DeepCorrect: Correcting DNN models against Image Distortions

Tejas Borkar  
lsborkar@asu.edu  
Arizona State University  
Arizona, USA
Lina Karam  
karam@asu.edu
Abstract

In recent years, the widespread use of deep neural networks (DNNs) has facilitated great improvements in performance for computer vision tasks like image classification and object recognition. In most realistic computer vision applications, an input image undergoes some form of image distortion such as blur and additive noise during image acquisition or transmission. Deep networks trained on pristine images perform poorly when tested on distorted images affected by image blur or additive noise. In this paper, we evaluate the effect of image distortions like Gaussian blur and additive noise on the outputs of pre-trained convolutional filters. We propose a metric to identify the most noise susceptible convolutional filters and rank them in order of the highest gain in classification accuracy upon correction. In our proposed approach called DeepCorrect, we apply small convolutional filter blocks on top of these ranked filters and train them to correct the worst noise and blur affected filter outputs. Applying DeepCorrect on the CIFAR-100 dataset, we significantly improve the robustness of DNNs against distorted images and also outperform the alternative approach of fine-tuning networks.

1 Introduction

Today, state-of-the-art algorithms for computer vision tasks employ some form of deep neural networks (DNNs). The ease of design for such networks, afforded by numerous open source deep learning libraries [3, 14], has established DNNs/CNNs as the go-to solution for many computer vision applications. Even challenging computer vision tasks like image classification [12, 18, 26, 29] and object recognition [9, 23, 24], which were previously considered to be extremely difficult, have seen great improvements in their state-of-the-art results due to the use of deep neural networks (DNNs). Training such deep architectures usually involves learning optimal values for millions of parameters iteratively over large datasets and the number of layers that need to be trained for achieving performance gains in classification has gone from just 8 layers [9] to almost 1000 layers [12] over the span of only 5 years. An important factor contributing to the success of such deep architectures in computer vision tasks is the availability of large scale annotated datasets [4, 20], consisting of high quality images devoid of any significant artifacts.

Although DNNs have been able to achieve an impressive performance on many semantic vision tasks, they have been shown to make erroneous predictions, when the network input
is perturbed by specific small magnitude noise \cite{28}. One case of particular interest is that of adversarial samples, where small input perturbations (imperceptible to humans) result in the network making erroneous predictions with very high confidence\cite{10}. Another equally interesting anomaly is the ability of DNNs to make high confidence predictions for input images that are completely unrecognizable to humans, i.e. the images contain no actual object but only have noise in them\cite{22}. Although these types of input image perturbations are not observed in natural images, they do highlight the inability of the DNN to make correct predictions when the test image statistics are different than those of the training images.

The visual quality of input images is an aspect very often overlooked while designing DNN based computer vision systems. In most realistic computer vision applications, an input image undergoes some form of image distortion including blur and additive noise during image acquisition, transmission or storage. Dogde and Karam \cite{6}, showed that even though such image distortions do not represent adversarial samples for a DNN, they do cause a considerable degradation in classification performance. Figure 1(a) shows example images from the ILSVRC2012\cite{4} validation dataset affected by two types of distortion —additive white Gaussian noise (AWGN) and Gaussian blur for different levels of distortion severity and the associated top prediction labels and confidence generated by a pre-trained network. Looking at the sample images in Figure 1, one can see that the image statistics for original and distorted images can be significantly different and the absence of a clear clustering for distorted image subsets in Figure 1(b), points to the degradation in classification accuracy also shown in Tables 1-2.

