Thesis

A Dissertation

Presented to

The Faculty of the Graduate School of Arts and Sciences

Brandeis University

Michtom School of Computer Science

James A. Storer, Advisor

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

Aaditya Prakash

May, 2019

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, Chair

Antonella DiLillo

Sadid Hasan

Abstract

Thesis

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

by Aaditya Prakash

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Introduction to Deep Learning

1.2 Convolutional Neural Networks

1.2.1 Object localization

1.3 Limitations of Convolutional Networks

1.3.1 Adversarial Images

1.3.2 Overalapping Filters

Chapter 2

Semantic Image Compression

2.1 Introduction

Several attempts have been made to improve upon the lossy image compression offered by JPEG [**xxx_ginesu2012objective**] [**xxx_toderici2016full**]. 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 fre-

quency regions are more tolerant of higher compression ratios, high frequency regions are not [**xxx_chandra1999jpegcompressionme**]. 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 [**xxx_konstantinides1998method**] as well as classification of each block according to a pre-determined look-up table [**xxx_memon2000metho** Soon et al [**xxx_tan1996classified**] 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 [**xxx_xiang2014adaptive**].

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 [**xxx_klein1992relevance**]. 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`[**xxx_ssim**], `MS-SSIM`[**xxx_msssim**], `VIFP`[**xxx_vifp**], `PSNR-HVS`[**xxx_psnrhvs**] and `PSNR-HVSM`[**xxx_psnrhvsm**]; these robust metrics have been shown to correlate better with subjective quality measurement than pixel-level metrics such as `MSE` or `PSNR` [**xxx_psnrhvsm**]. Yuri et al [**xxx_kerofsky2015perceptual**] showed that `PSNR` as an image comparison metric has severe limitations.

Convolutional Neural Networks (CNNs) have been successfully applied to a variety of com-

puter vision tasks [**xxx_he2015deep**] [**xxx_krizhevsky2012imagenet**]. Their feature extraction and transfer learning capabilities are now well known[**xxx_zeiler2014visualizing**] . 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 [**xxx_girshick2014rich**]. 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 [**xxx_jiang2015salicon**] [**xxx_kummerer2014deep**].

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 [**xxx_dai2016r**] [**xxx_girshick2014rich**] 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 [**xxx_mnih2014recurrent**] [**xxx_zund2013content**]. 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 [**xxx_stella2009image**], 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 [**xxx_liu2015predicting**].

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 [**xxx_reznik2013coding**] 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 semantic 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 [**xxx_ginesu2012objective**] and Residual-GRU [**xxx_toderici2016full**], 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 [**xxx_jiang2015salicon**]. 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 [**xxx_kerofsky2015perceptual**] showed that `PSNR` has severe limitations as an image comparison metric. Richter et al [**xxx_richter2009ms**][**xxx_richter2011ssim**] addressed structural similarity (`SSIM`[**xxx_ssim**] and `MS-SSIM`[**xxx_msssim**]) for JPEG and JPEG 2000. We evaluate on these metrics, as well as `VIFP`[**xxx_vifp**], `PSNR-HVS`[**xxx_psnrhvs**] and `PSNR-HVSM`[**xxx_psnrhvsm**], which have been shown to correlate with subjective visual

quality. Figure **??** 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 [**xxx_krizhevsky2012image**
The feature extraction and transfer learning capabilities of CNNs are well known [**xxx_zeiler2014visualizi**
as are their ability to classify images by their most prominent object [**xxx_he2015deep**],
and compute a bounding box [**xxx_girshick2014rich**]. Some success has been obtained in
predicting the visual saliency map of a given image [**xxx_jiang2015salicon**],[**xxx_kummerer2014deep**
Previous work has shown that semantic object detection has a variety of advantages over saliency maps [**xxx_mnih2014recurrent**], [**xxx_zund2013content**]. 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 [**xxx_kummerer2014deep**]. This is evident in the results obtained by Stella et al [**xxx_stella2009image**], 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 [**xxx_dai2016r**], [**xxx_girshick2014rich**] 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 [**xxx_liu2015predicting**]. 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.

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 [**xxx_lecun1995convolutional**]. 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 **??**, 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 [**xxx_lecun1995convolutional**] 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. 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 [**xxx_girshick2015fast**]. Faster RCNNs [**xxx_ren2015faster**] 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 [**xxx_long2015fully**], [**xxx_girshick2014rich**], [**xxx_zheng2015conditional**] 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) [**xxx_everingham2010pascal**]. 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 [**xxx_everingham2010pascal**]. Our approach only requires image-level labels of object classes, without pixel-level annotation or bounding-box localization.

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 [**xxx_oquab2015object**], [**xxx_zhou2015learning**] 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. 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. 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. 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 [**xxx_simonyan2014very**]. 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[**xxx_oquab2015object**]. Pooling of feature maps is probabilistic [**xxx_li2015beyond**], 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 [**xxx_zhou2015learning**] 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. Figure **??** (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. 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:

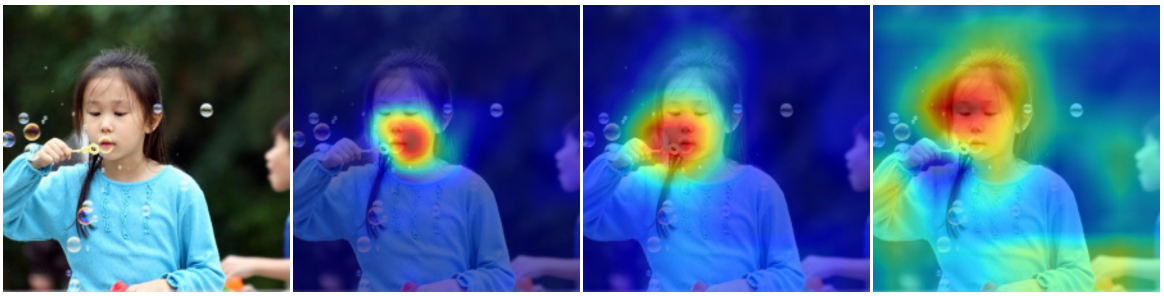
i Reduce the number of classes and increase the inter-class variance by combining similar classes

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 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 [**xxx_zeiler2014visualizing**], 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.

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



(a) Original (b) Human Fixation (c) CAM (d) MS-ROI

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 **??**. 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 **??**, we use sigmoid, which does not marginalize the posterior over the classes. Thus the likelihood of a class c is given by equation **??**.

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

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 1: JPEG Encoding with MSROI

Input : Image: I

Output: Image: O, same dimensions as I

Let $Q_{l \cdot h}$ 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 **??**.

2.5 Experimental Results

We trained our model with the Caltech-256 dataset [**xxx_griffin2007caltech**], 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 [**xxx_imagenet_cvpr09**]. 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 **??**. 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.

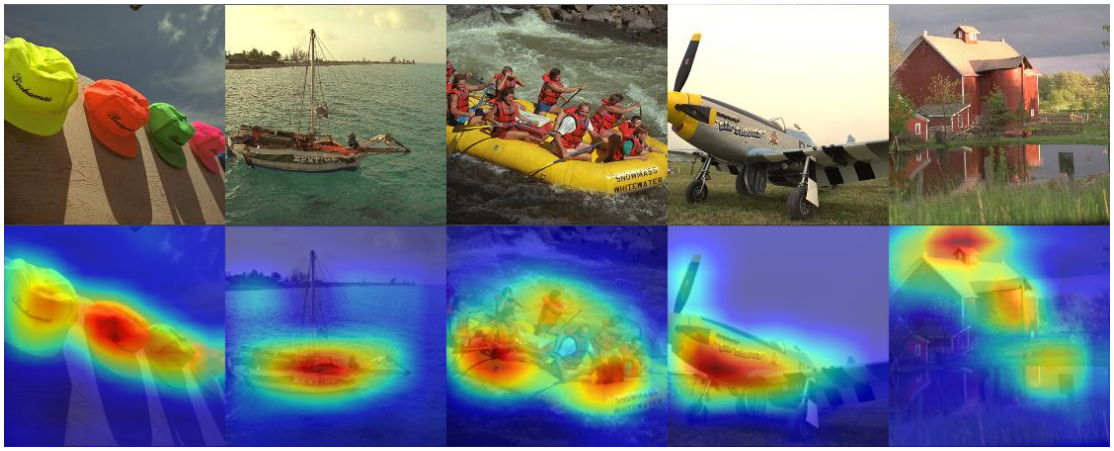


Figure 2.3: Sample of our map for five KODAK images.

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 **??** 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 **??** 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 **??** for the image sizes used in this experiment. Both Figure **??** and Figure **??** 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: **??** [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

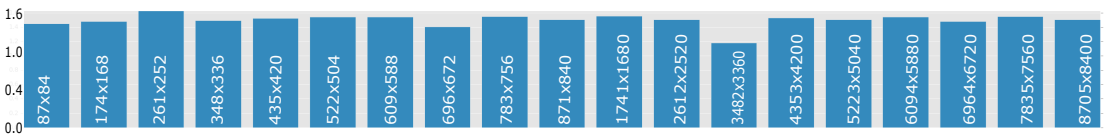


Figure 2.4: PSNR-HVS of our model - JPEG across various image size (higher is better).

