# SPECTRAL BIAS IN PRACTICE: THE ROLE OF FUNCTION FREQUENCY IN GENERALIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Despite their ability to represent highly expressive functions, deep learning models trained with SGD seem to find simple, constrained solutions that generalize surprisingly well. Spectral bias – the tendency of neural networks to prioritize learning low frequency functions – is one possible explanation for this phenomenon, but so far spectral bias has only been observed in theoretical models and simplified experiments. In this work, we propose methodologies for measuring spectral bias in modern image classification networks. We find that these networks indeed exhibit spectral bias, and that networks that generalize well strike a balance between having enough complexity (*i.e.* high frequencies) to fit the data while being simple enough to avoid overfitting. For example, we experimentally show that larger models learn high frequencies faster than smaller ones, but many forms of regularization, both explicit and implicit, amplify spectral bias and delay the learning of high frequencies. We also explore the connections between function frequency and image frequency and find that spectral bias is sensitive to the low frequencies prevalent in natural images. Our work enables measuring and ultimately controlling the spectral behavior of neural networks used for image classification, and is a step towards understanding why deep models generalize well.

## 1 INTRODUCTION

Two fundamental questions in machine learning are why overparameterized models generalize and how to make them more robust to distribution shift. Resolving both of these questions requires understanding how complex our models should be.

For instance, it is thought that overparameterized models generalize well because there are implicit regularizers that constrain the complexity of the learned functions. However, the precise nature of these implicit regularizers, and their importance in practice, remains unclear. As for how to achieve robustness, no consensus has emerged regarding function complexity. On the one hand, Cranko et al. (2018) and Leino et al. (2021) argue that Lipschitz smoothness of the learned function offers a guarantee of robustness; on the other, Madry et al. (2019) argues that a robust model must actually be more complex than its non-robust counterpart.

One window into function complexity is spectral bias – the tendency of neural networks to learn low frequency (simple and smooth) functions early in training, gradually increasing the frequency (complexity) of the learned function as training proceeds. Foundational work in this area has shown theoretical evidence of spectral bias by analyzing convergence rates of neural networks towards functions of different frequencies and showing that they converge to low frequency functions at a faster rate (Basri et al., 2019; Rahaman et al., 2019).

In practice, spectral bias is difficult to measure: the most direct method involves taking a Fourier transform with respect to the input, which is expensive to compute due to the high dimensionality of images. Early experimental work focused on low-dimensional synthetic data (Basri et al., 2019; 2020) or used proxy measurements of spectral bias by inserting label noise of various frequencies during training (Rahaman et al., 2019). However, the label noise method is limited to binary classification/regression and involves modifying the training data, making it difficult to disentangle how other changes to the training data (such as data augmentation) affect the spectral content of the

learned function. Thus, it remains an open question as to whether modern neural networks exhibit spectral bias and what role it plays in generalization.

In this work, we investigate model complexity through the lens of spectral bias, by introducing experimental methods to study the frequency decomposition of the functions learned by modern image classification networks. We find that neither simplicity nor complexity is purely beneficial or detrimental to performance; rather, a task-appropriate balance between the two is ideal.

**Contributions.** We extend the label noise procedure of Rahaman et al. (2019) to enable measuring spectral bias in multi-class classification and apply this technique to understand the function frequencies present in high-accuracy models on CIFAR-10 (Krizhevsky et al.). We also introduce a second method for measuring the smoothness of a learned function via linear interpolation between test examples. This offers a proxy measurement of spectral bias without the need to modify training labels, allowing us to probe the effects of the training data on the learned function frequencies.

Using these experimental methods, we find that increasing the model size decreases spectral bias; larger models learn high frequencies faster. However, we observe that more accurate models are not always higher frequency. Several forms of explicit and implicit regularization, including weight decay, increasing the dataset size, and applying Mixup (Zhang et al., 2018) data augmentation, increase spectral bias and produce a smoother learned function. Our experiments suggest that an ideal function should include high enough frequencies to fit the data but avoid unnecessary high frequencies that can harm generalization.

This perspective allows us to shed light on the mechanism behind self-distillation, in which a student model is trained to fit the predictions of a teacher model. Using our linear interpolation methodology, we observe that self-distillation produces a student model whose learned function is smoother than that of its teacher. This suggests that the teacher model acts as a sort of low-pass filter on the target function, perhaps making it more accessible to the spectrally-biased student.

Finally, we explore the relationship between image frequency and function frequency. By further extending the label noise methodology of Rahaman et al. (2019) to study spectral bias in directions of interest through the input space, we find that models are most sensitive to the low image frequencies common in natural images.

## 2 Related Work

**Implicit bias.** A common belief is that some form of implicit bias imposed by the training procedure may account for the generalization ability of overparameterized neural networks, and accordingly much research has been directed towards understanding these implicit biases (Soudry et al., 2018; Zhang et al., 2017; Keskar et al., 2017; Gunasekar et al., 2019; 2020; Hardt et al., 2016; Hoffer et al., 2018; Belkin et al., 2018; Neyshabur et al., 2018).