Before deep networks trained on high quality images can be used for vision tasks where the input images are distorted, it is important to address the performance gap that is observed in Tables 1-2, in terms of the network capacity, convolutional filter invariance and type of
| Dataset   | Distortion Level | Avg   |
|-----------|------------------|-------|
|           | Clean            | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | Level 6 |
| CIFAR 100 | 0.7024           | 0.6427 | 0.2221 | 0.0815 | 0.0444 | 0.0343 | 0.0290 |
| ImageNet  | 0.5694           | 0.4456 | 0.2934 | 0.1585 | 0.0786 | 0.0427 | 0.0256 |

Table 1: Top-1 accuracy of pre-trained networks for Gaussian blur affected images. The severity of distortion increases from Level 1 to 6, where 1 is the least severe and 6 is the most severe, while clean represents undistorted images.

| Dataset   | Distortion Level | Avg   |
|-----------|------------------|-------|
|           | Clean            | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | Level 6 |
| CIFAR 100 | 0.7024           | 0.6184 | 0.3862 | 0.2239 | 0.1324 | 0.0829 | 0.0569 |
| ImageNet  | 0.5694           | 0.5218 | 0.3742 | 0.1256 | 0.0438 | 0.019  | 0.009  |

Table 2: Top-1 accuracy of pre-trained networks for images distorted by additive white Gaussian noise. The severity of distortion increases from Level 1 to 6, where 1 is the least severe and 6 is the most severe, while clean represents undistorted images.

network architecture: 1) Are all convolutional filters in a network trained on undistorted images, equally susceptible to noise or blur? 2) Are networks able to learn some filters that are invariant to input distortions, even when such distortions are absent from the training set? 3) Is it possible to identify and rank the convolutional filters that are most susceptible to image distortions and recover the lost performance, by only correcting the outputs of such ranked filters?

In our proposed approach called DeepCorrect described in Section 3, we try to address questions 1), 2) and 3) by first evaluating the effect of image distortions like Gaussian blur and additive noise on the outputs of pre-trained convolutional filters. We observe that for every layer of convolutional filters in the deep network, certain filters are far more susceptible to input distortions than others and correcting the activations of these filters can help recover lost performance. We then propose a metric in Section 3.1 to rank the convolutional filters in order of the highest gain in classification accuracy upon correction. Finally, as described in Section 3.2, we append small stacks of convolutional filters called correction units to one fraction of the ranked filters in every convolutional layer of a pre-trained network and train them to correct filter activations against input distortions using a target-oriented loss. Applying DeepCorrect on the CIFAR100[17] dataset, we significantly improve the robustness of DNNs against image distortions and also outperform the alternative approach of fine-tuning networks as shown in Section 4.

Related work. Karam and Zhu[16] present QLFW, a face detection dataset consisting of images with five types of quality distortions. Basu et al.[3] present the n-MNIST dataset, which adds Gaussian noise, motion blur and reduced contrast to the original images of the MNIST dataset. Dodge and Karam[6] evaluate the impact of a variety of quality distortions such as Gaussian blur, AWGN and JPEG compression on various state-of-the-art DNNs and report a substantial drop in classification accuracy on the Imagenet(ILSVRC2012) dataset. Karahan et al.[15], present a similar evaluation for the task of face recognition.

An obvious approach to improve the resilience of networks trained on high quality images would be to perhaps fine-tune the network on images with observed distortion types or in cases where the network size and dataset is not too large, include the image distortion process as a form of data-augmentation process and retrain the entire network with such data.
Vasiljevic, et al. [30] study the effect of various types of blur on the classification and segmentation performance of DNNs. Even though DNN performance for the task of classification and segmentation drops in the presence of blur, they show that most of the lost performance can be effectively regained by fine tuning the DNN on a dataset comprising of both distorted and undistorted images. Zhou et al. [31] show that fine-tuning a network on distorted images helps to recover a part of the lost performance, when the degree of distortion is low, while retraining at least a part of the network is needed to partially recover lost performance when images are subjected to higher degrees of distortion.

Diamond et al. [5], propose a joint denoising, deblurring and classification pipeline that is able to outperform other approaches such as conventional image denoiser-based pre-processing and fine tuning of pre-trained networks. This involves an image pre-processing stage that denoises and deblurs the image in a manner that preserves image features optimal for classification rather than aesthetic appearance. The classification stage has to be finetuned using distorted and clean images, while the denoising and deblurring stages assume a priori knowledge of camera parameters and the blur kernel which may not be available at the time of testing. Such a pre-processing stage to denoise and deblur the input compliments our proposed work and can further improve robustness of networks.

