturb all regions of an image with equal probability. We have found that the regions chosen for perturations by attack methods do not strongly correlate with the presence of a semantic object. As a result, there often exists added noise outside the object of interest in the image. We target this portion of the noise for aggressive quantization. We tested the MSROI algorithm of [Pra+17], which identifies multiple salient regions of the image, and quantizes those regions which are not salient more aggressively. MSROI adaptive quantization dampens a significant portion of the adversarial noise, but does so without deteriorating the quality of salient regions, thereby disrupting the attack while allowing for recovery of the original classification. However, initial experiments demonstrated that the unmodified MSROI algorithm is also susceptible to adversarial noise, weakening the accuracy of the generated saliency maps. Therefore, we present a more robust method of generating a saliency map, which is able to recover object saliency even in the presence of adversarial noise, without significantly impacting performance on unperturbed images. We show that our proposed defense is successful against several state-of-the-art attack methods using an off-the-shelf classification model. We further demonstrate that our proposed technique greatly improves upon the defense provided by a standard JPEG

encoder.

3.2 Background

A deep neural network can be viewed as a parameterized function, F_θ, which maps $x \in \mathbb{R}^n$ to $y \in \mathbb{R}^m$ i.e. $F_\theta(x) = y$ where θ represents the model parameters. Feed-forward neural networks are composed of multiple layers of successive operations, each applied to the output of the previous layer. For a deep neural network containing l layers, the final network is a function composition of l simpler functions i.e $F = F_1 \circ F_2 \circ \cdots F_l$. These networks are trained by minimizing some loss function, $\mathcal{L}$, which is a differentiable measure of the classification accuracy on the training data. The parameters of the model, θ, are learned by stochastic gradient descent (SGD). Let $\widetilde{y}$ denote $F(x|\theta)$.

$$\min \mathcal{L}(y, \widetilde{y}) = -\frac{1}{N} \sum_{i=1}^{N} y^i \times \log$$

$$\theta^l := \theta^l - \eta \nabla \mathcal{L}(y, \widetilde{y})$$

Most adversarial attack methods use the gradient of the image pixels with respect to the the classification error to generate the desired perturbation. In order to compute this gradient, an attacker needs to have access to the internal model and parameters. These kinds of attacks are called white box attacks, as the adversary needs access to the targeted model. White box attacks can fool many state-of-the-art convolutional neural networks. Many popular deep learning models are publicly available, making them vulnerable in practice to these attacks. This increases their ability to be attacked using such techniques. These attacks are discussed in detail in section 3.2.1.

It is also possible to attack deep networks without access to the model parameters. Such

attacks are called black box attacks, and they begin by training a secondary model to match the outputs of the target model. This secondary model can then be attacked using white box methods, and the resulting adversarial images are transferable to the target network [Pap+16b]. Black box attacks are weaker than white box attacks; here we focus on white box attacks.

3.2.1 Adversarial Attacks

Formally, the paradigm of adversarial attacks can be described as follows. Let x be an input image to a classifier. Then, the class label (the model's evaluation of what object the image contains) for the image x is: $c = \text{argmax}_i F(x|\theta)$ where i is an index into the output vector.

An adversarial attacker takes the image x and adds a small perturbation δ_x, to obtain an adversarial image $x' = x + \delta_x$. The attack is considered to be successful if, for a small δ_x, the class label of the adversarial image (c'), is not the same as that of original image (c). There is no consensus on the exact definition of "small" for δ_x, but attacks are generally considered successful when the difference between x and x' is indistinguishable to the human eye. Some attacks are noticeable to humans, but do not change our perception of the class label of the image, and these attacks are also considered successful. Attacks can be targeted to induce a specific chosen class label, $\hat{c}$ such that $c' = \hat{c}$, or untargeted, meaning that they seek to induce any class label, as long as it does not match the original classification, i.e., $c' \neq c$.

In most cases, untargeted attacks are performed by iterating a targeted attack over all classes and choosing the class which requires the smallest perturbation. Targeted attacks tend to require a larger, more obvious perturbation, thereby making them easier to detect and defend against. As our work focuses on defense, we will consider untargeted attacks, and therefore our methods should also be effective against targeted attacks.

Adversarial attacks work by exploiting the design of the model and the limitations of SGD. Just as gradients of the model parameters (θ) with respect to the loss function are calculated during SGD training, an attacker can compute gradients of the input pixels with respect to the classification output. These gradients instruct the attacker on how to modify the image to change the model's classification. In the following sections, we will briefly describe each of the adversarial attacks used in our experiments.

3.2.2 Fast Gradient Sign Method (FGSM)

[GSS14] uses the sign of the gradient of the loss function to determine the direction of the perturbation δ_x.

$$x' = x + \epsilon \times \text{sign}(\nabla \mathcal{L}(F(x|\theta), \widetilde{y}))$$

Where ϵ is a small value which controls the magnitude of perturbation, chosen through trial and error. To ensure a sufficiently small δ_x, FGSM optimizes the L_∞ distance between x and x', and thus tries to minimize the maximum change to any of the pixels. It has been shown that with small values of ϵ, most deep neural networks trained on CIFAR-10 and MNIST can easily be fooled [GSS14]. FGSM is a particularly efficient attack method, as it requires only a single computation of the gradients of the input pixels. However, FGSM does not always produce robust results.

3.2.3 Iterative Gradient Sign Method

[KGB16] builds upon FGSM. Starting with the original image, FGSM is applied iteratively until a satisfactory image is found, and, at each step, the generated adversarial image is

clipped so that the generated image is within an L_∞ ϵ-neighborhood of the original image.

$$x'_0 = x, \qquad x'_{N+1} = \text{Clip}_{x,\epsilon}\Big\{x'_N + \alpha \times \text{sign}(\nabla\mathcal{L}(F(x|\theta), \widetilde{y}))\Big\}$$

Here, α is the amount of perturbation added per iteration, which is usually very small compared to the ϵ used in FGSM. This allows the attacker to find adversarial images which are closer to the original image than those found by FGSM [KGB16].

While IGSM generates better adversarial images than FGSM, it is a significantly more expensive process, because it requires multiple computations of the gradient. The Gradient Attack (GA) is a middle ground in which steps are taken according to magnitude of the normalized gradient, rather than just the sign.

3.2.4 Gradient Attack (GA)

Gradient Attack is where a single step is taken, as in FGSM, but the step is taken in the exact direction of the gradient, rather than the direction of the element-wise sign of the gradient. The gradient step is normalized to have unit length.

$$\mathcal{G} = \nabla\mathcal{L}(F(x|\theta), \widetilde{y}), \qquad x' = x + \epsilon \times \frac{\mathcal{G}}{||\mathcal{G}||_2} \qquad \text{where, } ||.||_2 \text{ denotes } L_2 \text{ norm}$$

3.2.5 Deep Fool

Deep Fool [MDFF16] makes the simplifying assumption that the classifier to be targeted is a linear model which uses a hyperplane to separate the classes. It tries to find an adversarial image by moving in the direction orthogonal to the linear decision boundary. Because the classifier is not actually linear, it is necessary to iterate this process until an adversarial image is found. Deep Fool is an untargeted attack and optimizes for L_2 (PSNR) instead of

L_∞, unlike FGSM. Experiments reveal that perturbations generated by Deep Fool are less perceptible than those generated by other methods.

3.2.6 Jacobian-based Saliency Map Attack

Jacobian-based Saliency Map Attack (JSMA) [CW17b] uses an adversarial saliency map, which indicates the pixels in the image that the adversary should increase in value in order to fool the classifier. Let $\mathcal{J}(x)$, where $\mathcal{J}_i(x) = \frac{\partial F_i(x)}{\partial x}$, be the Jacobian of the output of the classifier with respect to the image x and c' be the targeted class. Then, the adversarial saliency map, $S(x_j, c')$, is given by:

$$S(x_j, c') = \begin{cases} 0, & \text{if } \mathcal{J}'_c(x_j) < 0 \text{ or } \sum_{i \neq c'} \mathcal{J}_i(x_j) > 0 \\ \mathcal{J}_{c'}(x_j) | \sum_{i \neq c'} \mathcal{J}_i(x_j)|, & \text{otherwise} \end{cases}$$

The attacker tries to find the pixel x_j such that $S(x_j, c')$ is maximized. Then, that pixel is perturbed by ϵ. This process is iterated until the model's prediction for x is c'. JSMA produces alterations that are imperceptible to humans, but it is considerably slower than other techniques.

3.2.7 L-BFGS

L-BFGS [Sze+13a] is one of the earliest proposed techniques to generate adversarial images. The authors formulate the task of finding an adversarial image x' given an image x as a box-constrained optimization problem. The constraint is the class label and the function being minimized is the L_2 distance between the image x and the generated adversary x'. In practice, it is hard to find a satisfactory minimum under such strict constraints. Instead,

they reduce the minimization to:

$$\text{minimize} \quad \alpha||x - x'||_2^2 + \mathcal{L}((F(x'|\theta), c'), \quad \text{subject to } x' \in [0,1]^n$$

where α is an arbitrary value chosen by line search. This optimization is often carried out using L-BFGS, which is slower than other adversarial techniques.

3.2.8 Defenses

Defenses against adversarial systems are either specific to a given attack or attack-agnostic. We briefly discuss some well-known defenses from both categories.

3.2.9 Adversarial Training

One of the techniques to defend against adversarial images is to include such images as part of the training set, thereby encouraging the model to classify the perturbed images with same label as the original image. This method is very inefficient as the space of possible perturbations is very large and it is not practical to train a model long enough to become robust against all possible noise [Tra+17b]. Another limitation of this method is that adversarial images trained to fool other networks are not defended against. It has been shown that adversarial images are transferable between models [Liu+16], and such transferred images are generally difficult to defend against. Adversarial systems which include random perturbations to the image before applying an attack technique are generally robust against this kind of defense.

3.2.10 Defensive Distillation

One of the most effective techniques towards defending against adversarial perturbations is called Defensive Distillation. In this scheme, the outputs of the classifier are used to train another model, called the distilled model. The distilled model is trained to match the output of the last layer (softmax layer) of the original classifier. This distilled model benefits from a smoother distribution of target outputs as compared to the all-or-nothing ground truth labels. This model has been shown to be robust against some of the less efficient attacks. However, this comes at the expense of training a new model. Recently, quantization and other image processing techniques have been proposed as a first line of defense to counteract adversarial perturbations [Lia+17; Pra+18a; Guo+17a]. These defenses significantly alter the image and render them unfit for general purpose use. These techniques also reduce the classification accuracy of clean image, which is an undesirable effect of image transformations.

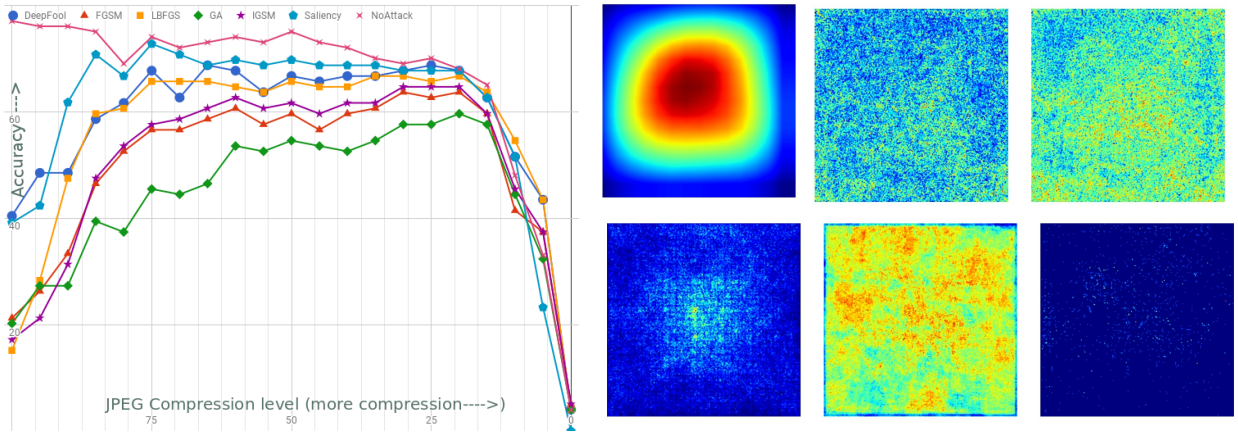


Figure 3.2: (a) Classification accuracy on adversarial images is recovered as quantization increases, but drops when quantization becomes too aggressive. *JPEG Compression level* refers to the scaling factor of the quantization matrix often denoted as Q. (b) Clockwise from top left; average MSROI saliency map for original images, average adversarial perturbations for five attack methods (IGSM, FGSM, JSMA, Deep Fool, GA). Note that noise intensity is not concentrated in salient regions.

Yet another defense against adversarial images is to either detect the presence of adversarial perturbations in the given image

3.2.11 Transformations

Recently, it has been shown that image transformations such as Gaussian blur, Gaussian noise, change of contrast and brightness, and image compression like JPEG are able to recover some portion of correct classifications when applied to adversarial images [KGB16]. The impact of JPEG compression was further studied by [DGR16] and [Das+17]. Their studies were limited to less effective attack techniques, such as FGSM, but demonstrated the baseline efficacy of JPEG. We show that standard JPEG does not defend against newer and more robust attacks, and analyze some of the limitations of pure JPEG for adversarial defense.

JPEG's success in defending against some attack models is due to the quantization of high-frequency signals in the image. The high frequencies will contain some portion of the perturbations added by the attacker, and quantization will lessen their impact. However, adversarial perturbations in lower frequencies remain, and the more aggressive quantization which is required to remove these perturbations results in lower image quality, making it difficult to recover the original classification.

Unlike adversary, a defensive model is not constrained by a L_p distance, regardless of that, losing integrity of the image would mean reduced classification accuracy. To maximize the effectiveness of our defense, while still preserving sufficient quality to allow for correct classification, we seek to strongly quantize those regions of the image which are not salient to the true class, while weakly quantizing salient regions. We present a method for achieving this in the following section.

3.3 Semantic Quantization

Our approach employs a transformation $T(x)$ of a given image x which quantizes the image in the frequency domain like standard JPEG. However, unlike JPEG, we determine the salient regions of x , i.e. regions for which the presence of an object is likely, and quantize those regions at higher bit-rates than non-salient regions. We use a convolutional neural network to generate a saliency heat map, which guides quantization.

The center plot on Figure 1 shows the map of the perturbations added by an adversarial system (IGSM). The adversary has made modifications to every image region (except where the image is saturated), even though the elephant is in the center of the image. Our method takes advantage of this, and strongly quantizes the image blocks outside of the salient region.

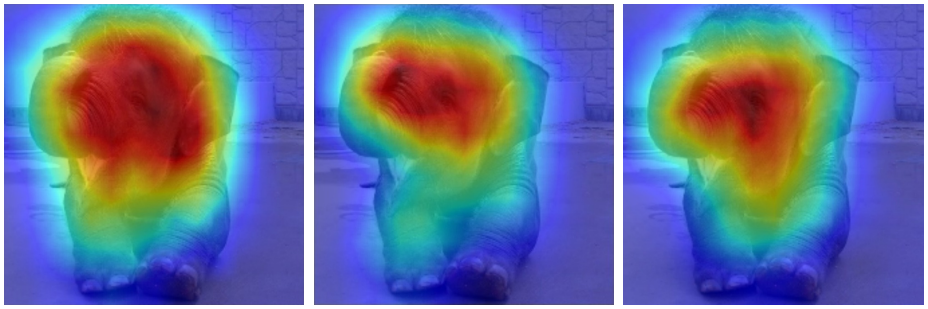


Figure 3.3: Left: MSROI on original image, Center: MSROI, Right: Aug-MSROI, on adversarial image

Convolutional neural networks (CNNs) have been used to detect and locate objects in a given image [Erh+14; Zha+15]. However, these works seek to generate a hard bounding box around the objects which does not take into consideration partial or occluded objects. They are also limited in the number of classes they can detect, which ranges between 20 to 40. Saliency detection techniques can solve these issues, but such techniques are limited in their ability to detect multiple objects, and the identified salient region may only contain a

limited subset of the objects in the image. To address these shortcomings, we utilize MSROI [Pra+17], a CNN designed to retrieve all salient regions and provide a soft boundary over the image.

MSROI generates a heat map based on its confidence over all classes and their corresponding activation values. At each layer, l, of the network, MSROI has independent filters for each class in the dataset. Thus, summing the filter responses of each class over their respective filters represents the confidence of that class. For an image x at a given location of (i, j) and C classes, MSROI map is defined as:

$$M(i,j) = \sum_{c \in C} \begin{cases} \sum_l f_l^c(i,j) & \text{if} \sum_l \sum_{x,y} f_d^c(i,j) > \mathcal{T} \\ 0 & \text{otherwise} \end{cases}$$

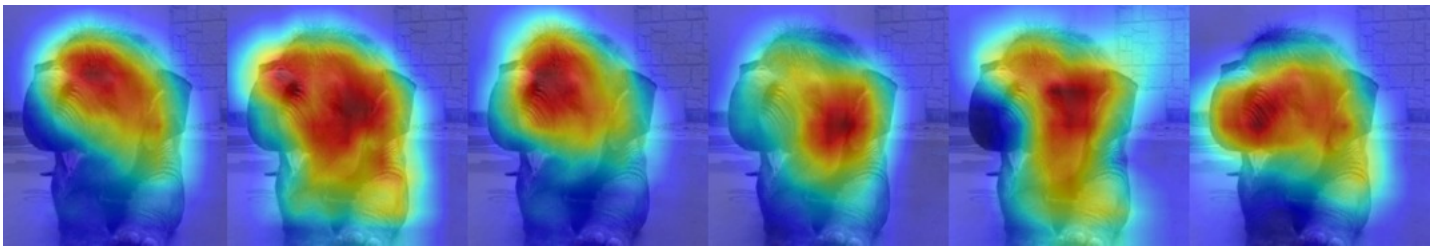


Figure 3.4: Combination of various jittered inputs

When the confidence of the model on a specific class is below a threshold $\mathcal{T}$ (determined through experimentation on validation data), the class is not represented.

3.4 Augmented MSROI

MSROI was trained on natural images, but in this task it will be applied to an adversarial image. This means that the incorrect adversarial classification will be represented in the

saliency map. This presents an issue, as the adaptive quantization will try to preserve those areas which are salient to the incorrect classification. Thus, performing standard MSROI retains some adversarial noise which could have been quantized away. To counteract this, we augment the MSROI algorithm by generating several small random perturbations of the image and the intermediate activations of the MSROI network, and calculating a saliency map $\widetilde{M}$ for each such perturbation:

$$\widetilde{M}(i,j) = \sum_{c \in \hat{C}} \sum_{l} (a_l^c(i,j)) \quad \text{and} \quad a_{l+1}^c = f_l(a_l^c(i,j) + \Delta)$$

where Δ is a random perturbation and $\hat{C}$ is subset of C found similarly using $\mathcal{T}$ as shown with MSROI. We take an average of $\widetilde{M}$ maps over several perturbations to generate a final map, which we term Augmented MSROI (Aug-MSROI). This process of averaging over many perturbations gives softer localization which is less vulnerable to attack. The (i, j) signifying the most salient objects gets adds over many times, thereby making the localization of object less precise but more robust. The number of perturbations to average over can be found empirically. Figure 4 shows several individual $\widetilde{M}$ maps which are combined to create the final heat map. In Figure 3 we show the difference between standard MSROI and Aug-MSROI. Most of the head of the baby elephant is retrieved in the Aug-MSROI heat map but not in the standard MSROI. Computation of MSROI relies heavily on the most prominent class even if it was designed to find all salient objects. Since an adversarial image's top-1 prediction has changed, it changes the saliency map built by MSROI.

Once the semantic heat map is obtained, we use the technique proposed in [Pra+17], which quantizes different 8×8 blocks at different levels and combines them to obtain a final JPEG image. This final image retains high quality in salient regions, and is heavily quantized in non-salient regions. Strong quantization in non-salient regions is often sufficient

to reverse the impact of adversarial perturbations, and high quality in salient regions allows recovery of the correct classification. The map has to be discretized so that different levels of quantization can be applied. We bin the heat map into K different levels. We then pick K different tables for quantization of these blocks. For simplicity we use different scaling factor on the same table, values of which were obtained from JPEG standard [Wal92], but these tables could be different from each other, as different quantization tables exist for JPEG blocks [Wan01]. This was done to keep the comparison against JPEG as fair as possible. After the quantization, inverse DCT is performed for each block.

Our modifications of the standard MSROI algorithm result in only a modest increase in encoding complexity, and, like standard MSROI, the resulting compressed image can be decompressed by a standard off-the-shelf JPEG decoder. Even though JSMA and L-BFGS attack selected pixels, the pixels changed by these attacks do not significantly correlate with the pixels selected by Aug-MSROI. Aug-MSROI extracts pixels belonging to the object while JSMA extracts pixels most likely to cause the change in image classification, which often lie in the negative space.

3.4.1 Quantization

Once the semantic heat map is obtained we bin the heat map into K slots. Every 8×8 block of the original image is then compressed using JPEG encoder at different quantization levels. Then the frequency components are quantized based on the values on the saliency map.

3.5 Experiments

We experimented with several adversarial attacks as described in Section 3.2.1. We used Residual Network (ResNet-50) [He+16b], a state-of-the-art deep CNN as the classification model. Our test images are scaled and cropped to 256×256, and are drawn from the ImageNet dataset, which contains 1000 classes. ResNet-50 is a fifty layer deep neural network which consists of convolutional layers with skip connections [He+16b]. This has been shown to improve classification accuracy considerably [He+16c] VGG-19 is comparatively less accurate but is known to generalized well in image transfer applications. [SZ14]. All the attacks were parametrized to have RMSE distance of 0.02 compared to the original image. Aug-MSROI, like MSROI, requires a one-time off-line training of the model. Encoding of images can be done in parallel for several images on a GPU. Computation of Aug-MSROI map requires multiple perturbations for the same image and thus is somewhat more resource intensive than computing MSROI map. However, these passes can be computed in parallel as the final map is an average of individual maps. On a Titan X Maxwell GPU with 12 GB of memory encoding 300 images, each with five perturbations, took 5 seconds total; the decoding process remains unchanged from standard JPEG. We compare our results against the original MSROI technique, as well as off-the-shelf JPEG.

Metrics for comparison Accuracy is the most common metric for assessing performance of deep neural networks on image classification tasks. Most publications report 'Top-5' accuracy, which is calculated by counting the number of instances for which the image's true label is in the top five predicted labels. For the purposes of adversarial attacks, we consider an attack to be successful if the model's top predicted class c' for adversarial image x' is not the same as the top predicted class for the original image x i.e 'Top-1' accuracy. This is a

very generous success metric for the adversary, as the predicted class label for most of the adversarial images are within the 'Top-5' of the original image even if not in 'Top-1'.

We achieved 76% accuracy with Residual Network (ResNet-50), which is in accordance with the reported numbers [He+16b]. In other words, after applying our defense, we recover a level of classification accuracy which is close to the accuracy obtained on non-adversarial images. Since the top-1 accuracy of classifier on the original image is limited the metric is biased towards adversary because in many cases adversary does not even have to work to produce a success.

Therefore, to measure the effectiveness of a transformation, T, we also report destruction rate as proposed in [KGB16]. Destruction rate is fraction of adversarial images which are no longer misclassified after the transformation.

Let C be a function defined as -

$$C(x, y) = \begin{cases} 1, & \text{if image } x \text{ is classified as } y \\ 0, & \text{otherwise} \end{cases}$$

Then, destruction rate is defined as –

$$d = \frac{\sum_{i=1}^{N} C(x_i, y_i) \times (1 - C(x'_i, y)) \times C(T(x'_i), y_i)}{\sum_{i=1}^{N} C(x_i, y_i) \times (1 - C(x'_i, y))}$$

	Accuracy↑	MSE↓	SSIM↑	PSNR↑
JPEG	52	16.55	0.62	24.33
MSROI	65	16.13	0.64	24.61
Aug-MSROI	82	10.92	0.86	27.36

Table 3.1: Results averaged across 300 images.

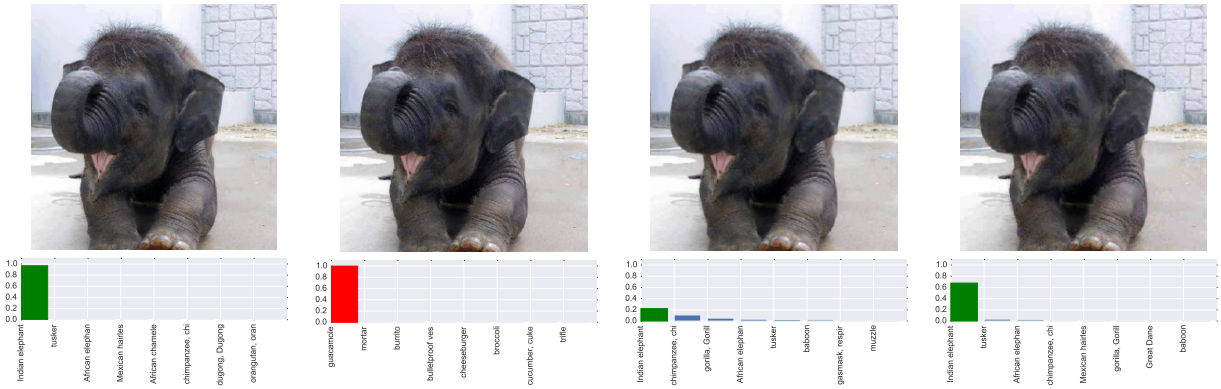


Figure 3.5: Original Image, Adversarial Image, JPEG, Aug-MSROI

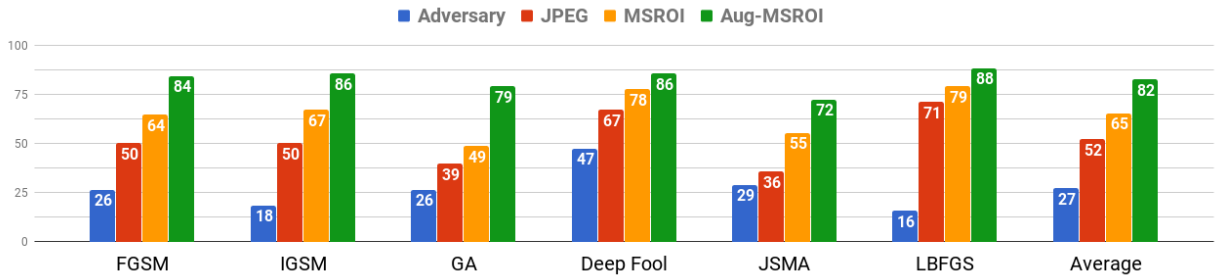


Figure 3.6: Classification accuracy of images on various attacks

3.6 Results

Figure 3.5 shows a typical example of the results for qualitative comparison. In this case, while JPEG was also able to retrieve the original class, it does so with significantly lower confidence. Table 1 shows the quantitative results averaged across 300 images of ImageNet.

The JPEG quantization level was chosen to achieve the highest accuracy. For MSROI, we applied the technique described in [Pra+17] and set the quantization level to keep the final image quality as close as possible to the JPEG version. The results are shown in Figure 2(a). Aug-MSROI improved classification accuracy while maintaining the perceptual quality as evaluated by PSNR, MSE and SSIM [Wan+04b].

3.7 Conclusion

We explored various vulnerabilities of neural networks against adversarial attacks. We investigated efficacy and limitations of standard JPEG and semantic JPEG against these minute perturbations. We have presented Aug-MSROI, an augmentation of MSROI [Pra+17] to employ semantic JPEG compression as an effective defense against state-of-the-art adversarial attacks. Compared to other defense techniques which require learning a new model, our method can be carried out as a image processing step. Images generated by our technique can easily be decoded by any standard JPEG decoder. In our experiments, JPEG compression employing Aug-MSROI produces compressed images, which, when decompressed, have high visual quality while at the same time preserving the accuracy of classification by a neural network. A key advantage of our approach is that, with only a modest increase in encoding time, like standard MSROI, it produces compressed images that can be decompressed by any off-the-shelf JPEG decoder.

Chapter 4

Pixel Deflection

4.1 Introduction

Image classification convolutional neural networks (CNNs) have become a part of many critical real-world systems. For example, CNNs can be used by banks to read the dollar amount of a check [BBL97], or by self-driving cars to identify stop signs [Pap+16b].

The critical nature of these systems makes them targets for adversarial attacks. Recent work has shown that classifiers can be tricked by small, carefully-crafted, imperceptible perturbations to a natural image. These perturbations can cause a CNN to misclassify an image into a different class (*e.g.* a "1" into a "9" or a stop sign into a yield sign).

Thus, defending against these vulnerabilities will be critical to the further adoption of advanced computer vision systems. Here, we consider *white-box* attacks, in which an adversary can see the weights of the classification model. Most of these attacks work by taking advantage of the differentiable nature of the classification model, *i.e.* taking the gradient of the output class probabilities with respect to a particular pixel. Several previous works propose defense mechanisms that are differentiable transformations applied to an image be-

fore classification. These differentiable defenses appear to work well at first, but attackers can easily circumvent these defenses by "differentiating through them", *i.e.* by taking the gradient of a class probability with respect to an input pixel through both the CNN and the transformation.

In this work, we present a defense method which combines two novel techniques for defending against adversarial attacks, which together modify input images in such a way that is (1) non-differentiable, and (2) frequently restores the original classification. The first component, *pixel deflection*, takes advantage of a CNN's resistance to the noise that occurs in natural images by randomly replacing some pixels with randomly selected pixels from a small neighborhood. We show how to weight the initial random pixel selection using a *robust activation map*. The second approach, *adaptive soft-thresholding* in the wavelet domain, which has been shown to effectively capture the distribution of natural images. This thresholding process smooths adversarially-perturbed images in such a way so as to reduce the effects of the attacks.

Experimentally, we show that the combination of these approaches can effectively defend against state-of-the-art attacks [Sze+13a; GSS14; CW17b; MDFF16; Pap+16c; KGB16] Additionally, we show that these transformations do not significantly decrease the classifier's accuracy on non-adversarial images.

In Section 4.2, we discuss the various attack techniques against which we will test our defense. In Sections 4.3 and 4.4 we discuss the established defense techniques against which we will compare our technique. In Sections 4.5, 4.6 and 4.7 we lay out the components of our defense and provide the intuition behind them. In Sections 4.9 and 4.10, we provide experimental results on a subset of ImageNet.

We propose a two-step process to defend against adversarial attacks on existing networks.

1. Apply targeted *pixel deflection* guided by robust activation maps, as described in Sections 4.5 and 4.6.

2. Apply adaptive soft thresholding to the image in a wavelet domain, as described in Section 4.7.

In a white-box attack model, the adversary is able to find the gradient of the image pixels with respect to the classification error. In this paradigm, defenses which employ a differentiable transformation are of limited effectiveness, as the adversary can simply incorporate that transformation into their training. Our method avoids this potential weakness by applying a non-differentiable transformation which stochastically alters pixel values. These factors reduce the adversary to working in a gray-box environment, in which gradients can only be computed for part of the classification process, and in which the adversary cannot be certain which perturbations will be preserved.

Previously proposed defenses which rely on a strong quantization [DGR16; Das+17] are often successful at avoiding the adversarial class, but this success comes at the cost of decreased performance on clean images. By contrast, our method uses an adaptive quantization scheme which preserves the classification of clean images at a much higher rate. Defenses which are based around a deterministic, differentiable transformation are of limited effectiveness, as the adversary can simply incorporate that transformation into their training. It is difficult to incorporate our technique as a an augmentation process for adversarial loss, because it decreases the space and availability of pixels for the attack.

White-box attacks are based on the ability to find the gradient of image with respect to the classification error. Our defense technique stochastically changes the pixel values, in a way which is non-differentiable, thereby reducing a white-box attack to a grey-box attack.

Most denoising based defense methods [DGR16; Das+17] come at cost of reduced accu-

racy of the classifier on the clean images. This is a limitation of hard and fixed quantization. Our method uses adaptive and soft quantization and therefore, accuracy of clean images remains unchanged.

4.2 Adversarial Attacks

It has been established that most image classification models can easily be fooled [Sze+13a; GSS14]. Several techniques have been proposed which can generate an image that is perceptually indistinguishable from another image but is classified differently. This can be done robustly when model parameters are known, a paradigm called *white-box attacks* [GSS14; KGB16; Mad+17; CW17b]. In the scenario where access to the model is not available, called *black-box attacks*, a secondary model can be trained using the model to be attacked as a guide. It has been shown that the adversarial examples generated using these substitute models are transferable to the original classifiers [Pap+16b; Liu+16].