For example, Neyshabur et al. (2015) observed that network capacity alone is insufficient to explain the generalization of deep models, and pointed to some other implicit bias or regularization that enables generalization. In followup work, Neyshabur et al. (2017) considered possible explanations for generalization of overparameterized networks, including Lipschitz constants (which generally have upper bounds too large to explain generalization) and width or flatness of optimization minima.

More recently, Mania et al. (2019) showed empirically that models make similar mistakes, more than we would expect given just their capacity and training data, which suggests that different models share similar implicit biases. Gunasekar et al. (2019) compared the biases of linear convolutional and fully connected networks, showing that convolutional structure encourages the learned function to be sparse in the frequency domain, whereas fully-connected networks seek only to maximize the margin. Nakkiran et al. (2019) studied a proxy for spectral bias, showing that the function learned early in training is nearly linear, and becomes increasingly nonlinear as training proceeds.

**Spectral bias.** Spectral bias is a form of implicit bias. Basri et al. (2019) studied spectral bias using a linear model of SGD training dynamics to show that models learn low frequency (simple) functions early in training and then gradually learn higher frequencies as training proceeds. This work assumed training data that is distributed uniformly on the hypersphere. Basri et al. (2020) extended the analysis to consider nonuniformly spaced training data, and found that learning is faster

where samples are denser; if sampling is nonuniform then at a given point in training the learned function will be higher frequency in regions with denser samples. Rahaman et al. (2019) used a more direct analysis to show the same spectral bias toward low frequency functions. This paper also suggested that, when training data lie on a manifold (as natural images are believed to), increasing the complexity of the manifold makes it easier for models to fit high frequency functions over the data. Additionally, Rahaman et al. (2019) posited that high frequency components of the learned function are most sensitive to perturbations in the model parameters, connecting back to the idea of flat optimization minima (Neyshabur et al., 2017). This paper proposed experimental methods to study spectral bias in image classification, but focused on a binary subset of the relatively simple MNIST dataset (Deng, 2012), with mean square error loss. Our work is a direct extension of this line of research on spectral bias, particularly Rahaman et al. (2019), which we extend to a more complex multiclass classification dataset with the more common cross-entropy loss.

**Model sensitivity to image frequency.** While we focus on *function frequency* in this work, prior research aimed to understand model sensitivity to image frequencies. Jo & Bengio (2017) found that CNNs are sensitive to Fourier statistics of the training data, even those irrelevant to human viewers. Ortiz-Jimenez et al. (2020) argued that models are sensitive primarily to discriminative Fourier directions in the training data, and have large margins in less-discriminative directions. They posit that adversarial training induces large margins in orthogonal directions, thereby improving robustness. Yin et al. (2020) introduced a procedure to measure the sensitivity of trained models to Fourier image perturbations, and showed that adversarial training and Gaussian data augmentation increase robustness to high frequency perturbations but increase sensitivity to low frequency perturbations. Our work adds to this line of research a study of the interplay between two notions of frequency, the function frequency involved in spectral bias and the image frequencies present in the data.

## 3 METHODOLOGY

The goal of our work is to measure the complexity of neural network functions through the lens of frequency. In low dimensions, measuring the frequency decomposition of a function is straightforward and tractable: evaluate the function at dense, uniform sampling positions and compute its discrete Fourier transform. However, the function of interest in image classification is unavoidably high-dimensional, mapping images (with thousands of pixel values) to object classes. It would be intractable even to collect sufficient samples of this function to compute a discrete Fourier transform, let alone compute the transform itself. Instead, we employ two complementary approaches to measure informative proxies of this frequency decomposition.

### 3.1 LABEL SMOOTHING

The core idea for measuring function frequency introduced by Rahaman et al. (2019) is to construct a sinusoid over the space of images, and to use that sinusoid as a form of label noise during training. Let $\mathcal{D} = \{(\mathbf{X}_i, \boldsymbol{y}_i)\}_{i=1}^{n_{train}}$ be the training examples and $\mathcal{D}_{\text{valid}} = \{(\mathbf{X}_j, \boldsymbol{y}_j)\}_{j=1}^{n_{val}}$ be the validation examples, with $\mathbf{X}_i$ an image and $\boldsymbol{y}_i$ a one-hot class encoding, where $n_{train}$ is the number of training examples, $n_{val}$ is the number of validation examples, $d$ is the side length of an image (assumed to be square for simplicity), $c$ is the number of color channels, and $k$ is the number of classes, so $\mathbf{X}_i \in \mathbb{R}^{d \times d \times c}$ and $\boldsymbol{y}_i \in \mathbb{R}^k$.

