High-frequency Component Helps Explain the Generalization of Convolutional Neural Networks

Haohan Wang, Xindi Wu, Zeyi Huang, Eric P. Xing
School of Computer Science
Carnegie Mellon University
{haohanw, epxing}@cs.cmu.edu, {xindiw, zeyih}@andrew.cmu.edu

Abstract

We investigate the relationship between the frequency spectrum of image data and the generalization behavior of convolutional neural networks (CNN). We first notice CNN’s ability in capturing the high-frequency components of images. These high-frequency components are almost imperceptible to a human. Thus the observation leads to multiple hypotheses that are related to the generalization behaviors of CNN, including a potential explanation for adversarial examples, a discussion of CNN’s trade-off between robustness and accuracy, and some evidence in understanding training heuristics.

1. Introduction

Deep learning has achieved many recent advances in predictive modeling in various tasks, but the community has nonetheless become alarmed by the unintuitive generalization behaviors of neural networks, such as the capacity in memorizing label shuffled data [65] and the vulnerability towards adversarial examples [54, 21].

To explain the generalization behaviors of neural networks, many theoretical breakthroughs have been made progressively, including studying the properties of stochastic gradient descent [31], different complexity measures [46], generalization gaps [50], and many more from different model or algorithm perspectives [30, 43, 7, 51].

In this paper, inspired by previous understandings that convolutional neural networks (CNN) can learn from confounding signals [59] and superficial signals [29, 19, 58], we investigate the generalization behaviors of CNN from a data perspective. Together with [27], we suggest that the unintuitive generalization behaviors of CNN as a direct outcome of the perceptual disparity between human and models (as argued by Figure 1): CNN can view the data at a much higher granularity than the human can.

However, different from [27], we provide an interpretation of this high granularity of the model’s perception: CNN can exploit the high-frequency image components that are not perceivable to human.

For example, Figure 2 shows the prediction results of eight testing samples from CIFAR10 data set, together with the prediction results of the high and low-frequency component counterparts. For these examples, the prediction outcomes are almost entirely determined by the high-frequency components of the image, which are barely perceivable to human. On the other hand, the low-frequency components, which almost look identical to the original image to human, are predicted to something distinctly different by the model.

Motivated by the above empirical observations, we further investigate the generalization behaviors of CNN and attempt to explain such behaviors via differential responses to the image frequency spectrum of the inputs (Remark 1). Our main contributions are summarized as follows:
2. Related Work

The remarkable success of deep learning has attracted a torrent of theoretical work devoted to explaining the generalization mystery of CNN.

- We reveal the existing trade-off between CNN’s accuracy and robustness by offering examples of how CNN exploits the high-frequency components of images to trade robustness for accuracy (Corollary 1).

- With image frequency spectrum as a tool, we offer hypothesis to explain several generalization behaviors of CNN, especially the capacity in memorizing label-shuffled data.

- We propose defense methods that can help improving the adversarial robustness of CNN towards simple attacks without training or fine-tuning the model.

The remainder of the paper is organized as follows. In Section 2, we first introduce related discussions. In Section 3, we will present our main contributions, including a formal discussion on that CNN can exploit high-frequency components, which naturally leads to the trade-off between adversarial robustness and accuracy. Further, in Section 4-6, we set forth to investigate multiple generalization behaviors of CNN, including the paradox related to capacity of memorizing label-shuffled data (§4), the performance boost introduced by heuristics such as Mixup and BatchNorm (§5), and the adversarial vulnerability (§6). We also attempt to investigate tasks beyond image classification in Section 7. Finally, we will briefly discuss some related topics in Section 8 before we conclude the paper in Section 9.

For example, ever since Zhang et al. [65] demonstrated the effective capacity of several successful neural network architectures is large enough to memorize random labels, the community sees a prosperity of many discussions about this apparent “paradox” [61, 15, 17, 15, 11]. Arpit et al. [3] demonstrated that effective capacity are unlikely to explain the generalization performance of gradient-based-methods trained deep networks due to the training data largely determine memorization. Kruger et al.[35] empirically argues by showing largest Hessian eigenvalue increased when training on random labels in deep networks.