Consider a given image x and a classifier $F_\theta(\cdot)$ with parameters θ. Then an adversarial example for $F_\theta(\cdot)$ is an image $\hat{x}$ which is close to x (*i.e.* $||x - \hat{x}||$ is small, where the norm used differs between attacks), but the classifier's prediction for each of them is different, *i.e.* $F(x) \neq F(\hat{x})$. *Untargeted attacks* are methods to produce such an image, given x and $F_\theta(\cdot)$. As long as the model mis-classifies $\hat{x}$, the attack is considered a success. Such attacks are called , in which the value of $F(\hat{x})$ is unconstrained beyond the requirement that it differ from $F(x)$. A stronger form of an attack is a targeted attack, in which the attacker is given x, $F(\cdot)$, and a specific target class $\hat{y}$, and produces $\hat{x}$ such that $F(\hat{x}) = \hat{y} \neq F(x)$. *Targeted attacks*, however, seek a $\hat{x}$ such that $F(\hat{x}) = \hat{y}$ for some specific choice of $\hat{y} \neq F(x)$, i.e. targeted attacks try to induce a specific class label, whereas untargeted attacks simply try to destroy the original class label.

Next, we present a brief overview of several well-known attacks, which form the basis for our experiments.

4.2.1 Fast Gradient Sign Method (FGSM)

[GSS14] is a single step attack process. It uses the sign of the gradient of the loss function, ℓ, with respect to to the image to find the adversarial perturbation. For a given value ϵ, FGSM is defined as:

$$\hat{x} = x + \epsilon \text{sign}(\nabla \ell(F(x), x)) \tag{4.1}$$

4.2.2 Iterative Gradient Sign Method (IGSM)

[KGB16] is an iterative version of FGSM. After each iteration the generated image is clipped to be within a ϵL_∞ neighborhood of the original and this process stops when an adversarial image has been discovered. Both FGSM and IGSM minimize the L_∞ norm with respect to to the original image. Let $x'_0 = x$, then after m iterations, the adversarial image is obtained by:

$$x'_{m+1} = \text{Clip}_{x,\epsilon}\Big\{x'_m + \alpha \times \text{sign}(\nabla \ell(F(x'_m), x'_m))\Big\} \tag{4.2}$$

4.2.3 L-BFGS

[Sze+13a] tries to find the adversarial input as a box-constraint minimization problem. L-BFGS optimization is used to minimize L_2 distance between the image and the adversarial example while keeping a constraint on the class label for the generated image. This is a slow and iterative targeted attack which generally produces images with small perturbations and strong classifier confidence in the misclassification.

4.2.4 Jacobian-based Saliency Map Attack (JSMA)

[Pap+16c] estimates the saliency of each image pixel with respect to to the classification output, and modifies those pixels which are most salient. This is a targeted attack, and saliency is designed to find the pixel which increases the classifier's output for the target class while tending to decrease the output for other classes.

4.2.5 Deep Fool (DFool)

[MDFF16] is an untargeted iterative attack. This method approximates the classifier as a linear decision boundary and then finds the smallest perturbation needed to cross that boundary. This attack minimizes L_2 norm with respect to to the original image.

4.2.6 Carlini & Wagner (C&W)

[CW17b] is a recently proposed adversarial attack, and one of the strongest. C&W updates the loss function, such that it jointly minimizes L_p and a custom differentiable loss function that uses the unnormalized outputs of the classifier (*logits*). Let Z_k denote the logits of a model for a given class k, and κ a margin parameter. Then C&W tries to minimize:

$$||x - \hat{x}||_p + c * max\left(Z(\hat{x}_y) - max\{Z(\hat{x})_k : k \neq y\}, -\kappa\right) \tag{4.3}$$

For our experiments, we use L_2 for the first term, as this makes the entire loss function differentiable and therefore easier to train. Limited success has been observed with L_0 and L_∞ for images beyond CIFAR and MNIST.

We have not included recently proposed attacks like 'Projected Gradient Descent' [Mad+17] and 'One Pixel Attack' [SVS17] because although they have been shown to be robust on

datasets of small images like CIFAR10 and MNIST, they do not scale well to large images. Our method is targeted towards large natural images where object localization is meaningful, i.e. that there are many pixels outside the region of the image where the object is located.

4.3 Defenses

When an image is provided, which may or may not have adversarial perturbations, the goal of the defense method is to classify the images with same predictions. If $F(x) = F(\hat{x})$ then defense is considered as successful. Defense strategies are based on either building a robust classifier such that it is harder to find an adversarial example, or they are based on some transformation $\mathcal{T}$ such that $F(\mathcal{T}(\hat{x})) = F(\mathcal{T}(x)) = F(x)$. Given a classification model F and an image $\tilde{x}$, which may either be an original image x, or an adversarial image $\hat{x}$, the goal of a defense method is to either augment either F as F' such that $F'(\tilde{x}) = F(x)$, or transform $\tilde{x}$ by a transformation $\mathcal{T}$ such that $F(\mathcal{T}(\tilde{x})) = F(x)$.

One method for augmenting F is called Ensemble Adversarial training [Tra+17a], which augments the training of deep convolutional networks to include various potential adversarial perturbations. This expands the decision boundaries around training examples to include some nearby adversarial examples, thereby making the task of finding an adversary within a certain ϵ harder than conventional models. Another popular technique uses distillation from a larger network by learning to match the softmax [Pap+16a]. This provides smoother decision boundaries and thus makes is harder to find an adversarial example which is imperceptible. There are methods that proposes to detect the adversarial images as it passes through the classifier model [MC17; ALM17].

Most transformation-based defense strategies suffer from accuracy loss with clean images [DGR16; KGB16], *i.e.* they produce $F(\mathcal{T}(x)) \neq F(x)$. This is an undesirable side effect of

the transformation process, and we propose a transformation which tries to minimize this loss while also recovering the classification of an adversarial image. Detailed discussion on various kinds of transformation based defenses is provided in section 4.4.

4.4 Related Work

Transformation-based defenses are a relatively recent and unexplored development in adversarial defense. The biggest obstacle facing most transformation-based defenses is that the transformation degrades the quality of non-adversarial images, leading to a loss of accuracy. Most transformations results in some undesirable loss of accuracy on clean images. Thus, transformations are not considered to be a strong defense against adversarial systems. This has limited the success of transformations as a practical defense, as even those which are effective at removing adversarial transformations struggle to maintain the model's accuracy on clean images. Our work is most similar to Guo *et al.*'s [Guo+17b] recently proposed transformation of image by quilting and Total Variance Minimization (TVM). Image quilting is the technique of patching regions of image from different source. Image quilting is performed by replacing patches of the input image with similar patches drawn from a bank of images. They collect one million image patches from clean images and use a k-nearest neighbor algorithm to find the best match. Image quilting in itself does not yield satisfactory results, so it is augmented with TVM. In Total Variance Minimization, a substitute image is constructed by optimization such that total variance is minimized. Total variation minimization has been widely used [Get12] as an image denoising technique. In section 4.10 we have compared the results. Our method uses semantic maps to obtain a better pixel to update and our update mechanism does not require any optimization and thus is significantly faster.

Another closely related work is from Luo *et al.* [Luo+15]. They propose a foveation-based mechanism. They obtain ground-truth data about object coordinates, and then crop the image to and scale the image back to the same dimensions. Using ground-truth data about object coordinates, they crop the image around the object, and then scale it back to the original size.

Our model shares the hypothesis that not all regions of the image are equally important to a classifier. However, we take a different approach to hardening an image such that salient parts are more representative of the image. Further, foveation-based methods can be fooled by finding an adversarial perturbation within the object bounding box. Our model does not rely on a ground-truth bounding box, and the stochastic nature of our approach means that it is not restricted to only modifying a particular region of the input.

Yet another similar work is from Xie *et al.* [Xie+18], in which they pad the image and take multiple random crops and evaluate ensemble classification. This method utilizes the randomness property that our model also exploits. However, our model tries to spatially define the probability of a presence of a perturbation and subsequently uses wavelet-based transform to denoise the perturbations.

4.5 Pixel Deflection

Much has been written about the lack of robustness of deep convolutional networks in the presence of adversarial inputs [NYC15; Sze+13b]. However, most deep classifiers are robust to the presence of natural noise, such as sensor noise [Dia+17].

We introduce a form of artificial noise and show that most models are similarly robust to this noise. We randomly sample a pixel from an image, and replace it with another randomly selected pixel from within a small square neighborhood. We also experimented with other

neighborhood types, including sampling from a Gaussian centered on the pixel, but these alternatives were less effective.

We term this process *pixel deflection*, and give a formal definition in Algorithm 3. Let R_p^r be a square neighborhood with apothem r centered at a pixel p. Let $\mathcal{U}(R)$ be the uniform distribution over all pixels within R. Let I_p indicate the value of pixel p in image I. Let K be the number of iterations, H and W be height and width of an image, respectively, and $\mathcal{U}(a, b)$ be the uniform distribution over the range $[a, b]$. We will randomly select a pixel and update its value from a close-by neighbor. Let $R_{x,y}$ be a square neighborhood with apothem r centered at x, y. If $r > \min(H, W)$, then whole image is the neighbor.

Algorithm 3: Pixel deflection transform

Input : Image I, neighborhood size r

Output: Image I' of the same dimensions as I

for $i \leftarrow 0$ **to** K **do**
  Let $p_i \sim \mathcal{U}(I)$
  Let $n_i \sim \mathcal{U}(R_p^r \cap I)$
  $I'[p_i] = I[n_i]$
end

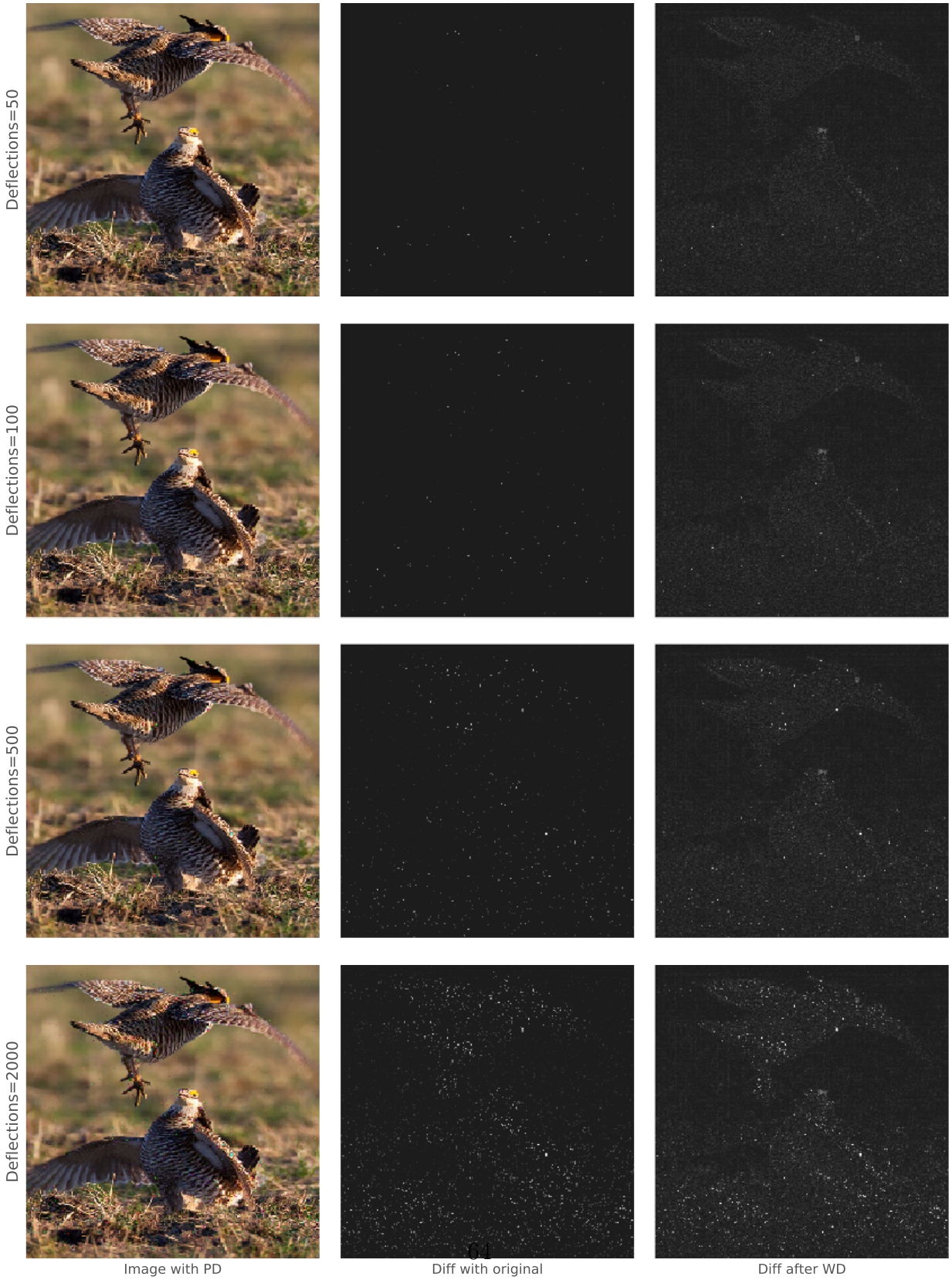


Figure 4.1: Left: Image with given number of pixels deflected. Middle: Difference between clean image and deflected image. Right: Difference after denoising.

Figure 4.2: Average classification probabilities for an adversarial image (top) and clean image (bottom) after pixel deflection (Image size: 299x299)

As shown in Figure 4.2, even changing as much as 1% (*i.e.* 10 times the amount changed in our experiments) of the original pixels does not alter the classification of a clean image. However, application of pixel deflection enables the recovery of a significant portion of correct classifications.

4.5.1 Distribution of Attacks

Most attacks search the entire image plane for adversarial perturbations, without regard for the location of the image content. The attack models have no special preference for changing pixels which belong to the object in the image. This is in contrast with the classification models, which show high activation in regions where an object is present [Yos+15; Cha+17]. This is especially true for attacks which aim to minimize the L_p norm of their changes for large values of p, as this gives little to no constraint on the total number of pixels perturbed. In fact, Lou *et al.* [Luo+15] use the object coordinates to mask out the background region and show that this defends against some of the known attacks.

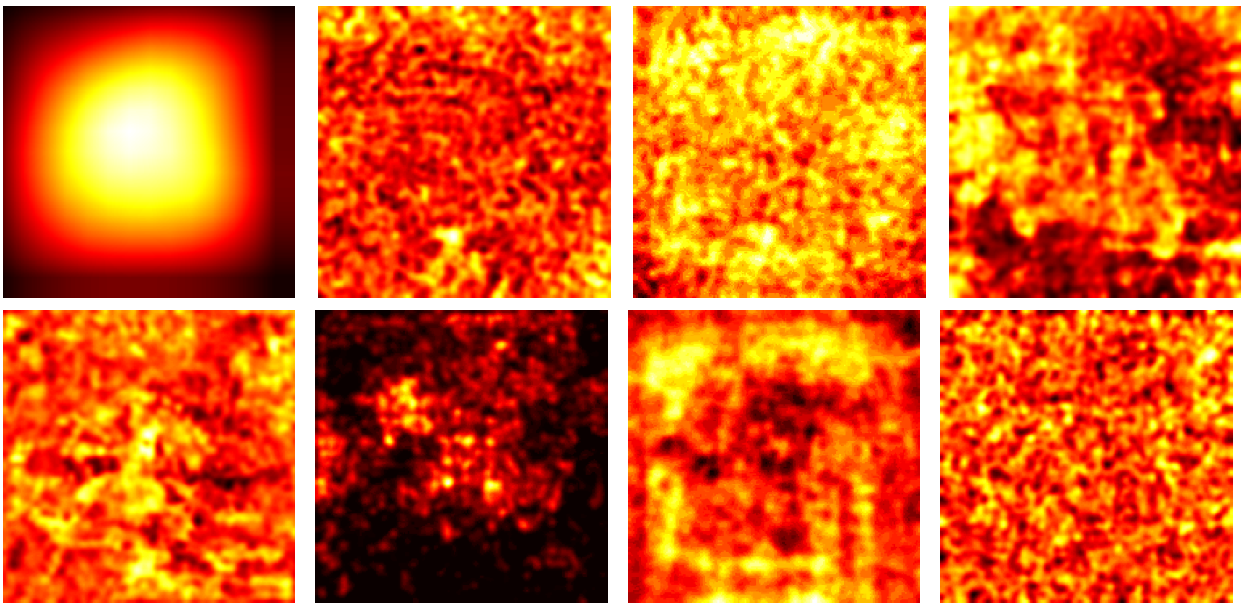


Figure 4.3: Visualization showing average location in the image where perturbation is added by an attacker. Clockwise from top left: Localization of most salient object in the image, FGSM, IGSM, FGSM-2 (higher ϵ), Deep Fool, JSMA, LBFGS and Carlini-Wagner attack.

In Figure 4.3 we show the average spatial distribution of perturbations for several attacks, as compared to the distribution of object locations (top left). Based on these ideas, we explore

the possibility of updating the pixels in the image such that the probability of that pixel being updated is inversely proportional to the likelihood of that pixel containing an object.

4.6 Targeted Pixel Deflection

As we have shown in section 4.5, image classification is robust against the loss of a certain number of pixels.

In natural images, many pixels do not correspond to a relevant semantic object and are therefore not salient to classification. Classifiers should then be more robust to pixel deflection if more pixels corresponding to the background are dropped as compared to the salient objects. Luo *et al.* [Luo+15] used this idea to mask the regions which did not contain the object, They use the ground truth co-ordinates from image localization dataset to obtain a mask. however, their method has two limitations which we will seek to overcome.

First, it requires ground-truth object coordinates and it is, therefore, difficult to apply to unlabeled inputs at inference time. We solve this by using a variant of class activation maps to obtain an approximate localization for salient objects. Class activation maps [Zho+16] are a weakly-supervised localization [Oqu+15a] technique in which the last layer of a CNN, often a fully connected layer, is replaced with a global average pooling layer. This results in a heat map which lacks pixel-level precision but is able to approximately localize objects by their class. We prefer to use weakly supervised localization over saliency maps [Hua+15], as saliency maps are trained on human eye fixations and thus do not always capture object classes [KTB14b]. Other weakly supervised localization techniques, such as regions-of-interest [Pra+17], capture more than a single object and thus are not suitable for improving single-class classification.

Second, completely masking out the background deteriorates classification of classes for

which the model has come to rely on the co-occurrence of non-class objects. For instance, airplanes are often accompanied by a sky-colored background, and most classifiers will have lower confidence when trying to classify an airplane outside of this context. We take a Bayesian approach to this problem and use stochastic re-sampling of the background. This preserves enough of the background to protect classification and drops enough pixels to weaken the impact of adversarial input.

4.6.1 Robust Activation Map

Class activation maps [Zho+16] are a valuable tool for approximate semantic object localization. Consider a convolutional network with k output channels on the final convolution layer (f) with spatial dimensions of x and y, and let $\boldsymbol{w}$ be a vector of size k which is the result of applying a global max pool on each channel. This reduces channel to a single value, $\boldsymbol{w_k}$. The class activation map, M_c for a class c is given by:

$$M_c(x, y) = \sum_k \boldsymbol{w}_k^c \; f_k(x, y) \tag{4.4}$$

Generally, one is interested in the map for the class for which the model assigns the highest probability. However, in the presence of adversarial perturbations to the input, the highest-probability class is likely to be incorrect. Fortunately, our experiments show that an adversary which successfully changes the most likely class tends to leave the rest of the top-k classes unchanged. Our experiments show that 38% of the time the predicted class of adversarial images is the second highest class of the model for the clean image. Figure 4.6 shows how the class of adversarial image relates to predictions on clean images. ImageNet has one thousand classes, many of which are fine-grained. Frequently, the second most likely class is a synonym or close relative of the main class (*e.g.* "Indian Elephant" and "African

Elephant"). To obtain a map which is robust to fluctuations of the most likely class, we take an exponentially weighted average of the maps of the top-k classes.

$$\widehat{M}(x,y) = \sum_{i}^{k} \frac{M_{c_i}(x,y)}{2^i} \tag{4.5}$$

We will refer to this robust activation maps as $\widehat{M}(x,y)$. We normalize the map by diving it by its max so that values are in the range of $[0,1]$. Even if the top-1 class is incorrect, this averaging reduces the impact of mis-localization of the object in the image.

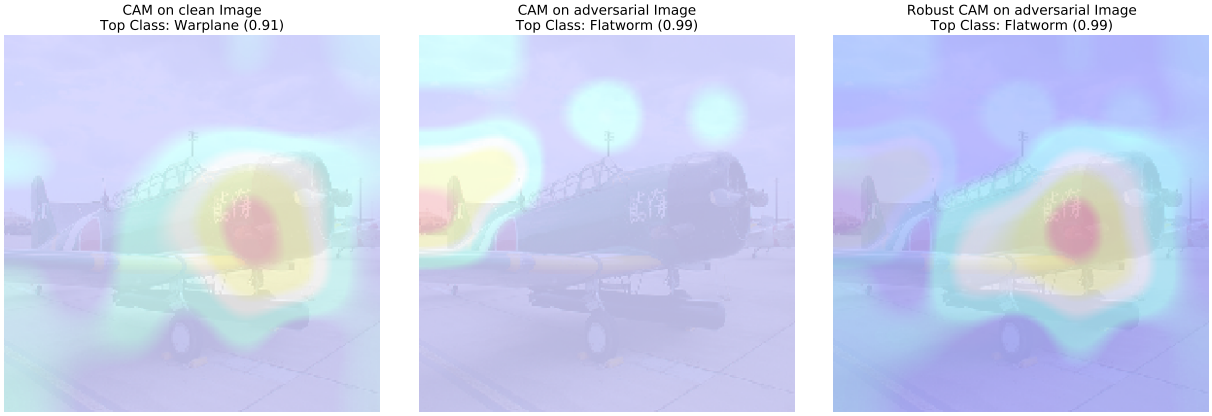


Figure 4.4: Difference between standard activation maps and robust maps under the presence of an adversary.

The appropriate number of classes k to average over depends on the total number of classes. For ImageNet-1000, we used a fixed $k = 5$. While each possible class has its own class activation map (CAM), only a single robust activation map is generated for a particular image, combining information about all classes. ImageNet covers wide variety of object classes and most structures found in other datasets are represented in ImageNet even if class names are not bijectional. Therefore, Robust Activation Map (R-CAM) is trained once on ImageNet but can also localize objects from Pascal-VOC or Traffic Signs.

4.7 Wavelet Denoising

Because both pixel deflection and adversarial attacks add noise to the image, it is desirable to apply a denoising transform to lessen these effects. Since adversarial attacks do not take into account the frequency content of the perturbed image, they are likely to pull the input away from the class of likely natural images in a way which can be detected and corrected using a multi-resolution analysis.

Works such as [CYV00; Sim99; Fie87] have shown that natural images exhibit regularities in their wavelet responses which can be learned from data and used to denoise images. These regularities can also be exploited to achieve better lossy image compression, the basis of JPEG2000. Many vision and neuroscience researchers [Mar80; Rus+05; HW59] have suggested that the visual systems of many animals take advantage of these priors, as the simple cells in the primary visual cortex have been shown to have Gabor-like receptive fields.

Swapping pixels within a window will tend to add noise with unlikely frequency content to the image, particularly if the window is large. This kind of noise can be removed by image compression techniques like JPEG, however, the quantization process in JPEG uses fixed tables that are agnostic to image content, and it quantizes responses at all amplitudes while the important image features generally correspond to large frequency responses. This quantization reduces noise but also gets rid of some of the signal.

Therefore, it is unsurprising that JPEG compression recovers correct classification on some of the adversarial images but also reduces the classification accuracy on clean images [KGB16; Das+17; DGR16; Guo+17b]. Dziugaite *et al.* [DGR16] reported loss of 8% accuracy on clean images after undergoing JPEG compression. We attribute the success of JPEG against adversarial images to the nonuniform quantization of spatially localized DCT

block coefficients, reflecting similar assumptions about the local frequency content of natural images to wavelet denoising.

We, therefore, seek filters with frequency response better suited to joint space-frequency analysis than the DCT blocks (and more closely matching representations in the early ventral stream, so that features which have a small filter response are less perceptible) and quantization techniques more suited to denoising. Wavelet denoising uses wavelet coefficients obtained using *Discrete Wavelet Transform* [Ant+92]. The wavelet transform represents the signal as a linear combination of orthonormal wavelets. These wavelets form a basis for the space of images and are separated in space, orientation, and scale. The Discrete Wavelet Transform is widely used in image compression [Ada01] and image denoising [CYV00; RVS02; Sim99].

It has benefits over Discrete Cosine Transform as it tries to recover signal regardless of the frequency content. While the noise introduced by dropping a pixel is mostly high-frequency, the same cannot be said about the adversarial perturbations. Several attempts have been made to quantify distribution of adversarial perturbations [Fei+17; Lin+17] but recent work by Carlini and Wagner [CW17a] has shown that most techniques fail to detect adversarial examples. We have observed that for the perturbations added by well-known attacks, wavelet denoising yields superior results as compared to block DCT.

4.7.1 Hard & Soft Thresholding

The process of performing a wavelet transform and its inverse is lossless and thus does not provide any noise reduction. In order to reduce adversarial noise, we need to apply thresholding to the wavelet coefficients before inverting the transform. Most compression techniques use a hard thresholding process, in which all coefficients with magnitude below

the threshold are set to zero: $Q(\hat{X}) = \hat{X} \; \forall \; |\hat{X}| > T_h$, where $\hat{X}$ is the wavelet transform of X, and T_h is the threshold value. The alternative is soft thresholding, in which we additionally subtract the threshold from the magnitude of coefficients above the threshold: $Q(\hat{X}) = \text{sign}(\hat{X}) \times \max(0, |\hat{X}| - T_h)$. Jansen *et al.* [Jan12] observed that hard thresholding results in over-blurring of the input image, while soft thresholding maintains better PSNR. By reducing all coefficients, rather than just those below the threshold, soft thresholding avoids introducing extraneous noise. This allows our method to preserve classification accuracy on non-adversarial images.

4.7.2 Adaptive Thresholding

Determining the proper threshold is very important, and the efficacy of our method relies on the ability to pick a threshold in an adaptive, image specific manner. The standard technique for determining the threshold for wavelet denoising is to use a universal threshold formula called *VisuShrink*. For an image X with N pixels, this is given by $\sigma\sqrt{2 \log N}$, where σ is the variance of the noise to be removed and is a hyper-parameter. However, we used *BayesShrink* [CYV00], which models the threshold for each wavelet coefficient as a Generalized Gaussian Distribution (GGD). The optimal threshold is then assumed to be the value which minimizes the expected mean square error *i.e.*

$$T_h * (\sigma_x, \beta) = \underset{T_h}{\text{argmin}} \, E(\hat{X} - X)^2 \; \approx \frac{\sigma^2}{\sigma_x} \tag{4.6}$$

where σ_x and β are parameters of the GGD for each wavelet sub-band. In practice, an approximation, as shown on right side of equation 4.6, is used. This ratio, also called T_{Bayes}, adapts to the amount of noise in the given image. Within a certain range of β values, BayesShrink has been shown to effectively remove artificial noise while preserving

the perceptual features of natural images [CYV00; RVS02]. As our experiments are carried out with images from ImageNet, which is a collection of natural images, we believe this is an appropriate thresholding technique to use. Yet another popular thresholding technique is Stein's Unbiased Risk Estimator (SUREShrink), which computes unbiased estimate of $E(\hat{X} - X)^2$. SUREShrink requires optimization to learn T_h for a given coefficient. We empirically evaluated results and SUREShrink did not perform as well as BayesShrink. Comparative results are shown in Table 4.8.

4.8 Method

The first step of our method is to corrupt the adversarial noise by applying targeted pixel deflection as follows:

(a) Generate a robust activation map $\widehat{M}$, as described in section 4.6.1.

(b) Uniformly sample a pixel location (x, y) from the image, and obtain the normalized activation map value for that location, $v_{x,y} = \widehat{M}(x, y)$.

(c) Sample a random value from a uniform distribution $\mathcal{U}(0, 1)$. If $v_{x,y}$ is lower than the random value, we deflect the pixel using the algorithm shown in Algorithm 3.

(d) Iterate this process K times.

The following steps are used to soften the impact of pixel deflection:

(a) Convert the image to YC_bC_r space to decorrelate the channels. YC_bC_r space is perceptually meaningful and thus has similar denoising advantages to the wavelets.

(b) Project the image into the wavelet domain using the discrete wavelet transform. We use the *db1* wavelet, but similar results were obtained with *db2* and *haar* wavelets.

(c) Soft threshold the wavelets using BayesShrink.

(d) Compute the inverse wavelet transform on the shrunken wavelet coefficients.

(e) Convert the image back to RGB.

4.9 Experimental Design

We tested our method on 1000 randomly selected images from the ImageNet [Den+09b] Validation set. We use ResNet-50 [He+16b] as our classifier. We obtain the pre-trained weights from TensorFlow's GitHub repository. These models achieved a Top-1 accuracy of 76% on our selected images. This is in agreement with the accuracy numbers reported in [He+16b] for a single-model single-crop inference.

By the definition set by adversarial attacks, an attack is considered successful by default if the original image is already mis-classified. In this case, the adversary simply returns the original image unmodified. However, these cases are not useful for measuring the effectiveness of an attack or a defense as there is no pixel level difference between the images. As such, we restrict our experiments to those images which are correctly classified in the absence of adversarial noise. Our attack models are based on the Cleverhans [NP17] library[1] with model parameters that aim to achieve the highest possible misclassification score with a normalized RMSE ($|\boldsymbol{L}_2|$) budget of $0.02 - 0.04$.

We will publicly release our implementation code. Impact of adversary can only be detected when there is a difference in predicted results. We reject the images from our set which yields incorrect classification on the clean image.

[1] https://github.com/tensorflow/cleverhans

4.9.1 Training

Unlike models that train a new network [Tra+17b; MC17] our model has very few hyper-parameters. Our defense model has three hyper-parameters, which is significantly fewer than the classification models it seeks to protect, making it preferable over defenses which require retraining of the classifier such as [Tra+17b; MC17]. These three hyper-parameters are: σ, a coefficient for *BayesShrink*, r, the window size for pixel deflection, and K, the number of pixel deflections to perform. Using a reduced set of 300 images from ImageNet Validation set, We perform a linear search over a small range of these hyper-parameters. We use 300 images from ImageNet Validation set to search for optimal hyper-parameters. These images are not part of the set used to show the results of our model. A particular set of hyper-parameters may be optimal for one attack model, but not for another. This is primarily because attacks seek to minimize different L_p norms, and therefore generate different types of noise. Different values of these parameters perform best on different attack strategies. Primarily because attacks are minimizing different L_p norms. To demonstrate the robustness of our defense, we select a single setting of the hyper-parameters to be used against all attack models. Figure 4.5 shows a visual indication of the variations in performance of each model across various hyper-parameter settings. In general, as the K and r increase, the variance of the resulting classification accuracy increases. This is primarily due to the stochastic nature of pixel deflection - as more deflections are performed over a wider window, a greater variety of transformed images can result. Variance in the accuracy increases with the increase in window size and number of iterations, this is due to stochastic nature of pixel deflection.