To extend this procedure to the multi-class setting, we add noise of various frequencies to a multi-dimensional label vector via label smoothing (Szegedy et al., 2015). Let $S : \mathbb{R}^{d \times d \times c} \to [0, 1]$ be a target function that maps an input image $\mathbf{X}_i$ to a scalar value between 0 and 1. We apply label smoothing to each label $\boldsymbol{y}_i$, mapping it to $\bar{\boldsymbol{y}}_i = \boldsymbol{y}_i(1 - S(\mathbf{X}_i)) + \frac{1}{k}$. We then train from scratch using the original examples $\mathbf{X}_i$ and their smoothed labels $\bar{\boldsymbol{y}}_i$. Finally, we evaluate on the validation images $\mathbf{X}_j$, comparing to both their original one-hot labels $\boldsymbol{y}_j$ and smoothed labels $\bar{\boldsymbol{y}}_j = \boldsymbol{y}_j(1 - S(\mathbf{X}_j)) + \frac{1}{k}$. Our experiments use the CIFAR-10 dataset (Krizhevsky et al.), where $n_{train} = 50000$, $d = 32$, $c = 3$, and $k = 10$; however, our experimental setup is sufficiently general that it should transfer easily to any image classification task.

As training proceeds, we see both the training loss (between predictions and smoothed labels $\bar{\boldsymbol{y}}_i$) and smoothed validation loss (between predictions and smoothed labels $\bar{\boldsymbol{y}}_j$) decrease. The clean validation loss (between predictions and one-hot labels $\boldsymbol{y}_j$), however, initially decreases, but at some

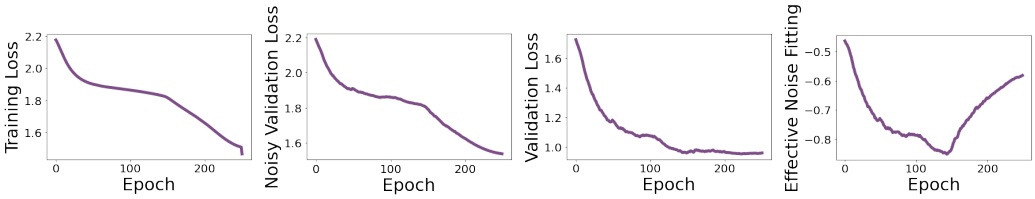



Figure 1: **Effective noise fitting shows when a network fits a target frequency function.** At roughly epoch 150, effective noise fitting *(Right)*, the difference between clean and perturbed validation loss, exhibits a clear "dip" when the model begins to fit the target function: training *(Left)* and validation *(Center Left)* loss on the perturbed function drop, while improvement stalls on clean validation data *(Center Right)*. Here, we train a shake_shake_96 with radial wave label smoothing at frequency 0.04. For visual clarity, we apply exponential averaging to all curves.

point in training, it plateaus or begins to increase. At this point, the model has begun to learn the target function $S$. Accordingly, we introduce **effective noise fitting**, defined as the difference between the validation loss on one-hot labels and the validation loss on labels smoothed with the same function $S$ that was applied to the training labels. Effective noise fitting is illustrated using shake_shake_96 in figure 1.

$$\textbf{effective noise fitting} = \textbf{clean validation loss} - \textbf{noisy validation loss} \tag{1}$$

By choosing different functions $S$, we can probe nuances of spectral bias. Typically (inspired by Rahaman et al. (2019)) we choose a radial wave: $S(\mathbf{X}) = \sin(2\pi f(\|\mathbf{X}\| - \mathbb{E}_{\mathcal{D}}\|\mathbf{X}\|))$, where we can vary the frequency $f$ to understand spectral bias at this global scale. We can also choose more targeted functions $S$; for instance, if $\mathbf{V}$ is a direction of interest through the space of images (*i.e.* $\mathbf{V}$ is an image-shaped vector of unit norm), we can construct $S(\mathbf{X}) = \sin(2\pi f \langle \mathbf{X}, \mathbf{V} \rangle)$. This allows us to vary both the frequency and direction of the target sinusoid, to understand model sensitivity along different directions through image space.

However, label smoothing has a few limitations. It requires retraining each model from scratch, with substantial investment of time and computational resources. Because it involves perturbing the training labels, questions about the interaction between the training dataset and spectral bias, such as the effects of data augmentation, are not directly answered by this technique. For the same reason, we cannot use this methodology to study the spectral decomposition of existing trained models.

### 3.2 Linear Interpolation

To complement the label smoothing approach, we also propose a measurement methodology based on linear interpolation between validation images. Although we cannot take dense, regularly-spaced samples of the learned function throughout the input space, we can do so along specific paths through input (image) space, and thereby glean glimpses into the spectral content of the learned function. We begin by choosing a random subset of the validation set $\mathcal{D}_{\text{valid}}$ according to some desired constraints (*e.g.* a specific class), and grouping the images into pairs. For each pair of images we evaluate the model at fixed, regular intervals $\Delta$ (in our experiments, $\Delta = 1$) along the linearly interpolating path between them. Note that the interpolations are done between the normalized images; figure 2 *(Bottom)* shows these interpolations rescaled to $[0, 1]$ to be visually easier to interpret.

Using these sampled interpolating paths, we compute a one-dimensional statistic of the softmax predictions that serves as a proxy for the local spectral content of the learned function. Let $\hat{\boldsymbol{y}}_t$ denote the softmax prediction at the $t$'th sample along the path starting from one image and linearly interpolating towards another. For each value of $t$, we compute the cumulative difference norm $\|\hat{\boldsymbol{y}}_t - \hat{\boldsymbol{y}}_0\|$. If we see rapid changes in the cumulative difference norm, this is proxy evidence of high frequency components in the learned function, as shown in figure 2 *(Top)*.