Dodge and Karam [7] propose the use of weighted ensemble of deep neural networks to make networks resilient to multiple conflicting image distortion types like gaussian blur and AWGN in the same dataset. They learn a gating function that assigns weights to each model in the ensemble by identifying the type or level of distortion in the input image. Each model in the ensemble is separately fine-tuned on images containing single distortions. Since the ensemble uses models fine-tuned separately on single distortions, this method is not expected to outperform a fine-tuned model, when test images consist of just a single previously known distortion type. Since our proposed approach seeks to improve robustness for single distortions, it can be used in the weighted ensemble in place of fine-tuning to add robustness against multiple distortions.

We believe, the work of Lenc and Vedaldi [19] on evaluating the invariance of CNN filters to affine transformations of the input is closely related to ours. However, their approach is proposed for simple geometric transformations, where every input image pixel is transformed in the same way. They propose a sparsity based regression solution for correcting filter outputs against input transformations, in a single CNN layer.

While fine-tuning tries to adapt to new training distributions while retaining most properties previously captured by the pre-trained model, it limits the ability of the network to learn invariance to severe levels of distortion, whereas retraining a network discards all previously captured properties and is able to achieve invariance to severe distortions, but this may come at the cost of sacrificing performance for clean images or images with minimal severity of distortion. By minimizing the noise in only those filters that are most susceptible to input distortions, we seek to find a balance between retaining previously learned properties and learning new properties of invariance.

2 Evaluation of pre-trained networks

Here we describe the various image distortions, datasets and network architectures used to evaluate the susceptibility of individual convolutional filters to input distortions.

Datasets. We use two popular image classification datasets, CIFAR-100[17] and ILSVRC-2012[4]. CIFAR-100 consists of 50000 training images and 10000 testing images covering
100 different object categories. The ILSVRC2012 dataset consists of around ~1.3 million training images covering 1000 object classes and 50000 validation images, with 50 validation images per class. We use the validation set of ISVRC2012 for our analysis.

**Distortions.** We use two conflicting types of image distortions: Gaussian blur and additive white Gaussian noise (AWGN) over 6 levels of distortion severity. Gaussian blur represents a distortion that eliminates object contours, whereas AWGN adds high frequency information to images and requires a smoothening filter to eliminate noise. Although trained and tested independently, any solution to improve performance should have the ability to recover lost information for both types of distortions.

For CIFAR-100, we use a noise standard deviation $\sigma_n \in \{5, 10, 15, 20, 25, 30\}$ for AWGN and blur standard deviation $\sigma_b \in \{0.5, 1, 1.5, 2.0, 2.5, 3.0\}$ for Gaussian blur. For ILSVRC-2012, a noise standard deviation $\sigma_n \in \{10, 20, 40, 60, 80, 100\}$ for AWGN and blur standard deviation $\sigma_b \in \{1, 2, 3, 4, 5, 6\}$ for Gaussian blur, is used. For both datasets, the size of the blur kernel is set to 4 times the blur standard deviation $\sigma_b$.

**Network Architectures.** We use a version of the All-Convolutional Net[27] shown in Figure 2(a) and Alexnet[18] for CIFAR-100 and ILSVRC2012 respectively. For the All-Convolutional Net[27], a batch normalization layer[13] is added for each convolutional layer. We initialize the network parameters using [11] and train the baseline network using stochastic gradient descent and a mini-batch size of 256, with a categorical cross-entropy loss and $L_2$ weight regularizer value of 0.0001. Starting with an initial learning rate of 0.1 and momentum equal to 0.9, the learning rate is decreased by a factor of 10 every time the validation set accuracy plateaus and training is terminated after around 90-100 epochs.

We use a pre-trained model of Alexnet available at [1], which is trained on the ILSVRC-2012 training data. The input images are first resized to 256x256 and then a central crop of size 227x227 is used for evaluation. We only use a single scale evaluation for simplicity.