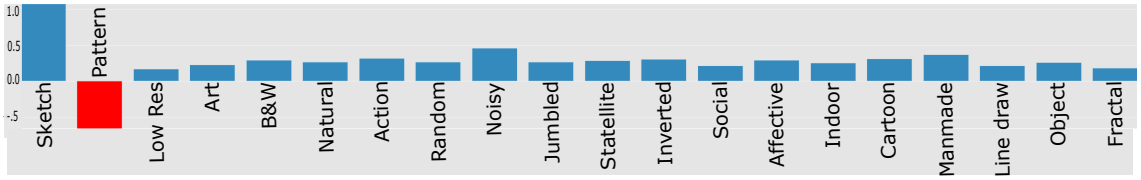


Figure 2.5: PSNR-HVS of our model - JPEG across various categories of MIT Saliency dataset (higher is better).

2.6 Discussion

While the comparison of metrics in Table **??** 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 **??** for qualitative evaluation.

The results in Table **??** 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 **??** 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 [**xxx_richter2009ms**] [**xxx_richter2011ssim**] has shown efficacy of these metric for image compression.

Figure **??** 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 **??** 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. 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

Adversarial Images

3.1 Introduction

Deep neural networks (DNNs) have shown tremendous success in image recognition tasks, even surpassing human capability [**He2016DeepRL**]. DNNs have become components of many critical systems, such as self-driving cars [**bojarski2016end**], medical image segmentation, surveillance, and malware classification [**pascanu2015malware**]. 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 [**Goodfellow2014ExplainingAH**]. In most cases, this change is imperceptible to humans (the resulting image is visually indistinguishable from the original).

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 [**Das2017KeepingTB**; **Dziugaite2016ASO**]. 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.

Several techniques have been proposed which attempt to reduce the feasibility of generating adversarial images [**Papernot2016DistillationAA**; **Miyato2015DistributionalSW**]. 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.

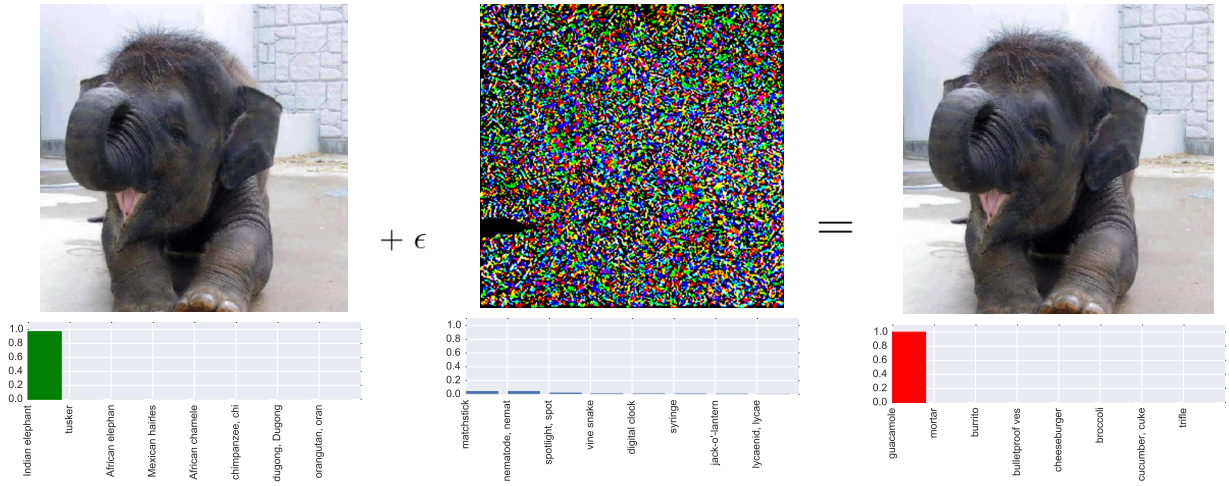


Figure 3.1: Left: Original Image, Center: Perturbed Noise, Right: Adversarial Image

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. 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 [**Prakash2017SemanticPI**], 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.

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 [**Papernot2016PracticalBA**]. Black box attacks are weaker than white box attacks; here we focus on white box attacks.

3.3 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. 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.

3.3.1 Fast Gradient Sign Method (FGSM)

[**Goodfellow2014ExplainingAH**] 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 [**Goodfellow2014ExplainingAH**]. 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.

3.3.2 Iterative Gradient Sign Method

[**Kurakin2016AdversarialEI**] 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 [**Kurakin2016AdversarialEI**].

3.3.3 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.

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

3.3.4 Deep Fool

Deep Fool [**MoosaviDezfooli2016DeepFoolAS**] 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.

3.3.5 Jacobian-based Saliency Map Attack

Jacobian-based Saliency Map Attack (JSMA) [**Carlini2017TowardsET**] 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.

3.3.6 L-BFGS

L-BFGS [**Szegedy2013IntriguingPO**] 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.

3.4 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.

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 [**Tramr2017TheSO**]. 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 [**Liu2016DelvingIT**], 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.

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 [**liang2017detecting**; **aadityaprakash2018**; **guo2017countering**]. 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.

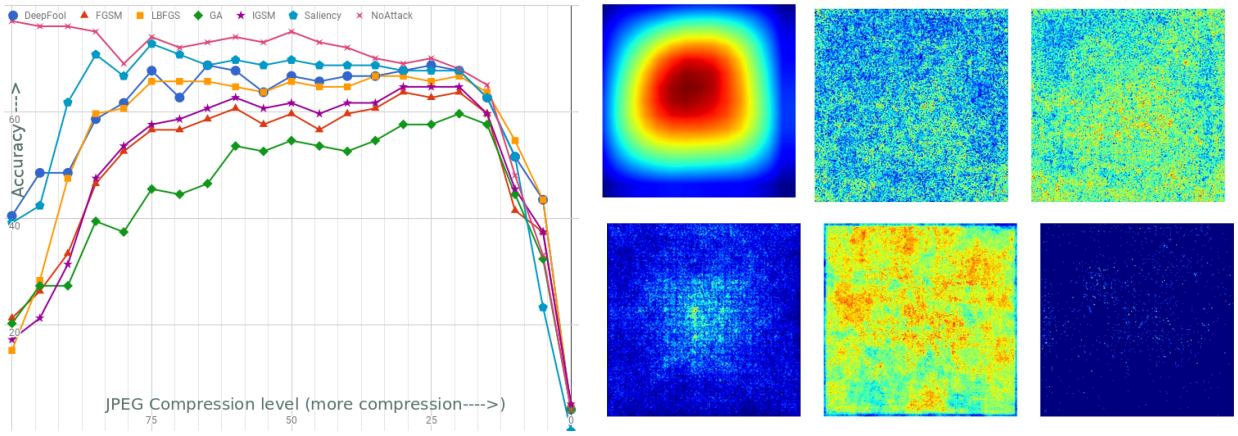


Figure 3.2: (a) Classification accuracy on adversarial images is recovered as quantization increases, but drops when quantization becomes too aggressive. *JPEG Compression level* refers to the scaling factor of the quantization matrix often denoted as Q. (b) Clockwise from top left; average MSROI saliency 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.

Limitations of JPEG - 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 [**Kurakin2016AdversarialEI**]. The impact of JPEG compression was further studied by [**Dziugaite2016ASO**] and [**Das2017KeepingTB**]. 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.

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.

3.5 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.

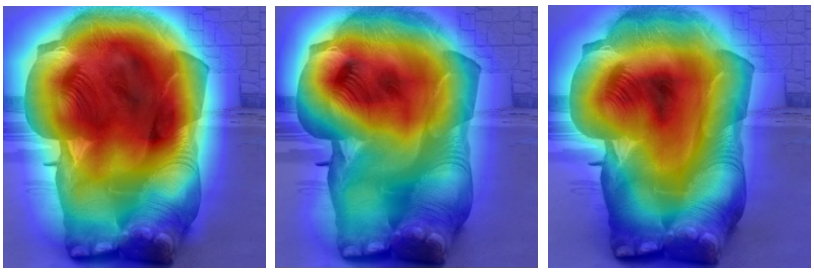


Figure 3.3: Left: MSROI on original image, Center: MSROI, Right: Aug-MSROI, on adversarial image

Convolutional neural networks (CNNs) have been used to detect and locate objects in a

given image [**Erhan2014ScalableOD**; **Zhao_2015_CVPR**]. 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 [**Prakash2017SemanticPI**], 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:

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

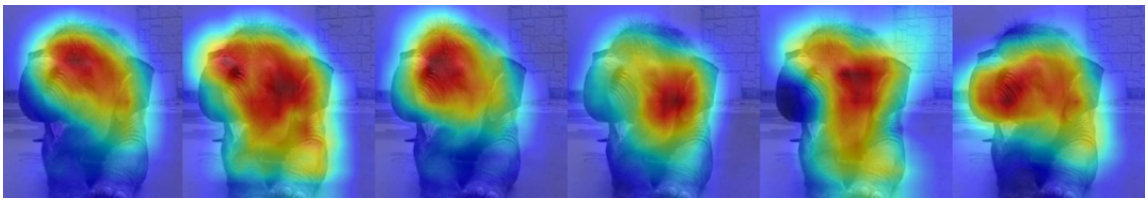


Figure 3.4: Combination of various jittered inputs

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.

3.6 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 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.

Once the semantic heat map is obtained, we use the technique proposed in [**Prakash2017SemanticPI**], 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.

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.

3.7 Experiments

We experimented with several adversarial attacks as described in Section **??**. We used Residual Network (ResNet-50) [**He2016DeepRL**], 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. 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.