The concept of adversarial example [54, 21] has become another intriguing direction relating to the behavior of neural networks. Along this line, researchers invented powerful methods such as FGSM [21], PGD [42], and many others [62, 9, 53, 36, 12] to deceive the models. This is known as attack methods. In order to defend the model against the deception, another group of researchers proposed a wide range of methods (known as defense methods) [1, 38, 44, 45, 24]. These are but a few highlights among a long history of proposed attack and defense methods. One can refer to comprehensive reviews for detailed discussions [2, 10].

However, while improving robustness, these methods may see a slight drop of prediction accuracy, which leads to another thread of discussion in the trade-off between robustness and accuracy. The empirical results in [49] demonstrated that more accurate model tend to be more robust over generated adversarial examples. While [25] argued that the seemingly increased robustness are mostly due to the increased accuracy, and more accurate models (e.g., VGG, ResNet) are actually less robust than AlexNet. Theoretical discussions have also been offered [56, 67], which also inspires new defense methods [67].
3. High-frequency Components & CNN’s Generalization

We first set up the basic notations used in this paper: \(\langle x, y \rangle\) denotes a data sample (the image and the corresponding label). \(f(\cdot; \theta)\) denotes a convolutional neural network whose parameters are denoted as \(\theta\). We use \(\mathcal{H}\) to denote a human model, and as a result, \(f(\cdot; \mathcal{H})\) denotes how human will classify the data \(\cdot\). \(l(\cdot, \cdot)\) denotes a generic loss function (e.g., cross entropy loss). \(\alpha(\cdot, \cdot)\) denotes a function evaluating prediction accuracy (for every sample, this function yields 1.0 if the sample is correctly classified, 0.0 otherwise). \(d(\cdot, \cdot)\) denotes a function evaluating the distance between two vectors. \(\mathcal{F}(\cdot)\) denotes the Fourier transform; thus, \(\mathcal{F}^{-1}(\cdot)\) denotes the inverse Fourier transform. We use \(z\) to denote the frequency component of a sample. Therefore, we have \(z = \mathcal{F}(x)\) and \(x = \mathcal{F}^{-1}(z)\).

Notice that Fourier transform or its inverse may introduce complex numbers. In this paper, we simply discard the imaginary part of the results of \(\mathcal{F}^{-1}(\cdot)\) to make sure the resulting image can be fed into CNN as usual.

3.1. CNN Exploit High-frequency Components

We decompose the raw data \(x = \{x_l, x_h\}\), where \(x_l\) and \(x_h\) denote the low-frequency component (shortened as LFC) and high-frequency component (shortened as HFC) of \(x\). We have the following four equations:

\[
\begin{align*}
z &= \mathcal{F}(x), \\
z_l, z_h &= t(z; r), \\
x_l &= \mathcal{F}^{-1}(z_l), \\
x_h &= \mathcal{F}^{-1}(z_h),
\end{align*}
\]

where \(t(\cdot; r)\) denotes a thresholding function that separates the low and high frequency components from \(z\) according to a hyperparameter, radius \(r\).

To define \(t(\cdot; r)\) formally, we first consider a grayscale (one channel) image of size \(n \times n\) with \(\mathcal{N}\) possible pixel values (in other words, \(x \in \mathcal{N}^{n \times n}\)), then we have \(z \in \mathcal{C}^{n \times n}\), where \(\mathcal{C}\) denotes the complex number. We use \(z(i, j)\) to index the value of \(z\) at position \((i, j)\), and we use \(c_i, c_j\) to denote the centroid. We have the equation \(z_l, z_h = t(z; r)\) formally defined as:

\[
\begin{align*}
z_l(i, j) &= \begin{cases} z(i, j), & \text{if } d((i, j), (c_i, c_j)) \leq r, \\
0, & \text{otherwise} \end{cases} \\
z_h(i, j) &= \begin{cases} 0, & \text{if } d((i, j), (c_i, c_j)) \leq r, \\
z(i, j), & \text{otherwise} \end{cases}
\end{align*}
\]

We consider \(d(\cdot, \cdot)\) in \(t(\cdot; r)\) as the Euclidean distance in this paper. If \(x\) has more than one channel, then the procedure operates on every channel of pixels independently.