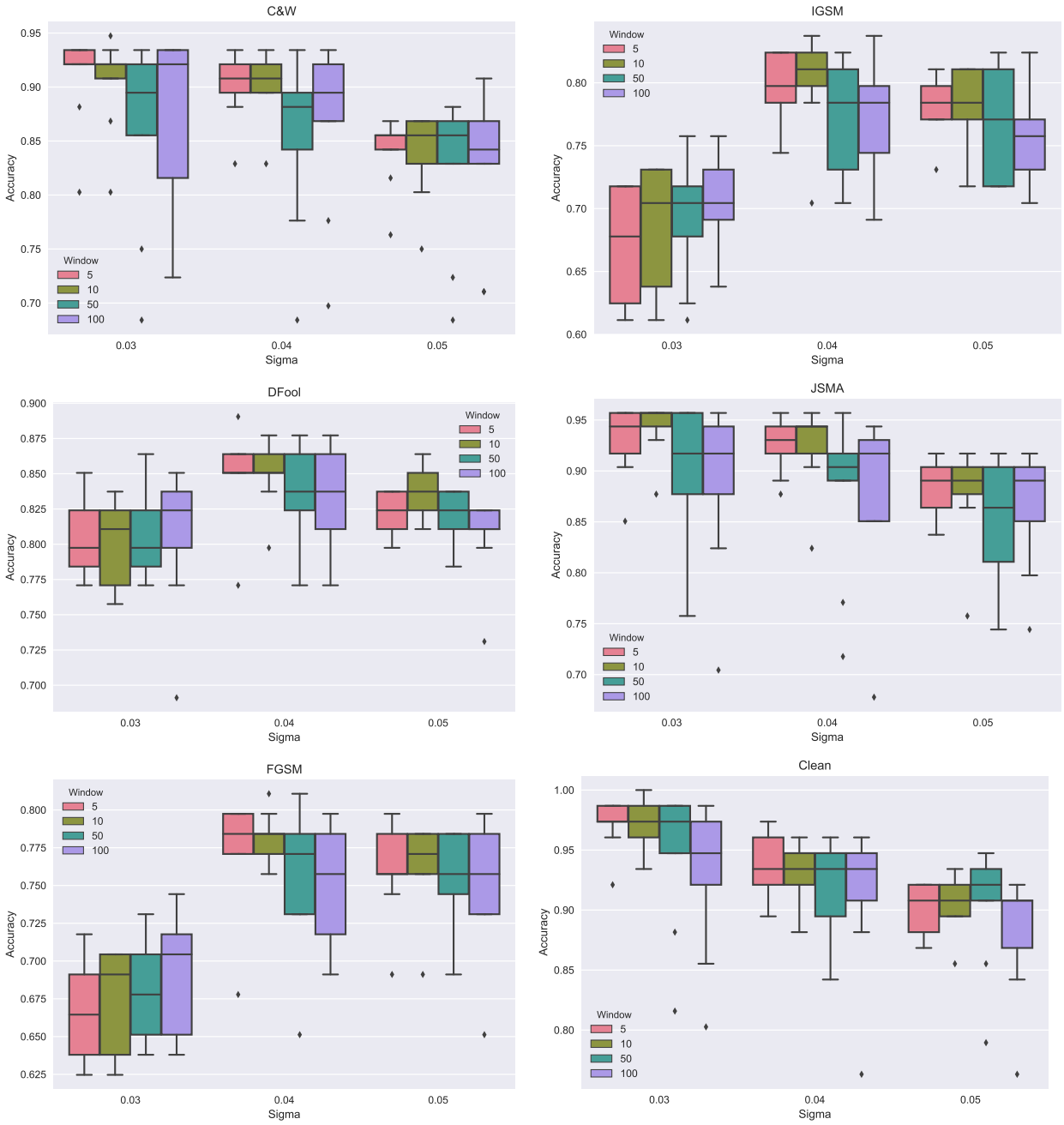


Figure 4.5: Linear search for model parameters

4.10 Results & Discussion

Model	$\lvert L_2 \rvert$	No Defense	With Defense	
			Single	Ens-10
Clean	0.00	100	98.3	**98.9**
FGSM	0.05	20.0	79.9	**81.5**
IGSM	0.03	14.1	83.7	**83.7**
DFool	0.02	26.3	86.3	**90.3**
JSMA	0.02	25.5	91.5	**97.0**
LBFGS	0.02	12.1	88.0	**91.6**
C&W	0.04	04.8	92.7	**98.0**
Large perturbations				
FGSM	0.12	11.1	61.5	**70.4**
IGSM	0.09	11.1	62.5	**72.5**
DFool	0.08	08.0	82.4	**88.9**
JSMA	0.05	22.1	88.9	**92.1**
LBFGS	0.04	12.1	77.0	**89.0**

Table 4.1: Params: $\sigma = 0.04$, Window=10, Deflections=100
Top-1 accuracy on applying pixel deflection and wavelet denoising across various attack models. We evaluate non-efficient attacks at larger $\lvert \boldsymbol{L}_P \rvert$ which leave visible perturbations to show the robustness of our model.

In Table 4.1 we present results obtained by applying our transformation against various untargeted white-box attacks. Our method is agnostic to classifier architecture, and thus shows

similar results across various classifiers. For brevity, we report only results on ResNet-50. Results for other classifiers are provided in Table 4.3. The accuracy on clean images without any defense is 100% because we didn't test our defense on images which were misclassified before any attack. We do not report results for targeted attacks as they are harder to generate [CW17b] and easier to defend. Due to the stochastic nature of our model, we benefit from taking the majority prediction over ten runs; this is reported in Table 4.1 as Ens-10. The benefits of doing majority ensemble are particularly pronounced against attacks such as JSMA which are based on activation intensity, thus different set of deflected pixels combine to weaken the adversary.

4.10.1 Results on larger samples

We randomly sampled 10K images from ILSVRC2012 validation set; this contained all 1000 classes with minimum of 3 images per class.

Attack	$\lvert L_2 \rvert$	**No Defense**	**With Defense**	
Window=10, Deflections=100			Single	Ens-10
Clean	0.00	100	98.1	**98.9**
FGSM	0.04	19.2	79.7	**81.2**
IGSM	0.03	11.8	81.7	**82.4**
DFool	0.02	18.0	87.7	**92.4**
JSMA	0.02	24.9	93.0	**98.1**
LBFGS	0.02	11.6	90.3	**93.6**
C&W	0.04	05.2	93.1	**98.3**

Table 4.2: Top-1 accuracy of our model on various attack models.

4.10.2 Results on various classifiers

Original classification accuracy of each classifier on selected 1000 images is reported in the table. However, we omit the images that were originally incorrectly classified, thus the accuracy of clean images without defense is always 100%. Weights for each classifier were obtained from Tensorflow GitHub repository [2].

Model	$\|L_2\|$	**No Defense**	**With Defense**	
			Single	Ens-10
ResNet-50, original classification 76%				
Clean	0.00	100	98.3	**98.9**
FGSM	0.05	20.0	79.9	**81.5**
IGSM	0.03	14.1	83.7	**83.7**
DFool	0.02	26.3	86.3	**90.3**
JSMA	0.02	25.5	91.5	**97.0**
LBFGS	0.02	12.1	88.0	**91.6**
C&W	0.04	04.8	92.7	**98.0**

Table 4.3: Top-1 accuracy of ResNet-50 with pixel deflection on various attack models.

[2] https://github.com/tensorflow/models/tree/master/research/slim#Pretrained

Model	$\lvert L_2 \rvert$	No Defense	With Defense	
			Single	Ens-10
VGG-19, original classification 71%				
Clean	0.00	100	99.8	**99.8**
FGSM	0.05	12.2	79.3	**81.3**
IGSM	0.04	9.79	79.2	**81.6**
DFool	0.01	23.7	83.9	**91.6**
JSMA	0.01	29.1	95.8	**98.5**
LBFGS	0.03	13.8	83.0	**93.9**
C&W	0.04	0.00	93.1	**97.6**

Table 4.4: Top-1 accuracy of VGG-19 with pixel deflection on various attack models.

Model	$\lvert L_2 \rvert$	No Defense	With Defense	
			Single	Ens-10
Inception-v3, original classification 78%				
Clean	0.00	100	98.1	**98.5**
FGSM	0.05	22.1	85.8	**87.1**
IGSM	0.04	15.5	**89.7**	89.1
DFool	0.02	27.2	82.6	**85.3**
JSMA	0.02	24.2	93.7	**98.6**
LBFGS	0.02	12.5	87.1	**91.0**
C&W	0.04	07.1	93.9	**98.5**

Table 4.5: Top-1 accuracy of Inception-v3 with pixel deflection on various attack models.

4.10.3 Comparison of results

- There are two main challenges when seeking to compare defense models. First, many attack and defense techniques primarily work on smaller images, such as those from CIFAR and MNIST. The few proposed transformation based defense techniques which work on larger-scale images are extremely recent, and currently under review [Xie+18; Guo+17b]. Second, because different authors target both different $|\boldsymbol{L}_P|$ norms and different perturbation magnitudes, it is difficult to balance the strength of various attacks. We achieved 98% recovery on C&W with $|\boldsymbol{L}_2|$ of 0.04 on ResNet-50, where Xie *et al.* [Xie+18] reports 97.1% on ResNet-101 and 98.8% on ens-adv-Inception-ResNet-v2. ResNet-101 is as stronger classifier than ResNet-50 and ens-adv-Inception-Resnet-v2 [Tra+17a] is an ensemble of classifiers specifically trained with adversarial augmentation. They do not report the $|\boldsymbol{L}_2|$ norm of the adversarial perturbations, and predictions are made on an ensemble of 21 crops. Guo *et al.* [Guo+17b] have reported (normalized) accuracy of 92.1% on C&W with $|\boldsymbol{L}_2|$ of 0.06, and their predictions are on an ensemble of 10 crops.

To present a fair comparison across various defenses we only measure the fraction of images which are no longer misclassified after the transformation. This ratio is known as Destruction Rate and was originally proposed in [24]. Value of 1 means all the misclassified images due to the adversary are correctly classified after the transformation.

Defense	FGSM	IGSM	DFool	C&W
Feature Squeezing (Xu et al [XEQ17])				
(a) Bit Depth (2 bit)	0.132	0.511	0.286	0.170
(b) Bit Depth (5 bit)	0.057	0.022	0.310	0.957
(c) Median Smoothing (2x2)	0.358	0.422	0.714	0.894
(d) Median Smoothing (3x3)	0.264	0.444	0.500	0.723
(e) Non-local Mean (11-3-2)	0.113	0.156	0.357	0.936
(f) Non-local Mean (13-3-4)	0.226	0.444	0.548	0.936
Best model (b) + (c) + (f)	0.434	0.644	0.786	0.915
Random resizing + padding (Xie *et al.* [Xie+18])				
Pixel padding	0.050	-	0.972	0.698
Pixel resizing	0.360	-	0.974	0.971
Padding + Resizing	0.478	-	**0.983**	0.969
Quilting + TVM (Guo *et al.* [Guo+17b])				
Quilting	0.611	0.862	0.858	0.843
TVM + Quilting	0.619	0.866	0.866	0.841
Cropping + TVM + Quilting	0.629	0.882	0.883	0.859
Our work: PD - Pixel Deflection, R-CAM: Robust CAM				
PD	0.735	0.880	0.914	0.931
PD + R-CAM	0.746	0.912	0.911	0.952
PD + R-CAM + DCT	0.737	0.906	0.874	0.930
PD + R-CAM + DWT	**0.769**	**0.927**	0.948	**0.981**

Table 4.6: Destruction Rate of various defense techniques. $|L_2|$ lies between $0.02 - 0.06$ and classifier accuracy is 76%. We only include the Black-box attacks, where the attack model is not aware of the defense techniques. Single Pattern Attack and Ensemble pattern attack as reported in Xie et al [Xie+18] are not reported.

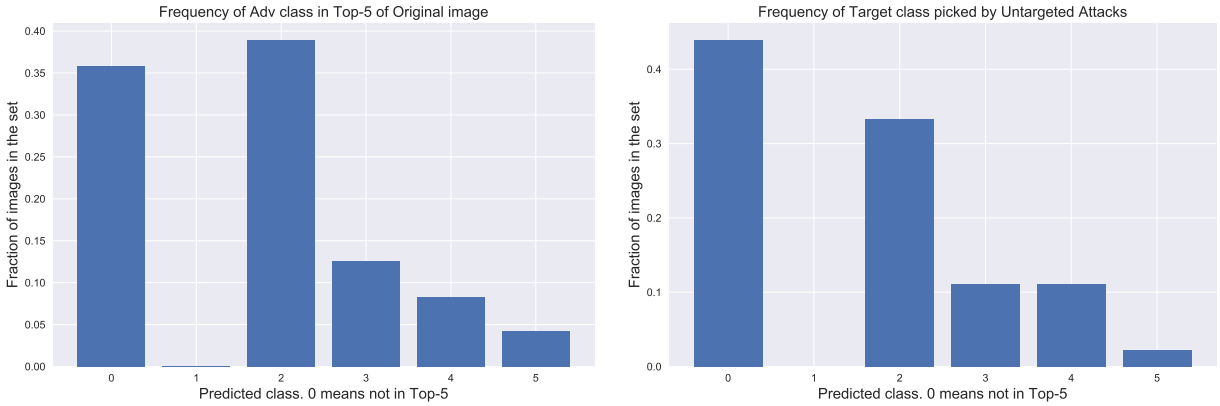


Figure 4.6: Left: Rank of adversarial class within the top-5 predictions for original images. Right: Rank of original class within the top-5 predictions for adversarial images. In both cases, 0 means the class was not in the top-5.

We do not report Top-5 accuracy for these evaluations as reported by Kurakin*et al.* [KGB16], as the adversary often targets the second most likely class, leaving top-5 accuracy unaffected. In 64% of the cases the mis-classified class was still within Top-5 picks barring the most likely class. As seen in Figure 4.6, the predicted class of the perturbed image is very frequently among the classifier's top-5 predictions for the original image. In fact, nearly 40% of the time, the adversarial class was the second most-probable class of the original image. Similarly, the original classification will often remain in the top-5 predictions for the adversarial image. Unlike Kurakin *et al.* [KGB16], our results are in terms of top-1 accuracy, as this matches the objective of the attacker. While top-1 accuracy is a more lenient metric for an attack method (due to the availability of nearly-synonymous alternatives to most classes in ImageNet-1000), it is a more difficult metric for a defense, as we must exactly recover the correct classification. These facts render top-5 accuracy an unsuitable metric for measuring the efficacy of a defense. While evaluating an adversarial system this might be considered as a weaker result, however for a defense model this is a harder task, as the semantic difference in the classes between clean image and adversarial image is very low. When evaluating

stochastic transformations, it is quite common to use an ensemble of predictions to get a more generalized version of the model's performance. We perform pixel deflection on a given image ten times and do a majority vote of the predicted classes to report the accuracy numbers. Results reported for Carlini & Wagner [CW17b] attacks are only for L_2 loss, even though they can be applied for L_0 and L_∞. Carlini & Wagner attack has been shown to be effective with MNIST and CIFAR but their efficacy against large images is limited due to expensive computation.

4.10.4 Ablation studies

Previous work [KGB16; DGR16] has demonstrated the efficacy of JPEG compression as a defense against adversarial attacks due to its denoising properties. Das *et al.* [Das+17] demonstrate that increasing the severity of JPEG compression defeats a larger percentage of attacks, but at the cost of accuracy on clean image. As our method employs a conceptually similar method to reduce adversarial noise via thresholding in a wavelet domain, we use JPEG as a baseline for comparison. Thus we consider JPEG as control to demonstrate efficacy of wavelet denoising.

In Table 4.7, we report accuracy with and without wavelet denoising with soft thresholding. While JPEG alone is effective against only a few attacks, the combination of JPEG and pixel deflection performs better than pixel deflection alone. The best results are obtained from pixel deflection and wavelet denoising. Adding JPEG on top of these leads to a drop in performance.

Model	JPG	WD	PD	PD JPG	WD PD JPG	WD PD
Clean	96.1	98.7	97.4	96.1	96.1	**98.9**
FGSM	49.1	40.6	79.7	81.1	78.8	**81.5**
IGSM	49.1	31.2	82.4	82.4	79.7	**83.7**
DFool	67.8	61.1	86.3	86.3	86.3	**90.3**
JSMA	91.6	89.1	95.7	93.0	93.0	**97.0**
LBFGS	71.8	67.2	90.3	89.1	88.9	**91.6**
C&W	85.5	95.4	95.4	94.1	93.4	**98.0**

Table 4.7: Params: $\sigma = 0.04$, Window=10, Deflections=100
Ablation study of pixel deflection (PD) in combination with wavelet denoising (WD) and JPEG compression.

Model	Hard	VISU	SURE	Bayes
Clean	39.5	96.1	92.1	**98.9**
FGSM	35.9	63.8	79.7	**81.5**
IGSM	42.5	67.8	81.1	**83.7**
DFool	37.2	78.4	87.7	**90.3**
JSMA	39.9	93.0	93.0	**97.0**
LBFGS	37.2	81.1	90.4	**91.6**
C&W	36.8	93.4	92.8	**98.0**

Table 4.8: Params: $\sigma = 0.04$, Window=10, Deflections=100
Comparison of various thresholding techniques, after application of pixel deflection.

In Table 4.8 we present a comparison of various shrinkage methods on wavelet coefficients

after pixel deflection. All the results reported are for applying the given thresholding after pixel deflection. For the impact of coefficient thresholding in the absence of pixel deflection, see Table 4.7. BayesShrink, which learns separate Gaussian parameters for each coefficient, does better than other soft-thresholding techniques. A brief overview of these shrinkage techniques are provided in Section 4.7.2, for more thorough review on BayesShrink, VisuShrink and SUREShrink we refer the reader to [CYV00] [DJ94] and [DJ92] respectively. VisuShrink is a faster technique as it uses a universal threshold but that limits its applicability on some images. SUREShrink has been shown to perform well with compression but as evident, in our results, it is less well suited to denoising.

Attack	$\|L_2$	**No Defense**	**With Defense**			
Window=10, Deflections $\longrightarrow$			**10**	**100**	**1K**	**10K**
Clean	0.00	100	**98.4**	98.1	94.7	80.3
FGSM	0.04	19.2	75.7	**79.7**	71.7	69.1
IGSM	0.03	13.8	78.4	**81.7**	75.2	71.2
DFool	0.02	25.0	83.7	**87.7**	81.0	77.0
JSMA	0.02	25.9	91.7	**93.0**	87.7	67.7
LBFGS	0.02	11.6	85.0	**90.3**	82.4	73.0
C&W	0.04	05.2	89.4	**93.1**	86.8	69.7

Table 4.9: Top-1 accuracy with different deflections.

Sampling technique (Random Pixel)				
Window ⟶	**5**	**10**	**50**	**100**
Uniform	**86.7**	**87.5**	**86.1**	**84.6**
Gaussian	80.0	81.4	79.0	76.4
Replacement technique (Uniform Sampling)				
Window ⟶	**5**	**10**	**50**	**100**
Min	73.0	64.4	49.1	44.3
Max	69.7	63.8	51.9	45.4
Mean	83.6	72.3	57.2	49.1
Random	**86.7**	**87.5**	**86.1**	**84.6**
Various Denoising Techniques				
Bilateral	**Anisotropic**	**TVM**	**Deconv**	**Wavelet**
78.1	84.1	77.26	85.12	**87.5**

Table 4.11: Top-1 accuracy averaged across all six attacks.

Attack	**L2**	**No Defense**	**With Defense**			
Deflections=100, Window ⟶			**5**	**10**	**50**	**100**
Clean	0.00	100	**98.6**	98.1	96.4	94.4
FGSM	0.04	19.2	**79.7**	**79.7**	78.4	76.7
IGSM	0.03	13.8	81.0	**81.7**	79.7	78.4
DFool	0.02	25.0	86.4	**87.7**	87.7	85.0
JSMA	0.02	25.9	92.3	**93.0**	91.7	90.3
LBFGS	0.02	11.6	89.4	**90.3**	89.0	88.1
C&W	0.04	05.2	91.8	**93.1**	90.5	89.2

Table 4.10: Top-1 accuracy with different window sizes.

For the following studies we present results averaged across all six attacks.

4.11 Conclusion

Motivated by the robustness of CNNs and the fragility of adversarial attacks, we have presented a technique which combines a computationally-efficient image transform, *pixel deflection*, with soft wavelet denoising. This combination provides an effective defense against state-of-the-art adversarial attacks. We show that most attacks are agnostic to semantic content, and using *pixel deflection* with probability inversely proportionate to robust activation maps (R-CAM) protects regions of interest. In ongoing work, we seek to improve our technique by adapting hyperparameters based on the features of individual images. Additionally, we seek to integrate our robust activation maps with wavelet denoising.

Chapter 5

RePr

5.1 Introduction

Convolutional Neural Networks have achieved state-of-the-art results in various computer vision tasks [He+16b; Lin+18]. Much of this success is due to innovations of a novel, task-specific network architectures [He+17; RFB15]. Despite variation in network design, the same core optimization techniques are used across tasks. All these networks are trained using similar optimization techniques, which are principally designed for a generic neural network which treats each weight as a single entity - *a neuron*. These techniques consider each individual weight as its own entity and update them independently. Limited progress has been made towards developing a training process specifically designed for convolutional networks, in which *filters* are the fundamental unit of the network. A filter is not a single weight parameter but a stack of spatial kernels. One of the implications of this is that most models are over-parameterized and a lot of the filters are redundant.

Because models are typically over-parameterized, a trained convolutional network will contain redundant filters [Cog+16; Li+17]. This is evident from the common practice of

pruning filters [HZS17; AHS17; Li+17; Mol+17; Liu+17b; LWL17], rather than individual parameters [HMD16], to achieve model compression. Most of these pruning methods are able to drop a significant number of filters with only a marginal loss in the performance of the model. However, a model with fewer filters cannot be trained from scratch to achieve the performance of a large model that has been pruned to be roughly the same size [Li+17; LWL17; ZG17]. Standard training procedures tend to learn models with extraneous and prunable filters, even for architectures without any excess capacity. This suggests that there is room for improvement in the training of Convolutional Neural Networks (ConvNets).

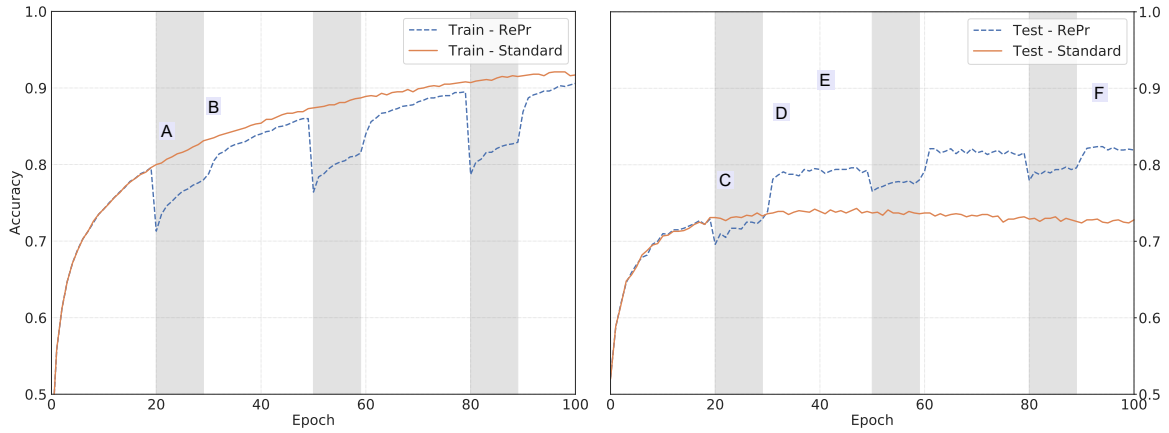


Figure 5.1: Performance of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme and our method - RePr on CIFAR-10. The shaded regions denote periods when only part of the network is trained. Left: Training Accuracy, Right: Test Accuracy. Annotations [A-F] are discussed in Section 5.4.

To this end, we propose a training scheme in which, after some number of iterations of standard training, we select a subset of the model's filters to be temporarily dropped. After additional training of the reduced network, we reintroduce the previously dropped filters, initialized with new weights, and continue standard training. We observe that following the reintroduction of the dropped filters, the model is able to achieve higher performance than was obtained before the drop. Repeated application of this process obtains models

which outperform those obtained by standard training as seen in Figure 5.1 and discussed in Section 5.4. We observe this improvement across various tasks and over various types of convolutional networks. This training procedure is able to produce improved performance across a range of possible criteria for choosing which filters to drop, and further gains can be achieved by careful selection of the ranking criterion. According to a recent hypothesis [FC18], the relative success of over-parameterized networks may largely be due to an abundance of initial sub-networks. Our method aims to preserve successful sub-networks while allowing the re-initialization of less useful filters.

In addition to our novel training strategy, the second major contribution of our work is an exploration of metrics to guide filter dropping. Our experiments demonstrate that standard techniques for permanent filter pruning are suboptimal in our setting, and we present an alternative metric which can be efficiently computed, and which gives a significant improvement in performance. We propose a metric based on the *inter*-filter orthogonality within convolutional layers and show that this metric outperforms state-of-the-art filter importance ranking methods used for network pruning in the context of our training strategy. We observe that even small, under-parameterized networks tend to learn redundant filters, which suggests that filter redundancy is not solely a result of over-parameterization, but is also due to ineffective training. Our goal is to reduce the redundancy of the filters and increase the expressive capacity of ConvNets and we achieve this by changing the training scheme rather than the model architecture.

5.2 Related Work

We have divided the discussion of related works into categories that broadly cover our area of research.

5.2.1 Training Scheme

Many changes to the training paradigm have been proposed to reduce over-fitting and improve generalization. Dropout [WG15] is widely used in training deep nets. By stochastically dropping the neurons it prevents co-adaption of feature detectors. A similar effect can be achieved by dropping a subset of activations [Wan+13]. Wu *et al.* [WG15] extend the idea of stochastic dropping to convolutional neural networks by probabilistic pooling of convolution activations. Yet another form of stochastic training recommends randomly dropping entire layers [Hua+16], forcing the model to learn similar features across various layers which prevent extreme overfitting. In contrast, our technique encourages the model to use a linear combination of features instead of duplicating the same feature. Han *et al.* [Han+16] propose Dense-Sparse-Dense (DSD), a similar training scheme, in which they apply weight regularization mid-training to encourage the development of sparse weights, and subsequently remove the regularization to restore dense weights. They recommend adding weight regularization to make sparse weights and removing the regularization subsequently to achieve dense network again. DSD works on individual parameters and uses weight norm to drop the weights. We designed our training scheme for specially convolutional filters and we show that it is important to allow each of the model (with reduced filters) to converge before switching over. While DSD works at the level of individual parameters, our method is specifically designed to apply to convolutional filters. We achieve this by allowing the sub-network to relearn any missing feature representations with existing filters. In a limited model capacity, doing so encourages more expressivity [Rag+17a] and as our results show it leads to efficient model.

5.2.2 Model Compression

Knowledge Distillation (KD) [HVD15] is a training scheme which uses soft logits from a larger trained model (teacher) to train a smaller model (student). Soft logits capture hierarchical information about the object and provide a smoother loss function for optimization. This leads to easier training and better convergence for small models. Stronger forms of KD where a model matches intermediate activations have also been proposed [Rom+15]. By forcing the student model to match the output statistics from the teacher, it provides the directions for weight updates which are not found in the gradients. In a surprising result, Born-Again-Network [Fur+18] shows that if the student model is of the same capacity as the teacher it can outperform the teacher. A few other variants of KD have been proposed [Rom+15] and all of them require training several models. Our training scheme does not depend on an external teacher and requires less training than KD. More importantly, when combined with KD, our method gives better performance than can be achieved by either technique independently (discussed in Section 5.7).

5.2.3 Neuron ranking

Interest in finding the least salient neurons/weights has a long history. LeCun [LDS89] and Hassibi*et al.* [HS92] show that using the Hessian, which contains second-order derivative, identifies the weak neurons and performs better than using the magnitude of the weights. Computing the Hessian is expensive and thus is not widely used. Han *et al.* [HMD16] show that the norm of weights is still effective ranking criteria and yields sparse models. The sparse models do not translate to faster inference, but as a neuron ranking criterion, they are effective. Computing the norm of the weights is easy and data-free, thus lends itself to pruning for transfer learning. Hu *et al.* [Hu+16] explore Average Percentage of Zeros (APoZ)

in the activations and use a data-driven threshold to determine the cut-off. It is a useful comparison metric and works well when the training data is available. ThinNet [LWL17] explores reconstruction error between the layers, this does not lead to a generalized filter ranking criteria. Molchanov *et al.* [Mol+17] recommend the second term from the Taylor expansion of the loss function.Currently, it is the state-of-the-art in filter pruning and works well for transfer pruning. Our filter ranking criteria, *inter*-filter orthogonality, is yet another approach and is designed especially for the purposes of temporary pruning. We provide detail comparison and show results on using these metrics with our training scheme in Section 5.5.

Chen *et al.* [Che+17] used privileged information, like segmentation maps, to encourage diversity in feature maps, and approximate group orthogonal filters. Unfortunately, such techniques restrict themselves to domains where segmentation maps are available for the task of image classification. Our method does not rely on any external information or a pretrained model.

5.2.4 Architecture Search

Neural architecture search [Liu+17a; Rea+18; ZL16] is where the architecture is modified during training, and multiple neural network structures are explored in search of the best architecture for a given dataset. Such methods do not have any benefits if the architecture is fixed ahead of time. Our scheme improves training for a given architecture by making better use of the available parameters. This could be used in conjunction with architecture search if there is flexibility around the final architecture or used on its own when the architecture is fixed due to certified model deployment, memory requirements, or other considerations. Our scheme is finds better weights given a fixed architecture and thus have a lower potential than the NAS models. Potential for finding a better generalizing architecture diminishes

by restricting to a fixed final architecture but asserts itself to several applications where predetermined architecture is warranted. This could be due to certified model deployment, memory requirements or better interpretability of simpler ConvNets.

5.2.5 Feature correlation

A well-known shortcoming of vanilla convolutional networks is their correlated feature maps [Cog+16; GB10]. Architectures like Inception-Net [Sze+15] are motivated by analyzing the correlation statistics of features across layers. They aim to reduce the correlation between the layers by using concatenated features from various sized filters, subsequent research shows otherwise [Rag+17b]. More recent architectures like ResNet [He+16b] and DenseNet [Hua+17] aim to implicitly reduce feature correlations by summing or concatenating activations from previous layers. That said, these models are computationally expensive and require large memory to store previous activations. Our aim is to induce decorrelated features without changing the architecture of the convolutional network. This benefits all the existing implementations of ConvNet without having to change the infrastructure. While our technique performs best with vanilla ConvNet architectures it still marginally improves the performance of modern architectures.

5.3 Motivation for Orthogonal Features

A feature for a convolutional filter is defined as the point-wise sum of the activations from individual kernels of the filter. A feature is considered useful if it helps to improve the generalization of the model. A model that has poor generalization usually has features that, in aggregate, capture limited directions in activation space [Mor+17]. On the other hand, if a model's features are orthogonal to one another, they will each capture distinct directions in

activation space, leading to improved generalization. For a trivially-sized ConvNet, we can compute the maximally expressive filters by analyzing the correlation of features across layers and clustering them into groups [Aro+14]. However, this scheme is computationally impractical for the deep ConvNets used in real-world applications. Alternatively, a computationally feasible option is the addition of a regularization term to the loss function used in standard SGD training which encourages the minimization of the covariance of the activations, but this produces only limited improvement in model performance [Rod+17; Cog+16]. A similar method, in which the regularization term instead encourages the orthogonality of filter weights, has also produced marginal improvements [Bro+17; PSDG13; XPX17; XXP17]. Shang *et al.* [Sha+16] discovered the low-level filters are duplicated with opposite phase. Forcing filters to be orthogonal will minimize this duplication without changing the activation function. In addition to improvements in performance and generalization, Saxe *et al.* [SMG14] show that the orthogonality of weights also improves the stability of network convergence during training. The authors of [XXP17; Xia+18] further demonstrate the value of orthogonal weights to the efficient training of networks. Orthogonal initialization is common practice for Recurrent Neural Networks due to their increased sensitivity to initial conditions [Vor+17], but it has somewhat fallen out of favor for ConvNets. These factors shape our motivation for encouraging orthogonality of features in the ConvNet and form the basis of our ranking criteria.

Collecting activations over a dataset in order to compute orthogonal features is slow and inefficient. Instead, we use orthogonality of weights, to represent feature overlap. Because features are dependent on the input data, determining their orthogonality requires computing statistics across the entire training set, and is therefore prohibitive. We instead compute the orthogonality of filter weights as a surrogate. Our experiments show that encouraging weight orthogonality through a regularization term is insufficient to promote the development of

features which capture the full space of the input data manifold. Our method of dropping overlapping filters acts as an implicit regularization and leads to the better orthogonality of filters without hampering model convergence.

We use Canonical Correlation Analysis [Hot36] (CCA) to study the overlap of features in a single layer. CCA finds the linear combinations of random variables that show maximum correlation with each other. It is a useful tool to determine if the learned features are overlapping in their representational capacity. Li *et al.* [Li+16] apply correlation analysis to filter activations to show that most of the well-known ConvNet architectures learn similar representations. Raghu *et al.* [Rag+17b] combine CCA with SVD to perform a correlation analysis of the singular values of activations from various layers. They show that increasing the depth of a model does not always lead to a corresponding increase of the model's dimensionality, due to several layers learning representations in correlated directions. We ask an even more elementary question - how correlated are the activations from various filters within a single layer? In an over-parameterized network like VGG-16, which has several convolutional layers with 512 filters each, it is no surprise that most of the filter activations are highly correlated. As a result, VGG-16 has been shown to be easily pruned - more than 50% of the filters can be dropped while maintaining the performance of the full network [Mol+17; Li+16]. Is this also true for significantly smaller convolutional networks, which under-fit the dataset?