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 [**He2016DeepRL**]. 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 [**Kurakin2016AdversarialEI**]. 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 –

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

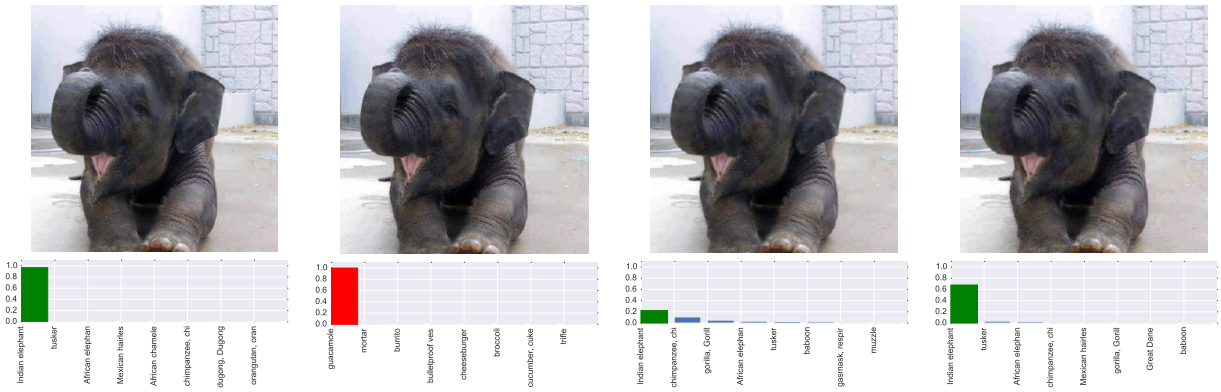


Figure 3.5: Original Image, Adversarial Image, JPEG, Aug-MSROI

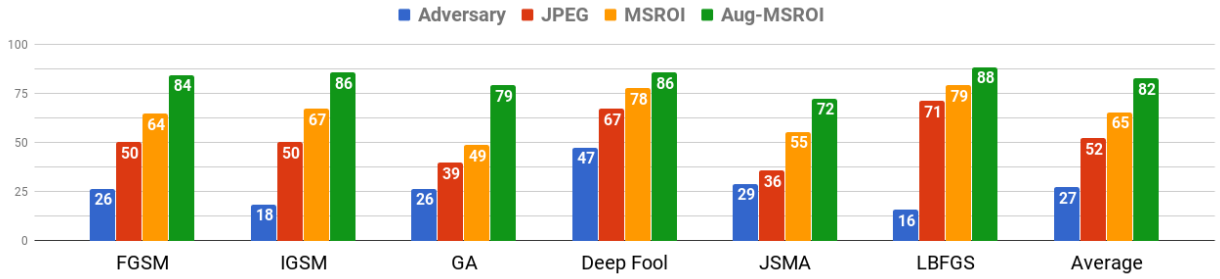


Figure 3.6: Classification accuracy of images on various attacks

3.8 Results

Figure **??** 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.

	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.

The JPEG quantization level was chosen to achieve the highest accuracy. For MSROI, we applied the technique described in [**Prakash2017SemanticPI**] 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 [**wang2004image**].

3.9 Conclusion

We have presented Aug-MSROI, an augmentation of MSROI [**Prakash2017SemanticPI**] to employ semantic JPEG compression as an effective defense against state-of-the-art adversarial attacks. 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 [**Bottou1997GlobalTO**], or by self-driving cars to identify stop signs [**Papernot2016PracticalBA**].

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 before 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 [**Szegedy2013IntriguingPO**; **Goodfellow2014ExplainingAH**; **Carlini2017TowardsET**; **MoosaviDezfooli2016DeepFoolAS**; **papernot2016limitations**; **Kurakin2016AdversarialEI**] Additionally, we show that these transformations do not significantly decrease the classifier's accuracy on non-adversarial images.

In Section **??**, we discuss the various attack techniques against which we will test our defense. In Sections **??** and **??** we discuss the established defense techniques against which we will compare our technique. In Sections **??**, **??** and **??** we lay out the components of our defense and provide the intuition behind them. In Sections **??** and **??**, 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 **??** and **??**.

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

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 [**Dziugaite2016ASO**; **Das2017KeepingTB**] 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.

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 [**Szegedy2013Intriguing** **Goodfellow2014ExplainingAH**]. 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* [**Goodfellow2014ExplainingAH**; **Kurakin2016AdversarialEI**; **Madry2017TowardsDL**; **Carlini2017TowardsET**]. 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 [**Papernot2016PracticalBA**; **Liu2016DelvingIT**].

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 $F_\theta(\cdot)$. *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)

[**Goodfellow2014ExplainingAH**] is a single step attack process. It uses the sign of the gradient of the loss function, ℓ, w.r.t. 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)

[**Kurakin2016AdversarialEI**] 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 w.r.t. 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

[**Szegedy2013IntriguingPO**] 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.

4.2.4 Jacobian-based Saliency Map Attack (JSMA)

[**papernot2016limitations**] estimates the saliency of each image pixel w.r.t. 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)

[**MoosaviDezfooli2016DeepFoolAS**] 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 w.r.t. to the original image.

4.2.6 Carlini & Wagner (C&W)

[**Carlini2017TowardsET**] 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' [**Madry2017Toward**

and 'One Pixel Attack' [**Su2017OnePA**] 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

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 [**Tramr2017EnsembleAT**], 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 [**Papernot2016DistillationAA**]. 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 [**Meng2017MagNetAT**; **akhtar2017defense**].

Most transformation-based defense strategies suffer from accuracy loss with clean images [**Dziugaite2016ASO**; **Kurakin2016AdversarialEI**], *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 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. 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 [**CounteringAIGuo17**] recently proposed transformation of image by quilting and Total Variance Minimization (TVM). 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 [**Getreuer2012RudinOsherFatemiTV**] as an image denoising technique. 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.* [**FoveationbasedMALuo2015**]. They propose a foveation-based mechanism. 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. 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.* [**MitigatingAnon208**], 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 [**EasilyFNguyen2015DeepNN**; **IntriguingSzegedy2013**]. However, most deep classifiers are robust to the presence of natural noise, such as sensor noise [**DirtyPODiamond2017**].

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 **??**. 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 2: 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

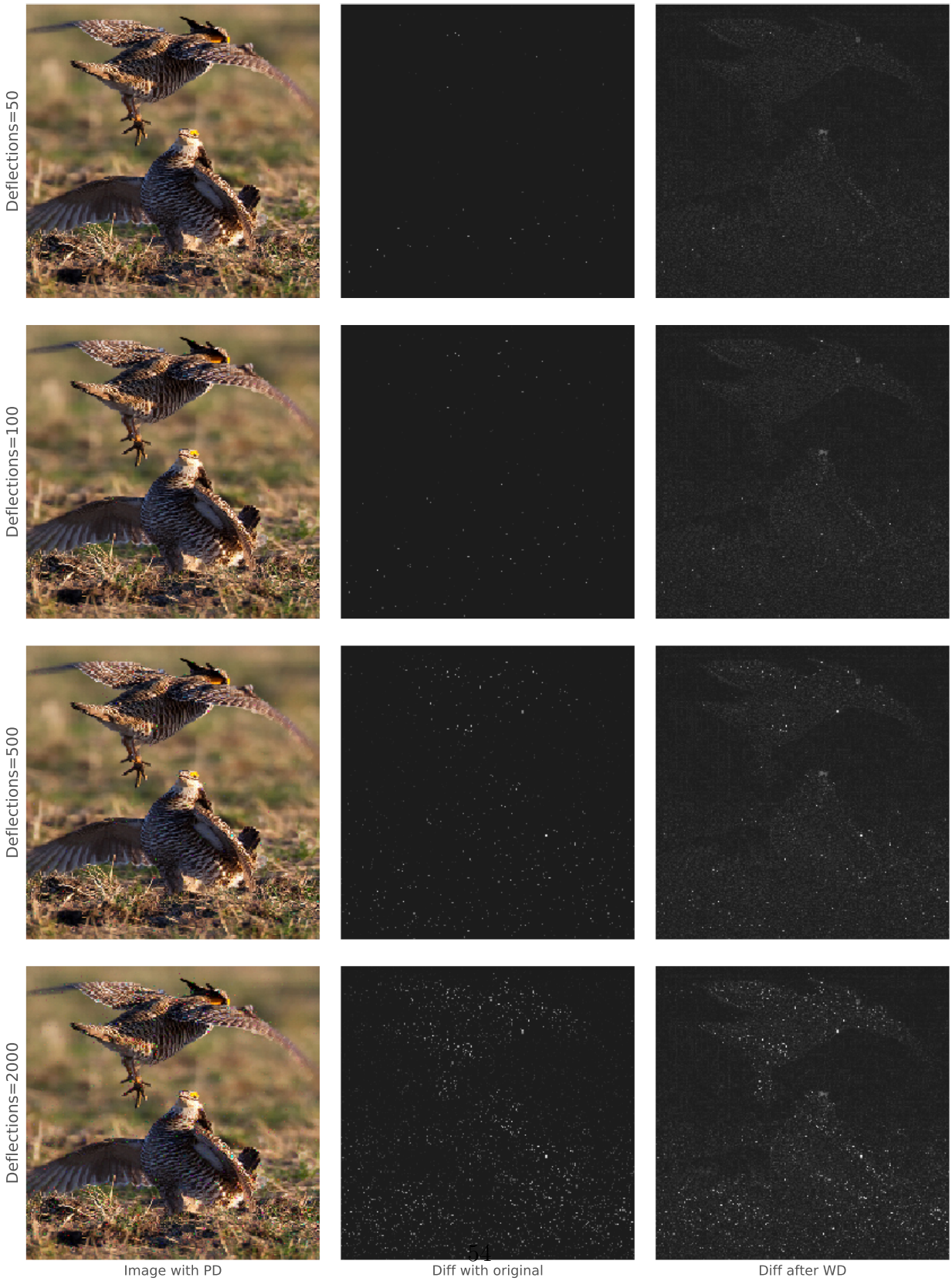


Figure 4.1: Left: Image with given number of pixels deflected. Middle: Difference between clean image and deflected image. Right: Difference after denoising.