Remark 1. With an assumption (referred to as A1) that presumes “only \(x_l\) is perceivable to human, but both \(x_l\) and \(x_h\) are perceivable to a CNN,” we have:

\[
y := f(x; \mathcal{H}) = f(x_l; \mathcal{H}),
\]

but when a CNN is trained with

\[
\arg \min_{\theta} l(f(x; \theta), y),
\]

which is equivalent to

\[
\arg \min_{\theta} l(f(\{x_l, x_h\}; \theta), y),
\]

CNN may learn to exploit \(x_h\) to minimize the loss. As a result, CNN’s generalization behavior appears unintuitive to a human.

Notice that “CNN may learn to exploit \(x_h\)” differs from “CNN overfit” because \(x_h\) can contain more information than sample-specific idiosyncrasy, and these more information can be generalizable across training, validation, and testing sets, but are just imperceptible to a human.

As Assumption A1 has been demonstrated to hold in some cases (e.g., in Figure 2), we believe Remark 1 can serve as one of the explanations to CNN’s generalization behavior. For example, the adversarial examples [54, 21] can be generated by perturbing \(x_h\); the capacity of CNN in reducing training error to zero over label shuffled data [65] can be seen as a result of exploiting \(x_h\) and overfitting sample-specific idiosyncrasy. We will discuss more in the following sections.

3.2. Trade-off between Robustness and Accuracy

We continue with Remark 1 and discuss CNN’s trade-off between robustness and accuracy given \(\theta\) from the image frequency perspective. We first formally state the accuracy of \(\theta\) as:

\[
E_{(x, y)}(f(x; \theta), y)
\]

and the adversarial robustness of \(\theta\) as in e.g., [8]:

\[
\begin{align*}
E_{(x, y)} \min_{x'\in\mathcal{N}^{n \times n}} \min_{x': d(x', x) \leq \epsilon} \alpha(f(x'; \theta), y)
\end{align*}
\]

where \(\epsilon\) is the upper bound of the perturbation allowed.

With another assumption (referred to as A2): “for model \(\theta\), there exists a sample \(\langle x, y \rangle\) such that:

\[
f(x; \theta) \neq f(x_l; \theta)
\]

we can extend our main argument (Remark 1) to a formal statement:

**Corollary 1.** With assumptions A1 and A2, there exists a sample \(\langle x, y \rangle\) that the model \(\theta\) cannot predict both accurately (evaluated to be 1.0 by Equation 1) and robustly (evaluated to be 1.0 by Equation 2) under any distance metric \(d(\cdot, \cdot)\) and bound \(\epsilon\) as long as \(\epsilon \geq d(x, x_l)\).
The proof is a direct outcome of the previous discussion and thus omitted. The Assumption A2 can also be verified empirically (e.g., in Figure 2), therefore we can safely state that Corollary 1 can serve as one of the explanations to the trade-off between CNN’s robustness and accuracy.

4. Rethinking Data before Rethinking Generalization

4.1. Hypothesis

Our first aim is to offer some intuitive explanations to the empirical results observed in [65]: neural networks can easily fit label-shuffled data. While we have no doubts that neural networks are capable of memorizing the data due to its capacity, the interesting question arises: “if a neural network can easily memorize the data, why it cares to learn the generalizable patterns out of the data, in contrast to directly memorizing everything to reduce the training loss?”