We aggregate across many paths based on the effective amount of interpolation: $\lambda$ in the interpolation formula $\lambda \mathbf{X}_1 + (1 - \lambda)\mathbf{X}_2$, where $\mathbf{X}_1$ and $\mathbf{X}_2$ are the images being interpolated. Since paths may vary in length, we normalize the contribution of each path based on its total number of samples so that each path is weighted equally in each $\lambda$ bin. When considering paths between images in the

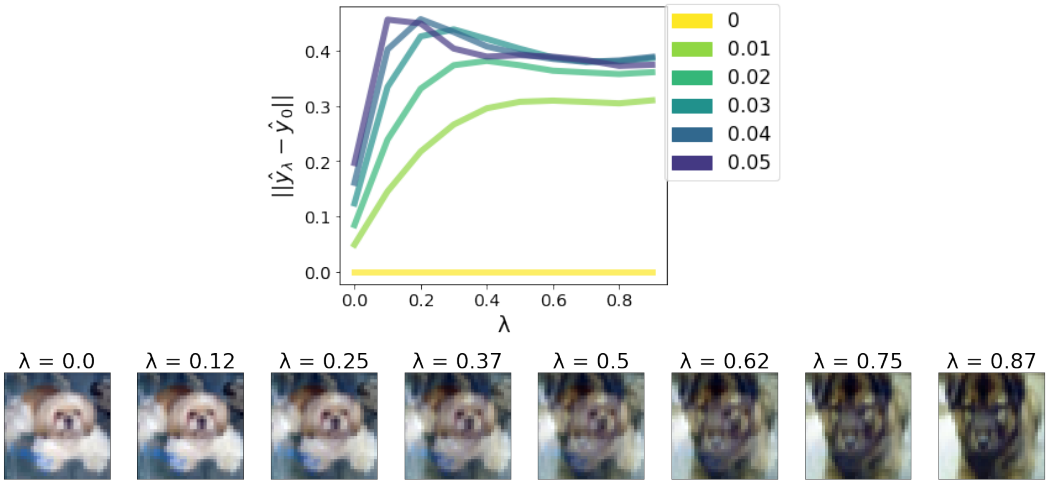



Figure 2: **Higher frequency functions exhibit more rapid changes in output.** *Top*: Interpolation experiment between images of the same class on a toy function: the true one-hot label perturbed by radial wave label smoothing of variable frequency. *Bottom*: Illustration of linear interpolation with a constant sampling distance $10\times$ that used in our experiments.

same class, we choose 200 random, distinct pairs of images from each of the 10 classes and average over the paths defined by these pairs. When interpolating between classes, we choose 200 random, distinct image pairs from each of the 45 (10 choose 2) pairs of distinct classes, and average over the paths defined by these pairs. To remove any unwanted artifacts from differing model accuracies, we exclude any of these pairs for which the model predicts either image incorrectly.

### 3.3 Data and Models

Our experiments use the CIFAR-10 (Krizhevsky et al.) dataset of natural images from ten classes, a mixture of animals and objects. Each image has 32 pixels per side and three color channels. There are 50000 training examples (5000 per class) and 10000 test examples (1000 per class). Images are pixel-wise normalized by the mean and standard deviation of the training images.

We consider six different convolutional neural networks that have achieved high accuracy on this task: two wide-resnets (Zagoruyko & Komodakis, 2017) of different sizes (wrn_32 and wrn_160), three shake-shake regularized networks (Gastaldi, 2017) of different sizes (shake_shake_32, shake_shake_96, and shake_shake_112), and pyramid_net, which uses the even stronger shake-drop regularization (Yamada et al., 2019). Our implementations are based on Cubuk et al. (2019a).

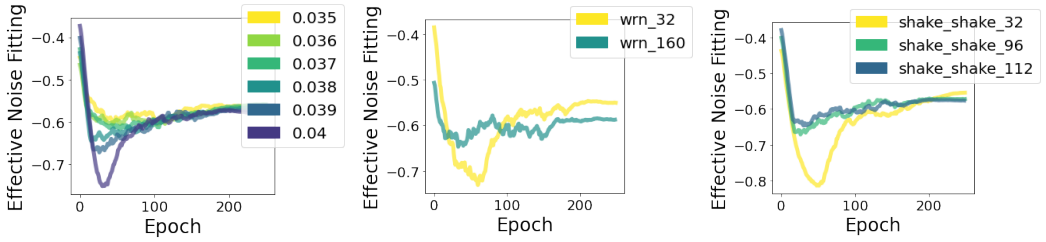
## 4 Results

Throughout this section we show representative examples from our experiments to highlight our main findings. Full results (on all six models we tested) are included in section A.

### 4.1 Spectral Bias and Model Architecture

We begin by applying our label smoothing methodology to find that modern image classification CNNs exhibit spectral bias, learning low frequency target functions early in training and learning higher frequency functions as training proceeds. Note that figure 3 *(Left)* shows spectral bias over a small but illustrative range of frequencies; target functions of sufficiently low frequency are fit almost immediately and target functions of sufficiently high frequency are never learned during the 250 epochs of training we tested.