**Performance.** We test these aforementioned models on both clean and 6 levels of noisy images denoted by Levels 1-6, with Level 1 being the least severe and Level 6 being the most severe distortion. It is important for a network to perform well on both clean and distorted images and an ideal performance curve should be flat maintaining the same accuracy for all levels of distortion. We report the top-1 accuracy for both datasets and the results are tabulated in Tables 1-2. From Tables 1-2, it is clear that the networks trained on clean images perform poorly when presented with input images that are distorted even at low distortion levels.

3 DeepCorrect

This section describes the main contributions of the paper: 1) measuring the susceptibility of convolutional filters to input distortion using a proposed objective metric in order of highest susceptibility to input distortion (Section 3.1) and 2) training small stacks of convolutional layers called correction units on top of these ranked filters to correct their activations against input distortion (Section 3.2).

3.1 Ranking and correction priority

Considering the time and resources involved in training DNNs on large scale high quality image datasets, it is crucial to assess just how much of the learned rich features are susceptible to input distortions, before attempting to update these features for a new training
distribution. Although pre-trained networks perform poorly on test images with significantly different image statistics than those used to train the networks (Tables 1-2), it is not obvious if only some convolutional filters in a network layer are responsible for most of the observed performance gap or if all convolutional filters in a layer contribute more or less equally to the performance degradation? If only a subset of the filters in a layer are responsible for most of the lost performance, it seems wasteful to modify all the remaining filters to fit better to the new input distribution.

We define the output of a single convolutional filter $\phi_{i,j}$ on the input $x_i$ to the kernel by $\phi_{i,j}(x_j)$, where $(i, j)$ corresponds to (layer number, filter number). If $g_i(\cdot)$ is a transformation that models the distortion acting on filter input $x_i$, then the output of a convolutional filter $\phi_{i,j}$ to the distortion affected input is given by $\phi'_{i,j}(x_i) = \phi_{i,j}(g_i(x_i))$. Since $\phi'_{i,j}(x_i)$ represents the filter activations generated by distorted inputs and $\phi_{i,j}(x_i)$ represents the filter activations for undistorted inputs, assuming we have access to $\phi_{i,j}(x_i)$ for a given set of input images, replacing $\phi'_{i,j}(x_i)$ with $\phi_{i,j}(x_i)$ in a deep network is akin to perfectly correcting the activations of the convolutional filter $\phi_{i,j}$ against input image distortions. Computing the output predictions by swapping distortion affected filter outputs with clean outputs would improve classification performance. The extent of improvement in performance is indicative of the susceptibility of a particular convolutional filter to input distortion and its contribution to the associated performance degradation. Although we do not have access to the ideal filter activation $\phi_{i,j}(x_i)$, while testing a random noise corrupted image, we can generate these pairs of noisy and clean filter activations $(\phi'_{i,j}(x_i), \phi_{i,j}(x_i))$ for a small set of clean and noisy input images, where the noisy image is generated by distorting the clean image with a known
Figure 3: Effect of varying percentage of corrected filters $\beta$ in first layer (conv-1) of DNNs for Gaussian blur affected inputs. Left plot shows results for CIFAR-100, while right plot shows ILSVRC2012.

distortion.

We now define the correction priority of a convolutional filter $\phi(i, j)$ as the improvement in classification performance on a validation set, generated by replacing $\phi^\prime(i, j)(x_i)$ with $\phi(i, j)(x_i)$ in the network. Let the baseline accuracy (accuracy for distorted images) for a network be $p_b$ and $p_{swp}(i, j)$ be the new accuracy of the network after swapping $\phi^\prime(i, j)(x_i)$ with $\phi(i, j)(x_i)$, then the correction priority for filter $\phi(i, j)$ is given as $p_{swp}(i, j) - p_b$. We now define our ranking metric for a convolutional filter $\phi(i, j)$, as the normalized correction priority given by:

$$
\tau(i, j) = \frac{p_{swp}(i, j) - p_b}{\sum_{j=1}^{N_i}(p_{swp}(i, j) - p_b)}
$$

where, $N_i$ represents the total number of convolutional filters in layer $i$. The higher the value of $\tau(i, j)$, the more susceptible the convolutional filter $\phi(i, j)$ is to input distortion. Using the proposed ranking measure in equation (1), we compute normalized correction priorities for every convolutional filter in the network and rank the filters in each layer in descending order of normalized correction priority.