Within the perspective introduced in Remark 1, our hypothesis is as follows: Despite the same outcome as a minimization of the training loss, the model considers different level of features in the two situations:

- In the original label case, the model will first pick up LFC, then gradually pick up the HFC to achieve higher training accuracy.
- In the shuffled label case, as the association between LFC and the label is erased due to shuffling, the model has to memorize the images when the LFC and HFC are treated equally.

4.2. Experiments

We set up the experiment to test our hypothesis. We use ResNet-18 [22] for CIFAR10 dataset [33] as the base experiment. The vanilla set-up, which we will use for the rest of this paper, is to run the experiment with 100 epoches with the ADAM optimizer [32] with learning rate set to be 10^{-4} and batch size set to be 100, when weights are initialized with Xavier initialization [20]. Pixels are all normalized to be [0, 1]. All these experiments are repeated in MNIST [14], FashionMNIST [63], and a subset of ImageNet [13]. These efforts are reported in the Appendix. We train two models, with the natural label setup and the shuffled label setup, denote as M_{natural} and M_{shuffle}, respectively; the M_{shuffle} needs 300 epoches to reach a comparative training accuracy. To test which part of the information the model picks up, for any x in the training set, we generate the low-frequency counterparts x_{r} with r set to 4, 8, 12, 16 respectively. We test the how the training accuracy changes for these low-frequency data collections along the training process.

The results are plotted in Figure 3. The first message is that the M_{shuffle} takes a longer training time than M_{natural} to reach the same training accuracy (300 epoches vs. 100 epoches), which suggests that memorizing the samples as an “unnatural” behavior in contrast to learning the generalizable patterns. By comparing the curves of the low-frequency training samples, we notice that M_{natural} learns more of the low-frequency patterns (i.e., when r = 4 or 8) than M_{shuffle}. Also, M_{shuffle} barely learns any LFC when r = 4, while on the other hand, even at the first epoch, M_{natural} already learns around 40% of the correct LFC when r = 4. This disparity suggests that when M_{natural} prefers to pick up the LFC, M_{shuffle} does not have a preference between LFC vs. HFC.

If a model can exploit multiple different sets of signals, then why M_{natural} prefers to learn LFC that happens to align well with the human perceptual preference? While there are explanations suggesting neural networks’ tendency towards simpler functions [48], we conjecture that this is simply because, since the data sets are organized and annotated by human, the LFC-label association is more “generalizable” than the one of HFC: picking up LFC-label association will lead to the steepest descent of the loss surface, especially at the early stage of the training.

To test this conjecture, we repeat the experiment of M_{natural}, but instead of the original train set, we use the x_{l} or x_{h} (normalized to have the standard pixel scale) and test how well the model can perform on original test set. Table 1 suggests that LFC is much more “generalizable” than HFC. Thus, it is not surprising if a model first picks up LFC as it leads to the steepest descent of the loss surface.

4.3. A Remaining Question

Finally, we want to raise a question: The coincidental alignment between networks’ preference in LFC and human perceptual preference might be a simple result of the “survival bias” of the many technologies invented one of the other along the process of climbing the ladder of the

Table 1. We test the generalization power of LFC and HFC by training the model with x_{l} or x_{h} and test on the original test set.

<table>
<thead>
<tr>
<th>r</th>
<th>train acc.</th>
<th>test acc.</th>
<th>r</th>
<th>train acc.</th>
<th>test acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.9668</td>
<td>0.6167</td>
<td>4</td>
<td>0.9885</td>
<td>0.2002</td>
</tr>
<tr>
<td>8</td>
<td>0.9786</td>
<td>0.7154</td>
<td>8</td>
<td>0.9768</td>
<td>0.092</td>
</tr>
<tr>
<td>12</td>
<td>0.9786</td>
<td>0.7516</td>
<td>12</td>
<td>0.9797</td>
<td>0.0997</td>
</tr>
<tr>
<td>16</td>
<td>0.9839</td>
<td>0.7714</td>
<td>16</td>
<td>0.9384</td>
<td>0.1281</td>
</tr>
</tbody>
</table>
5. Training Heuristics

We continue to reevaluate the heuristics that helped in climbing the ladder of state-of-the-art accuracy. We evaluate these heuristics to test the generalization performances towards LFC and HFC. Many renowned techniques in the ladder of accuracy seem to exploit HFC more or less.