Although all models we tested exhibit spectral bias, we found that the precise nature of the bias depends on the choice of model. For example, with all else fixed, increasing the width of a model




(a) Varying frequency

(b) Varying model width

Figure 3: **Modern CNNs exhibit spectral bias toward low frequencies, but larger networks learn high frequencies faster.** *Left*: Effective noise fitting for a shake_shake_96 model with radial wave label smoothing of varying frequency; lower frequencies are learned first. *Center*: Effective noise fitting for wide resnets of variable width at frequency 0.039; the larger model learns this frequency first. *Right*: Effective noise fitting for shake-shake models of variable width at frequency 0.039; larger models learn this frequency faster.

decreases its spectral bias, enabling it to fit target frequency functions faster. This trend is evident in figure 3 for the wide-resnet family *(Center)* and the shake-shake family *(Right)*. Results of our label smoothing experiment on all six models we tested are included in section A.1.

### 4.2 Sensitivity to Natural Image Directions

Although label smoothing with a radial wave offers a convenient global picture of spectral bias, by replacing the radial wave with other target functions we can use the same methodology to test more localized aspects of spectral bias. We take a step in this direction by considering the family of target label smoothing functions $S(\mathbf{X}; k) = \sin(2\pi f \langle \mathbf{X}, \mathbf{F}_k \rangle)$, where $\mathbf{F}_k$ is a diagonal Fourier basis image with frequency $k$ and the same dimensions as $\mathbf{X}$, visualized in figure 4 *(Bottom)*.

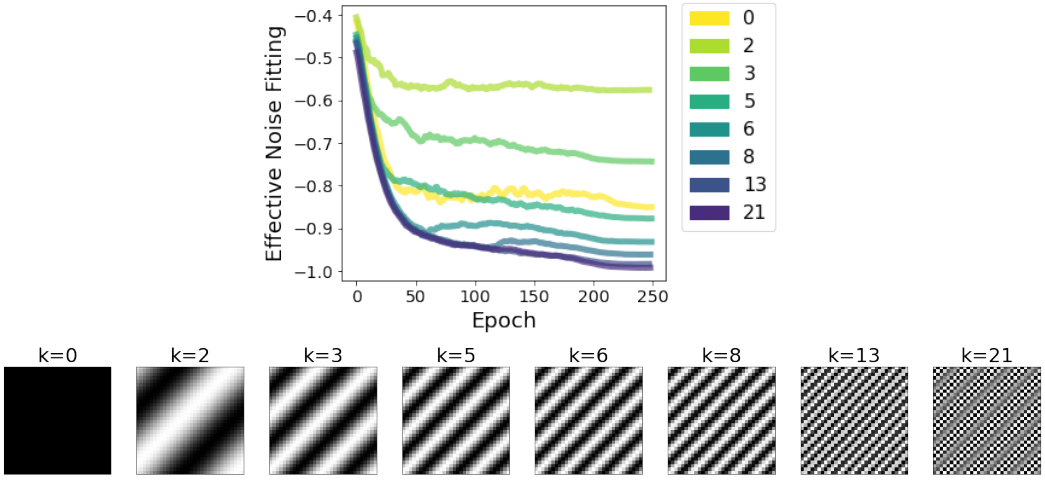



Figure 4: **Models are most sensitive to variations of low (but nonzero) image frequency, which are dominant in natural images.** *Top*: Effective noise fitting of a shake_shake_96 model with label smoothing of frequency 0.038 in various unit norm directions corresponding to Fourier basis images *(Bottom)* indexed by image frequency $k$ (scaled to [0, 1] for visualization).

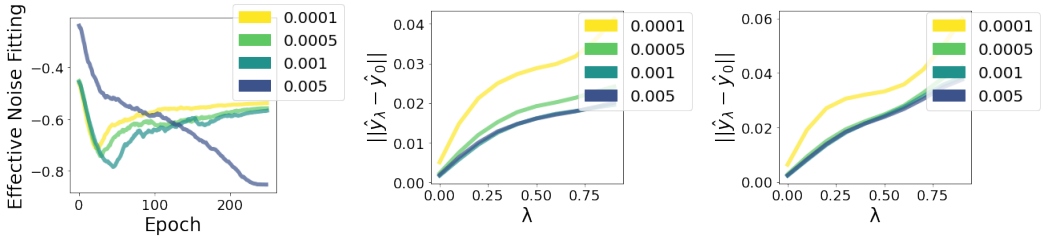
We consider (a subset of) Fourier basis images because the Fourier spectra statistics of natural images are well studied (see, *e.g.* Tolhurst et al. (1992)): natural images tend to be composed of Fourier basis images with amplitude proportional to their inverse spatial frequency. Indeed, in figure 4 we find that shake_shake_96 is more sensitive to label smoothing in low image frequency directions dominant in natural images. This finding is consistent with theoretical predictions of Basri et al.

(2020) that models learn faster in regions of higher density of training examples: since low image frequencies are more common in natural images, the effective sampling density enjoyed by a label smoothing target is higher in these directions.