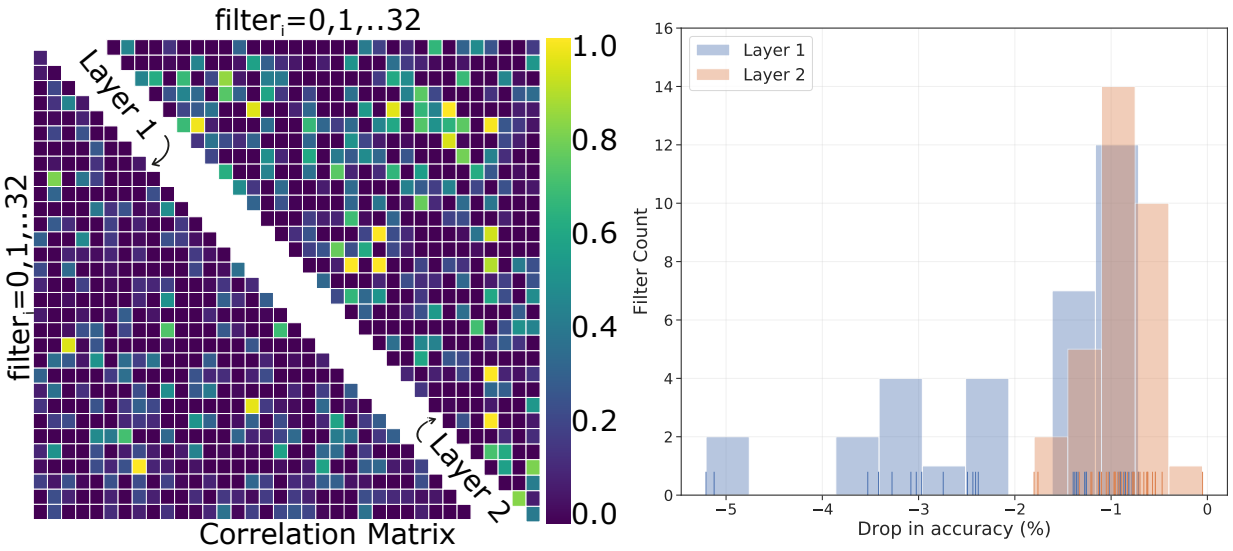


Figure 5.2: Left: Canonical Correlation Analysis of activations from two layers of a ConvNet trained on CIFAR-10. Right: Distribution of change in accuracy when the model is evaluated by dropping one filter at a time.

We will consider a simple network with two convolutional layers of 32 filters each, and a softmax layer at the end. Training this model on CIFAR-10 for 100 epochs with an annealed learning rate results in test set accuracy of 58.2%, far below the 93.5% achieved by VGG-16. Given the limited capacity of this model every filter is critical to its performance and any correlation in activation would mean limited expressivity of the features and therefore inefficiency in the training. In the case of VGG-16, we might expect that correlation between filters is merely an artifact of the over-parameterization of the model - the dataset simply does not have a dimensionality high enough to require every feature to be orthogonal to every other. On the other hand, our small network has clearly failed to capture the full feature space of the training data, and thus any correlation between its filters is due to inefficiencies in training, rather than over-parameterization.

Given a trained model, we can evaluate the contribution of each filter to the model's performance by removing *(zeroing out)* that filter and measuring the drop in accuracy on

the test set. We will call this metric of filter importance the *"greedy Oracle"*. We perform this evaluation independently for every filter in the model, and plot the distribution of the resulting drops in accuracy in Figure 5.2 (right). Most of the second layer filters contribute less than 1% in accuracy and with first layer filters, there is a long tail. Some filters are important and contribute over 4% of accuracy but most filters are around 1%. This implies that even a tiny and under-performing network could be filter pruned without significant performance loss. The model has not efficiently allocated filters to capture wider representations of necessary features. Figure 5.2 (left) shows the correlations from linear combinations of the filter activations (CCA) at both the layers. It is evident that in both the layers there is a significant correlation among filter activations with several of them close to a near perfect correlation of 1 (*bright yellow spots* ■). The second layer (upper right diagonal) has lot more overlap of features the first layer (lower right). For a random orthogonal matrix any value above 0.3 (*lighter than dark blue* ■) is an anomaly. The activations are even more correlated if the linear combinations are extended to kernel functions [HSST04] or singular values [Rag+17b]. Regardless, it suffices to say that standard training for convolutional filters does not maximize the representational potential of the network.

5.4 Our Training Scheme : RePr

We modify the training process by cyclically removing redundant filters, retraining the network, re-initializing the removed filters, and repeating. We consider each filter (3D *tensor*) as a single unit, and represent it as a long vector - (f). Let $\mathbf{M}$ denote a model with $\mathcal{F}$ filters spread across $\mathbf{L}$ layers. Let $\widehat{\mathcal{F}}$ denote a subset of $\mathcal{F}$ filters, such that $\mathbf{M}_{\mathcal{F}}$ denotes a complete network whereas, $\mathbf{M}_{\mathcal{F}-\widehat{\mathcal{F}}}$ denotes a sub-network without that $\widehat{\mathcal{F}}$ filters. Our training scheme alternates between training the complete network ($\mathbf{M}_{\mathcal{F}}$) and the sub-network ($\mathbf{M}_{\mathcal{F}-\widehat{\mathcal{F}}}$). This

introduces two hyper-parameters. First is the number of iterations to train each of the networks before switching over; let this be S_1 for the full network and S_2 for the sub-network. These have to be non-trivial values so that each of the networks learns to improve upon the results of the previous network. The second hyper-parameter is the total number of times to repeat this alternating scheme; let it be N. This value has minimal impact beyond certain range and does not require tuning. If S_1 and S_2 are equal to 1, then this scheme becomes stochastic dropout of filters.

The most important part of our algorithm is the metric used to rank the filters. Let $\mathcal{R}$ be the metric which associates some numeric value to a filter. This could be a norm of the weights or its gradients or our metric - *inter*-filter orthogonality in a layer. Here we present our algorithm agnostic to the choice of metric. Most sensible choices for filter importance results in an improvement over standard training when applied to our training scheme *(see Ablation Study 5.6)*.

Our training scheme operates on a macro-level and is not a weight update rule. Thus, is not a substitute for SGD or other adaptive methods like Adam [KB15] and RmsProp [TH12]. Our scheme works with any of the available optimizers and shows improvement across the board. However, if using an optimizer that has parameters specific learning rates *(like Adam)*, it is important to re-initialize the learning rates corresponding to the weights that are part of the pruned filters ($\widehat{\mathcal{F}}$). Corresponding Batch Normalization [IS15] parameters ($\gamma\&\beta$) must also be re-initialized. For this reason, comparisons of our training scheme with standard training are done with a common optimizer. We reinitialize the filters ($\widehat{\mathcal{F}}$) to be orthogonal to its value before being dropped and the current value of non-pruned filters ($\mathcal{F} - \widehat{\mathcal{F}}$).

Our algorithm is training interposed with **Re**-initializing and **Pr**uning - **RePr** *(pro-*

nounced: reaper). We summarize our training scheme in Algorithm 4.

Algorithm 4: RePr Training Scheme

for N *iterations* **do**
  for S_1 *iterations* **do**
    Train the full network: $\mathbf{M}_{\mathcal{F}}$
  end
  Compute the metric : $\mathcal{R}(f)\ \forall f \in \mathcal{F}$
  Let $\widehat{\mathcal{F}}$ be bottom $p_{\%}$ of $\mathcal{F}$ using $\mathcal{R}(f)$
  for S_2 *iterations* **do**
    Train the sub-network : $\mathbf{M}_{\mathcal{F}-\widehat{\mathcal{F}}}$
  end
  Reinitialize the filters ($\widehat{\mathcal{F}}$) s.t. $\widehat{\mathcal{F}} \perp \mathcal{F}$
  (and their training specific parameters
  from BatchNorm and Adam, if applicable)
end

We use a shallow model to analyze the dynamics of our training scheme and its impact on the train/test accuracy. A shallow model will make it feasible to compute the greedy Oracle ranking for each of the filters. This will allow us to understand the impact of training scheme alone without confounding the results due to the impact of ranking criteria. We provide results on larger and deeper convolutional networks in Section Results 5.8.

Consider a n layer vanilla ConvNet, without a skip or dense connections, with X filter each, as shown below:

$$\text{Img} \longmapsto \Big[\text{CONV}(X) \rightarrow \text{RELU}\Big]^{n} \longmapsto \text{FC} \longmapsto \text{Softmax}$$

We will represent this architecture as $C^n(X)$. Thus, a $C^3(32)$ has 96 filters, and when trained with SGD with a learning rate of 0.01, achieves test accuracy of 73%. Figure 5.1 shows training plots for accuracy on the training set (left) and test set (right). In this example, we use a RePr training scheme with $S_1 = 20, S_2 = 10, N = 3, p_{\%} = 30$ and the ranking criteria $\mathcal{R}$ as a greedy Oracle. We exclude a separate validation set of 5K images from the training set to compute the Oracle ranking. In the training plot, annotation [A] shows the point at which the filters are first pruned. Annotation [C] marks the test accuracy of the model at this point. The drop in test accuracy at [C] is lower than that of training accuracy at [A], which is not a surprise as most models overfit the training set. However, the test accuracy at [D] is the same as [C] but at this point, the model only has 70% of the filters. This is not a surprising result, as research on filter pruning shows that at lower rates of pruning most if not all of the performance can be recovered [Mol+17].

What is surprising is that test accuracy at [E], which is only a couple of epochs after re-introducing the pruned filters, is significantly higher than point [C]. Both point [C] and point [E] are same capacity networks, and higher accuracy at [E] is not due to the model convergence. In the standard training (orange line) the test accuracy does not change during this period. Models that first grow the network and then prune [DYJ17; Han+15], unfortunately, stopped shy of another phase of growth, which yields improved performance. In their defense, this technique defeats the purpose of obtaining a smaller network by pruning. However, if we continue RePr training for another two iterations, we see that the point [F], which is still at 70% of the original filters yields accuracy which is comparable to the point [E] (100% of the model size.

Another observation we can make from the plots is that training accuracy of RePr model is lower, which signifies some form of regularization on the model. This is evident in the Figure 5.4 (Right), which shows RePr with a large number of iterations ($N = 28$). While the

marginal benefit of higher test accuracy diminishes quickly, the generalization gap between train and test accuracy is reduced significantly.

5.5 Our Metric : *inter*-filter orthogonality

The goals of searching for a metric to rank least important filters are twofold - (1) computing the greedy Oracle is not computationally feasible for large networks, and (2) the greedy Oracle may not be the best criteria. If a filter which captures a unique direction, thus not replaceable by a linear combination of other filters, has a lower contribution to accuracy, the Oracle will drop that filter. On a subsequent re-initialization and training, we may not get back the same set of directions.

The directions captured by the activation pattern expresses the capacity of a deep network [Rag+17a]. Making orthogonal features will maximize the directions captured and thus expressiveness of the network. In a densely connected layer, orthogonal weights lead to orthogonal features, even in the presence of ReLU [Vor+17]. However, it is not clear how to compute the orthogonality of a convolutional layer.

A convolutional layer is composed of parameters grouped into spatial kernels and sparsely share the incoming activations. Should all the parameters in a single convolutional layer be considered while accounting for orthogonality? The theory that promotes initializing weights to be orthogonal is based on densely connected layers (FC-layers) and popular deep learning libraries follow this guide[1] by considering convolutional layer as one giant vector disregarding the sparse connectivity. A recent attempt to study orthogonality of convolutional filters is described in [Xia+18] but their motivation is the convergence of very deep networks (10K layers) and not orthogonality of the features. Our empirical study suggests a strong

[1] tensorflow:ops/init_ops.py#L543 & pytorch:nn/init.py#L350

preference for requiring orthogonality of individual filters in a layer (inter-filter & intra-layer) rather than individual kernels.

A filter of kernel size $k \times k$ is commonly a 3D tensor of shape $k \times k \times c$, where c is the number of channels in the incoming activations. Flatten this tensor to a 1D vector of size $k * k * c$, and denote it by f. A convolutional layer usually has more than one filter per layer. Let J_ℓ denote the number of filters in the layer ℓ, where $\ell \in \mathbf{L}$, and $\mathbf{L}$ is the number of layers in the ConvNet. Let $\boldsymbol{W}_\ell$ be a matrix, such that the individual rows are the flattened filters (f) of the layer ℓ.

Let $\hat{\boldsymbol{W}}_{\boldsymbol{\ell}} = \boldsymbol{W}_{\boldsymbol{\ell}}/||\boldsymbol{W}_{\boldsymbol{\ell}}||$ denote the normalized weights. Then, the measure of Orthogonality for filter f in a layer ℓ (denoted by O_ℓ^f) is computed as shown in the equations below.

$$\boldsymbol{P}_\ell = |\hat{\boldsymbol{W}}_{\boldsymbol{\ell}} \times \hat{\boldsymbol{W}}_{\boldsymbol{\ell}}^T - I| \tag{5.1}$$

$$O_\ell^f = \frac{\sum \boldsymbol{P}_{\boldsymbol{\ell}}[f]}{J_\ell} \tag{5.2}$$

$\boldsymbol{P}_\ell$ is a matrix of size $J_\ell \times J_\ell$ and $\boldsymbol{P}[i]$ denotes i^{th} row of $\boldsymbol{P}$. Off-diagonal elements of a row of $\boldsymbol{P}$ for a filter f denote projection of all the other filters in the same layer with f. The sum of a row is minimum when other filters are orthogonal to this given filter. We rank the filters least important (thus subject to pruning) if this value is largest among all the filters in the network. While we compute the metric for a filter over a single layer, the ranking is computed over all the filters in the network. We do not enforce per layer rank because that would require learning a hyper-parameter $p_{\%}$ for every layer and some layers are more sensitive than others. Our method prunes more filters from deeper layers compared to the earlier layers. This is in accordance with the distribution of contribution of each filter in a given network (Figure 5.2 right).

Computation of our metric does not require expensive calculations of the inverse of Hessian [LDS89] or the second order derivatives [HS92] and is feasible for any sized networks. The most expensive calculations are L matrix products of size $J_\ell \times J_\ell$, but GPUs are designed for fast matrix-multiplications. Still, our method is more expensive than computing norm of the weights or the activations or the Average Percentage of Zeros (APoZ).

Given the choice of Orthogonality of filters, an obvious question would be to ask if adding a soft penalty to the loss function improve this training? A few researchers [Bro+17; PSDG13; XPX17] have reported marginal improvements due to added regularization in the ConvNets used for task-specific models. We experimented by adding $\lambda * \sum_\ell \boldsymbol{P}_\ell$ to the loss function, but we did not see any improvement. Soft regularization penalizes all the filters and changes the loss surface to encourage random orthogonality in the weights without improving expressiveness.

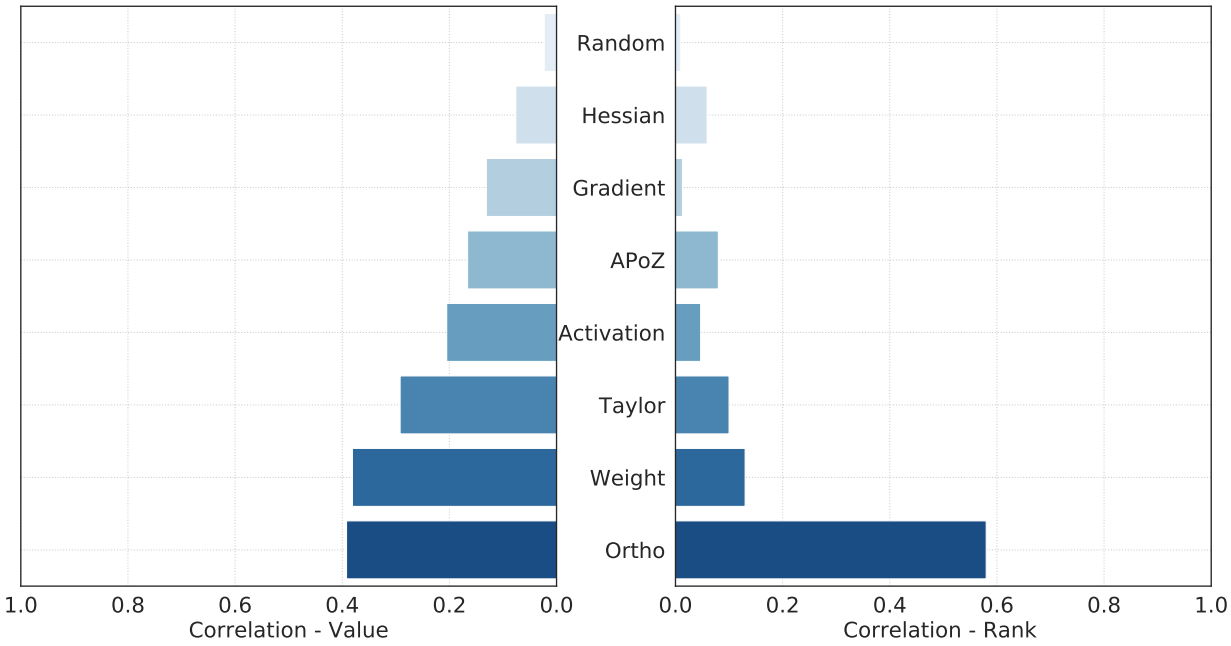


Figure 5.3: Left: Pearson correlation coefficient of various metric values with the accuracy values from greedy Oracle. Center: Pearson correlation coefficient of filter ranks using various metric with rank from greedy Oracle

CIFAR-10 - $C^3(32)$	
Standard	72.1
Random	73.4
Activations	74.1
APoZ [Hu+16]	74.3
Gradients [LDS89]	74.3
Taylor [Mol+17]	74.3
Hessian [LDS89]	74.4
Weights [HMD16]	74.6
Oracle	76.0
Ortho	**76.4**

Table 5.1: Test accuracy on CIFAR-10 using standard training and RePr training with various metrics

5.6 Ablation study

There are some hyper-parameters and function choices in our scheme. We present a short study of the parameters in our scheme.

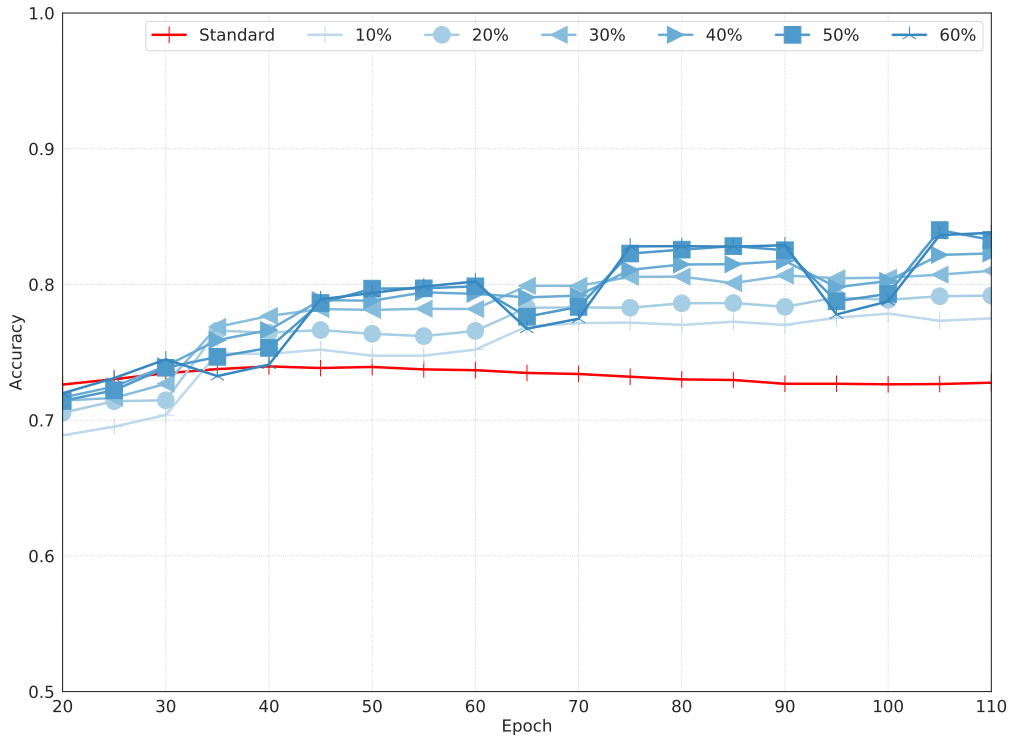


Figure 5.4: RePr training with various percentage of filters pruned. Shows average test accuracy over 5 epochs starting from epoch 20 for better visibility.

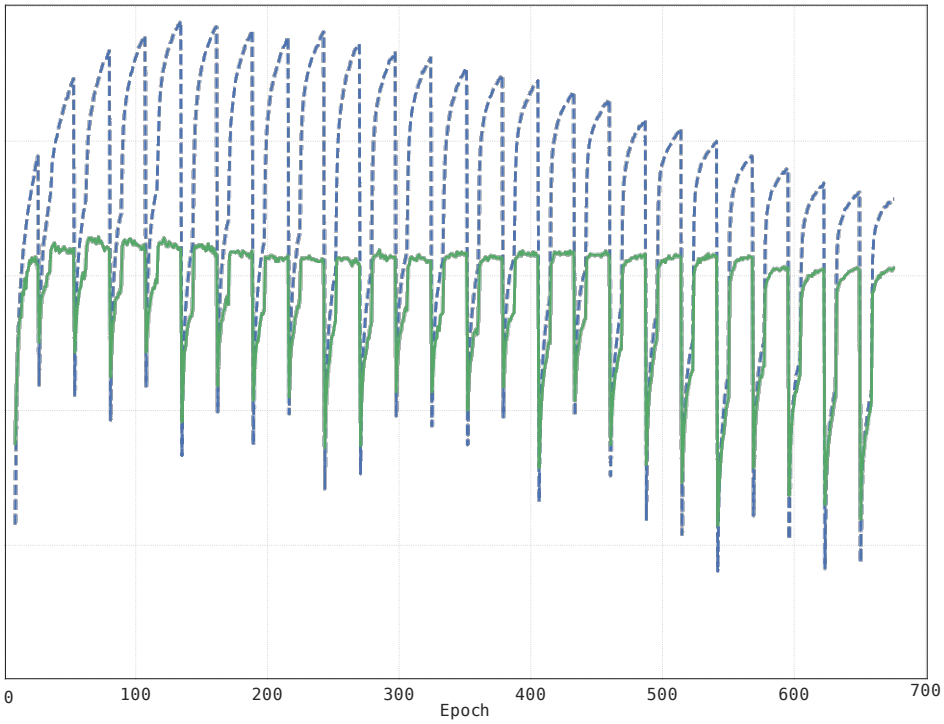


Figure 5.5: Marginal returns of multiple iterations of RePR - Training and Test accuracy on CIFAR-10

5.6.1 Comparison of pruning criteria

We measure the correlation of our metric with the Oracle to answer the question - how good a substitute is our metric for the filter importance ranking. Pearson correlation of our metric, henceforth referred to as Ortho, with the Oracle is 0.38. This is not a strong correlation, however, when we compare this with other known metrics, it is the closest. Molchanov *et al.* [Mol+17] report Spearman correlation of their criteria (Taylor) with greedy Oracle at 0.73. We observed similar numbers for Taylor ranking during the early epochs but the correlation diminished significantly as the models converged. This is due to low gradient value from filters that have converged. The Taylor metric is a product of the activation and the gradient. High gradients correlate with important filters during early phases of learning

but when models converge low gradient do not necessarily mean less salient weights. It could be that the filter has already converged to a useful feature that is not contributing to the overall error of the model or is stuck at a saddle point. With the norm of activations, the relationship is reversed. Thus by multiplying the terms together hope is to achieve a balance. But our experiments show that in a fully converged model, low gradients dominate high activations. Therefore, the Taylor term will have lower values as the models converge and will no longer be correlated with the inefficient filters. While the correlation of the values denotes how well the metric is the substitute for predicting the accuracy, it is more important to measure the correlation of the rank of the filters. Correlation of the values and the rank may not be the same, and the correlation with the rank is the more meaningful measurement to determine the weaker filters. Ortho has a correlation of 0.58 against the Oracle when measured over the rank of the filters. Other metrics show very poor correlation using the rank. Figure 5.3 (Left and Center) shows the correlation plot for various metrics with the Oracle. The table on the right of Figure 5.3 presents the test accuracy on CIFAR-10 of various ranking metrics. From the table, it is evident that Orthogonality ranking leads to a significant boost of accuracy compared to standard training and other ranking criteria.

5.6.2 Percentage of filters pruned

One of the key factors in our training scheme is the percentage of the filters to prune at each pruning phase ($p_{\%}$). It behaves like the Dropout parameter, and impacts the training time and generalization ability of the model *(see Figure: 5.4)*. In general the higher the pruned percentage, the better the performance. However, beyond 30%, the performances are not significant. Up to 50%, the model seems to recover from the dropping of filters. Beyond that, the training is not stable, and sometimes the model fails to converge.

5.6.3 Number of RePr iterations

Our experiments suggest that each repeat of the RePr process has diminishing returns, and therefore should be limited to a single-digit number (see Figure 5.4 (Right)). Similar to Dense-Sparse-Dense [Han+16] and Born-Again-Networks [Fur+18], we observe that for most networks, two to three iterations is sufficient to achieve the maximum benefit.

5.6.4 Optimizer and S1/S2

Figure 5.6 (left) shows variance in improvement when using different optimizers. Our model works well with most well-known optimizers. Adam and Momentum perform better than SGD due to their added stability in training. We experimented with various values of $S1$ and $S2$, and there is not much difference if either of them is large enough for the model to converge temporarily.

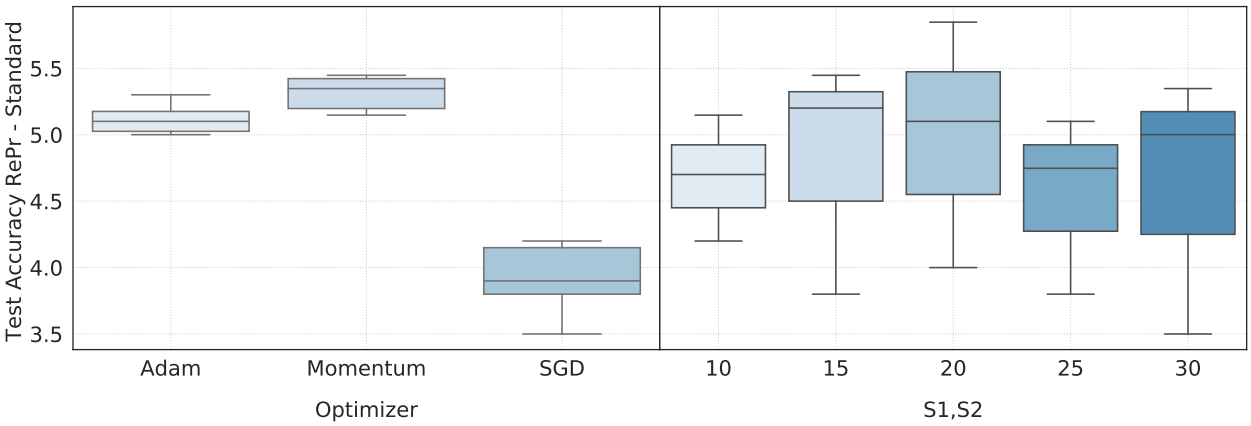


Figure 5.6: Left: Impact of using various optimizers on RePr training scheme. Right: Results from using different S1/S2 values. For clarity, these experiments only shows results with $S1 = S2$

5.6.5 Learning Rate Schedules

SGD with a fixed learning rate does not typically produce optimal model performance. Instead, gradually annealing the learning rate over the course of training is known to produce models with higher test accuracy. State-of-the-art results on ResNet, DenseNet, Inception were all reported with a predetermined learning rate schedule. However, the selection of the exact learning rate schedule is itself a hyperparameter, one which needs to be specifically tuned for each model. Cyclical learning rates [Smi17] can provide stronger performance without exhaustive tuning of a precise learning rate schedule. Figure 5.7 shows the comparison of our training technique when applied in conjunction with fixed schedule learning rate scheme and cyclical learning rate. Our training scheme is not impacted by using these schemes, and improvements over standard training is still apparent.

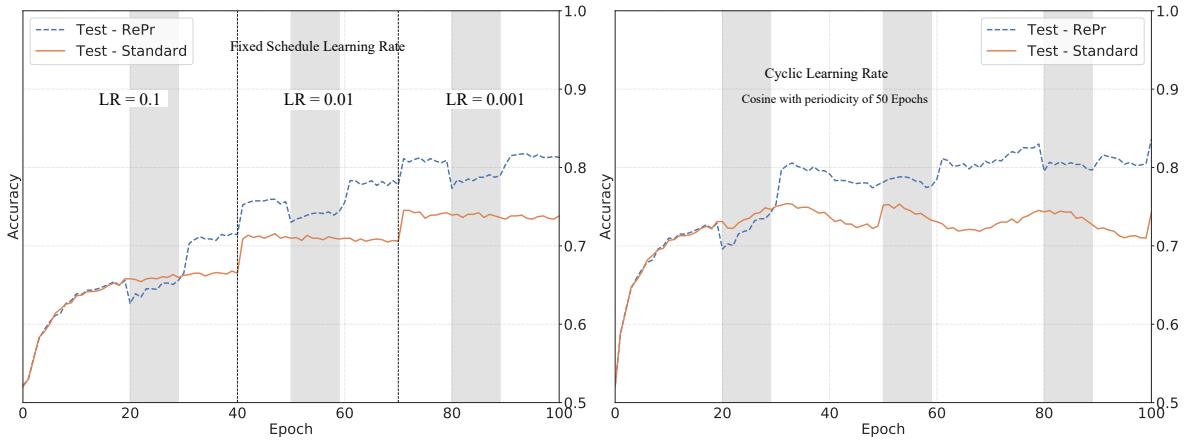


Figure 5.7: Test accuracy of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme and our method - RePr on CIFAR-10. The shaded regions denote periods when only part of the network is trained for RePr. Left: Fixed Learning Rate schedule of 0.1, 0.01 and 0.001.Right: Cyclic Learning Rate with periodicity of 50 Epochs, and amplitude of 0.005 and starting LR of 0.001.

5.6.6 Impact of Dropout

Dropout, while commonly applied in Multilayer Perceptrons, is typically not used for ConvNets. Our technique can be viewed as a type of non-random Dropout, specifically applicable to ConvNets. Unlike standard Dropout, out method acts on entire filters, rather than individual weights, and is applied only during select stages of training, rather than in every training step. Dropout prevents overfitting by encouraging co-adaptation of weights. This is effective in the case of over-parameterized models, but in compact or shallow models, Dropout may needlessly reduce already limited model capacity.

Figure 5.8 shows the performance of Standard Training and our proposed method (RePr) with and without Dropout on a three-layer convolutional neural network with 32 filters each. Dropout was applied with a probability of 0.5. We observe that the inclusion of Dropout lowers the final test accuracy, due to the effective reduction of the model's capacity by half. Our method produces improved performance with or without the addition of standard Dropout, demonstrating that its effects are distinct from the benefits of Dropout.

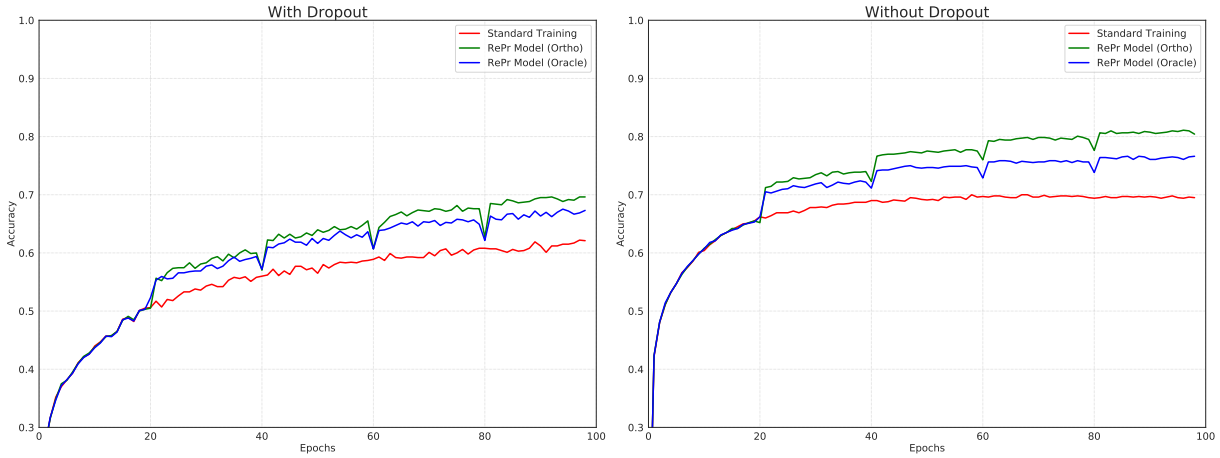


Figure 5.8: Test accuracy of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme, RePr with Oracle and RePr with Ortho on CIFAR-10. Left: With Dropout of 0.5. Right: No Dropout

Orthogonal Loss - OL Adding Orthogonality of filters (equation 1) as a regularization term as a part of the optimization loss does not significantly impact the performance of the model. Thus, the loss function will be -

$$\mathcal{L} = \text{Cross entropy} + \lambda * |\hat{\boldsymbol{W}}_\ell \times \hat{\boldsymbol{W}}_\ell^T - I|$$

where, λ is a hyper-parameter which balances both the cost terms. We experimented with various values of λ. Table 5.2 report the results with this loss term for the $\lambda = 0.01$, for which the validation accuracy was the highest. OL refers to addition of this loss term.