5.1. Comparison of Different Heuristics

We test multiple heuristics by inspecting the prediction accuracy over LFC and HFC with multiple choices of $r$ along the training process and plot the training curves.

**Batch Size:** We then investigate how the choices of batch size affect the generalization behaviors. We plot the results in Figure 4. As the figure shows, smaller batch size appears to excel in improving training and testing accuracy, while bigger batch size seems to stand out in closing the generalization gap. Also, it seems the generalization gap is closely related to the model’s tendency in capturing HFC: models trained with bigger epoch sizes are more invariant to HFC and introduce smaller differences in training accuracy and testing accuracy. The observed relation is intuitive because the smallest generalization gap will be achieved once the model behaves like a human (because it is the human who annotate the data).

The observation in Figure 4 also chips in the discussion in the previous section about “generalizable” features. Intuitively, with bigger epoch size, the features that can lead to steepest descent of the loss surface are more likely to be the “generalizable” patterns of the data, which are LFC.

**Heuristics:** We also test how different training methods react to LFC and HFC, including

- **Dropout [26]:** A heuristic that drops weights randomly during training. We apply dropout on fully-connected layers with $p = 0.5$.

- **Mix-up [66]:** A heuristic that linearly integrate samples and their labels during training. We apply it with standard hyperparameter $\alpha = 0.5$.

- **BatchNorm [28]:** A method that perform the normalization for each training mini-batch to accelerate Deep Network training process. It allows us to use a much higher learning rate and reduce overfitting, similar with Dropout. We apply it with setting scale $\gamma$ to 1 and offset $\beta$ to 0.

- **Adversarial Training [42]:** A method that augments the data through adversarial examples generated by a threat model during training. It is widely considered as one of the most successful adversarial robustness (defense) method. Following the popular choice, we use PGD with $\epsilon = 8/255 (\epsilon = 0.03)$ as the threat model.

We illustrate the results in Figure 5, where the first panel is the vanilla set-up, and then each one of the four heuristics are tested in the following four panels.

Dropout roughly behaves similarly to the vanilla set-up in our experiments. Mix-up delivers a similar prediction accuracy, however, it catches much more HFC, which is probably not surprising because the mix-up augmentation clearly does not encourage anything about LFC explicitly, and the performance gain is likely due to attention towards HFC.

Adversarial training mostly behaves as expected: it reports a lower prediction accuracy, which is likely due to the trade-off between robustness and accuracy. It also reports a smaller generalization gap, which is likely as a result of picking up “generalizable” patterns, as verified by its invariance towards HFC (e.g., $r = 12$ or $r = 16$). However, adversarial training seems to be sensitive to the HFC when $r = 4$, which is ignored even by the vanilla set-up.

The performance of BatchNorm is notable: compared to the vanilla set-up, BatchNorm picks more information in both LFC and HFC, especially when $r = 4$ and $r = 8$. This BatchNorm’s tendency in capturing HFC is also related to observations that BatchNorm encourages adversarial vulnerability [18].

**Other Tests:** We have also tested other heuristics or methods by only changing along one dimension while the rest is fixed the same as the vanilla set-up in Section 4.

Model architecture: We tested LeNet [37], AlexNet [34], VGG [52], and ResNet [23]. The ResNet architecture seems advantageous toward previous inventions at different levels: it reports better vanilla test accuracy, smaller generalization gap (difference between training and testing accuracy), and a weaker tendency in capturing HFC.

Optimizer: We tested SGD, ADAM [32], AdaGrad [16], AdaDelta [64], and RMSprop. We notice that SGD seems to be the only one suffering from the tendency towards significantly capturing HFC, while the rest are on par within our experiments.