### 4.3 Spectral Bias and Regularization

We also found a relationship between regularization and spectral bias, which is explored in figure 5 and figure 6. Some forms of regularization, like weight decay and dropout, are amenable to study by either of our methodologies; figure 5 demonstrates agreement between these methods in the context of varying weight decay. Further results showing agreement between these two measurement methodologies are presented in section A.3.

In total we consider 4 types of (explicit and implicit) regularization: weight decay, training set size, data augmentation (Cubuk et al., 2019b;a; Zhang et al., 2018), and the strength of Mixup augmentation (Zhang et al., 2018). The latter three types of implicit regularization involve changes to the training data and are therefore only directly investigated via our interpolation methodology, which we use in figure 6.




(a) Label smoothing

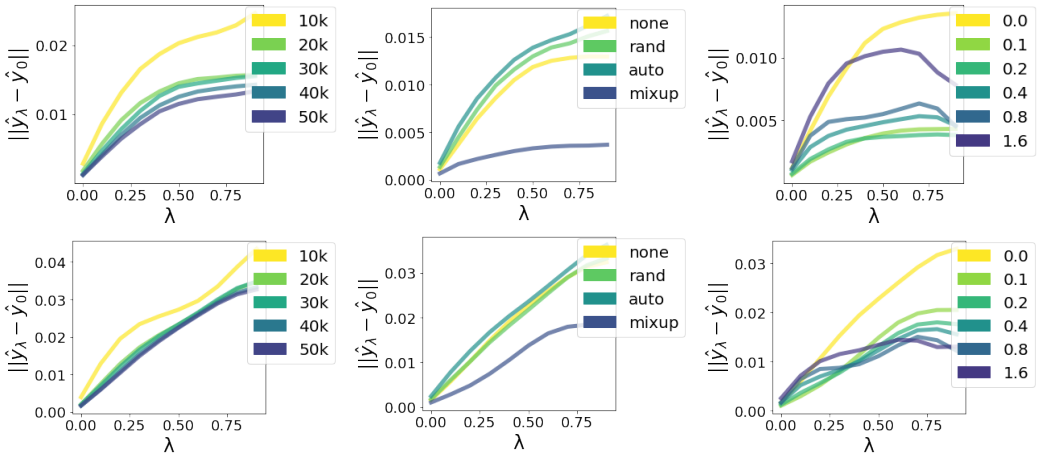
(b) Linear interpolation

Figure 5: **Weight decay inhibits learning high frequencies.** Results here are from shake_shake_32 using the radial wave label smoothing methodology at frequency 0.038 *(Left)* and linear interpolation within *(Center)* and between *(Right)* classes. Both measurement techniques show that increasing the penalty on the norm of the weights produces a stronger spectral bias, in the form of slower fitting of high frequency noise and a smoother learned function.

**Weight Decay.** Weight decay is a penalty on the square norm of the model parameters; a stronger penalty encourages a model to learn smaller-norm weights. Figure 5 shows that increasing the weight decay also encourages a model to learn high frequencies slower *(Left)* and to ultimately be smoother, both between examples of the same class *(Center)* and between examples from different classes *(Right)*. This result is consistent with the expectation that smaller-norm weights yield a learned function with a smaller Lipschitz constant and thus smoother changes in output (as discussed *e.g.* in Szegedy et al. (2014)). This experiment also demonstrates the nuanced relationship between the bandwidth or smoothness of the learned function and its accuracy; shake_shake_32 trained with weight decay 0.0005 or 0.001 achieves higher test accuracy compared to the same architecture trained with either more or less weight decay. Neither complexity nor smoothness is, in itself, purely beneficial or detrimental: the trick is to balance them appropriately for the dataset.

**Training Set Size.** As shown in figure 6 *(Left)* on wrn_160, we find that increasing the number of training examples produces a lower-frequency learned function. Note that all models in figure 6 *(Left)* were trained for the same number of epochs and with the same batch size, so the models trained with more examples were also trained for more gradient steps (since each epoch contained more batches). The models trained with more examples therefore had ample opportunity to fit the high frequencies present in their counterparts trained with fewer examples, and yet did not. This result supports the idea that additional examples serve as an implicit regularization by forcing the model away from more complex solutions that would predict the added examples incorrectly.

**Data Augmentation.** Data augmentation is a common strategy to increase the effective training set size without the expense of actually collecting additional examples. It also enables encoding some




(a) Varying training set size (b) Varying data augmentation (c) Varying Mixup strength

Figure 6: **Implicit regularization in the form of dataset size and Mixup augmentation generally produces a lower-frequency learned function, but RandAugment and AutoAugment can increase function frequency between examples from the same class.** *Left*: Interpolation experiment for wrn_160 trained with varying training set sizes: training with more examples produces a lower-frequency learned function, both within-class *(Top)* and between-class *(Bottom)*, although returns are diminishing with dataset size. *Center*: Training with Mixup (Zhang et al., 2018) makes wrn_160 lower-frequency both within-class *(Top)* and between-class *(Bottom)*, whereas training with RandAugment (Cubuk et al., 2019b) or AutoAugment (Cubuk et al., 2019a) makes it slightly less smooth within-class *(Top)*. *Right*: Modestly increasing the Mixup strength produces lower-frequency learned functions, but further increasing the Mixup can reintroduce high frequencies, particularly within-class *(Top)*.