$C^3(32)$	**Std**	**Std+OL**	**RePr**	**RePr+OL**
CIFAR-10	72.1	72.8	76.4	76.7
CIFAR-100	47.2	48.3	58.2	58.6

Table 5.2: Comparison of addition of Orthogonality loss to Standard Training and RePr

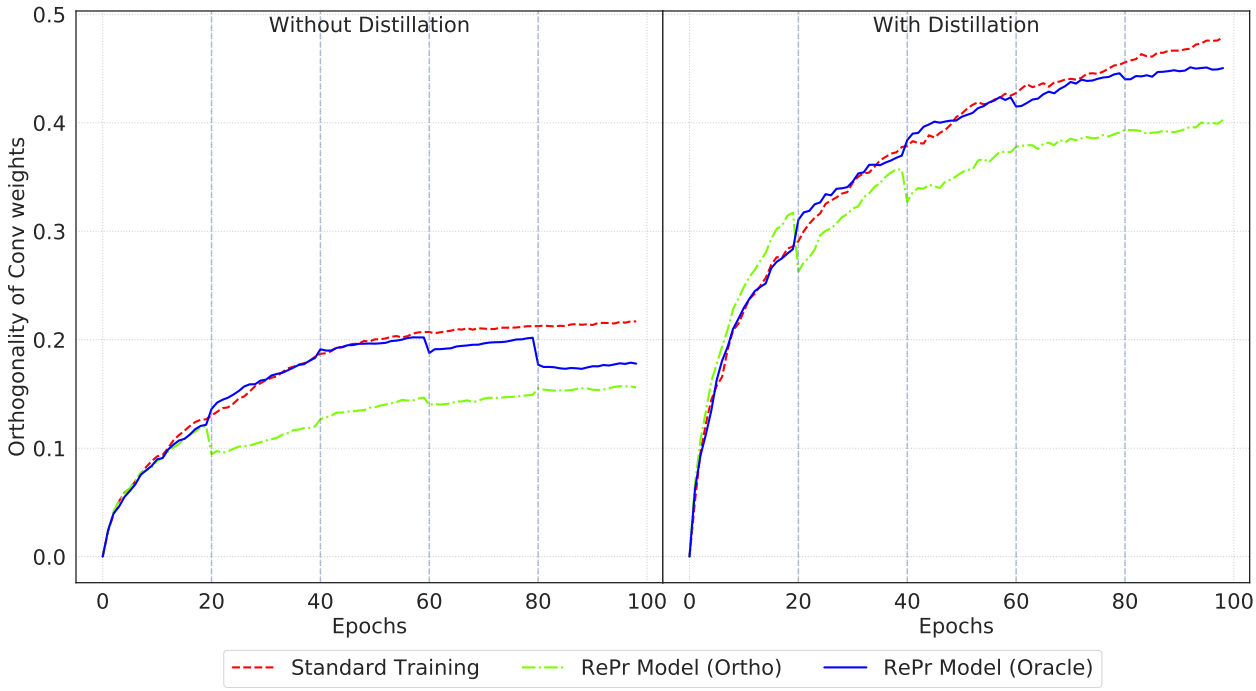


Figure 5.9: Comparison of orthogonality of filters (Ortho-sum - eq 2) in standard training and RePr training with and without Knowledge Distillation. **Lower value** signifies less overlapping filters. Dashed vertical lines denotes filter dropping.

$C^3(32)$	**Std**	**KD**	**RePr**	**KD+RePr**
CIFAR-10	72.1	74.8	76.4	**83.1**
CIFAR-100	47.2	56.5	58.2	**64.1**

Table 5.3: Comparison of Knowledge Distillation with RePr.

5.7 Orthogonality and Distillation

Our method, RePr and Knowledge Distillation (KD) are both techniques to improve performance of compact models. RePr reduces the overlap of filter representations and KD distills the information from a larger network. We present a brief comparison of the techniques and

show that they can be combined to achieve even better performance.

RePr repetitively drops the filters with most overlap in the directions of the weights using the *inter*-filter orthogonality, as shown in the equation 5.2. Therefore, we expect this value to gradually reduce over time during training. Figure 5.9 (left) shows the sum of this value over the entire network with three training schemes. Base Model is the standard training which does not involve our RePr scheme. We show RePr with two different filter ranking criteria - Ortho and Oracle. It is not surprising that RePr training scheme with Ortho ranking has lowest Ortho sum but it is surprising that RePr training with Oracle ranking also reduces the filter overlap, compared to the standard training. Once the model starts to converge, the least important filters based on Oracle ranking are the ones with the most overlap. And dropping these filters leads to better test accuracy *(table on the right of Figure 5.3)*. Does this improvement come from the same source as the that due to Knowledge Distillation? Knowledge Distillation (KD) is a well-proven methodology to train compact models. Using soft logits from the teacher and the ground truth signal the model converges to better optima compared to standard training. If we apply KD to the same three experiments (see Figure 5.9, right), we see that all the models have significantly larger Ortho sum. Even the RePr (Ortho) model struggles to lower the sum as the model is strongly guided to converge to a specific solution. This suggests that this improvement due to KD is not due to reducing filter overlap. Therefore, a model which uses both the techniques should benefit by even better generalization. Indeed, that is the case as the combined model has significantly better performance than either of the individual models, as shown in Table 5.3.

5.8 Results

5.8.1 Image Classification

We present the performance of our training scheme, RePr, with our ranking criteria, *inter*-filter orthogonality, Ortho, on different ConvNets [SZ14; He+16b; Sze+15; Sze+16; Hua+17]. For all the results provided RePr parameters are: $S_1 = 20$, $S_2 = 10$, $p_{\%} = 30$, and with three iterations, $N = 3$.

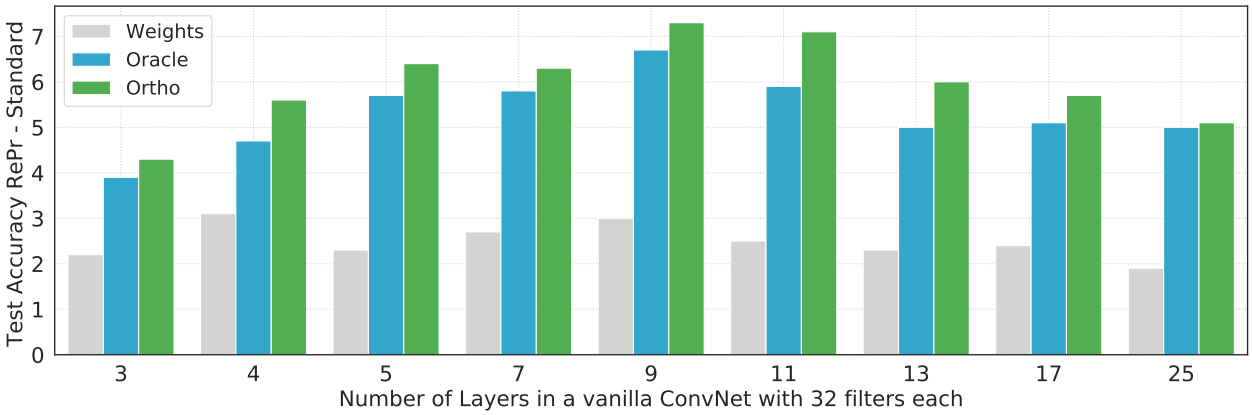


Figure 5.10: Accuracy improvement using RePr over standard training on Vanilla ConvNets across many layered networks $[C^n(32)]$

ResNet-20 on CIFAR-10					
Baseline		Various Training Schemes			
Original [He+16b]	**Our Impl**	**DSD [Han+16]**	**BAN [Fur+18]**	**RePr Weights**	**RePr Ortho**
8.7	8.4	7.8	8.2	7.7	6.9

Table 5.4: Comparison of test error from using various techniques.

We compare our training scheme with other similar schemes like BAN and DSD in table 5.4. All three schemes were trained for three iterations *i.e.* N=3. All models were trained for 150 epochs with similar learning rate schedule and initialization. DSD and RePr (Weights) perform roughly the same function - sparsifying the model guided by magnitude, with the difference that DSD acts on individual weights, while RePr (Weights) acts on entire filters. Thus, we observe similar performance between these techniques. RePr (Ortho) outperforms the other techniques and is significantly cheaper to train compared to BAN, which requires N full training cycles.

Compared to modern architectures, vanilla ConvNets show significantly more inefficiency in the allocation of their feature representations. Thus, we find larger improvements from our method when applied to vanilla ConvNets, as compared to modern architectures. Table 5.5 shows test errors on CIFAR 10 & 100. Vanilla CNNs with 32 filters each have high error compared to DenseNet or ResNet but their inference time is significantly faster. RePr training improves the relative accuracy of vanilla CNNs by 8% on CIFAR-10 and 25% on CIFAR-100. The performance of baseline DenseNet and ResNet models is still better than vanilla CNNs trained with RePr, but these models incur more than twice the inference cost. For comparison, we also consider a reduced DenseNet model with only 5 layers, which has similar inference time to the 3-layer vanilla ConvNet. This model has many fewer parameters (by a factor of 11×) than the vanilla ConvNet, leading to significantly higher error rates, but we choose to equalize inference time rather than parameter count, due to the importance of inference time in many practical applications. Figure 5.10 shows more results on vanilla CNNs with varying depth. Vanilla CNNs start to overfit the data, as most filters converge to similar representation. Our training scheme forces them to be different which reduces the overfitting (Figure 5.4 - right). This is evident in the larger test error of 18-layer vanilla CNN with CIFAR-10 compared to 3-layer CNN. With RePr training, 18 layer model shows

lower test error.

RePr is also able to improve the performance of ResNet and shallow DenseNet. This improvement is larger on CIFAR-100, which is a 100 class classification and thus is a harder task and requires more specialized filters. Similarly, our training scheme shows bigger relative improvement on ImageNet, a 1000 way classification problem. Table 5.6 presents top-1 test error on ImageNet [Rus+15] of various ConvNets trained using standard training and with RePr. RePr was applied three times (N=3), and the table shows errors after each round. We have attempted to replicate the results of the known models as closely as possible with suggested hyper-parameters and are within $\pm 1\%$ of the reported results. More details of the training and hyper-parameters are provided in the supplementary material. Each subsequent RePr leads to improved performance with significantly diminishing returns. Improvement is more distinct in architectures which do not have skip connections, like Inception v1 and VGG and have lower baseline performance.

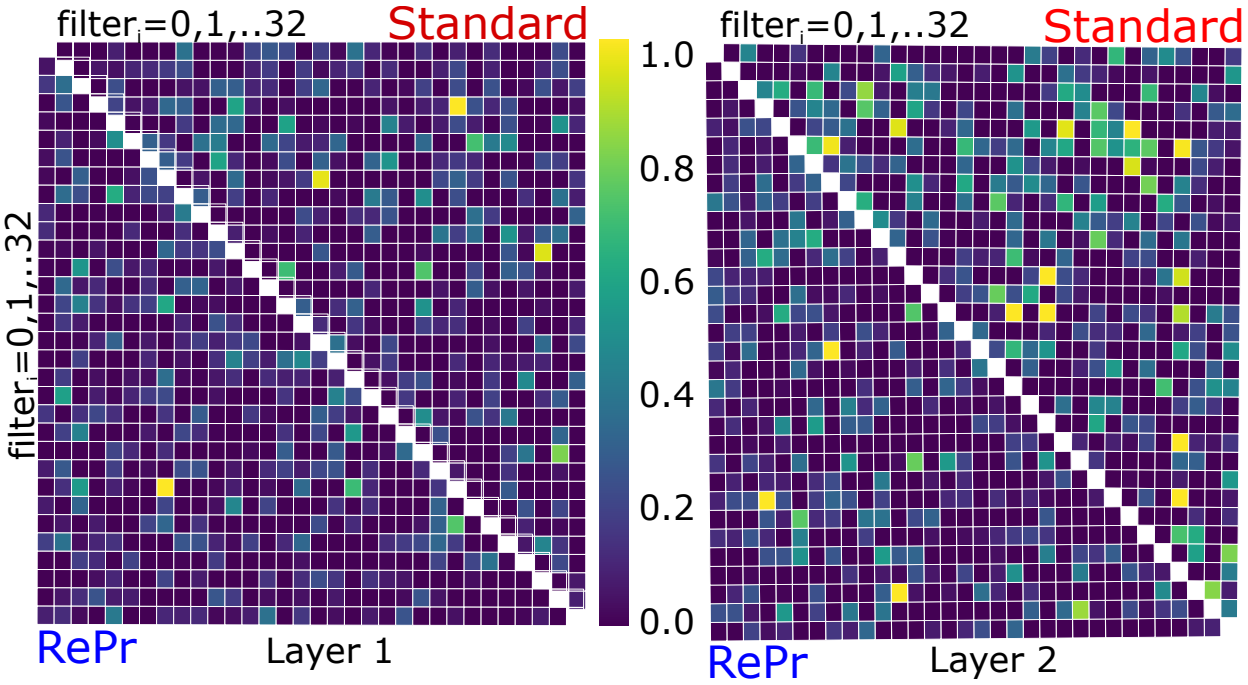


Figure 5.11: Canonical Correlation Analysis of activations from the 1st and 2nd layer of a C2(32) ConvNet trained on CIFAR-10.

Figure 5.11) provides a side-by-side comparison of correlation of activations on the layers of a two-layer ConvNet when trained with standard training scheme and when trained with RePr. While RePr is not able to remove all the high correlation filters, it significantly reduces the overlap of feature representation. For the first layer the differences are less pronounced due to less overlapping filters in the standard training.

			CIFAR-10		CIFAR-100	
Layers	Params (×000)	Inf. Time (relative)	Std	RePr	Std	RePr
Vanilla CNN [32 filters / layer]						
3	20	1.0	27.9	23.6	52.8	41.8
8	66	1.7	26.8	19.5	50.9	36.8
13	113	2.5	26.6	20.6	51.0	37.9
18	159	3.3	28.2	22.5	51.9	39.5
DenseNet [k=12]						
5	1.7	0.9	39.4	36.2	43.5	40.9
40	1016	8.0	6.8	6.2	26.4	25.2
100	6968	43.9	5.3	5.6	22.2	22.1
ResNet						
20	269	1.7	8.4	6.9	32.6	31.1
32	464	2.2	7.4	6.1	31.4	30.1
110	1727	7.1	6.3	5.4	27.5	26.4
182	2894	11.7	5.6	5.1	26.0	25.3

Table 5.5: Comparison of test error on Cifar-10 & Cifar-100 of various ConvNets using Standard training vs RePr Training. Inf. Time shows the inference times for a single pass. All time measurements are relative to Vanilla CNN with three layers. Parameter count does not include the last fully-connected layer.

ImageNet					
	Standard	RePr Training			Relative
Model	Training	N=1	N=2	N=3	Change
ResNet-18	30.41	28.68	27.87	27.31	-11.35
ResNet-34	27.50	26.49	26.06	25.80	-6.59
ResNet-50	23.67	22.79	22.51	22.37	-5.81
ResNet-101	22.40	21.70	21.51	21.40	-4.67
ResNet-152	21.51	20.99	20.79	20.71	-3.86
VGG-16	31.30	27.76	26.45	25.50	-22.75
Inception v1	31.11	29.41	28.47	28.01	-11.07
Inception v2	27.60	27.05	26.95	26.80	-2.99

Table 5.6: Comparison of test error (Top-1) on ImageNet with different models at various stages of RePr. N=1, N=2, N=3 are results after each round of RePr.

Our model improves upon other computer vision tasks that use similar ConvNets. We present a small sample of results from visual question answering and object detection tasks. Both these tasks involve using ConvNets to extract features, and RePr improves their baseline results.

5.8.2 Visual Question Answering

In the domain of visual question answering (VQA), a model is provided with an image and question (as text) about that image, and must produce an answer to that question. Most of the models that solve this problem use standard ConvNets to extract image features and an LSTM network to extract text features. These features are then fed to a third model which learns to select the correct answer as a classification problem. State-of-the-art models use an attention layer and intricate mapping between features. We experimented with a more standard model where image features and language features are fed to a Multi-layer Perceptron with a softmax layer at the end that does 1000-way classification over candidate answers. Table 5.8 provides accuracy on VQAv1 using VQA-LSTM-CNN model [Ant+15a]. Results are reported for Open-Ended questions, which is a harder task compared to multiple-choice questions. We extract image features from Inception-v1, trained using standard training and with RePr (Ortho) training, and then feed these image features and the language embeddings (GloVe vectors) from the question, to a two layer fully connected network. Thus, the only difference between the two reported results 5.8 is the training methodology of Inception-v1. Figure 5.7 shows qualitative results on VQA. Even in cases where the top-1 choice does not change RePR is able to improve the confidence of the prediction.

	What are they playing with?		What animal is in the picture?
Standard	0.4 - Soccer ball 0.3 - Frisbee 0.2 - Soccer	Standard	0.6 - Donkey 0.3 - Dog 0.1 - No
RePr	0.7 - Frisbee 0.2 - Soccer ball 0.1 - Baseball	RePr	0.5 - Dog 0.3 - Donkey 0.1 - Cow

Table 5.7: Sample of differences in confidence of selected answers between Standard Trainin and RePr training

	All	**Yes/No**	**Other**	**Number**
Standard	60.3	81.4	47.6	37.2
RePr (Ortho)	64.6	83.4	54.5	37.2

Table 5.8: Comparison of Standard Training and RePr on VQA using VQA-LSTM-CNN model

5.8.3 Object Detection

For object detection, we experimented with Faster R-CNN using ResNet 50 and 101 pre-trained on ImageNet. We experimented with both Feature Pyramid Network and baseline RPN with *c4* conv layer. We use the model structure from Tensorpack [Wu+16b], which is able to reproduce the reported mAP scores. The model was trained on 'trainval35k + minival' split of COCO dataset (2014). Mean Average Precision (mAP) is calculated at ten IoU thresholds from 0.5 to 0.95. mAP for the boxes obtained with standard training and RePr training is shown in the table 5.9.

	ResNet-50		**ResNet-101**	
	RPN	FPN	RPN	FPN
Standard	38.1	38.2	40.7	41.7
RePr (Ortho)	41.1	42.3	43.5	44.5

Table 5.9: mAP scores with Standard and RePr (Ortho) training for object detection with ResNet the ConvNet (RPN on C4)

5.9 Conclusion

We have introduced RePr, a training paradigm which cyclically drops and relearns some percentage of the least expressive filters. After dropping these filters, the pruned sub-model is able to recapture the lost features using the remaining parameters, allowing a more robust and efficient allocation of model capacity once the filters are reintroduced. Our model parameterizes the frequency and timing of filter dropping, and demonstrate significant improvement from proper settings for these parameters. We show that a reduced model needs

training before re-introducing the filters, and careful selection of this training duration leads to substantial gains. We also demonstrate that this process can be repeated with diminishing returns. By choosing the frequency and the timing of dropping the filters, we demonstrate significant improvement in generalization ability over standard training. Unlike previous approaches, our model takes a parametric approach to the frequency and the timing of the drop. This has a significant impact on performance, as the sub-model is able to learn the dropped representation leading to better allocation of reintroduced filters.

Motivated by prior research which highlights inefficiencies in the feature representations learned by convolutional neural networks, we further introduce a novel *inter*-filter orthogonality metric for ranking filter importance for the purpose of RePr training, and demonstrate that this metric outperforms established ranking metrics. We show that instead of using stochastic dropping or well-known filter pruning criteria, our proposed *inter*-filter orthogonality ranking yields the best result. Our ranking is motivated by the research in the inefficiency of feature representation in convolutional neural networks. Our training technique significantly improves the performance of smaller and simpler ConvNets and is complementary to Knowledge Distillation. Our training method is able to significantly improve performance in under-parameterized networks by ensuring the efficient use of limited capacity, and the performance gains are complementary to knowledge distillation. Even in the case of complex, over-parameterized network architectures, our method is able to improve performance across a variety of tasks.

Chapter 6

Multimodal Highway Networks

6.1 Introduction

While it is a common understanding in the field that deeper networks are better[BC14]; in practice training a very deep network for the same size data is prone to over-fitting and vanishing gradients. Recently, some success has been observed by techniques which minimize the attenuation of the information by either learning to represent the residues[He+16b] or by adding gating units which reinforce the original signal[SGS15]. This allows successful training of network with thousands of layers.

For the task of visual question answering, we propose a network which alters the gating units to be feed by multi dimensional representation of the words from the question. This allows us to learn common feature space in a single end to end training. When the length of question is shorter than the number of layers, it can be either padded with zeros or by repeating the words from the question. We present our model and result from the VQA dataset[Ant+15b].

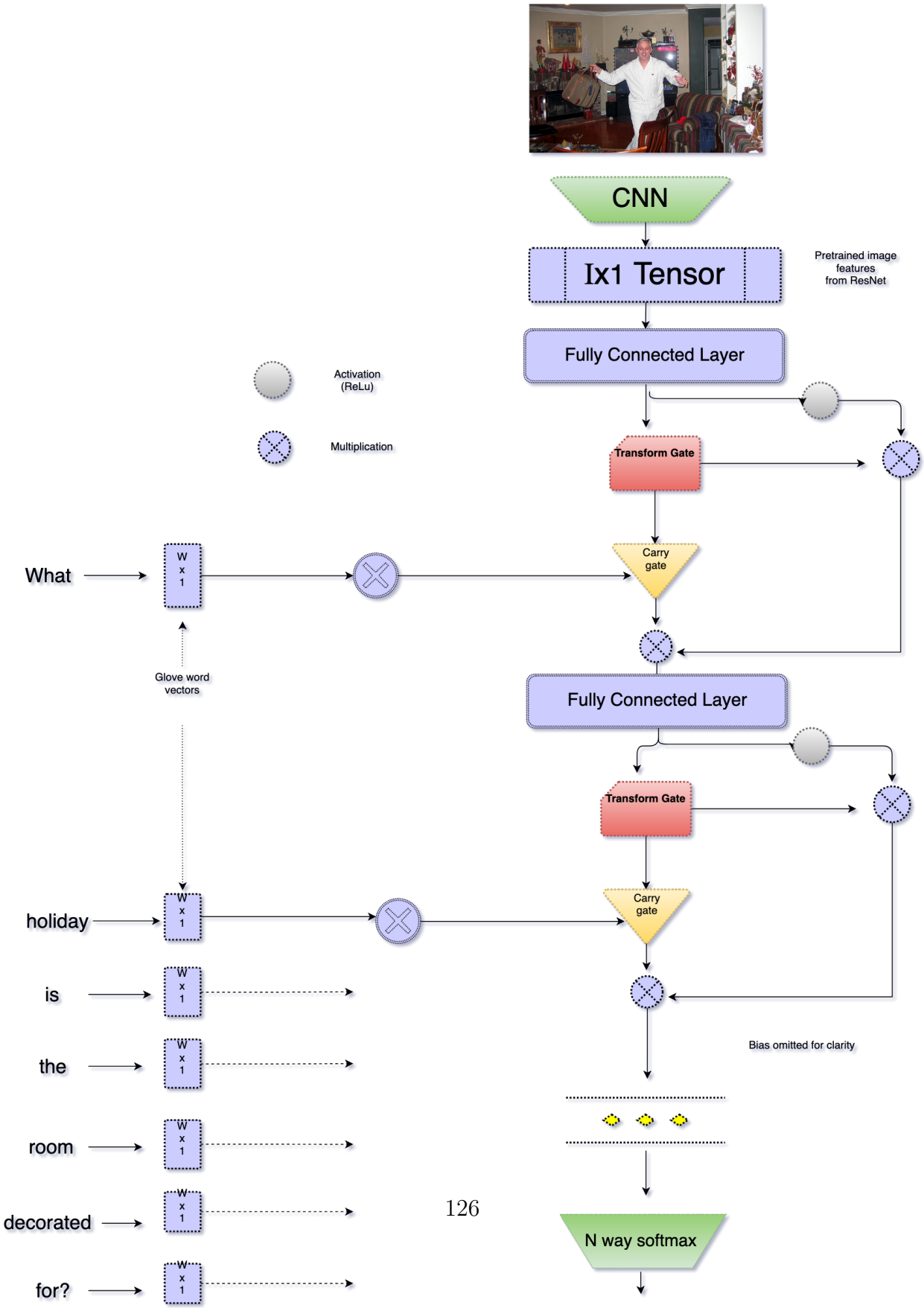


Figure 6.1: Architecture of multi-modal highway networks

6.1.1 Highway Networks

For the given input x, and weight matrix W, a feed forward network tries to learn the following representation –

$$y = H(x, W) + \mathbf{b}$$

where, H is a non-linear transformation and $\mathbf{b}$ is bias. Highway Network as described by Srivastava et al[SGS15], adds two gates to above equation, essentially transforming it to –

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_C)$$

where, T and C are *transform* gate and *carry* gate. It should be noted that x, y, H, T, and C should have same dimensionality.

6.1.2 Gating for Visual Question Answering

Since the original model was meant for a single learning task like image classification, the authors choose the carry gate $\boldsymbol{C = 1 - T}$, the same cannot be done for multimodal learning like visual question answering.

Our modification for the carry gate is to feed the word vectors of the words from the question, extracted from pre-trained models like Glove vectors[PSM14] . $\boldsymbol{K \times 1}$ features from word embeddings are translated to dimensions of '$\boldsymbol{x}$' by point-wise multiplication. At each layer vector from only one word or average of few words are feed. This allows us to learn temporal relationship in the words in question. However, for this task, individual words in any order seem to perform equally well.

6.1.3 Feature space

Most of the related work on visual question answering use some form of pre-trained image vector. All the models used are trained on ImageNet images, and the last layer before the softmax is extracted. It is common to use GoogLeNet [Jia+15a] [XS15] [Zho+15] [Wu+16a] or VGGNet [Wu+15] [XMS16] [Yan+15] [SSH15] [RKZ15] [And+16] [JLP15][Zhu+15] [NSH15]. We use ResNet[He+16b] for their ability to extract better features. Features obtained from VGGNet, GoogLeNet or ResNet are easily transferable to MS COCO Images and number of images in the VQA dataset are not big enough to train a independent image model. However, some of the models do fine-tuning[NSH15], but with our model that is redundant because the *transfer gate* alters the weights of image features effectively doing fine-tuning constrained upon carry gate weights.

6.1.4 Attention

Most of the recent models [XMS16] [XS15] [Yan+15] tackling the task of visual question answering have taken inspiration from success of soft attention models for image captioning[KFF15] [Xu+15]. This has led to some improvement in VQA but comes at a cost of slower training process due to recurrent network. While Noh et. al's [NSH15] use of parameter prediction to obtain attention from hashing weights into smaller dimension and Xiong et. al's[XMS16] use of episodic memory for attention avoids recurrence but are limited by size of learnable parameters. Our model achieves the effect of attention by learning the parameters of the gate. And thus it can be extended by adding more layers. *Carry gate* changes the network structure based on the improvement obtained from merging the weights from current word vector. In a control experiment where question words were replaced by random words, the network performance was diminished significantly.

6.1.5 Bi-directional Recurrence

As an experimental model we appended with input questions in reverse order. This improved the accuracy of the model only slightly but the computational cost was doubled and training took twice the time. Thus we believe there is not much gain in bi-directional recurrence as far as visual question answering is concerned. Similar results were reported for bi-directional LSTM by Ren et al [RKZ15].

6.2 Results

Table 6.1: Results of VQAv1

	Open Ended					MC
	Test-Dev				Test-Std	
Method	**All**	**Y/N**	**Other**	**Num**	**All**	**All**
Image only	28.1	64.0	3.8	0.4	-	30.5
Question only	48.1	75.7	27.1	36.7	-	53.6
Q+I	52.6	75.6	37.4	33.7	-	58.9
LSTM Q+I	53.7	78.9	36.4	35.2	54.1	57.1
CMV [Jia+15a]	52.6	78.3	35.9	34.4	-	-
AMA [Wu+15]	55.7	79.2	40.1	36.1	56.0	-
iBOW [Zho+15]	55.7	76.5	42.6	35.0	55.9	61.9
DPPNet [NSH15]	57.2	80.7	41.7	37.2	57.4	62.6
LCN [And+16]	57.9	80.5	43.1	37.4	58.0	-
AAA [XS15]	57.9	80.8	43.2	37.3	58.2	-
dLSTM+ [JLP15]	57.7	80.5	43.0	36.7	58.1	63.0
SAN [Yan+15]	58.7	79.3	46.1	36.6	58.9	-
DMN+ [XMS16]	60.3	80.5	48.3	36.8	60.4	-
OUR	**60.4**	**81.5**	**47.6**	**37.2**	**60.7**	**65.0**

Chapter 7

Residual LSTM

7.1 Introduction

Paraphrasing, the act of using textual alternatives to express the same meaning as the source content, is an important subtask in various Natural Language Processing (NLP) applications such as question answering, information extraction, information retrieval, summarization and natural language generation. Research on paraphrasing methods typically aims at solving three related problems: (1) recognition (i.e. to identify if two textual units are paraphrases of each other), (2) extraction (i.e. to extract paraphrase instances from a thesaurus or a corpus), and (3) generation (i.e. to generate a reference paraphrase given a source textual unit) [MD10]. In this paper, we focus on the paraphrase generation problem.

Paraphrase generation has been used to gain performance improvements in several NLP applications, for example, by generating query variants or pattern alternatives for information retrieval, information extraction or question answering systems, by creating reference paraphrases for automatic evaluation of machine translation and document summarization systems, and by generating concise or simplified information for sentence compression or

sentence simplification systems [MD10].

Traditional paraphrase generation methods exploit hand-crafted rules [McK83] or automatically learned complex paraphrase patterns [Zha+09], use thesaurus-based [Has+07] or semantic analysis driven natural language generation approaches [KMVS03], or leverage statistical machine learning theory [QBD04; WBK10]. In this paper, we propose to use deep learning principles to address the paraphrase generation problem. Recently, techniques like sequence to sequence learning [SVL14] have been applied to various NLP tasks with promising results, for example, in the areas of machine translation [Cho+14; BCB15], speech recognition [LW15], language modeling [Vin+15], and dialogue systems [Ser+16]. Although paraphrase generation can be formulated as a sequence to sequence learning task, not much work has been done in this area with regard to applications of state-of-the-art deep neural networks. There are several works on paraphrase recognition [Soc+11; YS15; Kir+15], but those employ classification techniques and do not attempt to generate paraphrases. Another recent related work uses attention-based Long Short-Term Memory (LSTM) networks for textual entailment generation [KRR16]; however, paraphrase generation is a type of bidirectional textual entailment generation and no prior work has proposed a deep learning-based formulation of this task. In the light of the lack of such prior work, we explore various types of sequence to sequence models for paraphrase generation. We test these models on three different datasets and evaluate these models using various metrics. Along with the application of various existing sequence to sequence models for the paraphrase generation task, in this paper we also propose a new model that allows for training multiple stacked LSTM networks by introducing a residual connection between the layers. This is inspired by the recent success of such connections in a deep Convolutional Neural Network (CNN) for the image recognition task [He+16b]. Our experiments demonstrate that the proposed model can outperform various other techniques we have explored. Most of the deep learning mod-

els in NLP use Recurrent Neural Networks (RNNs). RNNs differ from normal perceptrons as they allow gradient propagation in time to model sequential data with variable-length input and output [SMH11]. In practice, RNNs often suffer from the vanishing/exploding gradient problems while learning long-range dependencies [BSF94]. LSTM [HS97] and GRU [Cho+14] are known to be successful remedies to these problems. It has been observed that increasing the depth of a deep neural network can improve the performance of the model [SZ14; He+16b] as deeper networks learn better representations of features [Far+13]. In the vision related tasks where CNNs are more widely used, adding many layers of neurons is a common practice. For tasks like speech recognition [LW15] and also in machine translation, it is useful to stack layers of LSTM or any other variants of RNN. So far this has been limited to only a few layers due to difficulty in training deep RNN networks. We propose to add residual connections between multiple stacked LSTM networks and show that this allows us to stack more layers of LSTM successfully.

The rest of the paper is organized as follows: Section 2 presents a brief overview of the sequence to sequence models followed by a description of our proposed residual deep LSTM model, Section 3 describes the datasets used in this work, Section 4 explains the experimental setup, Section 5 presents the evaluation results and analyses, Section 6 discusses the related work, and finally, in Section 7 we conclude and discuss future work.