5.2. A hypothesis on Batch Normalization

Based on the observation, we hypothesized that one of BatchNorm’s advantage is, through normalization, to align the distributional disparities of different predictive signals. For example, HFC usually shows smaller magnitude than...
LFC, so a model trained without BatchNorm may not easily pick up these HFC. Therefore, the higher convergence speed may also be considered as a direct result of capturing different predictive signals simultaneously.

To verify this hypothesis, we compare the performance of models trained with vs. without BatchNorm over LFC data and plot the results in Figure 6.

As Figure 6 shows, when the model is trained with only LFC, BatchNorm does not always help improve the predictive performance, either tested by original data or by corresponding LFC data. Also, the smaller the radius is, the less the BatchNorm helps. Also, in our setting, BatchNorm does not generalize as well as the vanilla setting, which may raise a question about the benefit of BatchNorm.

However, BatchNorm still seems to at least boost the convergence of training accuracy. Interestingly, the acceleration is the smallest when \( r = 4 \). This observation further aligns with our hypothesis: if one of BatchNorm’s advantage is to encourage the model to capture different predictive signals, the performance gain of BatchNorm is the most limited when the model is trained with LFC when \( r = 4 \).

6. Adversarial Attack & Defense

As one may notice, our observation of HFC can be directly linked to the phenomenon of “adversarial example”: if the prediction relies on HFC, then perturbation of HFC will significantly alter the model’s response, but such perturbation may not be observed to human at all, creating the unintuitive behavior of neural networks.

This section is devoted to study the relationship between adversarial robustness and model’s tendency in exploiting HFC. We first discuss the linkage between the “smoothness” of convolutional kernels and model’s sensitivity towards HFC (§6.1), which serves the tool for our follow-up analysis. With such tool, we first show that adversarially robust models tend to have “smooth” kernels (§6.2), and then demonstrate that directly smoothing the kernels (without training) can help improve the adversarial robustness towards some attacks (§6.3).

6.1. Kernel Smoothness vs. Image Frequency

As convolutional theorem [6] states, the convolution operation of images is equivalent to the element-wise multiplication of image frequency domain. Therefore, roughly, if a convolutional kernel has negligible weight at the high-end of the frequency domain, it will weigh HFC accordingly. This may only apply to the convolutional kernel at the first layer because the kernels at higher layers do not directly interact with the data, thus the relationship is not clear.

Therefore, we argue that, to push the model to ignore the HFC, one can consider to force the model to learn the convolutional kernels that have only negligible weights at the high-end of the frequency domain.

Intuitively (from signal processing knowledge), if the convolutional kernel is “smooth”, which means that there is no dramatics fluctuations between adjacent weights, the corresponding frequency domain will see a negligible amount of high-frequency signals. The connections have been mathematically proved [47, 55], but these proved exact relationships are out of the scope of this paper.

6.2. Robust Models Have Smooth Kernels

To understand the connection between “smoothness” and adversarial robustness, we visualize the convolutional kernels at the first layer of the models trained in the
6.3. Smoothing Kernels Improves Adversarial Robustness

The intuitive argument in §6.1 and empirical findings in §6.2 directly lead to a question of whether we can improve the adversarial robustness of models by smoothing the convolutional kernels at the first layer.

Following the discussion, we introduce an extremely simple method that appears to improve the adversarial robustness of models by smoothing the convolutional kernels at the first layer.

For a convolutional kernel $w$, we use $i$ and $j$ to denote its column and row indices, thus $w_{i,j}$ denotes the value at $i^{th}$ row and $j^{th}$ column. If we use $\mathcal{N}(i, j)$ to denote the set of the spatial neighbors of $(i, j)$, our method is simply:

$$w_{i,j} = w_{i,j} + \sum_{(h,k)\in\mathcal{N}(i,j)} \rho w_{h,k},$$

where $\rho$ is a hyperparameter of our method. We fix $\mathcal{N}(i, j)$ to have eight neighbors. If $(i, j)$ is at the edge, then we simply generate the out-of-boundary values by duplicating the values on the boundary.