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 **??**, 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. This is in contrast with the classification models, which show high activation in regions where an object is present [**Yosinski2015UnderstandingNN**; **Chattopadhyay2017GradCAMGG**]. 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.* [**FoveationbasedMALuo2015**] use the object coordinates to mask out the background region and show that this defends against some of the known attacks.

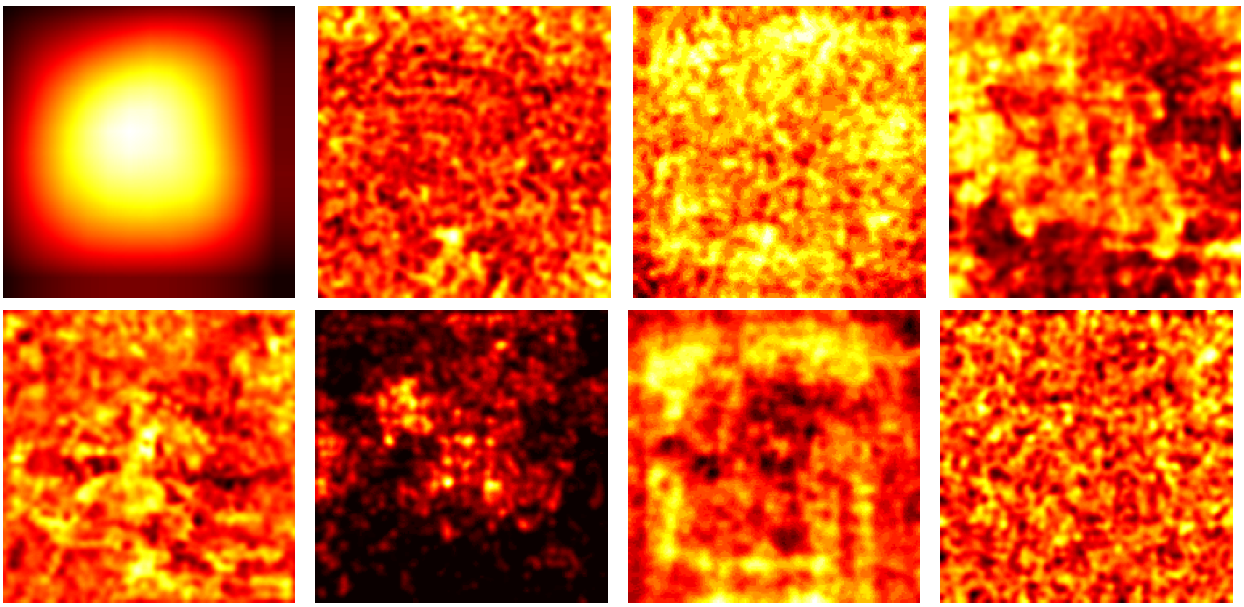


Figure 4.3: Visualization showing average location in the image where perturbation is added by an attacker. Clockwise from top left: Localization of most salient object in the image, FGSM, IGSM, FGSM-2 (higher ϵ), Deep Fool, JSMA, LBFGS and Carlini-Wagner attack.

In Figure **??** 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 **??**, 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.* [**FoveationbasedMALuo2015**] used this idea to mask the regions which did not contain the object, 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 [**CAMZhou2016LearningDF**] are a weakly-supervised localization [**WeaklyOquab2015IsOL**] 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 [**SALICONHuang2015**], as saliency maps are trained on human eye fixations and thus do not always capture object classes [**DeepGazeKmmerer2014DeepGI**]. Other weakly supervised localization techniques, such as regions-of-interest [**Prakash2017SemanticPI**], 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 [**CAMZhou2016LearningDF**] 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 **??** 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 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 [**BayesShrinkChang2000**; **Simoncelli1999BayesianDO**; **field1987relations**] 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 [**Marcelja1980MathematicalDO**; **Rust2005SpatiotemporalEO**; **Hubel1959ReceptiveFO**] 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

[**Kurakin2016AdversarialEI**; **Das2017KeepingTB**; **Dziugaite2016ASO**; **CounteringAIGuo17**]. Dziugaite *et al.* [**Dziugaite2016ASO**] reported loss of 8% accuracy on clean images after undergoing JPEG compression.

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* [**DWTantonini1992image**]. 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 [**JPEG2000Adams2001**] and image denoising [**BayesShrinkChang2000**; **WaveletDenoisingRangarajan2002**; **Simoncelli1999BayesianDO**].

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 [**DropoutFeinman2017**; **ForesightLin2017DetectingAA**] but recent work by Carlini and Wagner [**EasilyDetectedCarlini2017**] 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.* [**ThresholdingJansen2012noise**] 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* [**BayesShrinkChang2000**], 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 **??**, 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 [**BayesShrinkChang2000**; **WaveletDenoisingRangarajan2002**]. 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 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 **??**.

(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 **??**.

(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 [**Deng2009ImageNetAL**] Validation set. We use ResNet-50 [**He2016DeepRL**] 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 [**He2016DeepRL**] 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 [**papernot2017cleverhans**] 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.

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

4.9.1 Training

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 [**Tramr2017TheSO**; **Meng2017MagNetAT**]. 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. 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. To demonstrate the robustness of our defense, we select a single setting of the hyper-parameters to be used against all attack models. Figure **??** 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.

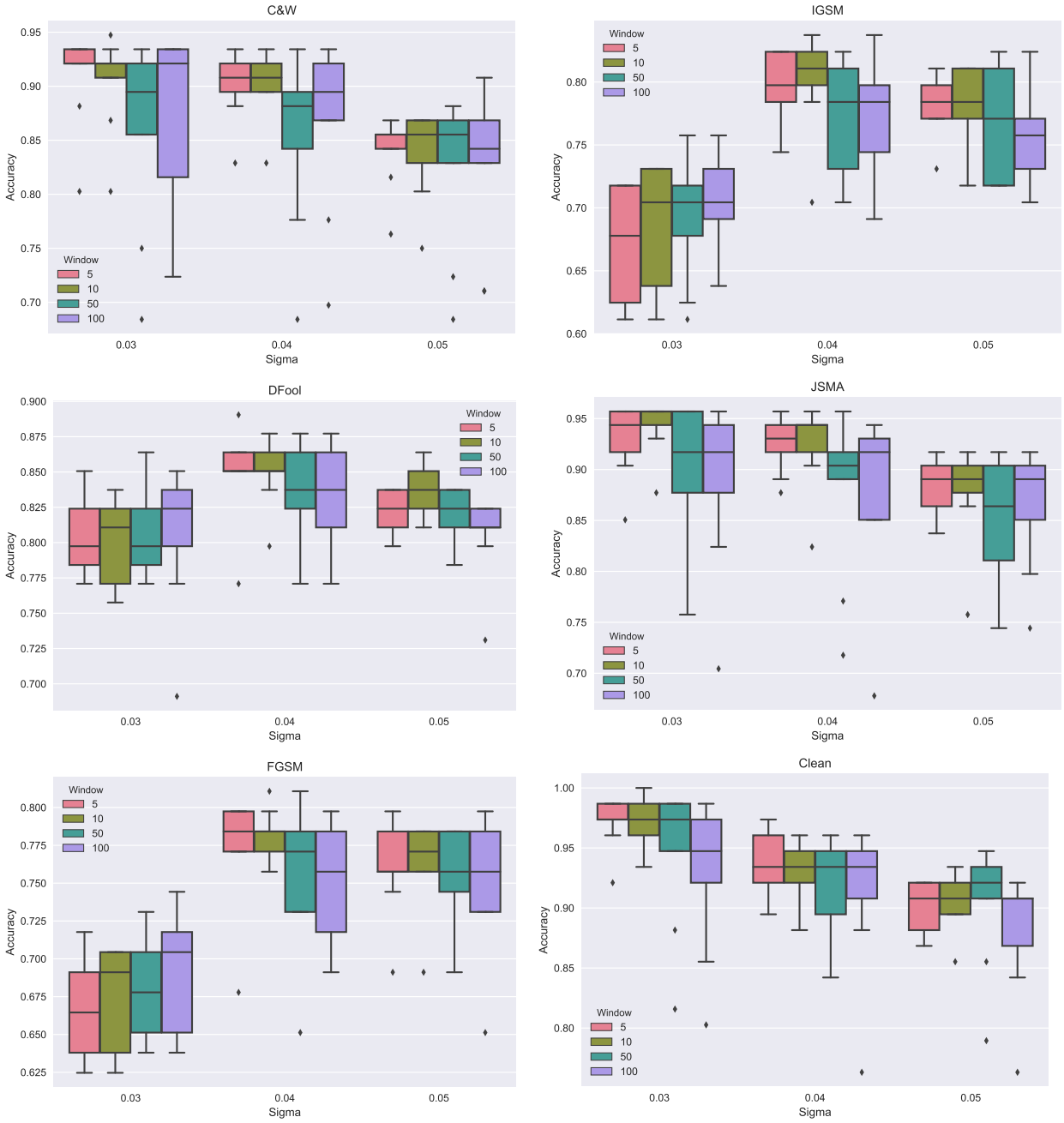


Figure 4.5: Linear search for model parameters

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 $|\boldsymbol{L}_P|$ which leave visible perturbations to show the robustness of our model.