7.2 Model Description

7.2.1 Encoder-Decoder Model

A neural approach to sequence to sequence modeling proposed by Sutskever et al. [SVL14] is a two-component model, where a source sequence is first encoded into some low dimensional

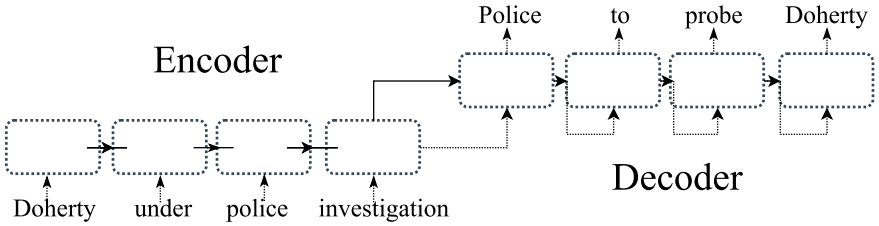


Figure 7.1: Encoder-Decoder framework for sequence to sequence learning

representation (Figure 7.1) that is later used to reproduce the sequence back to a high dimensional target sequence (i.e. decoding). In machine translation, an encoder operates on text written in the source language and encodes the meaning of the sentence to be translated before the decoder can take that vector (which represents the meaning) and generate a sentence in the target language. These encoder-decoder blocks can be either a vanilla RNN or its variants. While producing the target sequence, the generation of each new word depends on the model and the last word generated. For the first word, this is done by appending a special '`EOS`' (end-of-sentence) token to the source. The training objective is to maximize the log probability of the target sequence given the source sequence. Therefore, the best possible decoded target is the one that has the maximum score over the length of the sequence. To find this, a small set of hypotheses (candidate set) called *beam size* is used and the total score for all these hypotheses are computed. In the original work by Sutskever et al. [SVL14], they observe that even beam size of **1** gives a very good result but a higher beam size always does better. This is because for some of the hypotheses the first word may not have the highest score.

7.2.2 Deep LSTM

LSTM (Figure 7.2) is a RNN which adds an internal memory cell $\boldsymbol{c_t} \in \mathbb{R}^{\boldsymbol{n}}$ at every time step. An LSTM unit takes three inputs $\boldsymbol{x_t}, \boldsymbol{h_{t-1}}, \boldsymbol{c_{t-1}}$ at every time step and produces the hidden

state, $\boldsymbol{h_t}$ and the internal memory state, $\boldsymbol{c_t}$ at time step $\boldsymbol{t}$. The memory cell is controlled via three learned gates: input $\boldsymbol{i}$, forget $\boldsymbol{f}$, and output $\boldsymbol{o}$. These memory cells use the addition of gradient with respect to time and thus minimize the gradient explosion. In most NLP tasks, LSTM outperforms vanilla RNN [SSN12]. Therefore, for our model we only explore LSTM as a basic unit in the encoder and decoder. Here, we describe the basic computations in an LSTM unit, which will provide the grounding to understand the residual connections between stacked LSTM layers later. In the equations below, $\boldsymbol{W_{x\_}}, \boldsymbol{W_{h\_}}$ are the learned parameters for $\boldsymbol{x}$ and $\boldsymbol{h}$ respectively. $\boldsymbol{\sigma(.)}$ and $\boldsymbol{\tanh(.)}$ denote element-wise sigmoid and hyperbolic tangent functions. $\odot$ is the element-wise multiplication operator and $\boldsymbol{b}$ denotes the added bias.

1. Gates

   $$\boldsymbol{i_t} = \boldsymbol{\sigma}(\boldsymbol{W_{xi}x_t} + \boldsymbol{W_{hi}h_{t-1}} + \boldsymbol{b_i})$$
   $$\boldsymbol{f_t} = \boldsymbol{\sigma}(\boldsymbol{W_{xf}x_t} + \boldsymbol{W_{hf}h_{t-1}} + \boldsymbol{b_f})$$
   $$\boldsymbol{o_t} = \boldsymbol{\sigma}(\boldsymbol{W_{xo}x_t} + \boldsymbol{W_{ho}h_{t-1}} + \boldsymbol{b_o})$$

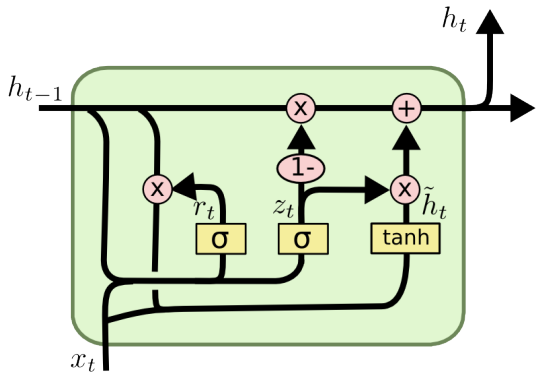
2. Input transform

   $$\boldsymbol{c\_in_t} = \tanh(\boldsymbol{W_{xc}x_t} + \boldsymbol{W_{hc}h_{t-1}} + \boldsymbol{b_{c\_in}})$$

3. State Update

   $$\boldsymbol{c_t} = \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \odot \boldsymbol{c\_in_t}$$
   $$\boldsymbol{h_t} = \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t})$$

Graves [Gra13] explored the advantages of deep LSTMs for handwriting recognition and text generation. There are multiple ways of combining one layer of LSTM with another. For example, Pascanu et al. [Pas+13] explored multiple ways of combining them and discussed various difficulties in training deep LSTMs. In this work, we employ vertical stacking where



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 7.2: LSTM Cell. Fig reproduced with permission from Chris Olah

only the output of the previous layer of LSTM is fed to the input, as compared to the stacking technique used by Sutskever et al. [SVL14], where hidden states of all LSTM layers are fully connected. In our model, all but the first layer input at time step $\boldsymbol{t}$ is passed from the hidden state of the previous layer $\boldsymbol{h_t^l}$, where $\boldsymbol{l}$ denotes the layer. This is similar to stacked RNN proposed by Bengio et al. [BSF94] but with LSTM units. Thus, for a layer $\boldsymbol{l}$ the activation is described by:

$$\boldsymbol{h_t^{(l)} = f_h^l(h_t^{(l-1)}, h_{t-1}^{(l)})}$$

where hidden states $\boldsymbol{h}$ are recursively computed and $\boldsymbol{h_t^{(l)}}$ at $\boldsymbol{t = 0}$ and $\boldsymbol{l = 0}$ is given by the LSTM equation of $\boldsymbol{h_t}$.

7.2.3 Stacked Residual LSTM

We take inspiration from a very successful deep learning network *ResNet* [He+16b] with regard to adding residue for the purpose of learning. With theoretical and empirical reasoning, He et al. [He+16b] have shown that the explicit addition of the residue $\boldsymbol{x}$ to the function being learned allows for deeper network training without overfitting the data.

When stacking multiple layers of neurons, often the network suffers through a *degradation*

problem [He+16b]. The degradation problem arises due to the low convergence rate of training error and is different from the vanishing gradient problem. Residual connections can help overcome this issue. We experimented with four-layers of stacked LSTM for each of the model. Residue connections is added at layer two as the pointwise addition (see Figure 7.3), and thus it requires the input to be in the same dimension as the output of $\boldsymbol{h_t}$. Principally because of this reason, we use a simple last hidden unit stacking of LSTM instead of a more intricate way as shown by Sutskever et al. [SVL14]. This allowed us to clip the $\boldsymbol{h_t}$ to match the dimension of $\boldsymbol{x_{t-2}}$ where they were not the same. Similar results could be achieved by padding $\boldsymbol{x}$ to match the dimension instead. The function $\hat{\boldsymbol{h}}$ that is being learned for the layer with residue connection is therefore:

$$\hat{\boldsymbol{h}}_{\boldsymbol{t}}^{(\boldsymbol{l})} = \boldsymbol{f}_{\boldsymbol{h}}^{\boldsymbol{l}}(\boldsymbol{h}_{\boldsymbol{t}}^{(\boldsymbol{l-1})}, \boldsymbol{h}_{\boldsymbol{t-1}}^{(\boldsymbol{l})}) + \boldsymbol{x_{l-n}}$$

where $\hat{\boldsymbol{h}}$ for layer $\boldsymbol{l}$ is updated with residual value $\boldsymbol{x_{l-n}}$ and $\boldsymbol{x_i}$ represents the input to layer $\boldsymbol{i+1}$. Residual connection is added after every $\boldsymbol{n}$ layers. However, for stacked LSTM $\boldsymbol{n > 3}$ is very expensive in terms of computation. In this paper we experimented with $\boldsymbol{n = 2}$. Note that when $\boldsymbol{n = 1}$, the resulting function learned is a standard LSTM with bias that depends on the input $\boldsymbol{x}$. That is why, it is not necessary to add the residue connection after every stacked layer of LSTM. Addition of residual connection does not add any learnable parameters. Therefore, this does not increase the complexity of model unlike bi-directional models which doubles the number of LSTM units.

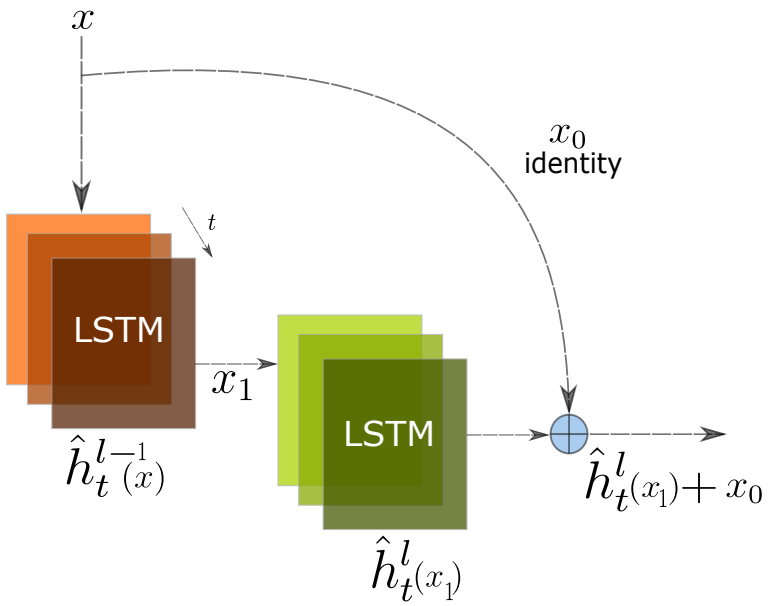


Figure 7.3: A unit of stacked residual LSTM

7.3 Datasets

We present the performance of our model on three datasets, which are significantly different in their characteristics. **PPDB** [Pav+15] is a well known dataset used for various NLP tasks. It comes in different sizes and the precision of the paraphrases degrades with the size of the dataset. We use the size ***L*** dataset from PPDB 2.0, which comes with over **18*M*** paraphrases including lexical, phrasal and syntactic types. We have omitted the syntactic paraphrases and the instances which contain numbers, as they increase the vocabulary size significantly without giving any advantage of a larger dataset. This dataset contains relatively short paraphrases (**86%** of the data is less than four words), which makes it suitable for synonym generation and phrase substitution to address lexical and phrasal paraphrasing [MD10]. For some phrases, PPDB has one-to-many paraphrases. All such phrases were collected to make a set of paraphrases and sampling without replacement was used to obtain the source and

reference phrases.

WikiAnswers [FZE13] is a large question paraphrase corpus created by crawling the WikiAnswers website, where users can post questions and answers about any topic. The paraphrases are different questions, which were tagged by the users as similar questions. The dataset contains approximately 18M word-aligned question pairs. There is some loss of specialization between the given source question (a paraphrase to be tagged as similar to the reference question) and the reference question. For example, "prepare a *three month* cash budget" is tagged to "how to prepare a cash budget". This happens because it is more likely that a general question has been answered as they are more popular and specific ones are redirected to the general ones due to comparative lack of interest in a very specific question. It should be noted that this dataset has been lemmatized and some processing has been done in order to sanitize the large corpus. We refer the reader to the original paper for more details. **MSCOCO** [Lin+14] dataset contains human annotated captions of over **120*K*** images. Each image contains five captions from five different annotators. While there is no guarantee that the human annotations are paraphrases, but because of the nature of the image (which tends to focus on only a few objects and in most cases one prominent object or action) most annotators describe the obvious thing in the image. In fact, this is the main reason why neural networks for generating captions obtain very good `BLEU` scores [Vin+14], which allows us to use this dataset for the paraphrase generation task.

Dataset	Training	Test	Vocabulary Size
PPDB	4,826,492	20,000	38,279
WikiAnswers	4,826,492	20,000	50,000
MSCOCO	331,163	20,000	30,332

Table 7.1: Dataset details

7.4 Experimental Settings

7.4.1 Data Selection

For PPDB we remove the phrases that contain numbers, which includes all syntactic phrases. This step gives us a total of $\mathbf{5.3}\boldsymbol{M}$ paraphrases from which we randomly select **90%** instances for training and $\mathbf{20}\boldsymbol{K}$ pairs from the rest **10%** data are selected similarly for testing. Although WikiAnswers comes with $\mathbf{29,139,992}$ instances, we randomly select $\mathbf{4,826,492}$ for training to keep the training size similar to PPDB (see Table 1). $\mathbf{20}\boldsymbol{K}$ instances are randomly selected from the remaining data for testing. Note that, for the WikiAnswers dataset, we had to clip the vocabulary size[1] to $\mathbf{50}\boldsymbol{K}$ and use the special *UNK* symbol for the words outside the vocabulary. MSCOCO dataset has five captions for every image. This dataset comes with separate subsets for training and validation: *Train 2014* contains over $\mathbf{82}\boldsymbol{K}$ images and *Val 2014* contains over $\mathbf{40}\boldsymbol{K}$ images. From each set of five captions, we randomly select two pairs to use as training instances. The remaining caption is disregarded. Because of the free form nature of the caption generation task [Vin+14], some captions were very long. We clipped those captions to the size of **15** words in order to make the training feasible.

[1] WikiAnswers dataset had many spelling errors yielding a very large vocabulary size (approximately $\mathbf{250}\boldsymbol{K}$). Hence, we selected the most frequent $\mathbf{50}\boldsymbol{K}$ words in the vocabulary to reduce the computational complexity.

Models	Reference
Sequence to Sequence	[SVL14]
With Attention	[BCB15]
Bi-directional LSTM	[GJM13]
Residual LSTM	Our proposed model

Table 7.2: Models

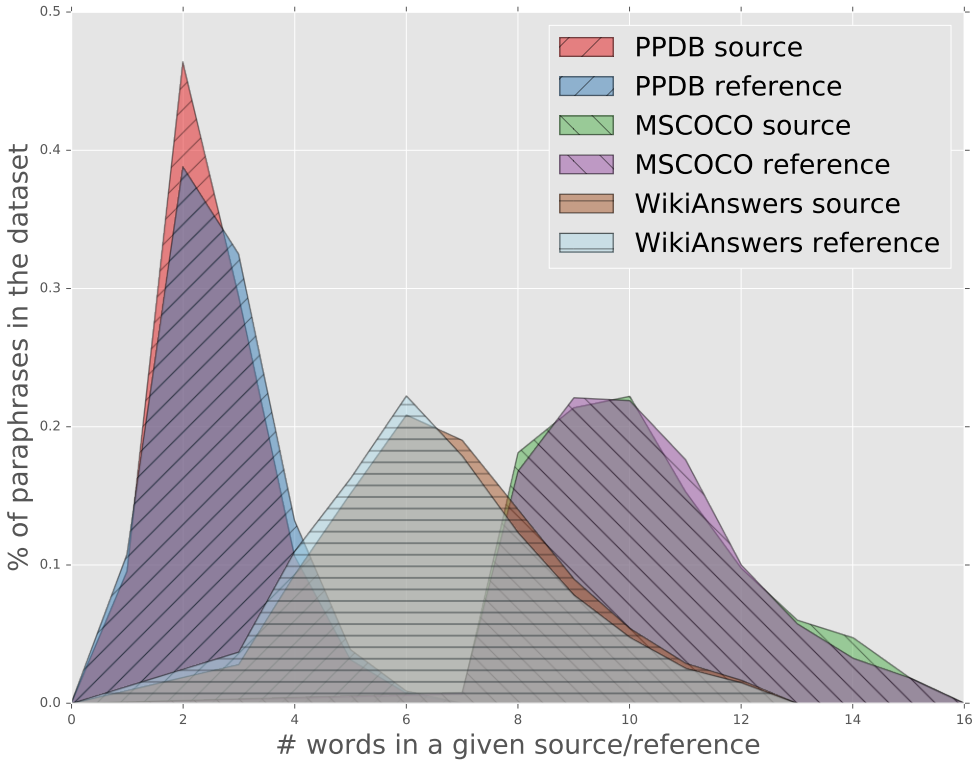


Figure 7.4: Distribution of word length across datasets

7.4.2 Models

In this work, we experimented with four different models (see Table 2). For each model, we experimented with two- and four-layers of stacked LSTMs. This was motivated from the state-of-the-art speech recognition systems that also use three to four layers of stacked LSTMs [LW15]. In encoder-decoder models, the size of the beam search used during inference is very important. Larger beam size always give higher accuracy but comes at a computational cost. We experimented with beam sizes of **5** and **10** to compare the models, as these are the most common beam sizes used in the literature [SVL14]. The bi-directional model used half of the number of layers shown for other models. This was done to ensure similar parameter sizes across the models.

7.4.3 Training

We used a one-hot vector approach to represent the words in all models. Models were trained with a stochastic gradient descent (SGD) algorithm. The learning rate began at **1.0**, and was halved after every third training epoch. Each network was trained for ten epochs. A standard dropout [Sri+14a] of 50% was applied after every LSTM layer. The number of LSTM units in each layer was fixed to **512** across all models. In order to allow exploration of a wide variety of models, training was restricted to a limited number of epochs, and no hyper-parameter search was performed. Training time ranged from **36** hours for `WikiAnswers` and `PPDB` to **14** hours for `MSCOCO` on a *Titan X* with *CuDNN 5* using Theano version **0.9.0*dev*1** [The16].

A beam search algorithm was used to generate optimal paraphrases by exploiting the trained models in the testing phase [SVL14]. We used perplexity as the loss function during training. Perplexity measures the uncertainty of the language model, corresponding to how many bits on average would be needed to encode each word given the language model.

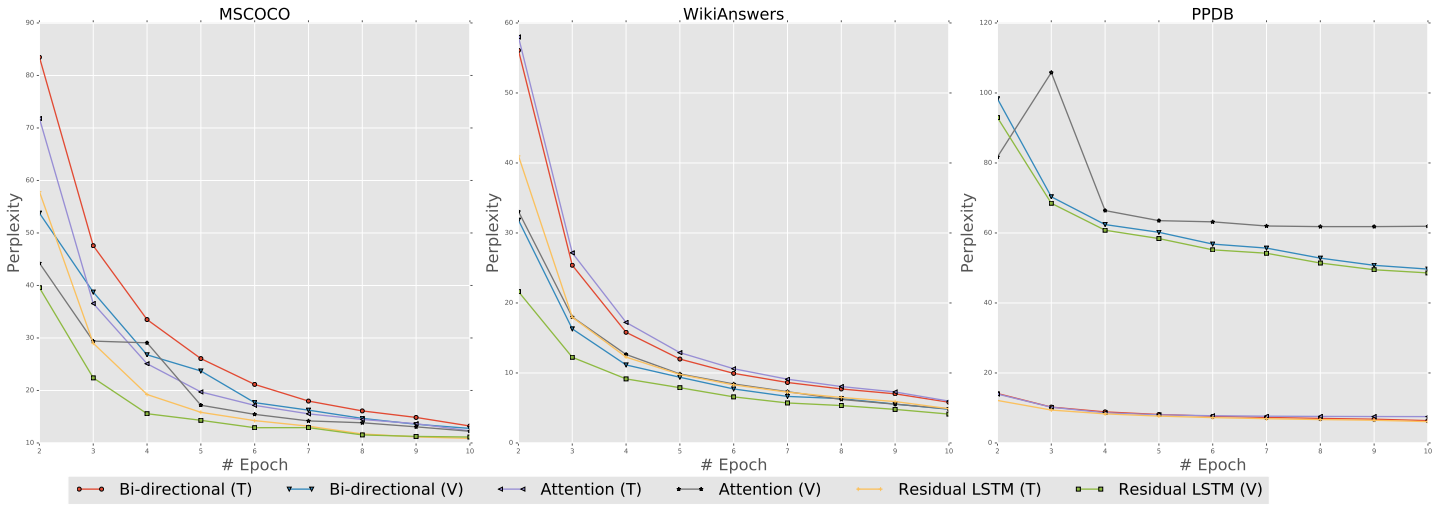


Figure 7.5: Perplexity (↓) during training and validation for various models [shared legend]

A lower `PPLX` indicates a better score. While `WikiAnswers` and `MSCOCO` had a very good correlation of training and validation perplexity, `PPDB` easily overfitted and did not yield good validation perplexity (see Figure 7.5).

7.5 Evaluation

7.5.1 Metrics

To quantitatively evaluate the performance of our paraphrase generation models, we use the well-known automatic evaluation metrics[2] for comparing parallel corpora: `BLEU` [Pap+02], `METEOR` [LA07], and Translation Error Rate (`TER`) [Sno+06]. Even though these metrics were designed for machine translation, previous works have shown that they can perform well for the paraphrase recognition task [MTC12] and correlate well with human judgments in evaluating generated paraphrases [WBK10].

[2]We used the software available at `https://github.com/jhclark/multeval`

Although there exists a few automatic evaluation metrics that are specifically designed for paraphrase generation, such as PEM (Paraphrase Evaluation Metric) [LDN10] and PINC (Paraphrase In N-gram Changes) [CD11], they have certain limitations. PEM relies on large in-domain bilingual parallel corpora along with sample human ratings for training the metric and it can only model paraphrasing up to the phrase-level granularity. PINC attempts to solve these limitations by proposing a method that is essentially the inverse of BLEU, as it calculates the n-gram difference between the source and the target sentences. Although PINC correlates well with human judgments in lexical dissimilarity assessment, BLEU have been shown to correlate better for semantic equivalence agreements at the sentence-level when a sufficiently large number of reference sentences are available for each source sentence [CD11].

BLEU considers exact matching between reference paraphrases and system generated paraphrases by considering n-gram overlaps while METEOR improves upon this measure via stemming and synonymy using WordNet. TER measures the number of edits required to change a system output into one of the references. As suggested in [Cla+11], we used a stratified approximate randomization (AR) test. AR calculates the probability of a metric score providing the same reference sentence by chance. We report our p-values at 95% Confidence Intervals (CI). The major limitation of these evaluation metrics is that they do not consider the meaning of the paraphrases, and hence, are not able to capture paraphrases of entities. For example, these metrics will not reward the paraphrasing of *Coke* to *Pepsi*. However, in Word2Vec embeddings they will be considered as similar entities. Therefore, we also evaluate our models on a sentence similarity metric[3] proposed by Rus et al. [RL12]. This metric uses word embeddings to compare the phrases. In our experiments, we used Word2Vec embeddings pre-trained on the Google News Corpus [Mik+14]. This is referred as *'Emb Greedy'* in

[3]We used the software available at https://github.com/julianser/hed-dlg-truncated/

our results table.

7.5.2 Results

	PPDB	WikiAnswers	MSCOCO
Src	south eastern	what be the symbol of magnesium sulphate	a small kitten is sitting in a bowl
Ref	the eastern part	chemical formulum for magnesium sulphate	a cat is curled up in a bowl
Gen	south east	do magnesium sulphate have a formulum	a cat that is sitting on a bowl
Src	organized	what be the bigggest galaxy know to man	an old couple at the beach during the day
Ref	managed	how many galaxy be there in you known universe	two people sitting on dock looking at the ocean
Gen	arranged	about how many galaxy do the universe contain	a couple standing on top of a sandy beach
Src	counselling	what do the ph of acid range to	a little baby is sitting on a huge motorcycle
Ref	be kept informed	a acid have ph range of what	a little boy sitting alone on a motorcycle
Gen	consultations	how do acid affect ph	a baby sitting on top of a motorcycle

Table 7.3: Samples generated using Residual LSTM with beam size 5 for each dataset.Src: Source, Ref: Reference, Gen: Generated

We present the results from various models across different datasets (Table 3). Although our focus is on stacked residual LSTM, which is applicable only when there are more than two layers, we still present the scores from two layer LSTM as a baseline. This will provide a good comparison against deeper models. The results demonstrate that our proposed model outperforms other models on BLEU and TER for all datasets. On *Emb Greedy* our model outperforms other models in all datasets except against the Attention model when beam size is 10. On METEOR our model outperforms other models on MSCOCO and WikiAnswers however, for PPDB the simple sequence to sequence model has better scores. Note that, these results were obtained from using single models and no ensemble of any models was used. To calculate BLEU and METEOR, four references were used for MSCOCO, and five for PPDB and WikiAnswers. In some instances WikiAnswers did not have five references for every source, hence, those were calculated on reduced references. In Table 4, we present the variance due to the test set selection. This is calculated using bootstrap re-sampling for each optimizer run [Cla+11]. Variance due to optimizer instability was less than 0.1 in all cases. p-value of

#Layers	Model	Beam size = 5 BLEU↑	METEOR↑	Emb Greedy↑	TER↓	Beam size = 10 BLEU↑	METEOR↑	Emb Greedy↑	TER↓
				PPDB					
2	Sequence to Sequence	12.5	21.3	32.55	82.9	12.9	20.5	32.65	83.0
	With Attention	13.0	21.2	32.95	82.2	13.8	20.6	32.29	81.9
4	Sequence to Sequence	18.3	**23.5**	33.18	82.7	18.8	**23.5**	33.78	82.1
	Bi-directional	19.2	23.1	34.39	77.5	19.7	23.2	34.56	84.4
	With Attention	19.9	23.2	34.71	83.8	20.2	22.9	**34.90**	77.1
	Residual LSTM	**20.3**	23.1	**34.77**	**77.1**	**21.2**	23.0	34.78	**77.0**
				WikiAnswers					
2	Sequence to Sequence	19.2	26.1	62.65	35.1	19.5	26.2	62.95	34.8
	With Attention	21.2	22.9	63.22	37.1	21.2	23.0	63.50	37.0
4	Sequence to Sequence	33.2	29.6	73.17	28.3	33.5	29.6	73.19	28.3
	Bi-directional	34.0	30.8	73.80	27.3	34.3	30.7	73.95	27.0
	With Attention	34.7	31.2	73.45	27.1	34.9	31.2	73.50	27.1
	Residual LSTM	**37.0**	**32.2**	**75.13**	**27.0**	**37.2**	**32.2**	**75.19**	**26.8**
				MSCOCO					
2	Sequence to Sequence	15.9	14.8	54.11	66.9	16.5	15.4	55.81	67.1
	With Attention	17.5	16.6	58.92	63.9	18.6	16.8	59.26	63.0
4	Sequence to Sequence	28.2	23.0	67.22	56.7	28.9	23.2	67.10	56.3
	Bi-directional	32.6	24.5	68.62	53.8	32.8	24.9	68.91	53.7
	With Attention	33.1	25.4	69.10	54.3	33.4	25.2	**69.34**	53.8
	Residual LSTM	**36.7**	**27.3**	**69.69**	**52.3**	**37.0**	**27.0**	69.21	**51.6**

Table 7.4: Evaluation results on PPDB, WikiAnswers, and MSCOCO (Best results are in **bold**)

these tests are less than **0.05** in all cases. Thus, comparison between two models is significant at 95% CI if the difference in their score is more than the variance due to test set selection [Table 4].

7.5.3 Analysis

Scores on various metrics vary a lot across the datasets, which is understandable because they are of different natures. PPDB contains very small phrases and thus does not score well

Dataset	σ^2[BLEU]	σ^2[METEOR]	σ^2[TER]	σ^2[Emb Greedy]
PPDB	2.8	0.2	0.4	0.000100
WikiAnswers	0.3	0.1	0.1	0.000017
MSCOCO	0.2	0.1	0.1	0.000013

Table 7.5: Variance due to test set selection for the results obtained

with metrics like BLEU and METEOR which penalize short phrases. As shown in Figure 7.4 more than fifty percent of PPDB contains two or fewer works. This leads to big difference in training and validation errors, as shown in Figure 7.5. For WikiAnswers and MSCOCO the boost in the result compared to other sequence to sequence model is statistically significant (Table 3). From the results it is evident that deeper LSTMs always improve upon shallow ones. For the beam size of 5 our model beats other models in all datasets. For beam size of 10, attention model has marginally better *Emb Greedy* score than our model. When we look at the qualitative result we notice that the bias in the dataset is exploited by the system which is a side effect of any form of learning on a limited dataset. We can see this effect in Table 5. For example *a OBJECT* is always paraphrased with *a OBJECT*. Shorter sentences always generate shorter paraphrases and same is true for longer sequences. Based on our results we see that the embedding-based metric correlates well with statistical metrics. Looking at Figure 7.5 and the results from Table 5, we can conclude that perplexity is a good loss function for training paraphrase generation models. However, a more ideal metric to fully encode the objective of paraphrasing should reward novelty and penalize redundancy during paraphrase generation.

7.6 Related Work

Prior work for paraphrase generation have applied relatively different methodologies, typically using knowledge-driven approaches or statistical machine translation (SMT) principles. The knowledge-driven approaches for paraphrase generation [MD10] generally utilize hand-crafted rules [McK83] or automatically learned complex paraphrase patterns [Zha+09]. Other related work uses thesaurus-based [Has+07] or semantic analysis-driven natural language generation approaches [KMVS03] to generate paraphrases. In contrast, Quirk et al., [QBD04] show the effectiveness of SMT techniques for paraphrase generation given adequate monolingual parallel corpus extracted from comparable news articles. Wubben et al., [WBK10] propose a phrase-based SMT framework for sentential paraphrase generation by using a large aligned monolingual corpus of news headlines. Zhao et al., [Zha+08] propose a combination of multiple resources to learn phrase-based paraphrase tables and corresponding feature functions to devise a log-linear SMT model. Other models generate application-specific paraphrases [Zha+09], leverage bilingual parallel corpora [BCB05] or apply a multi-pivot approach to output candidate paraphrases [Zha+10].

Not much work has been done with regard to applications of deep learning for paraphrase generation. We explored several sources as potential large datasets. Recently, Weiting et al. [Wie+15] took the `PPDB` dataset (size ***XL***) and annotated phrases based on their paraphrasability. This dataset is called *Annotated*-`PPDB` and contains **3000** pairs in total. In the same work, they also introduced another dataset called *ML-Paraphrase* for the purpose of evaluating bigram paraphrases. This dataset contains **327** instances. Microsoft Research Paraphrase Corpus (MSRP) [DBQ05] is another widely used dataset for paraphrase detection. MSRP contains **5800** pairs of sentences (obtained from various news sources) accom-

panied with human annotations. These datasets are too small and therefore not suitable for deep learning models.

7.7 Conclusion

In this paper, we described a novel technique to train stacked LSTM networks for paraphrase generation. This is an extension to sequence to sequence learning, which has shown great promise for NLP tasks. Our model is able to outperform the state-of-the-art models for sequence to sequence learning. We have shown that stacking of residual LSTM layers is useful for paraphrase generation, but it may not perform equally well for machine translation because not every word in a source sequence needs to be substituted for paraphrasing. Residual connections help retain important words in the generated paraphrases.

We experimented on three different large scale datasets and reported results using various automatic evaluation metrics. We showed the use of the well-known `MSCOCO` dataset for paraphrase generation and demonstrated that the models can be trained effectively without the advantage of having the images. Regardless of this limitation, our model performed equally good on this dataset. We believe that we have set strong baselines for neural paraphrase generation on these datasets. This will enable future researchers to easily compare and evaluate subsequent works in paraphrase generation.

Recent advances in neural network with regard to learnable memory [SWF+15; GWD14a] have enabled models to get one step closer to learn comprehension. It will be helpful to explore such networks for the paraphrase generation task. Also, it remains to be explored how unsupervised deep learning could be harnessed for paraphrase generation. It would be interesting to see if researchers working on image-captioning can employ neural paraphrase generation to augment their dataset.

Chapter 8

Condensed Memory Networks

8.1 Introduction

Clinicians perform complex cognitive processes to infer the probable diagnosis after observing several variables such as the patient's past medical history, current condition, and various clinical measurements. The cognitive burden of dealing with complex patient situations could be reduced by having an automated assistant provide suggestions to physicians of the most probable diagnostic options for optimal clinical decision-making.

Some work has been done in building Artificial Intelligence (AI) systems that can support clinical decision making [Lip+15; CBS15; Cho+16]. These works have primarily focused on the use of various biosignals as features. EHRs typically store such structured clinical data (e.g. physiological signals, vital signs, lab tests etc.) about the patients' clinical encounters in addition to unstructured textual notes that contain a complete picture of the associated clinical events. Structured clinical data generally contain raw signals without much interpretation, whereas unstructured free-text clinical notes contain detailed description of the overall clinical scenario. In this paper, we explore the discriminatory capability of the

unstructured free-text clinical notes to correctly infer the most probable diagnoses from a complex clinical scenario. In the Clinical Decision Support track[1] of the recent Text REtrieval Conference (TREC) series, clinical diagnostic inferencing from unstructured free texts has been shown to play a significant role in improving the accuracy of relevant biomedical article retrieval [Has+14; Has+15; Has+16]. We also explore the use of an external knowledge source like Wikipedia from which the model can extract relevant information, such as signs and symptoms for various diseases. Our goal is to combine such an external clinical knowledge source with the free-text clinical notes and use the learning capability of memory networks to correctly infer the most probable diagnosis.