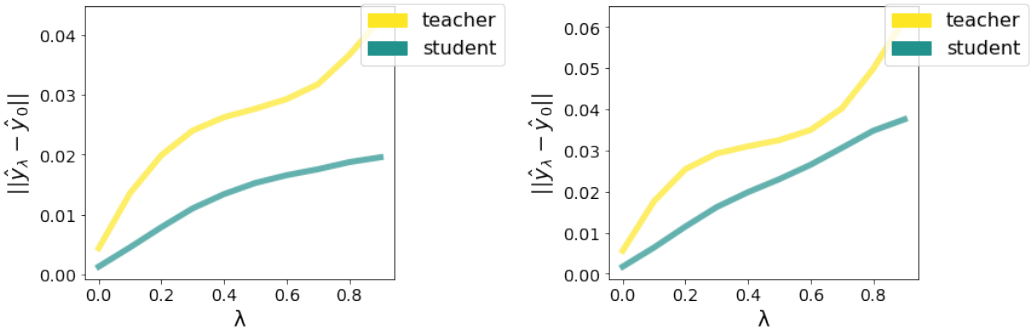
prior knowledge into the dataset, such as invariance to small geometric or lighting transformations. In figure 6 *(Center)* we consider the effects of RandAugment (Cubuk et al., 2019b), AutoAugment (Cubuk et al., 2019a), and Mixup (Zhang et al., 2018) data augmentation strategies, each of which tends to improve the final accuracy of the trained model (with RandAugment producing the most substantial benefits in our experiments). We find that training with RandAugment or AutoAugment tends to make the learned function slightly less smooth between examples of the same class *(Top)*. Mixup (at strength 0.1) makes the learned function substantially smoother, both within-class *(Top)* and between-class *(Bottom)*. This experiment reinforces the nuanced relationship between function complexity and performance; a model can be improved by the addition of necessary complexity (as in RandAugment and AutoAugment) or by the removal of unnecessary complexity (as in Mixup).

**Mixup Strength.** Mixup augmentation (Zhang et al., 2018) perturbs each batch of training data as follows: each example is matched with a partner based on a random permutation of the batch, and each example is then perturbed towards its partner by an interpolation amount $\lambda$ drawn from a symmetric beta distribution (with the same $\lambda$ used for all examples in the batch). We refer to the parameter of this beta distribution as the Mixup strength or amount of Mixup, as it controls the degree to which the augmented images tend to lie close to an original training image or close to the average of two training images. A parameter of 0 corresponds to no augmentation ($\lambda = 0$ or $\lambda = 1$, always using the original images), a parameter of 1 corresponds to the uniform distribution over $\lambda \in [0, 1]$, and a parameter of $\infty$ corresponds to $\lambda = 0.5$, the exact midpoint between a pair of training images. Figure 6 *(Center)* uses Mixup strength 0.1 (one of the values recommended by Zhang et al. (2018)). Figure 6 *(Right)* shows that, as expected, increasing the strength of the Mixup augmentation generally makes the learned function smoother, particularly between examples from different classes *(Bottom)*, but that Mixup that is too strong can induce high frequencies within-class *(Top)*. Here again we find the nuanced relationship that, while some smoothness (or spectral bias toward low frequencies) is beneficial in preventing overfitting, too much causes detrimental underfitting.

### 4.4 SELF-DISTILLATION

In a sense another form of regularization, we finally consider the effect of self-distillation on the smoothness of the learned function. In self-distillation, a 'teacher' model is trained with some form of strong regularization, in our case a combination of weight decay and early stopping based on training loss. A 'student' model with the same architecture is then trained from scratch to fit the pseudolabels produced by the teacher, instead of the original one-hot training labels (alternatively, an interpolation between the pseudolabels and the original labels may be used). Prior research (Furlanello et al., 2018) found that this procedure can train student models that outperform both their teachers and a baseline (the same architecture trained to completion on the one-hot labels).

Prior research has also sought to understand the mechanism behind knowledge distillation and self-distillation in particular. Mobahi et al. (2020) finds that self-distillation acts as a regularizer by limiting the basis functions available to the student to learn. We complement their theoretical work with our interpolation experimental methodology in figure 7, where indeed we find that the student model learns a lower-frequency function than its teacher, when both have the same training loss. We conjecture that this effect is analogous to low-pass prefiltering common in digital signal processing: a high-frequency target function that we cannot adequately sample is first smoothed (in this case by being approximated by a regularized teacher) and then it can be modeled via samples without further loss in fidelity. Without this prefiltering step, our samples are inadequate to capture the true complexity of the target function, so we reconstruct an imperfect version corrupted by aliasing.




(a) Within-class

(b) Between-class

Figure 7: **Self-distillation produces a student model with a lower-frequency learned function than its teacher.** The teacher shake_shake_96 model is trained on one-hot labels and stopped early when training loss reaches a threshold. The student is trained to fit the pseudolabels produced by the teacher until the same training loss threshold is achieved, at which point it has higher validation accuracy than the teacher.