We evaluate the effect of correcting different percentages $\beta_i$ of the ranked filters ($\beta_i \in \{10\%, 25\%, 50\%, 75\%, 90\%\}$), in a given layer $i$ by swapping $\phi^\prime(i, j)(x_i)$ with $\phi(i, j)(x_i)$ for every filter in the subset, where $j$ represents the filter index. For the CIFAR-100 model, it is possible to recover a large part of the lost performance by correcting a small percentage of the filters in the first layer of the DNNs, as shown in Figure 3. However for Alexnet, the number of filters in the first layer that need correction is higher.

Next, we evaluate the improvement in performance for different layers of the network by keeping the percentage of corrected filters $\beta_i$ fixed for a layer $i$. From Figure 4, we see that for CIFAR-100 model, the best performance is achieved by correcting filters of the earlier convolutional layers and as we go deeper in the network, improvement in accuracy diminishes. This indicates that the as we go deeper in the network, all the convolutional filters become equally susceptible to distortion and correcting just a subset of the filter activations is not enough to recover lost performance and is consistent with observations made by Lenc and Vedaldi in [19]. The initial convolutional layers of DNNs are more generic and as a consequence, are more resilient to input distortion and due to the sequential nature of feed-forward DNNs, the higher layer convolutional filters are less robust to input distortion.
Figure 4: Effect of correcting 50% filters for different layers in DNNs for images affected by Gaussian Blur, where conv-i represents $i^{th}$ network layer. Left plot shows results for CIFAR-100, while right plot shows ILSVRC2012.

3.2 Correcting ranked filter outputs

In this work, we proposed a novel approach, which we refer to as DeepCorrect. In our approach, we try and learn a corrective transform on top of the original object specific properties learned by networks, such that the corrective transform acts as a distortion masker or denoiser. We try to learn such a corrective transform for convolutional filters that are most susceptible to input distortion, while leaving all the other pre-trained filters in the layer unchanged. Let $R_i$ represent a set consisting of the $N_i$ ranked filter indices in the $i^{th}$ layer of the network computed using the procedure in Section 3.1. Also let $R_i, \beta$ represent a sub-set of $R_i$ consisting of the top $\beta\%$ ranked filter indices in network layer $i$. If $\Phi_i$ represents the set of convolutional filters in the $i^{th}$ layer, the objective is to learn a transform $F_{corr_i}(\cdot)$, such that $F_{corr_i}(\Phi_{R_i, \beta}(g_i(x_i))) \approx \Phi_{R_i, \beta}(x_i)$, where $x_i$ is the undistorted input to the $i^{th}$ layer of convolutional filters, $g_i(\cdot)$ is a transformation that models the distortion acting on $x_i$.

Since we do not assume any specific form for the image distortion process, we let the corrective transform $F_{corr_i}(\cdot)$ take the form of a multi-layer convolutional filter block, such as the one shown in Figure 2(c). We call these convolutional filter blocks as correction units. Using a small multi-layer convolutional block to model $F_{corr_i}(\cdot)$ makes $F_{corr_i}(\Phi_{R_i, \beta}(g_i(x_i)))$ a differentiable operation and $F_{corr}(\cdot)$ can now be estimated using a target-oriented loss such as the one used to train the original network, through backpropogation[25]. Consider an $N$ layered DNN $\Phi$ that has been pre-trained for an image classification task using clean images. $\Phi$ can be interpreted as a function that maps network input $x$ to an output vector $\Phi(x) \in \mathbb{R}^d$, such that $\Phi = \Phi_N \circ \Phi_{N-1} \circ \ldots \circ \Phi_2 \circ \Phi_1$, where $\Phi_i$ is the mapping function representing the $i^{th}$ DNN layer and $d$ is the dimensionality of the network output. If we add a correction unit that acts on the top $\beta\%$ ranked filters in the first network layer, then the resultant network $\Phi_{corr}$ is given by: $\Phi_{corr} = \Phi_N \circ \Phi_{N-1} \circ \ldots \circ \Phi_2 \circ \Phi_1'$, where $\Phi_1'$ represents the new mapping function for the first layer, in which the corrective transform $F_{corr_1}(\cdot)$ acts on the filter subset $\Phi_{R_1, \beta}$ and all the remaining filters are left unchanged. If $W_{corr}$ represents the trainable parameters in $F_{corr_1}$, then $F_{corr_1}$ can be estimated by minimizing:

$$E(W_{corr}) = \lambda R(W_{corr}) + \frac{1}{n} \sum_{i=1}^{n} L(y_i, \Phi_{corr}(x_i))$$

(2)

where $\lambda$ is a constant, $R$ is a regularizer such as $l_1$ or $l_2$, $L$ is a standard classification loss, $y_i$ is the target output label for input image $x_i$ and $n$ represents the total number of images.
| Method       | Clean | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | Level 6 | Avg  |
|--------------|-------|---------|---------|---------|---------|---------|---------|------|
| Pre-trained  | 0.7024| 0.6427  | 0.2221  | 0.0815  | 0.0444  | 0.0343  | 0.0290  | 0.2509|
| Finetune-3   | 0.6885| 0.6841  | 0.5832  | 0.5029  | 0.4381  | 0.3802  | 0.3256  | 0.5146|
| Finetune-all  | 0.6934| 0.6907  | 0.6507  | 0.6007  | 0.5471  | 0.5007  | 0.4386  | 0.5888|
| Retrain      | 0.6819| 0.6743  | 0.6429  | 0.6130  | **0.5835** | **0.5427** | **0.4933** | **0.6045** |
| DeepCorr-1   | 0.6942| 0.6804  | 0.6460  | 0.6016  | 0.5532  | 0.4929  | 0.4083  | 0.5824|
| DeepCorr-3   | 0.6960| 0.6867  | 0.6537  | 0.6166  | 0.5779  | 0.5234  | 0.4568  | 0.6016|
| DeepCorr-6   | 0.6954| 0.6900  | **0.6591** | 0.6217  | 0.5739  | 0.5230  | 0.4529  | 0.6023|

Table 3: Top-1 accuracy for CIFAR-100 images distorted by Gaussian blur. The severity of distortion increases from Level 1 to 6, where 1 is the least severe and 6 is the most severe, while clean represents undistorted images. Best accuracy is shown in bold, while underlined text indicates second best accuracy.

in the training set. Since we train on a collection of both distorted and clean images, \( x_i \) in equation (2) represents a clean or a distorted image. The trainable parameters in equation (2) are \( W_{corr} \), while all other network parameters are fixed. Although equation (2) shows correction unit estimation for only the first layer, it is possible to add such units at the output of distortion susceptible filters in each layer. Figure 2(b) shows distorted filter outputs being corrected in the first 3 layers of the pre-trained network in Figure 2(a). The correction unit shown in Figure 2(c), is constructed using a residual formulation similar to the one in [12].

4 Experiments

As mentioned in Section 2, we test the classification accuracy of DNNs on both the original clean images as well as distorted versions of the images in CIFAR-100. We evaluate the results for fine-tuning, re-training and our proposed approach, for each of the two distortion types, Gaussian blur and AWGN, independently.

**Fine-tuning.** We start with the pre-trained model for the network shown in Figure 2(a), and fine-tune this model on a mix of noisy and clean images. Similar to the procedure followed in [30], we use a mini-batch size of 256 samples, such that half of the input samples in each mini-batch are randomly chosen and distorted with a distortion level that is also chosen randomly. We start with an initial learning rate that is 100 times lower than the initial learning rate used to train the original model, i.e 0.001. The learning rate is dropped by a factor of 10 every time the validation set accuracy is seen to plateau, until the training is eventually terminated after 100 epochs. The data augmentation process used during training is the same as in [12]. We generate two variants of fine-tuned models, 1) Finetune-all (all layers are fine-tuned) 2) Finetune-3 (only the final 3 layers including the softmax layers are fine-tuned, leaving the previous 6 layers unchanged).