In other words, we try to smooth the kernel through simply reducing the adjacent differences by mixing the adjacent values. The method barely has any computational load, but appears to improve the adversarial robustness of $M_{\text{natural}}$ and $M_{\text{adv}}$ towards FGSM and PGD, even when $M_{\text{adv}}$ is trained with PGD as the threat model.

In Figure 7, we visualize the convolutional kernels with our method applied to $M_{\text{natural}}$ and $M_{\text{adv}}$ with $\rho = 1.0$, denoted as $M_{\text{natural}}(\rho = 1.0)$ and $M_{\text{adv}}(\rho = 1.0)$, respectively. As the visualization shows, the resulting kernels tend to show a significantly smoother pattern.

We test the robustness of the models smoothed by our method against FGSM and PGD with different choices of $\epsilon$, where the maximum of perturbation is 1.0. As Table 2 shows, when our smoothing method is applied, the performance of clean accuracy directly plunges, but the performance of adversarial robustness improves. In particular, our method helps when the perturbation is allowed to be relatively large. For example, when $\epsilon = 0.09$ (roughly 23/255), $M_{\text{natural}}(\rho = 1.0)$ even outperforms $M_{\text{adv}}$. In general, our method can easily improve the adversarial robustness of $M_{\text{natural}}$, but can only improve upon $M_{\text{adv}}$ in the case where $\epsilon$ is larger, which is probably because the $M_{\text{adv}}$ is trained with PGD ($\epsilon = 0.03$) as the threat model.

7. Beyond Image Classification

We aim to explore more than image classification tasks. We investigate in the object detection task. We use RetinaNet [40] with ResNet50 [23] + FPN [39] as the backbone. We train the model with COCO detection train set [41] and perform inference in its validation set, which includes 5000 images, and achieve an MAP of 35.6%.

Then we choose $r = 128$ and maps the images into $x_l$ and $x_h$, and test with the same model and get 27.5% MAP with LFC and 10.7% MAP with HFC. The performance drop from 35.6% to 27.5% intrigues us so we further study whether the same drop should be expected from human.

7.1. Performance Drop on LFC

The performance drop from the $x$ to $x_l$ may be expected because $x_l$ may not have the rich information from the original images when HFC are dropped. In particular, different from image classification, HFC may play a significant role in depicting some objects, especially the smaller ones.

Figure 8 illustrates a few examples, where some objects are recognized worse in terms of MAP scores when the input images are replaced by the low-frequent counterparts. This disparity may be expected because the low-frequent images tend to be blurry and some objects may not be clear to a human either (as the left image represents).
8. Discussion: Are HFC just Noises?

To answer this question, we experiment with another frequently used image denoising method: truncated singular value decomposition (SVD). We decompose the image and separate the image into one reconstructed with dominant singular values and one with trailing singular values. With this set-up, we find much fewer images supporting the story in Figure 2. Our observations suggest the signal CNN exploits is more than just random “noises”.

9. Conclusion & Outlook

We investigated how image frequency spectrum affects the generalization behavior of CNN, leading to multiple interesting explanations of the generalization behaviors of neural networks from a new perspective: there are multiple signals in the data, and not all of them align with human’s visual preference. As the paper comprehensively covers many topics, we briefly reiterate the main lessons learned:

- CNN may capture HFC that are misaligned with human visual preference (§3), resulting in generalization mysteries such as the paradox of learning label-shuffled data (§4) and adversarial vulnerability (§6).
- Heuristics that improve accuracy (e.g., Mix-up and BatchNorm) may encourage capturing HFC (§5). Due to the trade-off between accuracy and robustness (§3), we may have to rethink the value of them.
- Adversarially robust models tend to have smooth convolutional kernels, the reverse is not always true (§6).
- Similar phenomena are noticed in the context of object detection (§7), with more conclusions yet to be drawn.

Looking forward, we hope our work serves as a call towards future era of computer vision research, where the state-of-the-art is not as important as we thought.