In Table **??** 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 **??**. 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 [**Carlini2017TowardsET**] 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 **??** as Ens-10.

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.

4.10.1 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	$\lvert L_2 \rvert$	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	$\|L_2\|$	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	$\|L_2\|$	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.

4.10.2 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 [**MitigatingAnon208**; **CounteringAIGuo17**]. 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.* [**MitigatingAnon208**] 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 [**Tramr2017EnsembleAT**] 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.* [**CounteringAIGuo17**] 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 [**FeatureSqueezingXu2017**])				
(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.* [**MitigatingAnon208**])				
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.* [**CounteringAIGuo17**])				
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 [**MitigatingAnon208**] are not reported.

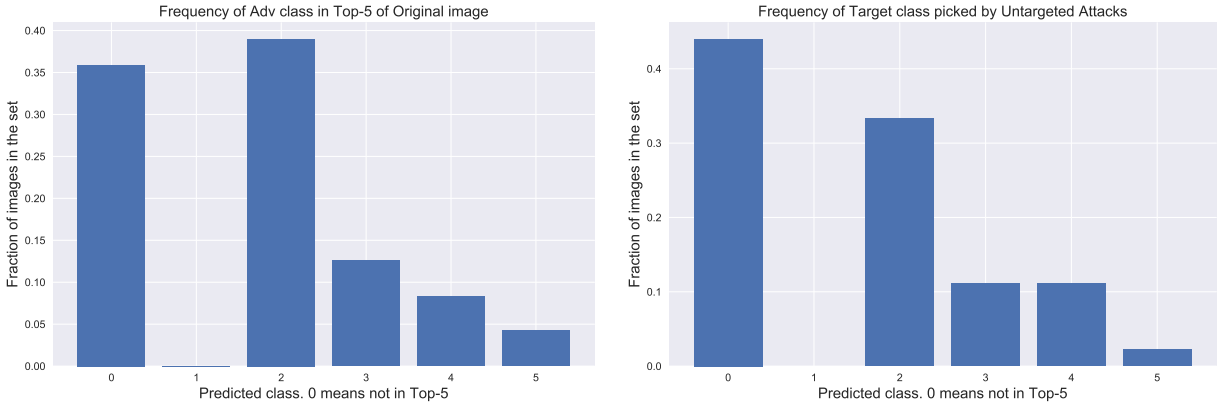


Figure 4.6: Left: Rank of adversarial class within the top-5 predictions for original images. Right: Rank of original class within the top-5 predictions for adversarial images. In both cases, 0 means the class was not in the top-5.

As seen in Figure **??**, 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.* [**Kurakin2016AdversarialEI**], 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. 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 [**Carlini2017TowardsET**] 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.

4.10.3 Ablation studies

Previous work [**Kurakin2016AdversarialEI**; **Dziugaite2016ASO**] has demonstrated the efficacy of JPEG compression as a defense against adversarial attacks due to its denoising properties. Das *et al.* [**Das2017KeepingTB**] 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. In Table **??**, 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 **??** we present a comparison of various shrinkage methods on wavelet coefficients after pixel deflection. For the impact of coefficient thresholding in the absence of pixel deflec-

tion, see Table **??**. 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 **??**, for more thorough review on BayesShrink, VisuShrink and SUREShrink we refer the reader to [**BayesShrinkChang2000**] [**VISUDonoho1994IdealSA**] and [**SUREDonoho1992AdaptingTU**] 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.

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.

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.

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

Efficient Training

5.1 Introduction

Convolutional Neural Networks have achieved state-of-the-art results in various computer vision tasks [**He2016DeepRL**; **Lin2018FocalLF**]. Much of this success is due to innovations of a novel, task-specific network architectures [**He2017MaskR**; **Ronneberger2015UNetCN**]. Despite variation in network design, the same core optimization techniques are used across tasks. 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.

Because models are typically over-parameterized, a trained convolutional network will contain redundant filters [**Cogswell2015ReducingOI**; **Li2016PruningFF**]. This is evident from the common practice of pruning filters [**He2017ChannelPF**; **Anwar2017StructuredPO**; **Li2016PruningFF**; **Molchanov2016PruningCN**; **Liu2017LearningEC**; **Luo2017ThiNetAF**], rather than individual parameters [**Han2015DeepCC**], 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 [**Li2016PruningFF**; **Luo2017ThiNetAF**; **Zhu2017ToPO**]. 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).

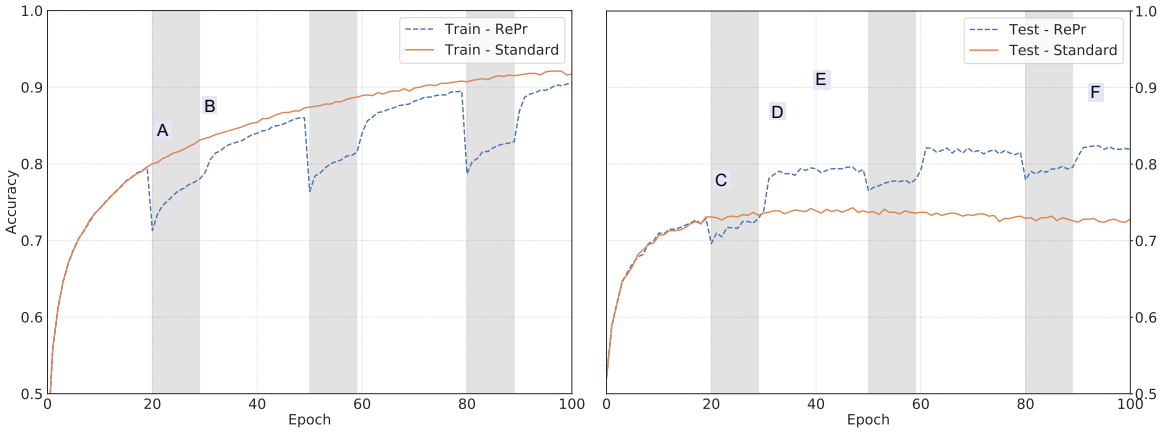


Figure 5.1: Performance of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme and our method - RePr on CIFAR-10. The shaded regions denote periods when only part of the network is trained. Left: Training Accuracy, Right: Test Accuracy. Annotations [A-F] are discussed in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

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 **??** and discussed

in Section **??**. 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 [**Frankle2018TheLT**], 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

5.2.1 Training Scheme

Many changes to the training paradigm have been proposed to reduce over-fitting and improve generalization. Dropout [**Wu2015TowardsDT**] 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 [**Wan2013RegularizationON**]. Wu *et al.* [**Wu2015TowardsDT**] 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 [**Huang2016DeepNW**], forcing the model to learn similar features across various layers which prevent extreme over-fitting. In contrast, our technique encourages the model to use a linear combination of features instead of duplicating the same feature. Han *et al.* [**Han2016DSDDT**] 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 [**Raghu2017OnTE**] and as our results show it leads

to efficient model.

5.2.2 Model Compression

Knowledge Distillation (KD) [**Hinton2015DistillingTK**] 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 [**Romero2014FitNetsHF**]. 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 [**Furlanello2018BornAN**] 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 [**Romero2014FitNetsHF**] 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 **??**).

5.2.3 Neuron ranking

Interest in finding the least salient neurons/weights has a long history. LeCun [**LeCun1989OptimalBD**] and Hassibi*et al.* [**Hassibi1992SecondOD**] 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.* [**Han2015DeepCC**] 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.* [**Hu2016NetworkTA**] 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 [**Luo2017ThiNetAF**] explores reconstruction error between the layers, this does not lead to a generalized filter ranking criteria. Molchanov *et al.* [**Molchanov2016PruningCN**] 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 **??**.