**Retraining.** We also evaluate the effect of retraining the entire model in Figure 2(a) on a mix of clean and noisy images. The model is trained with an initial learning rate of 0.1, momentum equal to 0.9 and the learning rate is dropped by 10 every time the validation set accuracy saturates. The learning rate is dropped 4 times before termination. All other training parameters such as mini-batch size, data augmentation are the same as fine-tuning.

**DeepCorrect.** As shown in Section 3.1, it is best to start correcting filter outputs from the earliest layers in the network, i.e. first convolutional layer in Figure 2. We can generate a
Table 4: Top-1 accuracy for CIFAR-100 images distorted by AWGN. The severity of distortion increases from Level 1 to 6, where 1 is the least severe and 6 is the most severe, while clean represents undistorted images. Best accuracy is shown in bold, while underlined text indicates second best accuracy.

| Method         | Distortion Level |        |        |        |        |        | Avg    |
|----------------|------------------|--------|--------|--------|--------|--------|--------|
|                | Clean            | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 | Level 6 |        |
| Pre-trained    | 0.7024           | 0.6184 | 0.3862 | 0.2239 | 0.1324 | 0.0829 | 0.0569 | 0.3147 |
| Finetune-3     | 0.6828           | 0.6813 | 0.6543 | 0.6068 | 0.5553 | 0.5051 | 0.4462 | 0.5902 |
| Finetune-all    | 0.691            | 0.685  | 0.6761 | 0.6559 | 0.6351 | 0.6074 | 0.5754 | 0.6465 |
| Retrain        | 0.6856           | 0.6786 | 0.6691 | 0.6538 | 0.6425 | 0.6245 | 0.5955 | 0.6499 |
| DeepCorr-1     | 0.7021           | 0.6920 | 0.6776 | 0.6540 | 0.6268 | 0.5983 | 0.5595 | 0.6443 |
| DeepCorr-3     | 0.7013           | 0.6919 | 0.6766 | 0.6508 | 0.6279 | 0.5999 | 0.5616 | 0.6443 |
| DeepCorr-6     | 0.7009           | 0.6936 | 0.6795 | 0.6560 | 0.6316 | 0.6068 | 0.5785 | 0.6495 |

Analysis. The results of evaluating the above mentioned approaches can be seen in Tables 3-4. The performance of a pre-trained network on distorted images is shown in Tables 1-2. We can see that even by fine-tuning just the final 3 layers on a mix of clean and noisy images, we improve the average classification rate of the network from 0.25 to 0.514 for Gaussian blur and 0.314 to 0.590 for AWGN. Our deep correct models are able to outperform fine-tuning on both distortion types, by only correcting 50% of the filters in a layer. Our simplest model DeepCorr-1, which trains almost 4x lesser parameters than a fine-tuned model is able to almost match or outperform the fine-tuned model on less severely distorted images. For the case of Gaussian blur, DeepCorr-3 and DeepCorr-6 are able to outperform a fine-tuned model over almost all distortion levels without updating as many parameters as fine-tuning. Even in the case of AWGN, DeepCorr-6 is able to outperform a fine-tuned model on almost all levels of distortion levels. Although one would expect a model completely re-trained on a mixture of noisy and clean data to outperform all other approaches, what we observe from Tables 3-4 is that re-training is outperformed by DeepCorrect models on almost half of the distortion levels. Although a retrained model is able to achieve a higher accuracy for the worst distortion levels than our proposed model, this comes at the cost number of different corrected models by varying: 1) number of filters corrected per layer, and 2) number of network layers corrected. In general, the choice of these parameters would depend on the original pre-trained network used. For simplicity, we set the number of filters to be corrected in each layer to 50% of the total filters in a layer, however it is possible to correct a different number of filters in each layer. Depending on the number of network layers used for the correction and the number of filters corrected in each layer, we represent our models as DeepCorr-(i,β), where i represents the number of network layers used in correction, starting from the first convolutional layer and β represents the fraction of filters corrected in each layer. As an example, for the model in Figure 2(a), DeepCorr-(9,1) would represent retraining the entire network. For our proposed approach, we generate 3 models: 1) DeepCorr-(1,0.5), 2) DeepCorr-(3,0.5) and 3) DeepCorr-(6,0.5). For brevity, we just denote these as DeepCorr-1, DeepCorr-3 and DeepCorr-6 in our results. We train the correction units using the target-oriented loss described in Section 3.2. Starting with an initial learning rate of 0.1 and momentum equal to 0.9, we follow the same learning rate policy and data augmentation used for retraining or fine-tuning the network. Since the corrective transform cannot exactly recover original clean outputs, for each model, we also let the last 3 layers update their weights with a much smaller learning rate, instead of keeping them fixed.
of sacrificing performance on lower levels of distortion severity. The ideal approach that improves robustness of networks against distortion should perform equally well on images with almost no distortion or even clean images. DeepCorrect models are able to improve robustness against more severe levels of input distortion without sacrificing performance on clean images or less severe distortions.