Memory Networks (MemNNs) [Suk+15; WCB14] are a class of models which contain an external memory and a controller to read from and write to the memory. Memory Networks read a given input source and a knowledge source several times (hops) while updating an internal memory state. The memory state is the representation of relevant information from the knowledge base optimized to solve the given task. This allows the network to remember useful features. The notion of neural networks with memory was introduced to solve AI tasks that require complex reasoning and inferencing. These models have been successfully applied in the Question Answering domain on datasets like *bAbi* [Wes+15], *MovieQA* [Tap+15], and *WikiQA* [Suk+15; Mil+16]. Memory networks are harder to train than traditional networks and they do not scale easily to a large memory. End-to-End Memory Networks [Suk+15] and Key-Value Memory Networks (KV-MemNNs) [Mil+16] try to solve these problems by training multiple hops over memory and compartmentalizing memory slots into hashes, respectively.

When the memory is large, hashing can be used to selectively retrieve only relevant information from the knowledge base, however not much work has been done to improve the

[1] http://www.trec-cds.org/

information content of the memory state. If the network were trained for factoid question answering, the memory state might be trained to represent relevant facts and relations from the underlying domain. However, for real world tasks, a large amount of memory is required to achieve state-of-the-art results. In this paper, we introduce Condensed Memory Networks (C-MemNNs), an approach to efficiently store condensed representations in memory, thereby maximizing the utility of limited memory slots. We show that a condensed form of memory state which contains some information from earlier hops learns efficient representation. We take inspiration from human memory for this model. Humans can learn new information and yet remember even very old memories as abstractions. We also experiment with a simpler form of knowledge retention from previous hops by taking a weighted **average** of memory states from all the hops (A-MemNN). Even this simpler alternative which does not add any extra parameter is able to outperform standard memory networks. Empirical results on the `MIMIC-III` dataset reveal that C-MemNN improves the accuracy of clinical diagnostic inferencing over other classes of memory networks. To the best of our knowledge, this is the first empirical study to classify diagnosis from EHR free-text clinical notes using memory networks.

8.2 Related Work

8.2.1 Memory Networks

Memory Networks (MemNNs) [WCB14] and Neural Turing Machines (NTMs) [GWD14b] are the two classes of neural network models with an external memory component. MemNN stores all information (e.g. knowledge base, background context) into the external memory, assigns a relevance probability to each memory slot using content-based addressing schemes,

and reads contents from each memory slot by taking their weighted sum. End-to-End Memory Networks introduced multi-hop training [Suk+15] and do not require strong supervision unlike MemNN. Key-Value Memory Networks [Mil+16] have a key-value paired memory and are built upon MemNN. The key-value paired structure is a generalized way of storing content in the memory. The contents in the key-memory are used to calculate the relevance probabilities whereas the contents in the value-memory are read into the model to help make the final prediction.

In contrast to MemNN, which uses a content-based mechanism to access the external memory, the NTM controller uses both content- and location-based mechanisms. The fundamental difference between these models is that MemNN does not have a mechanism for the content to be changed in the memory, while NTM can modify the content in each episode. This makes MemNN easier to train compared to NTM.

Recently, there has been an attempt to incorporate longer contextual memory into the basic Recurrent Neural Networks (RNNs) framework. Stack-Augmented RNN [JM15] proposes interconnecting RNN modules using a push-down stack in order to learn long-term dependencies. They are able to reproduce complicated sequence patterns. Chung, Ahn, and Bengio (2016) explore multi-scale RNN, which is able to learn a latent hierarchical structure by using temporal representation at different time-scales. These methods are well-suited for learning long-term temporal dependencies, but do not scale well to large memory. Hierarchical Memory Networks [Cha+16] study the use of maximum inner product search (MIPS) to store memory slots in a hierarchy. Our aim is to improve the efficiency and knowledge density of the standard memory slots by exploring a hierarchy of internal memory representations over multiple hops.

Another related class of models are the attention-based neural networks. These models are trained to learn an attention mechanism so that they can focus on the important

information on a given input. Applying an attention mechanism on the machine reading comprehension task [Her+15; Dhi+16; Cui+16; SBB16] has shown promising results. In tasks where inferencing is governed by the input source e.g. sentence-level machine translation [BCB14], image caption generation [Xu+15], and visual question answering [Lu+16], the use of attention-based models has proven to be very effective. As attention is learned by the iterative finding of the highly-activated input regions, this is not feasible for a large-scale external memory; however, more research is required in order to achieve attention over memory.

8.2.2 Neural Networks for Clinical Diagnosis

The application of neural networks to medical tasks dates back more than twenty years [Bax90]. The recent success of deep learning has drawn broader interest in building AI systems to support clinical decision making. Lipton et al. (2015) train Long Short-Term Memory Networks (LSTMs) to classify 128 diagnoses from 13 frequently but irregularly sampled clinical measurements extracted from the EHR. DoctorAI [CBS15] and RETAIN [Cho+16] also use time series data for diagnosis classification. Similar to these works, we formulate the problem as multi-label classification, since each medical note might be associated with multiple diagnoses. However, there are two important differences between our work and the previous work. We only consider discharge notes for our experiments, which are unstructured free-texts and do not contain time series data, while they rely on time series datasets where each time series has a fixed number of clinical measurements. Moreover, we train our model in an end-to-end fashion and do not extract any hand-engineered features from the notes, while they resample all time series data to an hourly rate and fill in the gaps created by window-based resampling in clinical measurements. We propose the use of memory networks instead

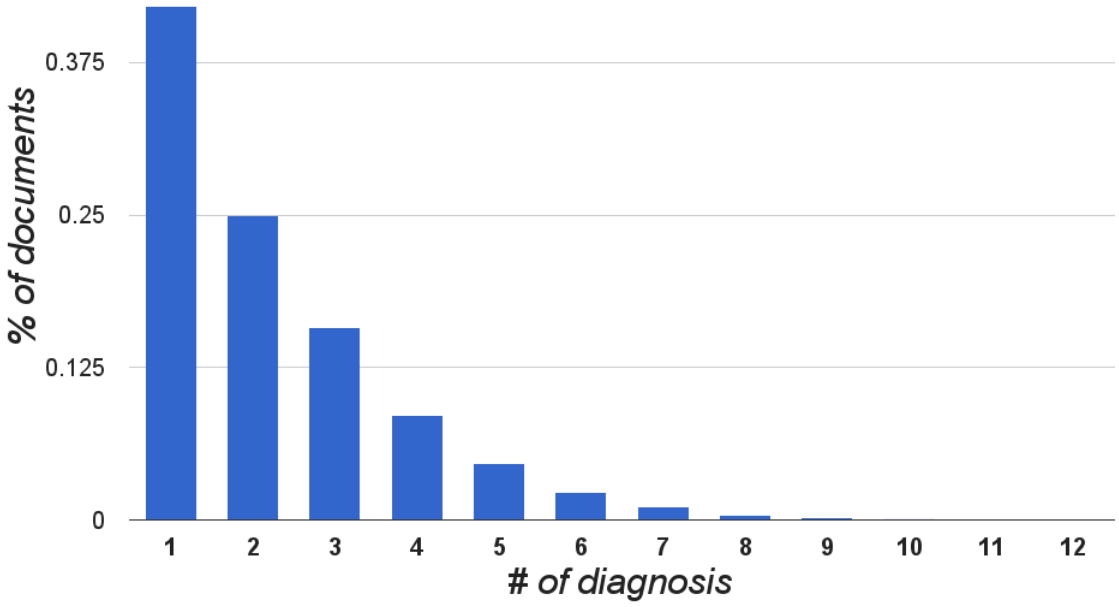


Figure 8.1: Distribution of number of diagnosis in a note.

of LSTMs to classify the diagnoses since the memory component provides the flexibility to learn from an external knowledge source.

8.3 Dataset

`MIMIC-III` (Multiparameter Intelligent Monitoring in Intensive Care) [Joh+16] is a large freely-available clinical database. It contains physiological signals and various measurements captured from patient monitors, and comprehensive clinical data obtained from hospital medical information systems for over $\mathbf{58}\boldsymbol{K}$ Intensive Care Unit (ICU) patients. We use the *noteevents* table from `MIMIC-III`: v1.3, which contains the unstructured free-text clinical notes for patients. We use 'discharge summaries', instead of 'admission notes', as former contains actual ground truth and free-text. Since discharge summaries are written after diagnosisdecision, we sanitize the notes by removing any mention of class-labels in the text.

Medical Note (partially shown)
Date of Birth: [********] Sex: M Service: Medicine **Chief Complaint:** Admitted from rehabilitation for hypotension (systolic blood pressure to the 70s) and decreased urine output. **History of present illness:** The patient is a 76-year-old male who had been hospitalized at the [**Hospital1 3007**] from [******] through [******] of 2002 after undergoing a left femoral-AT bypass graft and was subsequently discharged to a rehabilitation facility. On [*****], he presented again to the [**Hospital1 3087**] after being found to have a systolic blood pressure in the 70s and no urine output for 17 hours.
Diagnosis
Cardiorespiratory arrest. Non-Q-wave myocardial infarction. Acute renal failure.

Table 8.1: An example `MIMIC-III` note.

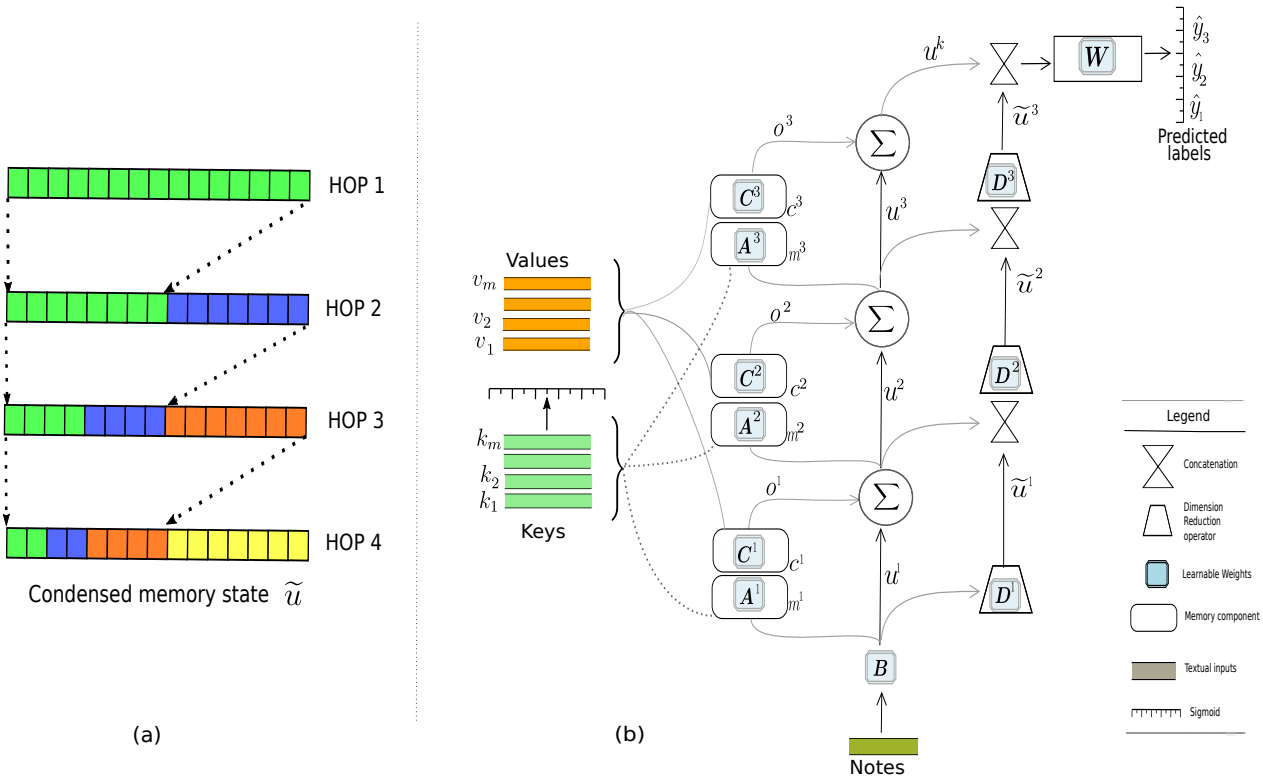


Figure 8.2: (a) Abstract view of transformation of memory representation over multiple hops. (b) Structural overview of end-to-end model for condensed memory networks.

As shown in Table 1, medical notes contain several details about the patient but the sections are not uniform. We do not separate the sections other than the *DIAGNOSIS*, which is our label. There are multiple labels (diagnoses) for a given note, and a note can belong to multiple classes of diagnoses, thus we formulate our task as a multiclass-multilabel classification problem. The number of diagnoses per note is also not consistent and shows a long tail (Figure 8.1). We have taken measures to counteract these issues, which are discussed in the *Memory addressing* section.

Some diagnoses are less frequent in the data set. Without enough training instances, a model is not able to learn to recognize these diagnoses. Therefore, we experiment with a varying number of labels in this work (see details in the *Experiments* section).

8.3.1 Knowledge Base

We use Wikipedia pages (see Table 2) corresponding to the diagnoses in the `MIMIC-III` notes as our external knowledge source. WikiProject Medicine is dedicated to improving the quality of medical articles on Wikipedia and the information presented in these pages are generally shown to be reliable [Tre11]. Since some diagnosis terms from `MIMIC-III` don't always match a Wikipedia title, we use the Wikipedia API with the diagnoses as the search terms to find the most relevant Wikipedia pages. In most cases we find an exact match while in the rest we pick the most relevant page. We use the first paragraph and the paragraphs corresponding to the *Signs and symptoms* sections for our experiments. In cases where such a section is not available, we use the second and third paragraphs of the page. This happens for the obscure diseases, which have a limited content.

Cardiac arrest
Cardiac arrest is a sudden stop in effective blood circulation due to the failure of the heart to contract effectively or at all[1]. A cardiac arrest is different from (but may be caused by) a myocardial infarction (also known as a heart attack), where blood flow to the muscle of the heart is impaired such that part or all of the heart tissue dies… **Signs and symptoms** Cardiac arrest is sometimes preceded by certain symptoms such as fainting, fatigue, blackouts, dizziness, chest pain, shortness of breath, weakness, and vomiting. The arrest may also occur with no warning …

Table 8.2: Partially shown example of a relevant Wikipedia page.

8.4 Condensed Memory Networks

The basic structure of our model is inspired by MemNN. Our model tries to learn memory representation from a given knowledge base. Memory is organized as some number of slots $\boldsymbol{m_1, \ldots m_t}$. For the given input text i.e. medical notes $\boldsymbol{x_1, \ldots, x_n}$, the external knowledge base (wiki pages, wiki titles) $\boldsymbol{(k_1, v_1), (k_2, v_2), \ldots, (k_m, v_m)}$, and the diagnoses of those notes $\boldsymbol{y}$, we aim to learn a model $\boldsymbol{\mathcal{F}}$ such that

$$\boldsymbol{\mathcal{F}(x_n, (k_m, v_m)) = \hat{y} \rightarrow y} \tag{8.1}$$

We break down this function $\boldsymbol{\mathcal{F}}$, in four parts *I, G, O, R* which are the standard components of Memory Networks.

- **I: Input memory representation** is the transformation of the input $\boldsymbol{x}$ to some

internal representation $\boldsymbol{u}$ using learned weights $\boldsymbol{B}$. This is the internal state of the model and is similar to the hidden state of RNN-based models. In this paper, we propose the addition of a condensed memory state $\widetilde{\boldsymbol{u}}$, which is obtained via the iterative concatenation of successively lower dimensional representations of the input memory state $\boldsymbol{u}$.

- **G: Generalization** is the process of updating the memory. MemNN updates all slots in the memory, but this is not feasible when the size of the knowledge source is very large. Therefore, we organize the memory as key-value pairs as described in Miller et al. (2016). We use hashing to retrieve a small portion of keys for the given input.

- **O: Output memory representation** is the transformation of the knowledge $(\boldsymbol{k}, \boldsymbol{v})$ to some internal representation $\boldsymbol{m}$ and $\boldsymbol{c}$. While End-to-End MemNN uses an embedding matrix to convert memories to learned feature space, our model uses a two-step process because we represent wiki-titles and wikipages as different learned spaces. We learn matrix $\boldsymbol{A}$ to transform wikipages (keys) and $\boldsymbol{C}$ to transform wiki-titles (values). Our choice of wikipages as keys and wiki-titles as values is deliberate – the input "medical notes" more closely match the text of the wikipages and the diagnoses more closely match the wiki-titles. This allows for a better mapping of features; our empirical results validate this idea.

  Let $\boldsymbol{k}$ represents the hop number. The output memory representation is obtained by:

$$\boldsymbol{o}^{\boldsymbol{k}} = \sum_{\boldsymbol{i}} \text{Addressing}(\boldsymbol{u}^{\boldsymbol{k}}, \boldsymbol{m}_{\boldsymbol{i}}^{\boldsymbol{k}}) \cdot \boldsymbol{c}_{\boldsymbol{i}}^{\boldsymbol{k}} \tag{8.2}$$

  where *Addressing* is a function which takes the given input memory state $\boldsymbol{u}$ and provides the relevant memory representation $\boldsymbol{m}$.

- **R: Response** combines the internal state $\boldsymbol{u}$, internal condensed state $\widetilde{\boldsymbol{u}}$ and the output representation $\boldsymbol{o}$ to provide the predicted label $\hat{\boldsymbol{y}}$. We sum $\boldsymbol{u}$ and $\boldsymbol{o}$ and then take the dot product with another learned matrix $\boldsymbol{W}$. We then concatenate this value with condensed memory state $\widetilde{\boldsymbol{u}}$. This value is then passed through sigmoid to obtain the likelihood of each class. We use sigmoid instead of softmax in order to obtain multiple predicted labels, $\hat{\boldsymbol{y}}_{\mathbf{1}}, \ldots \hat{\boldsymbol{y}}_{\boldsymbol{r}}$ among possible $\boldsymbol{R}$ labels. Both the memory states $\boldsymbol{u}$ and $\widetilde{\boldsymbol{u}}$ are computed as:

$$\boldsymbol{u}^{\boldsymbol{k+1}} = \boldsymbol{u}^{\boldsymbol{k}} + \boldsymbol{o}^{\boldsymbol{k}} \tag{8.3}$$

$$\widetilde{\boldsymbol{u}}^{\boldsymbol{k+1}} = \boldsymbol{u}^{\boldsymbol{k+1}} \oplus \boldsymbol{D}_{\mathbf{1}} \cdot \widetilde{\boldsymbol{u}}^{\boldsymbol{k}} \tag{8.4}$$

where $\oplus$ denotes concatenation of vectors.

Our major contribution to memory networks is the use of condensed memory state $\widetilde{\boldsymbol{u}}$ in combination with input memory state $\boldsymbol{u}$ to do the inference. As shown in Figure 2(a), $\widetilde{\boldsymbol{u}}$ is transformed to include the information of previous hops, but in lower dimensional feature space. This leads to a longer term memory representation, which is better able to represent hierarchy in memory. The class prediction is obtained using:

$$\hat{\boldsymbol{y}}_{\boldsymbol{r}} = \underset{\boldsymbol{r} \epsilon \boldsymbol{R}}{\arg\max} \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-\mathbf{1} * (\widetilde{\boldsymbol{u}}^{\boldsymbol{k+1}} \cdot \boldsymbol{W})}} \tag{8.5}$$

8.4.1 Network Overview

Figure 2(b) shows the overview of the C-MemNN structure. The input $\boldsymbol{x}$ is converted to internal state $\boldsymbol{u}^{\mathbf{1}}$ using the transformation matrix $\boldsymbol{B}$. This is combined with memory

key slots $\boldsymbol{m^1}$ using matrix $\boldsymbol{A}$. Memory addressing is used to retrieve the corresponding memory value $\boldsymbol{c^1}$. This value is transformed using matrix $\boldsymbol{C}$ to output memory representation $\boldsymbol{o^1}$. In parallel, memory state $\boldsymbol{u}$ is condensed to half of its original dimension using the transformation matrix $\boldsymbol{D}$. If $\boldsymbol{u}$ is of size $\boldsymbol{1 \times K}$ then $\boldsymbol{D}$ is of size $\boldsymbol{K \times \frac{K}{2}}$. We call this reduced representation of $\boldsymbol{u}$ the condensed memory state, $\widetilde{\boldsymbol{u}}$. This is the end of the first hop. This process is then repeated for a desired number of hops. After each hop, the condensed memory state $\widetilde{\boldsymbol{u}}$ becomes the concatenation of its previous state and its current state, each reduced to half of its original dimension.

8.4.2 Averaged Memory Networks

In C-MemNN, the transformation of $\widetilde{\boldsymbol{u}}$ at every hop adds more parameters to the model, which is not always desirable. Thus, we also present a simpler alternative model, which we call A-MemNN, to capture hierarchy in memory representation without adding any learned parameters. In this alternative model, we compute the weighted average of $\widetilde{\boldsymbol{u}}$ across multiple hops. Instead of concatenating previous $\widetilde{\boldsymbol{u}}$ values, we simply maintain an exponential moving average from different hops:

$$\widetilde{\boldsymbol{u}}^{\boldsymbol{k+1}} = \widetilde{\boldsymbol{u}}^{\boldsymbol{k}} + \frac{\widetilde{\boldsymbol{u}}^{\boldsymbol{k-1}}}{\boldsymbol{2}} + \frac{\widetilde{\boldsymbol{u}}^{\boldsymbol{k-2}}}{\boldsymbol{4}} + \boldsymbol{\ldots} \tag{8.6}$$

where, the starting condensed memory state is same as the input memory state $\widetilde{\boldsymbol{u}}^{\boldsymbol{1}} = \boldsymbol{u}^{\boldsymbol{1}}$.

8.4.3 Memory Addressing

Key-Value addressing as described in KV-MemNN uses softmax on the product of question embeddings and retrieved keys to learn a relevance probability distribution over memory slots. The representation obtained is then the sum of the output memory representation

$\boldsymbol{o}$, weighted by those probability values. KV-MemNN was designed to pick the single most relevant answer given a set of candidate answers. The use of softmax significantly decreases the estimated relevance of all but the most probable memory slot. This presents a problem for multi-label classification in which several memory slots may be relevant for different target labels. We experimented with changing softmax to sigmoid to alleviate this problem, but this was not sufficient to allow the incorporation of the condensed form of the internal state $\boldsymbol{u}$ arising from earlier hops. Thus, we explore a novel alternate addressing scheme, which we call gated addressing. This addressing method uses a multi-layer feed-forward neural network (FNN) with a sigmoid output layer to determine the appropriate weights for each memory slot. The network calculates a weight value between 0 and 1 for each memory slot, and a weighted sum of memory slots is obtained as before.

8.4.4 Document Representation

There are a variety of models to represent knowledge in key-value memories, and the choice of a model can have an impact on the overall performance. We use a simple bag-of-words (BoW) model which transforms each word $\boldsymbol{w_{ij}}$ in the document $\boldsymbol{d_i} = \boldsymbol{w_{i1}}, \boldsymbol{w_{i2}}, \boldsymbol{w_{i3}}, \ldots, \boldsymbol{w_{in}}$ to embeddings, and sums these together to obtain the vectors $\boldsymbol{\Phi(d_i)} = \sum_j \boldsymbol{A w_{ij}}$, with $\boldsymbol{A}$ being the embedding matrix. Medical notes, memory keys and memory values are all represented in this way.

8.5 Experiments

The distribution of diagnoses in our training data has a very long tail. There are 4,186 unique diagnoses in `MIMIC-III` discharge notes. However, many diagnoses (labels) occur in only a single note. This is not sufficient to efficiently train on these labels. The 50 most-common

labels cover 97% of the notes and the 100 most-common labels cover 99.97%. Thus, we frame this task as multi-label classification for top-N labels. We present experiments for both the 50 most-common and 100-most common labels. For all experiments, we truncate both notes and wiki-pages to 600 words. We reduce the trained vocabulary to 20K after removing common stop-words. We use a common dimension of 500 for all embedding matrices. We use a memory slot of dimension 300. A smaller embedding of dimension 32 is used to represent the wiki-titles.

We present experiments for end-to-end memory networks [Suk+15], Key-Value Memory Networks (KV-MemNNs) [Mil+16] and our models, Condensed Memory Networks (C-MemNN) and Averaged Memory Networks (A-MemNN). We separately train models for three, four and five hops. The strength of our model is the ability to make effective use of several memory hops, and so we do not present results for one or two hops. We did not consider standard text classification models like CNN, LSTM or Fasttext [Jou+16] as they do not allow incorporating external memory, which is one of the main goals of this paper.

8.5.1 Training

We use Adam [KB15] stochastic gradient descent for optimizing the learned parameters. The learning rate is set to 0.001 and batch size for each iteration to 100 for all models. For the final prediction layer, we use a fully connected layer on top of the output from equation 8.5 with a sigmoid activation function. The loss function is the sum of cross entropy from prediction labels and prediction memory slots using addressing schema. Complexity of the model was penalized by adding $\boldsymbol{L2}$ regularization to the cross entropy loss function. We use dropout [Sri+14b] with probability 0.5 on the output-to-decision sigmoid layer and limit the norm of the gradients to be below 20. Models are trained on **80%** of the data and validated

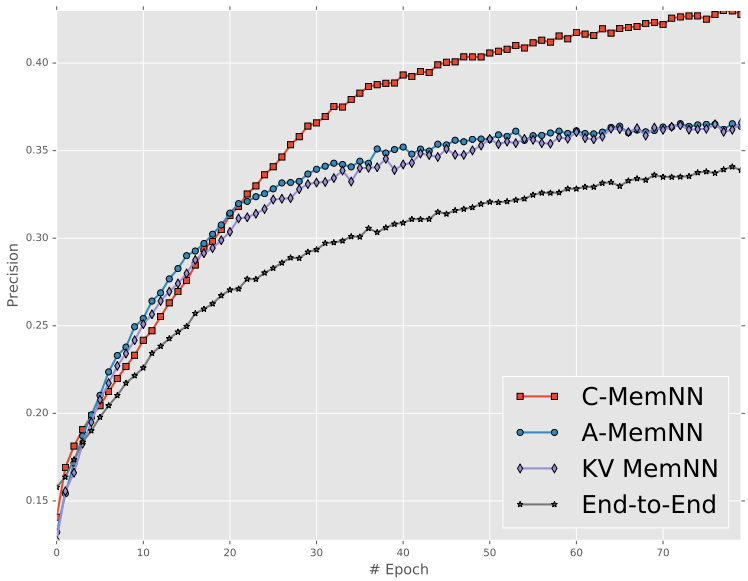


Figure 8.3: Precision@5 plot for various models on validation data (at 4 hops).

		# classes = 50			# classes = 100		
# Hops	Model	AUC (macro) ↑	Average Precision @5 ↑	Hamming Loss ↓	AUC (macro) ↑	Average Precision @5 ↑	Hamming Loss ↓
3	End-to-End	0.759	0.32	0.06	0.664	0.23	0.15
	KV MemNN	0.761	0.36	**0.05**	0.679	0.24	0.14
	A-MemNN	0.762	0.36	0.06	0.675	0.23	0.14
	C-MemNN	**0.785**	**0.39**	**0.05**	**0.697**	**0.27**	**0.12**
4	End-to-End	0.760	0.33	0.04	0.672	0.24	0.15
	KV MemNN	0.776	0.35	0.04	0.683	0.24	0.13
	A-MemNN	0.775	0.37	0.03	0.689	0.23	0.11
	C-MemNN	**0.795**	**0.42**	**0.02**	**0.705**	**0.27**	**0.09**
5	End-to-End	0.761	0.34	0.04	0.683	0.25	0.14
	KV MemNN	0.775	0.36	0.03	0.697	0.25	0.11
	A-MemNN	0.804	0.40	0.02	0.720	0.29	0.11
	C-MemNN	**0.833**	**0.42**	**0.01**	**0.767**	**0.32**	**0.05**

Table 8.3: Evaluation results of various memory networks on MIMIC-III dataset.

on **10%**. The remaining **10%** is used as test set which is evaluated only once across all experiments with different models.

8.5.2 Results and Analysis

We present experiments in which performance is evaluated using three metrics: the area under the ROC curve (AUC), the average precision over the top five predictions, and the hamming loss. The AUC is calculated by taking unweighted mean of the AUC values for each label - this is also known as the macro AUC. Average precision over the top five predictions is reported because it is a relevant metric for real world applications. Hamming loss is reported instead of accuracy because it is a better measure for multi-label classification [EW01].

As shown in Table 3, C-MemNN is able to exceed the results of various other memory networks across all experiments. The improvement is more pronounced with a higher number of memory hops. This is because of the learning saturation of vanilla memory networks over multiple hops. While A-MemNN has better results for higher hops it does not improve upon KV-MemNN at lower hops. The strength of our model lies at higher hops, as the condensed memory state $\widetilde{\boldsymbol{u}}$ after several hops contains more information than the same size input memory state $\boldsymbol{u}$. Across all models, results improve as the number of hops increases, although with diminishing returns. The AUC value of C-MemNN with five memory hops for 100 labels is higher than the AUC value for End-to-End models trained only for three hops, which shows efficient training of higher hops produces good results.

Most documents do not have five labels (Figure 1) and thus precision obtained for five predictions is poor across all models. Hamming Loss correlates very well with other metrics along with the cross-entropy loss function, which was used for training.

Our model has 30% more parameters than standard Memory Networks ($\boldsymbol{D}$ for every

pair of $\boldsymbol{A}$ and $\boldsymbol{C}$). Figure 3 shows that adding more memory does not lead to overfitting. Training time of our model for GPU implementation is same as other memory networks, as condensed memory state ($\widetilde{\boldsymbol{u}}$) is trained almost in parallel with standard memory state ($\boldsymbol{u}$) for every hop.

8.6 Conclusion

Weston, Chopra, and Bordes (2014) discussed the possibility of a better memory representation for complex inferencing tasks. We achieved a better memory representation by condensing the previous hops in a novel way to obtain a hierarchical representation of the internal memory. We have shown the efficacy of the proposed memory representation for clinical diagnostic inferencing from raw textual data. We discussed the limitations of memory networks for multi-label classification and explored gated addressing to achieve a better mapping between the clinical notes and the memory slots. We have shown that training multiple hops with condensed representation is helpful, but this is still computationally expensive. We plan to investigate asynchronous memory updating, which will allow for faster training of memory networks. In the future, we will explore other knowledge sources and the recently proposed word vectors for the biomedical domain [Chi+16].

Bibliography

[Ada01] Michael D. Adams. "The JPEG - 2000 Still Image Compression Standard (Last Revised June 30 , 2001)". In: 2001.

[AHS17] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. "Structured Pruning of Deep Convolutional Neural Networks". In: *JETC* 13 (2017), pp. 321–3218.

[ALM17] Naveed Akhtar, Jian Liu, and Ajmal Mian. "Defense against Universal Adversarial Perturbations". In: *arXiv preprint arXiv1711.05929* (2017).

[And+16] Jacob Andreas et al. "Learning to Compose Neural Networks for Question Answering". In: *arXiv preprint arXiv:1601.01705* (2016).

[Ant+15a] Stanislaw Antol et al. "VQA Visual Question Answering". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 2425–2433.

[Ant+15b] Stanislaw Antol et al. "VQA: Visual question answering". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2425–2433.

[Ant+92] Marc Antonini et al. "Image coding using wavelet transform". In: *IEEE Transactions on image processing* 1.2 (1992), pp. 205–220.

[Aro+14] Sanjeev Arora et al. "Provable Bounds for Learning Some Deep Representations". In: *ICML*. 2014.

[Bax90] William G Baxt. "Use of an artificial neural network for data analysis in clinical decision-making: the diagnosis of acute coronary occlusion". In: *Neural computation* 2.4 (1990), pp. 480–489.

[BBL97] Léon Bottou, Yoshua Bengio, and Yann LeCun. "Global Training of Document Processing Systems Using Graph Transformer Networks". In: *CVPR*. 1997.

[BC14] Jimmy Ba and Rich Caruana. "Do deep nets really need to be deep?" In: *Advances in neural information processing systems*. 2014, pp. 2654–2662.

[BCB05] C. Bannard and C. Callison-Burch. "Paraphrasing with Bilingual Parallel Corpora". In: *Proceedings of ACL*. Ann Arbor, Michigan, 2005, pp. 597–604.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[BCB15] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proceedings of ICLR*. San Diego, CA, 2015, pp. 1–15.

[Boj+16] Mariusz Bojarski et al. "End to End Learning for Self-Driving Cars". In: *arXiv preprint arXiv1604.07316* (2016).

[Bro+17] Andrew Brock et al. "Neural Photo Editing with Introspective Adversarial Networks". In: *ICLR* abs/1609.07093 (2017).

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. "Learning Long-Term Dependencies with Gradient Descent is Difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[CAB16] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. "Hierarchical Multiscale Recurrent Neural Networks". In: *arXiv preprint arXiv:1609.01704* (2016).

[CBS15] Edward Choi, Mohammad Taha Bahadori, and Jimeng Sun. "Doctor AI: Predicting Clinical Events via Recurrent Neural Networks". In: *arXiv preprint arXiv:1511.05942* (2015).