Chen *et al.* [**Chen2017TrainingGO**] 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 [**Liu2017ProgressiveNA**; **Real2018RegularizedEF**; **Zoph2016NeuralAS**
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 [**Cogswell2015**
Glorot2010UnderstandingTD]. Architectures like Inception-Net [**Szegedy2015GoingDW**] 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 [**Raghu2017SVCCASV**]. More recent architectures like ResNet [**He2016DeepRL**] and DenseNet [**Huang2017DenselyCC**] 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 [**Morcos2017OnTI**]. 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 [**Arora2014ProvableBF**]. 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 [**Rodrguez2016RegularizingCW**; **Cogswell2015ReducingOI**]. A similar method, in which the regularization term instead encourages the orthogonality of filter weights, has also produced marginal improvements [**Brock2016NeuralPE**; **Poole2014AnalyzingNI**; **Xie2017NearOrthogonalityRI**; **Xie2017AllYN**]. Shang *et al.* [**Shang2016UnderstandingAI**] 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.* [**Saxe2013ExactST**] show that the orthogonality of weights also improves the stability of network convergence during training. The authors of [**Xie2017AllYN**; **Xiao2018DynamicalIA**] further demonstrate the value of orthogonal weights to the effi-

cient training of networks. Orthogonal initialization is common practice for Recurrent Neural Networks due to their increased sensitivity to initial conditions [**Vorontsov2017OnOA**], 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 [**hotelling1936relations**] (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.* [**Li2015ConvergentLD**] apply correlation analysis to filter activations to show that most of the well-known ConvNet architectures learn similar representations. Raghu *et al.* [**Raghu2017SVCCASV**] 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

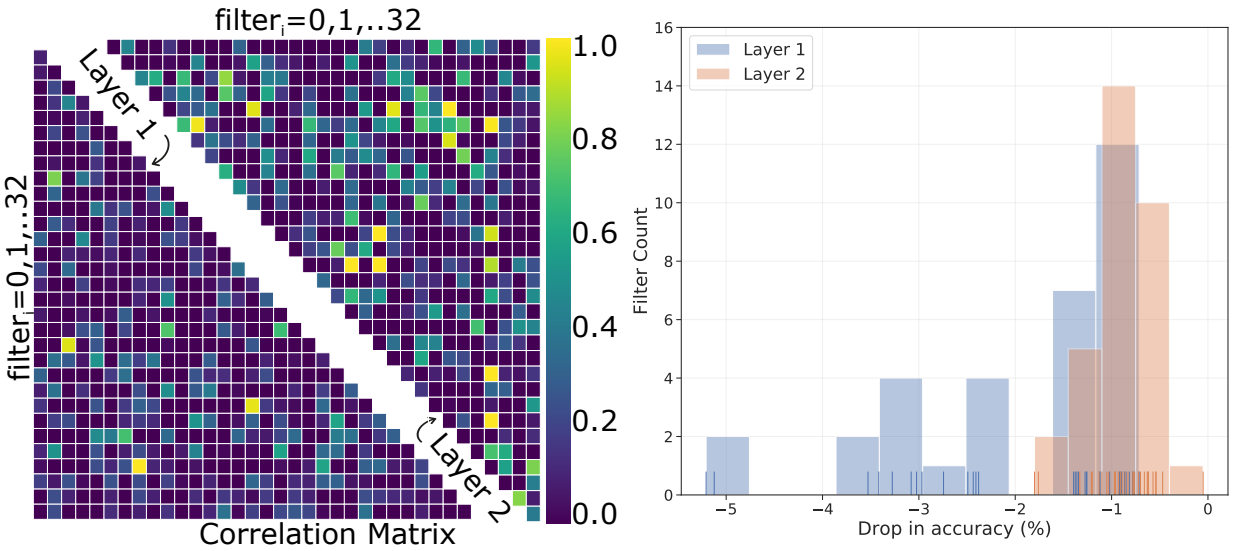


Figure 5.2: Left: Canonical Correlation Analysis of activations from two layers of a ConvNet trained on CIFAR-10. Right: Distribution of change in accuracy when the model is evaluated by dropping one filter at a time.

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 [**Molchanov2016PruningCN**; **Li2015ConvergentLD**]. Is this also true for significantly smaller convolutional networks, which under-fit the dataset?

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. 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 **??** (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 **??** (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 [**Hardoon2004CanonicalCA**] or singular values [**Raghu2017SVCCASV**]. 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.

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* **??***)*.

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 [**Kingma2014AdamAM**] and RmsProp [**Tieleman2012**]. 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 [**Ioffe2015BatchNA**] 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.

Our algorithm is training interposed with **Re**-initializing and **Pr**uning - **RePr** *(pronounced: reaper)*. We summarize our training scheme in Algorithm **??**.

Algorithm 3: 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 **??**.

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 **??** 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 [**Molchanov2016PruningCN**].

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 [**Dai2017NeSTAN**; **Han2015LearningBW**], 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 **??** (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 [**Raghu2017OnTE**]. 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 [**Vorontsov2017OnOA**]. 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 [**Xiao2018DynamicalIA**] but their motivation is the convergence of very deep networks (10K layers) and not orthogonality of the features. Our empirical study suggests

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

a strong 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. 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^f_ℓ) is computed as shown in the equations below.

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

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

$\boldsymbol{P}_\ell$ 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 **??** right).

Computation of our metric does not require expensive calculations of the inverse of Hes-

sian [**LeCun1989OptimalBD**] or the second order derivatives [**Hassibi1992SecondOD**] 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 [**Brock2016NeuralPE**; **Poole2014AnalyzingNI**; **Xie2017NearOrthogonalityRI**] 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.

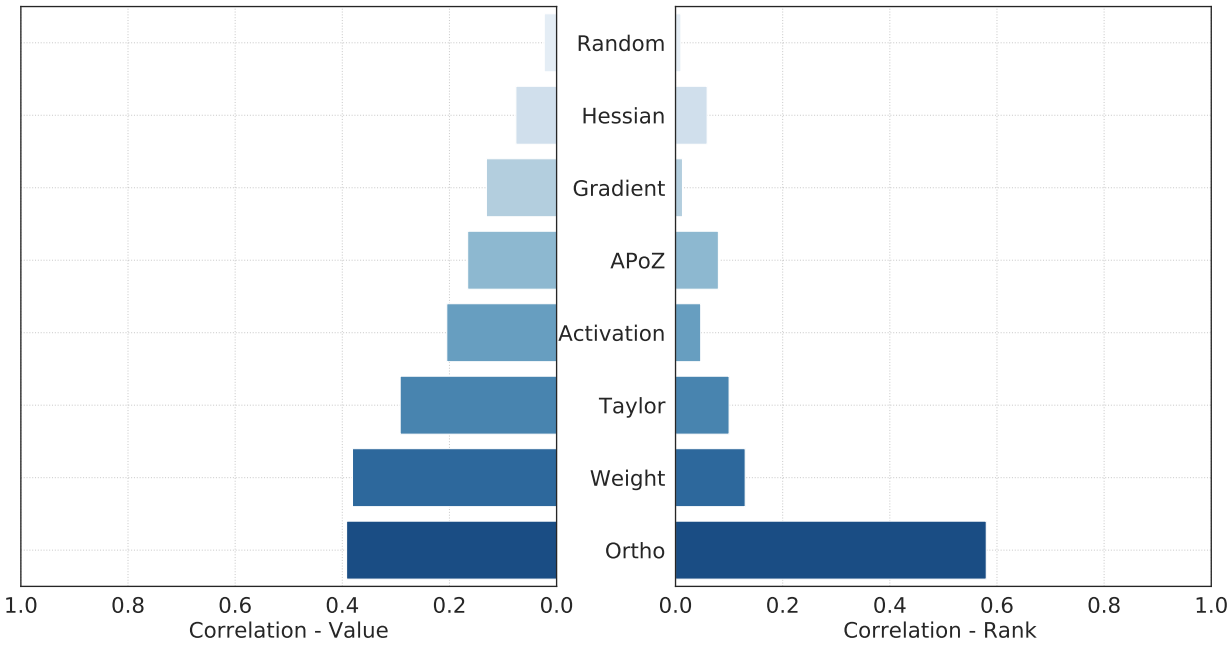


Figure 5.3: Left: Pearson correlation coefficient of various metric values with the accuracy values from greedy Oracle. Center: Pearson correlation coefficient of filter ranks using various metric with rank from greedy Oracle

CIFAR-10 - $C^3(32)$	
Standard	72.1
Random	73.4
Activations	74.1
APoZ [**Hu2016NetworkTA**]	74.3
Gradients [**LeCun1989OptimalBD**]	74.3
Taylor [**Molchanov2016PruningCN**]	74.3
Hessian [**LeCun1989OptimalBD**]	74.4
Weights [**Han2015DeepCC**]	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

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.* [**Molchanov2016PruningCN**] 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

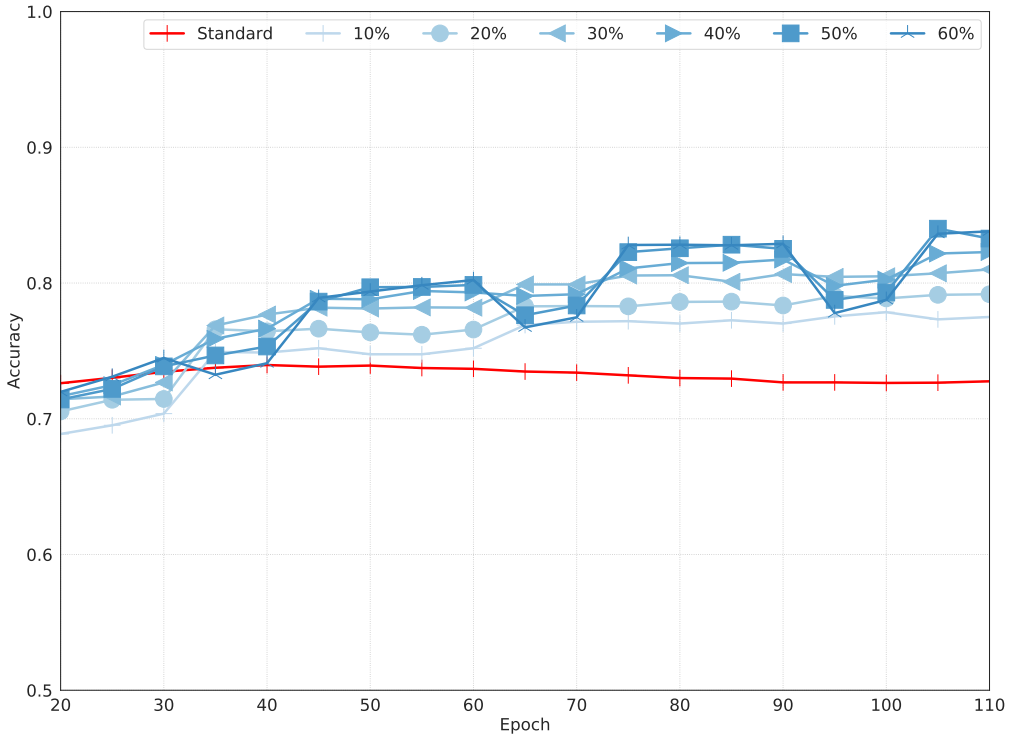


Figure 5.4: RePr training with various percentage of filters pruned. Shows average test accuracy over 5 epochs starting from epoch 20 for better visibility.