5 Conclusions

We proposed a novel objective metric to identify convolutional filters which are most susceptible to image distortions. By training stacks of convolutional filters called correction units on top of these ranked filters in a network, we significantly improve the robustness of DNNs against image distortions and also outperform the alternative approach of fine-tuning networks. Fine-tuning limits a network’s ability to add robustness against severe image distortions, whereas a re-trained network sacrifices performance on clean images. By minimizing the noise in convolutional filters that are most susceptible to input distortions, we are able to make a DNN robust to severe distortions and still maintain very good performance on clean images.

Although we focus on image classification tasks, our proposed approach is generic enough to apply to a wide selection of tasks that use DNN models such as object detection [9][8] or semantic segmentation[20]. Denoising approaches like [5] can directly be used as a preprocessing stage for our approach to add more robustness to pre-trained DNNs, while ensemble based approaches like [7] can be combined with our approach to make DNNs simultaneously robust to a wide range of distortions.

References

[1] Convnets-keras. https://github.com/heuritech/convnets-keras.

[2] Saikat Basu, Manohar Karki, Sangram Ganguly, Robert DiBiano, Supratik Mukhopadhyay, Shreekant Gayaka, Rajgopal Kannan, and Ramakrishna Nemani. Learning sparse feature representations using probabilistic quadtrees and deep belief nets. Neural Processing Letters, pages 1–13, 2015.

[3] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009.

[5] Steven Diamond, Vincent Sitzmann, Stephen P. Boyd, Gordon Wetzstein, and Felix Heide. Dirty pixels: Optimizing image classification architectures for raw sensor data. CoRR, abs/1701.06487, 2017. URL http://arxiv.org/abs/1701.06487.

[6] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on, pages 1–6. IEEE, 2016.

[7] Samuel Dodge and Lina Karam. Quality resilient deep neural networks. arXiv preprint arXiv:1703.08119, 2017.
[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[10] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[15] Samil Karahan, Merve Kilinc Yildirum, Kadir Kirtac, Ferhat Sukru Rende, Gultekin Butun, and Hazim Kemal Ekenel. How image degradations affect deep cnn-based face recognition? In *Biometrics Special Interest Group (BIOSIG), 2016 International Conference of the*, pages 1–5. IEEE, 2016.

[16] Lina J. Karam and Tong Zhu. Quality labeled faces in the wild (qlfw): a database for studying face recognition in real-world environments, 2015.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
[22] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[25] David E. Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation. chapter Backpropagation: The Basic Theory, pages 1–34. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1995.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of ICLR*, 2015.

[27] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[28] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[30] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *CoRR*, abs/1611.05760, 2016. URL http://arxiv.org/abs/1611.05760.

[31] Yiren Zhou, Sibo Song, and Ngai-Man Cheung. On classification of distorted images with deep convolutional neural networks. *arXiv preprint arXiv:1701.01924*, 2017.