[CD11] D. Chen and W. B. Dolan. "Collecting Highly Parallel Data for Paraphrase Evaluation". In: *Proceedings of ACL-HLT*. Portland, Oregon, 2011, pp. 190–200.

[CE99] Surendar Chandra and Carla Schlatter Ellis. "JPEG compression metric as a quality aware image transcoding". In: *2nd USENIX Symposium on Internet Technologies and Systems (USITS99)*. 1999.

[Cha+16] Sarath Chandar et al. "Hierarchical Memory Networks". In: *arXiv preprint arXiv:1605.07427* (2016).

[Cha+17] Aditya Chattopadhyay et al. "Grad-CAM++ Generalized Gradient-based Visual Explanations for Deep Convolutional Networks". In: *CoRR* abs/1710.11063 (2017).

[Che+17] Yunpeng Chen et al. "Training Group Orthogonal Neural Networks with Privileged Information". In: *IJCAI*. 2017.

[Chi+16] Billy Chiu et al. "How to train good word embeddings for biomedical NLP". In: *ACL 2016* (2016), p. 166.

[Cho+14] K. Cho et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of EMNLP*. Doha, Qatar, 2014, pp. 1724–1734.

[Cho+16] Edward Choi et al. "RETAIN: Interpretable Predictive Model in Healthcare using Reverse Time Attention Mechanism". In: *CoRR* abs/1608.05745 (2016).

[Cla+11] J. H. Clark et al. "Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability". In: *Proceedings of ACL-HLT*. Portland, Oregon, 2011, pp. 176–181.

[Cog+16] Michael Cogswell et al. "Reducing Overfitting in Deep Networks by Decorrelating Representations". In: *ICLR* abs/1511.06068 (2016).

[Cui+16] Yiming Cui et al. "Attention-over-Attention Neural Networks for Reading Comprehension". In: *CoRR* abs/1607.04423 (2016).

[CW17a] Nicholas Carlini and David A. Wagner. "Adversarial Examples Are Not Easily Detected Bypassing Ten Detection Methods". In: *AISec@CCS*. 2017.

[CW17b] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *SSP* (2017).

[CYV00] S. Grace Chang, Bin Yu, and Martin Vetterli. "Adaptive wavelet thresholding for image denoising and compression". In: *IEEE transactions on image processing a publication of the IEEE Signal Processing Society* 9 9 (2000), pp. 1532–46.

[Dai+16] Jifeng Dai et al. "R-FCN Object Detection via Region-based Fully Convolutional Networks". In: *arXiv preprint arXiv1605.06409* (2016).

[Das+17] Nilaksh Das et al. "Keeping the Bad Guys Out Protecting and Vaccinating Deep Learning with JPEG Compression". In: *CoRR* abs/1705.02900 (2017).

[DBQ05] B. Dolan, C. Brockett, and C. Quirk. "Microsoft Research Paraphrase Corpus". In: *Retrieved March* 29 (2005), p. 2008.

[Den+09a] J. Deng et al. "ImageNet A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[Den+09b] Jia Deng et al. "ImageNet A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255.

[DGR16] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. "A study of the effect of JPG compression on adversarial images". In: *CoRR* abs/1608.00853 (2016).

[Dhi+16] Bhuwan Dhingra et al. "Gated-Attention Readers for Text Comprehension". In: *CoRR* abs/1606.01549 (2016).

[Dia+17] Steven Diamond et al. "Dirty Pixels Optimizing Image Classification Architectures for Raw Sensor Data". In: *CoRR* abs/1701.06487 (2017).

[DJ92] David L. Donoho and Iain Johnstone. "Adapting to Unknown Smoothness via Wavelet Shrinkage". In: 1992.

[DJ94] David L. Donoho and Iain Johnstone. "Ideal Spatial Adaptation by Wavelet Shrinkage". In: 1994.

[DYJ17] Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. "NeST A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm". In: *CoRR* abs/1711.02017 (2017).

[Egi+06] Karen Egiazarian et al. "New full-reference quality metrics based on HVS". In: *CD-ROM proceedings of the second international workshop on video processing and quality metrics, Scottsdale, USA*. Vol. 4. 2006.

[Erh+14] Dumitru Erhan et al. “Scalable Object Detection Using Deep Neural Networks”. In: *CVPR* (2014).

[Eve+10] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010).

[EW01] André Elisseeff and Jason Weston. “A kernel method for multi-labelled classification”. In: *NIPS*. 2001, pp. 681–687.

[Far+13] C. Farabet et al. “Learning Hierarchical Features for Scene Labeling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1915–1929.

[FC18] Jonathan Frankle and Michael Carbin. “The Lottery Ticket Hypothesis Training Pruned Neural Networks”. In: *CoRR* abs/1803.03635 (2018).

[Fei+17] Reuben Feinman et al. “Detecting Adversarial Samples from Artifacts”. In: *CoRR* abs/1703.00410 (2017).

[Fie87] David J Field. “Relations between the statistics of natural images and the response properties of cortical cells”. In: *Josa a* 4.12 (1987), pp. 2379–2394.

[Fur+18] Tommaso Furlanello et al. “Born Again Neural Networks”. In: *ICML*. 2018.

[FZE13] A. Fader, L. S Zettlemoyer, and O. Etzioni. “Paraphrase-Driven Learning for Open Question Answering”. In: *ACL*. ACL. 2013, pp. 1608–1618.

[GB10] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *AISTATS*. 2010.

[Get12] Pascal Getreuer. “Rudin-Osher-Fatemi Total Variation Denoising using Split Bregman”. In: *IPOL Journal* 2 (2012), pp. 74–95.

[GHP07] Gregory Griffin, Alex Holub, and Pietro Perona. "Caltech-256 object category dataset". In: (2007).

[Gir+14] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[Gir15] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[GJM13] A. Graves, N. Jaitly, and A. Mohamed. "Hybrid Speech Recognition with Deep Bidirectional LSTM". In: *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE. 2013, pp. 273–278.

[GPG12] Giaime Ginesu, Maurizio Pintus, and Daniele D Giusto. "Objective assessment of the WebP image coding algorithm". In: *Signal Processing Image Communication* 27.8 (2012).

[Gra13] A. Graves. "Generating Sequences with Recurrent Neural Networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[GSS14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *CoRR* abs/1412.6572 (2014).

[Guo+17a] Chuan Guo et al. "Countering Adversarial Images using Input Transformations". In: *arXiv preprint* (2017).

[Guo+17b] Chuan Guo et al. "Countering Adversarial Images using Input Transformations". In: 2017.

[GWD14a] A. Graves, G. Wayne, and I. Danihelka. "Neural Turing Machines". In: *arXiv:1410.5401*. 2014.

[GWD14b] Alex Graves, Greg Wayne, and Ivo Danihelka. "Neural turing machines". In: *arXiv preprint arXiv:1410.5401* (2014).

[Han+15] Song Han et al. "Learning both Weights and Connections for Efficient Neural Networks". In: *NIPS*. 2015.

[Han+16] Song Han et al. "DSD Dense-Sparse-Dense Training for Deep Neural Networks". In: 2016.

[Has+07] S. Hassan et al. "UNT: SubFinder: Combining Knowledge Sources for Automatic Lexical Substitution". In: *Proceedings of SemEval*. Prague, Czech Republic, 2007, pp. 410–413.

[Has+14] S. A. Hasan et al. "A Hybrid Approach to Clinical Question Answering". In: *TREC*. 2014.

[Has+15] S. A. Hasan et al. "Using Neural Embeddings for Diagnostic Inferencing in Clinical Question Answering". In: *TREC*. 2015.

[Has+16] S. A. Hasan et al. "Clinical Question Answering using Key-Value Memory Networks and Knowledge Graph". In: *TREC*. 2016.

[He+16a] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[He+16b] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[He+16c] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[He+17] Kaiming He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988.

[Her+15] Karl Moritz Hermann et al. "Teaching Machines to Read and Comprehend". In: *NIPS*. 2015.

[HMD16] Song Han, Huizi Mao, and William J. Dally. "Deep Compression Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding". In: *ICLR* abs/1510.00149 (2016).

[Hot36] Harold Hotelling. "Relations between two sets of variates". In: *Biometrika* 28.3/4 (1936), pp. 321–377.

[HS92] Babak Hassibi and David G. Stork. "Second Order Derivatives for Network Pruning Optimal Brain Surgeon". In: *NIPS*. 1992.

[HS97] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[HSST04] David R. Hardoon, Sándor Szedmák, and John Shawe-Taylor. "Canonical Correlation Analysis An Overview with Application to Learning Methods". In: *Neural Computation* 16 (2004), pp. 2639–2664.

[Hu+16] Hengyuan Hu et al. "Network Trimming A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures". In: *CoRR* abs/1607.03250 (2016).

[Hua+15] Xun Huang et al. "SALICON Reducing the Semantic Gap in Saliency Prediction by Adapting Deep Neural Networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 262–270.

[Hua+16] Gao Huang et al. "Deep Networks with Stochastic Depth". In: *ECCV*. 2016.

[Hua+17] Gao Huang et al. "Densely Connected Convolutional Networks". In: *CVPR* (2017), pp. 2261–2269.

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *CoRR* abs/1503.02531 (2015).

[HW59] David H. Hubel and Torsten N. Wiesel. "Receptive fields of single neurones in the cat's striate cortex." In: *The Journal of physiology* 148 (1959), pp. 574–91.

[HZS17] Yihui He, Xiangyu Zhang, and Jian Sun. "Channel Pruning for Accelerating Very Deep Neural Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 1398–1406.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML*. 2015.

[Jan12] Maarten Jansen. *Noise reduction by wavelet thresholding*. Vol. 161. Springer Science & Business Media, 2012.

[Jia+15a] Aiwen Jiang et al. "Compositional Memory for Visual Question Answering". In: *arXiv preprint arXiv:1511.05676* (2015).

[Jia+15b] Ming Jiang et al. "SALICON Saliency in context". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015.

[JLP15] Dhruv Batra Jiasen Lu Xiao Lin and Devi Parikh. *Deeper LSTM and normalized CNN Visual Question Answering model*. https://github.com/VT-vision-lab/VQA_LSTM_CNN. 2015.

[JM15] Armand Joulin and Tomas Mikolov. "Inferring algorithmic patterns with stack-augmented recurrent nets". In: *NIPS*. 2015, pp. 190–198.

[Joh+16] Alistair EW Johnson et al. "MIMIC-III, a freely accessible critical care database". In: *Scientific data* 3 (2016).

[Jou+16] Armand Joulin et al. "Bag of Tricks for Efficient Text Classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam A Method for Stochastic Optimization". In: *ICLR* abs/1412.6980 (2015).

[KFF15] Andrej Karpathy and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3128–3137.

[KGB16] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *CoRR* abs/1607.02533 (2016).

[Kir+15] R. Kiros et al. "Skip-Thought Vectors". In: *Advances in Neural Information Processing Systems*. 2015, pp. 3294–3302.

[KMVS03] R. Kozlowski, K. F. McCoy, and K. Vijay-Shanker. "Generation of Single-sentence Paraphrases from Predicate/Argument Structure Using Lexico-grammatical Resources". In: *Proceedings of the 2nd International Workshop on Paraphrasing*. Sapporo, Japan, 2003, pp. 1–8.

[KRR16] V. Kolesnyk, T. Rocktäschel, and S. Riedel. "Generating Natural Language Inference Chains". In: *CoRR* abs/1606.01404 (2016).

[KSC92] Stanley A Klein, D Amnon Silverstein, and Thom Carney. "Relevance of human vision to JPEG-DCT compression". In: *SPIE/IS&T 1992 Symposium on Electronic Imaging Science and Technology*. International Society for Optics and Photonics. 1992.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012.

[KT98] Konstantinos Konstantinides and Daniel Tretter. "A method for variable quantization in JPEG for improved text quality in compound documents". In: *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*. Vol. 2. IEEE. 1998.

[KTB14a] Matthias Kúmmerer, Lucas Theis, and Matthias Bethge. "Deep gaze i Boosting saliency prediction with feature maps trained on imagenet". In: *arXiv preprint arXiv1411.1045* (2014).

[KTB14b] Matthias Kúmmerer, Lucas Theis, and Matthias Bethge. "Deep Gaze I Boosting Saliency Prediction with Feature Maps Trained on ImageNet". In: *CoRR* abs/1411.1045 (2014).

[KVR15] Louis Kerofsky, Rahul Vanam, and Yuriy Reznik. "Perceptual adaptation of Objective video quality metrics". In: *Proc. Ninth International Workshop on Video Processing and Quality Metrics (VPQM)*. 2015.

[LA07] A. Lavie and A. Agarwal. "METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments". In: *Proceedings of the Second Workshop on Statistical Machine Translation*. Prague, Czech Republic, 2007, pp. 228–231.

[LB95] Yann LeCun and Yoshua Bengio. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995).

[LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. "Network In Network". In: *CoRR* abs/1312.4400 (2014).

[LDN10] C. Liu, D. Dahlmeier, and H. T. Ng. "PEM: A Paraphrase Evaluation Metric Exploiting Parallel Texts". In: *Proceedings of EMNLP*. Cambridge, Massachusetts, 2010, pp. 923–932.

[LDS89] Yann LeCun, John S. Denker, and Sara A. Solla. "Optimal Brain Damage". In: *NIPS*. 1989.

[Li+16] Yixuan Li et al. "Convergent Learning Do different neural networks learn the same representations?" In: *ICLR*. 2016.

[Li+17] Hao Li et al. "Pruning Filters for Efficient ConvNets". In: *ICLR* abs/1608.08710 (2017).

[Lia+17] Bin Liang et al. "Detecting Adversarial Examples in Deep Networks with Adaptive Noise Reduction". In: *arXiv preprint* (2017).

[Lin+14] T. Lin et al. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755.

[Lin+17] Yen-Chen Lin et al. "Detecting Adversarial Attacks on Neural Network Policies with Visual Foresight". In: *CoRR* abs/1710.00814 (2017).

[Lin+18] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

[Lip+15] Zachary Chase Lipton et al. "Learning to Diagnose with LSTM Recurrent Neural Networks". In: *CoRR* abs/1511.03677 (2015).

[Liu+15] Nian Liu et al. "Predicting eye fixations using convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[Liu+16] Yanpei Liu et al. "Delving into Transferable Adversarial Examples and Black-box Attacks". In: *CoRR* abs/1611.02770 (2016).

[Liu+17a] Chenxi Liu et al. "Progressive Neural Architecture Search". In: *CoRR* abs/1712.00559 (2017).

[Liu+17b] Zhuang Liu et al. "Learning Efficient Convolutional Networks through Network Slimming". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2755–2763.

[LRH15] Chi Li, Austin Reiter, and Gregory D Hager. "Beyond spatial pooling fine-grained representation learning in multiple domains". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[Lu+16] Jiasen Lu et al. "Hierarchical Question-Image Co-Attention for Visual Question Answering". In: *arXiv preprint arXiv:1606.00061* (2016).

[Luo+15] Yan Luo et al. "Foveation-based Mechanisms Alleviate Adversarial Examples". In: *CoRR* abs/1511.06292 (2015).

[LW15] X. Li and X. Wu. "Constructing Long Short-term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition". In: *2015*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. 2015, pp. 4520–4524.

[LWL17] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. "ThiNet A Filter Level Pruning Method for Deep Neural Network Compression". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 5068–5076.

[Mad+17] Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *CoRR* abs/1706.06083 (2017).

[Mar80] Stjepan Marčelja. "Mathematical description of the responses of simple cortical cells." In: *Journal of the Optical Society of America* 70 11 (1980), pp. 1297–300.

[MC17] Dongyu Meng and Hao Chen. "MagNet A Two-Pronged Defense against Adversarial Examples". In: *CCS*. 2017.

[McK83] K. R. McKeown. "Paraphrasing Questions Using Given and New Information". In: *Computational Linguistics* 9.1 (1983), pp. 1–10.

[MD10] N. Madnani and B. J. Dorr. "Generating Phrasal and Sentential Paraphrases: A Survey of Data-driven Methods". In: *Computational Linguistics* 36.3 (2010), pp. 341–387.

[MDFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool A Simple and Accurate Method to Fool Deep Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2574–2582.

[MHG+14] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. "Recurrent models of visual attention". In: *Advances in Neural Information Processing Systems*. 2014.

[Mik+14] T. Mikolov et al. *Word2Vec*. https://code.google.com/p/word2vec. Online; accessed 2014-04–15. 2014.

[Mil+16] Alexander Miller et al. "Key-Value Memory Networks for Directly Reading Documents". In: *CoRR* abs/1606.03126 (2016).

[Miy+15] Takeru Miyato et al. "Distributional Smoothing with Virtual Adversarial Training". In: 2015.

[Mol+17] Pavlo Molchanov et al. "Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning". In: *ICLR* abs/1611.06440 (2017).

[Mor+17] Ari S. Morcos et al. "On the importance of single directions for generalization". In: *CoRR* abs/1803.06959 (2017).

[MT00] Nasir D Memon and Daniel R Tretter. "Method for variable quantization in JPEG for improved perceptual quality". In: *Electronic Imaging*. International Society for Optics and Photonics. 2000.

[MTC12] N. Madnani, J. Tetreault, and M. Chodorow. "Re-examining Machine Translation Metrics for Paraphrase Identification". In: *Proceedings of NAACL-HLT*. Montreal, Canada, 2012, pp. 182–190.

[NP17] et. al Nicolas Papernot Nicholas Carlini. "cleverhans v2.0.0 an adversarial machine learning library". In: *arXiv preprint arXiv1610.00768* (2017).

[NSH15] Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. "Image Question Answering using Convolutional Neural Network with Dynamic Parameter Prediction". In: *arXiv preprint arXiv:1511.05756* (2015).

[NYC15] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled High confidence predictions for unrecognizable images". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 427–436.

[Oqu+15a] Maxime Oquab et al. "Is object localization for free? - Weakly-supervised learning with convolutional neural networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 685–694.

[Oqu+15b] Maxime Oquab et al. "Is object localization for free?-weakly-supervised learning with convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[Pap+02] K. Papineni et al. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of ACL*. Philadelphia, Pennsylvania, USA, 2002, pp. 311–318.

[Pap+16a] Nicolas Papernot et al. "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: *SSP* (2016).

[Pap+16b] Nicolas Papernot et al. "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples". In: *CoRR* abs/1602.02697 (2016).

[Pap+16c] Nicolas Papernot et al. "The limitations of deep learning in adversarial settings". In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE. 2016.

[Pas+13] R. Pascanu et al. "How to Construct Deep Recurrent Neural Networks". In: *arXiv preprint arXiv:1312.6026* (2013).

[Pas+15] Razvan Pascanu et al. “Malware classification with recurrent networks”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2015*. IEEE. 2015.

[Pav+15] E. Pavlick et al. “PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification”. In: *Proceedings of ACL-IJCNLP)*. Beijing, China, 2015, pp. 425–430.

[PB16] Aaditya Prakash and James Storer Brandeis. “Highway Networks for Visual Question Answering”. In: 2016.

[Pon+07] Nikolay Ponomarenko et al. “On between-coefficient contrast masking of DCT basis functions”. In: *Proceedings of the third international workshop on video processing and quality metrics*. Vol. 4. 2007.

[Pra+16a] Aaditya Prakash et al. “Condensed Memory Networks for Clinical Diagnostic Inferencing”. In: *AAAI*. 2016.

[Pra+16b] Aaditya Prakash et al. “Neural Paraphrase Generation with Stacked Residual LSTM Networks”. In: *COLING*. 2016.

[Pra+17] Aaditya Prakash et al. “Semantic Perceptual Image Compression Using Deep Convolution Networks”. In: *2017 Data Compression Conference (DCC)* (2017).

[Pra+18a] Aaditya Prakash et al. “Deflecting Adversarial Attacks with Pixel Deflection”. In: *arXiv preprint arXiv1801.08926* (2018).

[Pra+18b] Aaditya Prakash et al. “Protecting JPEG Images Against Adversarial Attacks”. In: *2018 Data Compression Conference* (2018), pp. 137–146.

[Pra+19] Aaditya Prakash et al. “RePr: Improved Training of Convolutional Filters”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* abs/1811.07275 (2019).

[PSDG13] Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. "Analyzing noise in autoencoders and deep networks". In: *NIPS Workshop on Deep Learning* abs/1406.1831 (2013).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global Vectors for Word Representation." In: *EMNLP*. Vol. 14. 2014, pp. 1532–1543.

[QBD04] C. Quirk, C. Brockett, and W. Dolan. "Monolingual Machine Translation for Paraphrase Generation". In: *Proceedings of EMNLP*. Barcelona, Spain, 2004, pp. 142–149.

[Rag+17a] Maithra Raghu et al. "On the expressive power of deep neural networks". In: *ICML*. 2017.

[Rag+17b] Maithra Raghu et al. "SVCCA Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability". In: *NIPS*. 2017.

[Rea+18] Esteban Real et al. "Regularized Evolution for Image Classifier Architecture Search". In: *CoRR* abs/1802.01548 (2018).

[Ren+15] Shaoqing Ren et al. "Faster R-CNN Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015.

[Rez+13] Yuriy Reznik et al. *Coding of feature location information*. US Patent 8,571,306. 2013.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net Convolutional Networks for Biomedical Image Segmentation". In: *MICCAI*. 2015.

[Ric11] Thomas Richter. "SSIM as global quality metric a differential geometry view". In: *Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on*. IEEE. 2011.

[RK09] Thomas Richter and Kil Joong Kim. "A ms-ssim optimal jpeg 2000 encoder". In: *2009 Data Compression Conference*. IEEE. 2009.

[RKZ15] Mengye Ren, Ryan Kiros, and Richard Zemel. "Exploring models and data for image question answering". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2935–2943.

[RL12] V. Rus and M. Lintean. "A Comparison of Greedy and Optimal Assessment of Natural Language Student Input using Word-to-Word Similarity Metrics". In: *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. ACL. 2012, pp. 157–162.

[Rod+17] Pau Rodríguez et al. "Regularizing CNNs with Locally Constrained Decorrelations". In: *ICLR* abs/1611.01967 (2017).

[Rom+15] Adriana Romero et al. "FitNets Hints for Thin Deep Nets". In: *ICLR* (2015).

[Rus+05] Nicole C. Rust et al. "Spatiotemporal Elements of Macaque V1 Receptive Fields". In: *Neuron* 46 (2005), pp. 945–956.

[Rus+15] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115 (2015), pp. 211–252.

[RVS02] Raghuram Rangarajan, Ramji Venkataramanan, and Siddharth Shah. "Image Denoising Using Wavelets". In: 2002.

[SB06] Hamid R Sheikh and Alan C Bovik. "Image information and visual quality". In: *IEEE Transactions on Image Processing* 15.2 (2006).

[SBB16] Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. "Iterative Alternating Neural Attention for Machine Reading". In: *CoRR* abs/1606.02245 (2016).

[Ser+16] I. V. Serban et al. "Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation". In: *arXiv preprint arXiv:1606.00776* (2016).

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).

[Sha+16] Wenling Shang et al. "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units". In: *ICML*. 2016.

[Sim99] Eero P. Simoncelli. "Bayesian Denoising of Visual Images in the Wavelet Domain". In: 1999.

[SL09] X Yu Stella and Dimitri A Lisin. "Image compression based on visual saliency at individual scales". In: *International Symposium on Visual Computing*. Springer. 2009.

[SMG14] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *ICLR* abs/1312.6120 (2014).

[SMH11] I. Sutskever, J. Martens, and G. E. Hinton. "Generating Text with Recurrent Neural Networks". In: *Proceedings of ICML*. Bellevue, WA, 2011, pp. 1017–1024.

[Smi17] Leslie N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017), pp. 464–472.

[Sno+06] M. Snover et al. "A Study of Translation Edit Rate with Targeted Human Annotation". In: *Proceedings of Association for Machine Translation in the Americas*. 2006, pp. 223–231.

[Soc+11] R. Socher et al. "Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1–9.

[Sri+14a] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[Sri+14b] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[SSH15] Kevin J Shih, Saurabh Singh, and Derek Hoiem. "Where To Look: Focus Regions for Visual Question Answering". In: *arXiv preprint arXiv:1511.07394* (2015).

[SSN12] M. Sundermeyer, R. Schlüter, and H. Ney. "LSTM Neural Networks for Language Modeling". In: *Interspeech*. 2012, pp. 194–197.

[Suk+15] Sainbayar Sukhbaatar et al. "End-To-End Memory Networks". In: *NIPS*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 2440–2448.

[SVL14] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Annual Conference on Neural Information Processing Systems*. Montreal, Canada, 2014, pp. 3104–3112.

[SVS17] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. "One pixel attack for fooling deep neural networks". In: *CoRR* abs/1710.08864 (2017).

[SWF+15] S. Sukhbaatar, J. Weston, R. Fergus, et al. "End-to-End Memory Networks". In: *Advances in neural information processing systems*. 2015, pp. 2440–2448.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014).

[Sze+13a] Christian Szegedy et al. "Intriguing properties of neural networks". In: *CoRR* abs/1312.6199 (2013).

[Sze+13b] Christian Szegedy et al. "Intriguing properties of neural networks". In: *CoRR* abs/1312.6199 (2013).

[Sze+15] Christian Szegedy et al. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1–9.

[Sze+16] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2818–2826.

[Tap+15] Makarand Tapaswi et al. "Movieqa: Understanding stories in movies through question-answering". In: *arXiv preprint arXiv:1512.02902* (2015).

[TH12] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp Divide the gradient by a running average of its recent magnitude*. COURSERA Neural Networks for Machine Learning. 2012.

[Tod+16] George Toderici et al. "Full Resolution Image Compression with Recurrent Neural Networks". In: *arXiv preprint arXiv1608.05148* (2016).

[TPN96] Soon Hie Tan, Khee K Pang, and King N Ngan. "Classified perceptual coding with adaptive quantization". In: *IEEE transactions on circuits and systems for video technology* 6.4 (1996).

[Tra+17a] Florian Tramèr et al. "Ensemble Adversarial Training Attacks and Defenses". In: *CoRR* abs/1705.07204 (2017).

[Tra+17b] Florian Tramèr et al. "The Space of Transferable Adversarial Examples". In: *CoRR* abs/1704.03453 (2017).

[Tre11] Lyndal Trevena. "WikiProject medicine". In: *BMJ* 342 (2011), p. d3387.

[Vin+14] O. Vinyals et al. "Show and Tell: A Neural Image Caption Generator". In: *CoRR* abs/1411.4555 (2014).

[Vin+15] O. Vinyals et al. "Grammar as a Foreign Language". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2773–2781.

[Vor+17] Eugene Vorontsov et al. "On orthogonality and learning recurrent networks with long term dependencies". In: *ICML*. 2017.

[Wal92] Gregory K Wallace. "The JPEG still picture compression standard". In: *IEEE transactions on consumer electronics* 38.1 (1992), pp. xviii–xxxiv.

[Wan+04a] Zhou Wang et al. "Image quality assessment from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004).

[Wan+04b] Zhou Wang et al. "Image quality assessment from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[Wan+13] Li Wan et al. "Regularization of Neural Networks using DropConnect". In: *ICML*. 2013.

[Wan01] Shouhong Wang. "Designing information systems for electronic commerce". In: *Industrial Management & Data Systems* 101.6 (2001), pp. 304–315.

[WBK10] S. Wubben, A. van den Bosch, and E. Krahmer. "Paraphrase Generation As Monolingual Translation: Data and Evaluation". In: *Proceedings of INLG*. Trim, Ireland, 2010, pp. 203–207.

[WCB14] Jason Weston, Sumit Chopra, and Antoine Bordes. "Memory Networks". In: *CoRR* abs/1410.3916 (2014).

[Wes+15] Jason Weston et al. "Towards ai-complete question answering: A set of prerequisite toy tasks". In: *arXiv preprint arXiv:1502.05698* (2015).

[WG15] Haibing Wu and Xiaodong Gu. "Towards Dropout Training for Convolutional Neural Networks". In: *Neural networks the official journal of the International Neural Network Society* 71 (2015), pp. 1–10.

[Wie+15] J. Wieting et al. "From Paraphrase Database to Compositional Paraphrase Model and Back". In: *Transactions of the ACL (TACL)* (2015).

[WSB03] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. "Multiscale structural similarity for image quality assessment". In: *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*. Vol. 2. Ieee. 2003.

[Wu+15] Qi Wu et al. "Ask Me Anything: Free-form Visual Question Answering Based on Knowledge from External Sources". In: *arXiv preprint arXiv:1511.06973* (2015).

[Wu+16a] Qi Wu et al. "Image Captioning and Visual Question Answering Based on Attributes and Their Related External Knowledge". In: *arXiv preprint arXiv:1603.02814* (2016).

[Wu+16b] Yuxin Wu et al. *Tensorpack*. https//github.com/tensorpack/. 2016.

[XEQ17] Weilin Xu, David Evans, and Yanjun Qi. "Feature Squeezing Detecting Adversarial Examples in Deep Neural Networks". In: *CoRR* abs/1704.01155 (2017).

[Xia+14] Guoqing Xiang et al. "An Adaptive Perceptual Quantization Algorithm Based on Block-Level JND for Video Coding". In: *Pacific Rim Conference on Multimedia*. Springer. 2014.

[Xia+18] Lechao Xiao et al. "Dynamical Isometry and a Mean Field Theory of CNNs How to Train 10,000-Layer Vanilla Convolutional Neural Networks". In: *ICML*. 2018.

[Xie+18] Cihang Xie et al. "Mitigating Adversarial Effects Through Randomization". In: *International Conference on Learning Representations*. 2018.

[XMS16] Caiming Xiong, Stephen Merity, and Richard Socher. "Dynamic Memory Networks for Visual and Textual Question Answering". In: *arXiv preprint arXiv:1603.01417* (2016).

[XPX17] Pengtao Xie, Barnabás Póczos, and Eric P. Xing. "Near-Orthogonality Regularization in Kernel Methods". In: *UAI*. 2017.

[XS15] Huijuan Xu and Kate Saenko. "Ask, Attend and Answer: Exploring Question-Guided Spatial Attention for Visual Question Answering". In: *arXiv preprint arXiv:1511.05234* (2015).

[Xu+15] Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *arXiv preprint arXiv:1502.03044* 2.3 (2015), p. 5.

[XXP17] Di Xie, Jiang Xiong, and Shiliang Pu. "All You Need is Beyond a Good Init Exploring Better Solution for Training Extremely Deep Convolutional Neural Networks with Orthonormality and Modulation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5075–5084.

[Yan+15] Zichao Yang et al. "Stacked Attention Networks for Image Question Answering". In: *arXiv preprint arXiv:1511.02274* (2015).

[Yos+15] Jason Yosinski et al. "Understanding Neural Networks Through Deep Visualization". In: *CoRR* abs/1506.06579 (2015).

[YS15] W. Yin and H. Schütze. "Convolutional Neural Network for Paraphrase Identification". In: *Proceedings of NAACL-HLT*. Denver, Colorado, 2015, pp. 901–911.

[ZF14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European Conference on Computer Vision*. Springer. 2014.

[ZG17] Michael Zhu and Suyog Gupta. "To prune, or not to prune exploring the efficacy of pruning for model compression". In: *NIPS Workshop on Machine Learning of Phones and other Consumer Devices* abs/1710.01878 (2017).

[Zha+08] S. Zhao et al. "Combining Multiple Resources to Improve SMT-based Paraphrasing Model". In: *Proceedings of ACL-HLT*. Columbus, Ohio, 2008, pp. 1021–1029.

[Zha+09] S. Zhao et al. "Application-driven Statistical Paraphrase Generation". In: *Proceedings of ACL-IJCNLP*. Suntec, Singapore, 2009, pp. 834–842.

[Zha+10] S. Zhao et al. “Leveraging Multiple MT Engines for Paraphrase Generation”. In: *Proceedings of COLING*. Beijing, China, 2010, pp. 1326–1334.

[Zha+15] Rui Zhao et al. “Saliency Detection by Multi-Context Deep Learning”. In: *CVPR*. 2015.

[Zhe+15] Shuai Zheng et al. “Conditional random fields as recurrent neural networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

[Zho+15] Bolei Zhou et al. “Simple Baseline for Visual Question Answering”. In: *arXiv preprint arXiv:1512.02167* (2015).

[Zho+16] Bolei Zhou et al. “Learning Deep Features for Discriminative Localization”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2921–2929.

[Zhu+15] Yuke Zhu et al. “Visual7W: Grounded Question Answering in Images”. In: *arXiv preprint arXiv:1511.03416* (2015).

[ZL16] Barret Zoph and Quoc V. Le. “Neural Architecture Search with Reinforcement Learning”. In: *CoRR* abs/1611.01578 (2016).

[Zün+13] Fabio Zünd et al. “Content-aware compression using saliency-driven image retargeting”. In: *2013 IEEE International Conference on Image Processing*. IEEE. 2013.

[The16] Theano Development Team. “Theano: A Python Framework for Fast Computation of Mathematical Expressions”. In: *arXiv e-prints* abs/1605.02688 (2016). URL: http://arxiv.org/abs/1605.02688.