- A single numeric on the leaderboard, while can significantly boost the research towards a direction, does not reliably reflect the alignment between models and human, while such an alignment is arguably paramount.
- We hope our work will set forth towards a new testing scenario where the performance of low-frequent counterparts needs to be reported together with the performance of the original images.
- Explicit inductive bias considering how a human views the data (e.g., [58, 57]) may play a significant role in the future. In particular, neuroscience literature have shown that human tend to rely on low-frequent signals in recognizing objects [4, 5], which may inspire development of future methods.

Table 2. Prediction performance of models against different adversarial attacks with different $\epsilon$.

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>Clean $\epsilon = 0.03$</th>
<th>FGSM $\epsilon = 0.06$</th>
<th>PGD $\epsilon = 0.06$</th>
<th>FGSM $\epsilon = 0.09$</th>
<th>PGD $\epsilon = 0.09$</th>
<th>FGSM $\epsilon = 0.03$</th>
<th>PGD $\epsilon = 0.03$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{\text{natural}}$</td>
<td>0.856</td>
<td>0.107</td>
<td>0.099</td>
<td>0.044</td>
<td>0.003</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>$M_{\text{natural}}(\rho = 0.10)$</td>
<td>0.815</td>
<td>0.149</td>
<td>0.105</td>
<td>0.073</td>
<td>0.009</td>
<td>0.002</td>
<td>0.001</td>
</tr>
<tr>
<td>$M_{\text{natural}}(\rho = 0.25)$</td>
<td>0.743</td>
<td>0.16</td>
<td>0.11</td>
<td>0.079</td>
<td>0.021</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>$M_{\text{natural}}(\rho = 0.50)$</td>
<td>0.674</td>
<td>0.17</td>
<td>0.11</td>
<td>0.083</td>
<td>0.031</td>
<td>0.016</td>
<td>0.014</td>
</tr>
<tr>
<td>$M_{\text{natural}}(\rho = 1.0)$</td>
<td>0.631</td>
<td>0.171</td>
<td>0.14</td>
<td>0.127</td>
<td>0.086</td>
<td>0.078</td>
<td>0.078</td>
</tr>
<tr>
<td>$M_{\text{adversarial}}$</td>
<td>0.707</td>
<td>0.435</td>
<td>0.232</td>
<td>0.137</td>
<td>0.403</td>
<td>0.138</td>
<td>0.038</td>
</tr>
<tr>
<td>$M_{\text{adversarial}}(\rho = 0.10)$</td>
<td>0.691</td>
<td>0.412</td>
<td>0.192</td>
<td>0.109</td>
<td>0.379</td>
<td>0.13</td>
<td>0.047</td>
</tr>
<tr>
<td>$M_{\text{adversarial}}(\rho = 0.25)$</td>
<td>0.667</td>
<td>0.385</td>
<td>0.176</td>
<td>0.097</td>
<td>0.352</td>
<td>0.116</td>
<td>0.04</td>
</tr>
<tr>
<td>$M_{\text{adversarial}}(\rho = 0.50)$</td>
<td>0.653</td>
<td>0.365</td>
<td>0.18</td>
<td>0.106</td>
<td>0.334</td>
<td>0.121</td>
<td>0.062</td>
</tr>
<tr>
<td>$M_{\text{adversarial}}(\rho = 1.0)$</td>
<td>0.638</td>
<td>0.356</td>
<td>0.223</td>
<td>0.186</td>
<td>0.337</td>
<td>0.175</td>
<td>0.131</td>
</tr>
</tbody>
</table>

Code used in the paper: https://github.com/HaohanWang/HFC
Acknowledgements

This material is based upon work supported by NIH R01GM114311, NIH P30DA035778, and NSF IIS1617583. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Institutes of Health or the National Science Foundation.

References


[24] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defense: Ensembles of weak defenses are not strong. In 11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17), 2017. 2


[27] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. arXiv preprint arXiv:1905.02175, 2019. 1