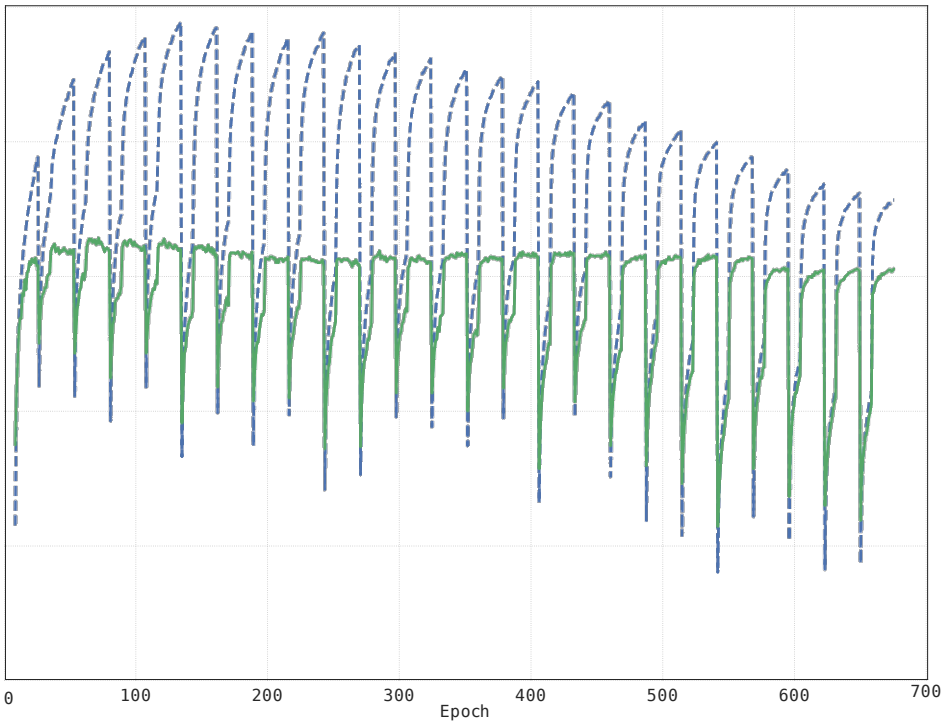


Figure 5.5: Marginal returns of multiple iterations of RePR - Training and Test accuracy on CIFAR-10

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 **??** (Left and Center) shows the correlation plot for various metrics with the Oracle. The table on the right of Figure **??** 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:* **??***)*. 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 **??** (Right)). Similar to Dense-Sparse-Dense [**Han2016DSDDT**] and Born-Again-Networks [**Furlanello2018BornAN**], 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 **??** (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.

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 [**Smith2017CyclicalLR**] can provide stronger

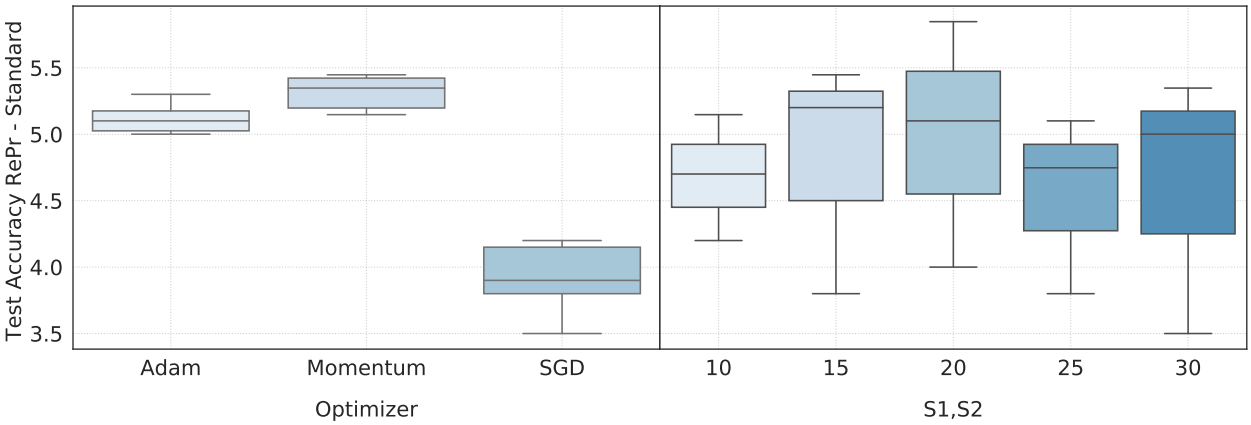


Figure 5.6: Left: Impact of using various optimizers on RePr training scheme. Right: Results from using different S1/S2 values. For clarity, these experiments only shows results with $S1 = S2$

performance without exhaustive tuning of a precise learning rate schedule. Figure **??** 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.

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 **??** 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.

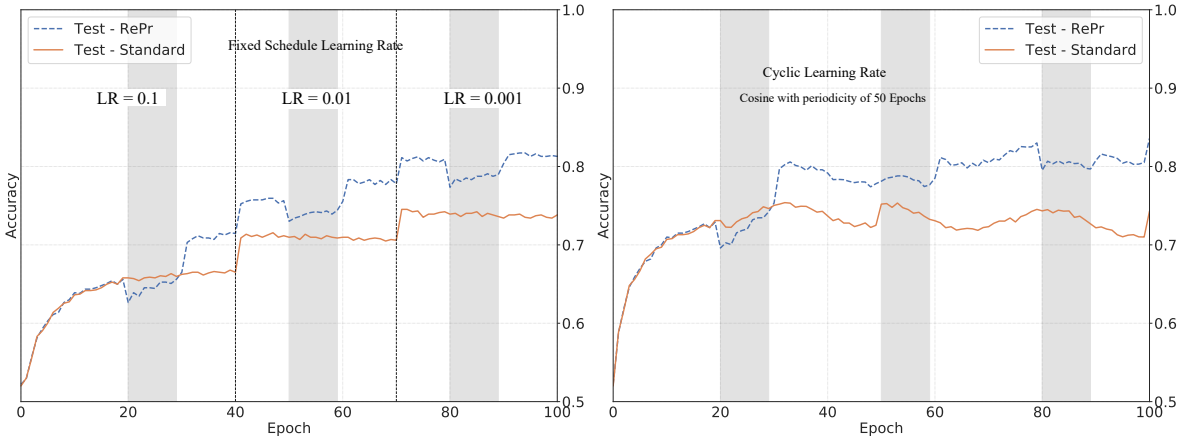


Figure 5.7: Test accuracy of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme and our method - RePr on CIFAR-10. The shaded regions denote periods when only part of the network is trained for RePr. Left: Fixed Learning Rate schedule of 0.1, 0.01 and 0.001.Right: Cyclic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

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.

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 **??** 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.

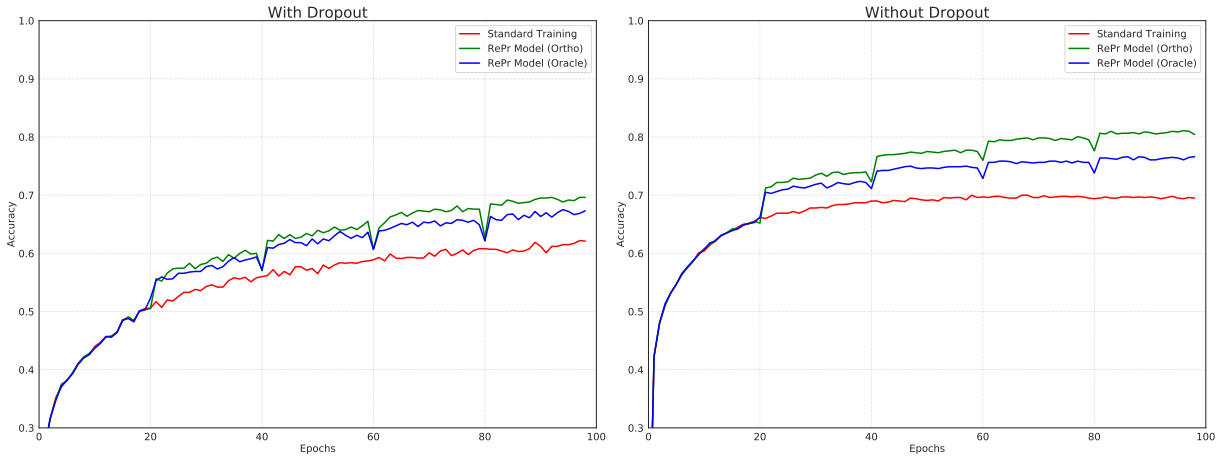


Figure 5.8: Test accuracy of a three layer ConvNet with 32 filters each over 100 epochs using standard scheme, RePr with Oracle and RePr with Ortho on CIFAR-10. Left: With Dropout of 0.5. Right: No Dropout

$C^3(32)$	**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.

$C^3(32)$	**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

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

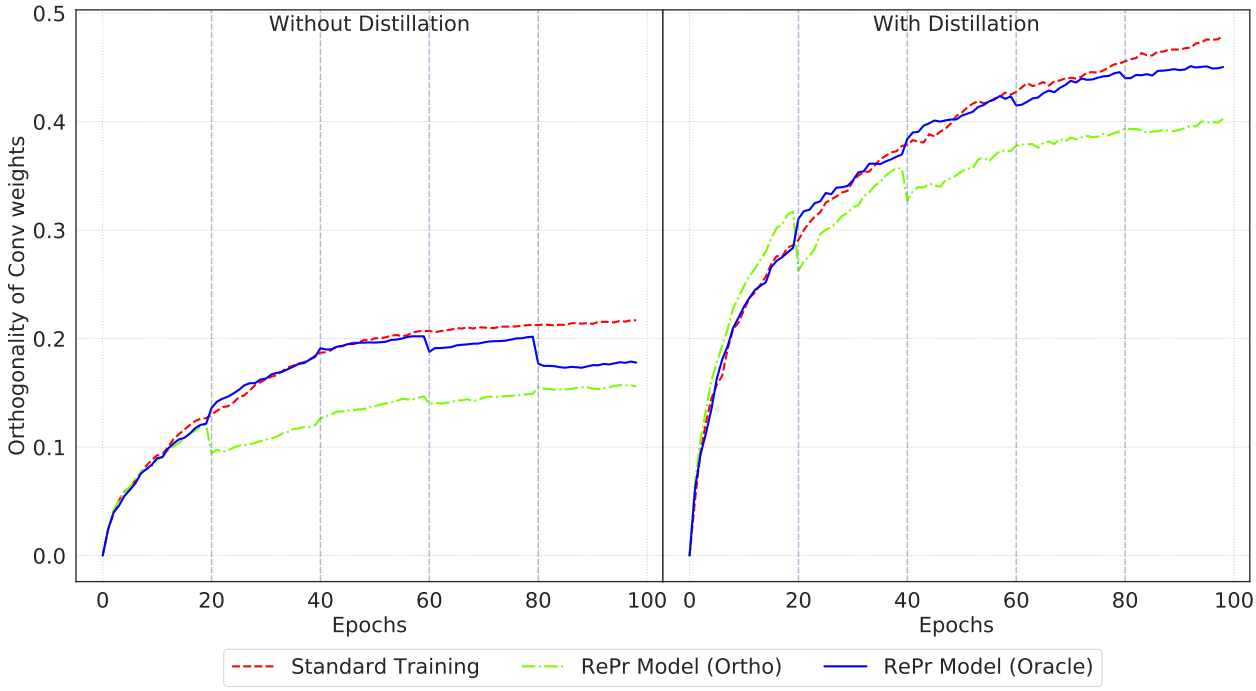


Figure 5.9: Comparison of orthogonality of filters (Ortho-sum - eq 2) in standard training and RePr training with and without Knowledge Distillation. **Lower value** signifies less overlapping filters. Dashed vertical lines denotes filter dropping.

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 **??**. Therefore, we expect this value to gradually reduce over time during training. Figure **??** (left) shows the sum of this value over the entire network with three training schemes. 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* **??***).* 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 **??**, 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 **??**.

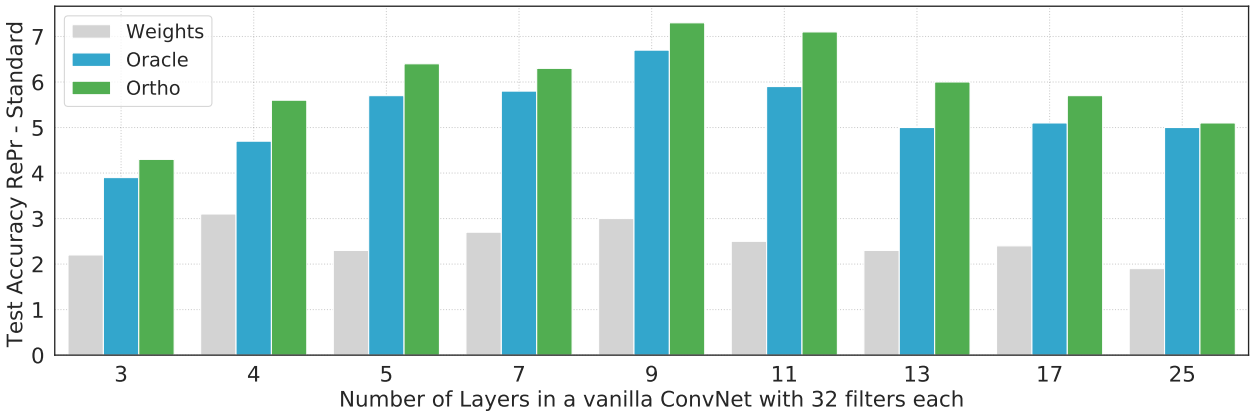


Figure 5.10: Accuracy improvement using RePr over standard training on Vanilla ConvNets across many layered networks $[C^n(32)]$

ResNet-20 on CIFAR-10					
Baseline		Various Training Schemes			
Original [He2016DeepRL]	**Our Impl**	**DSD [Han2016DSDDT]**	**BAN [Furlanello2018BornAN]**	**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.

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 [**Simonyan2014VeryDC**; **He2016DeepRL**; **Szegedy2015GoingDW**; **Szegedy2016RethinkingTI**; **Huang2017DenselyCC**]. For all the results provided RePr parameters are: $S_1 = 20$, $S_2 = 10$, $p_{\%} = 30$, and with three iterations, $N = 3$.

We compare our training scheme with other similar schemes like BAN and DSD in table **??**. 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 **??** 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 **??** 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 **??** - 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 **??** presents top-1 test error on ImageNet [**Russakovsky2015ImageNetLS**] 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 ±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.

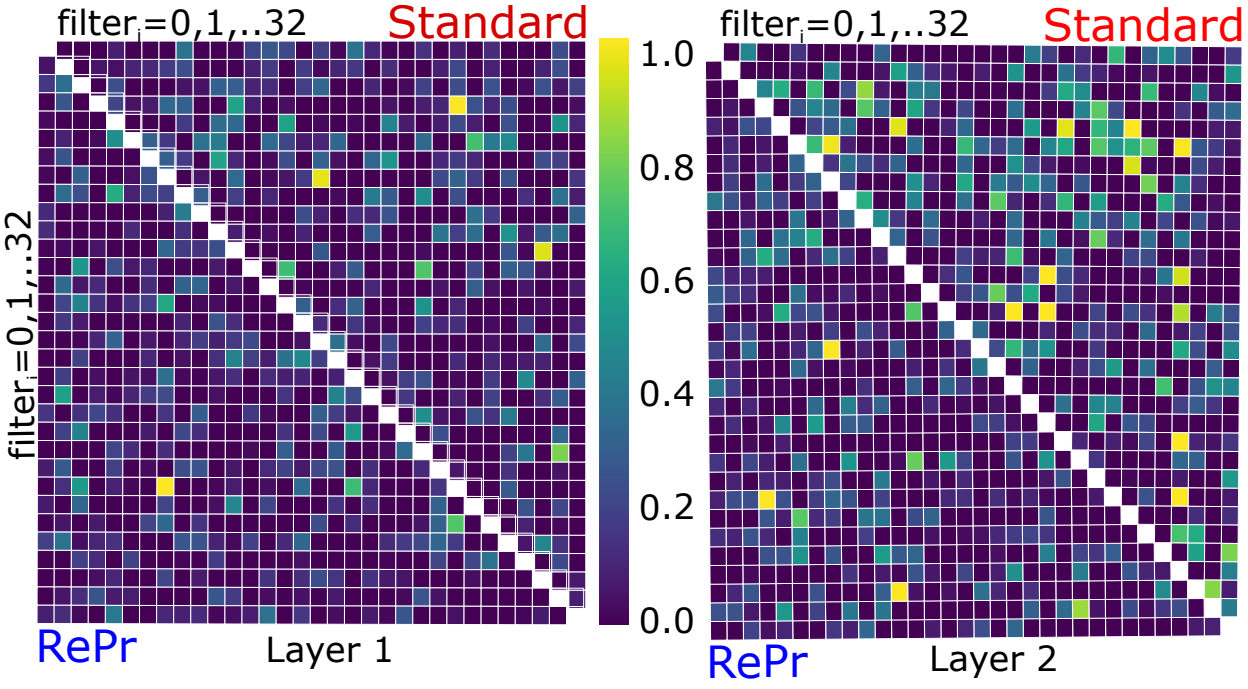


Figure 5.11: Canonical Correlation Analysis of activations from the 1st and 2nd layer of a C2(32) ConvNet trained on CIFAR-10.

Figure **??**) 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	27.15	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 **??** provides accuracy on VQAv1 using VQA-LSTM-CNN model [**Antol2015VQAVQ**]. 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 **??** is the training methodology of Inception-v1. Figure **??** 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 [**wu2016tensorpack**], 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 **??**.

	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. 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.

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. 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.