## 5 CONCLUSIONS

In this paper, we introduced two methods to measure spectral bias in modern image classification neural networks, and applied these methods towards the central question:

*What kinds of function frequencies are needed for modern neural networks to generalize?*

Specifically, we applied these methods to examine the impact of a variety of training choices on the learned frequencies. Among training choices that improve validation accuracy, using a larger model, training longer, and using RandAugment (Cubuk et al., 2019b) or AutoAugment (Cubuk et al., 2019a) tend to increase the frequency of the learned function. However, regularizing with modest weight decay, training with more examples, using modest Mixup augmentation (Zhang et al., 2018), and performing self-distillation all improve validation accuracy while decreasing the frequency of the learned function. Our findings support the common wisdom that, while some complexity is necessary to fit the data, unnecessary complexity can harm generalization. Our experimental methods offer a valuable window into precisely what kind of function complexity we should strive for.

### REPRODUCIBILITY STATEMENT

Our code will be released upon publication. Our work uses the publicly-available CIFAR-10 dataset (Krizhevsky et al.).

### ETHICS STATEMENT

Since our work is foundational rather than applied in nature, we do not anticipate any direct ethical concerns. However, as with any foundational research in machine learning, our work has the potential to improve the performance and controllability of machine learning models, particularly in the domain of image classification. These improvements in turn may find use in ethical (*e.g.* medical diagnostics, autonomous vehicles) or ethically questionable (*e.g.* surveillance) applications.

As for direct ethical impacts of our work, we acknowledge the energy costs of training and evaluating the models used in our experiments. One of our key contributions is proposing a method (via linear interpolation) to measure the frequency content of a learned function without retraining, which should enable future experimental research in spectral bias to be less energy-intensive.

## REFERENCES

Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies, 2019.

Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density, 2020.

Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *International Conference on Machine Learning*, pp. 541–549. PMLR, 2018.

Zac Cranko, Simon Kornblith, Zhan Shi, and Richard Nock. Lipschitz networks and distributional robustness, 2018.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2019a.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019b.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks, 2018.

Xavier Gastaldi. Shake-shake regularization, 2017.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks, 2019.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry, 2020.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent, 2016.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks, 2018.

Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical regularities, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima, 2017.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks, 2021.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

Horia Mania, John Miller, Ludwig Schmidt, Moritz Hardt, and Benjamin Recht. Model similarity mitigates test set overuse, 2019.

Hossein Mobahi, Mehrdad Farajtabar, and Peter L. Bartlett. Self-distillation amplifies regularization in hilbert space, 2020.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity, 2019.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning, 2017.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks, 2018.

Guillermo Ortiz-Jimenez, Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Hold me tight! influence of discriminative features on deep network boundaries, 2020.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

DJ Tolhurst, Y_ Tadmor, and Tang Chao. Amplitude spectra of natural images. *Ophthalmic and Physiological Optics*, 12(2):229–232, 1992.

Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019. ISSN 2169-3536. doi: 10.1109/access.2019.2960566. URL http://dx.doi.org/10.1109/ACCESS.2019.2960566.

Dong Yin, Raphael Gontijo Lopes, Jonathon Shlens, Ekin D. Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision, 2020.

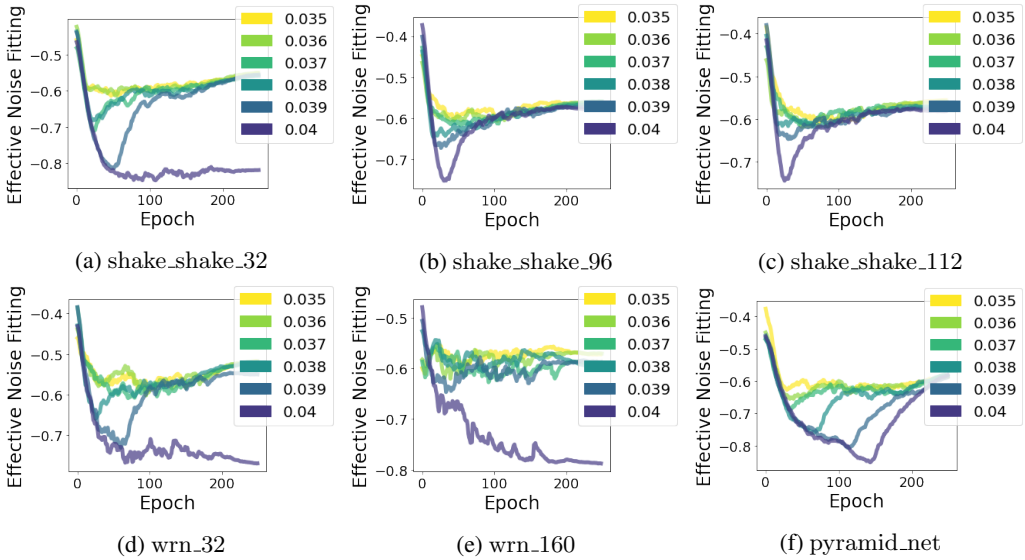
Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

# A APPENDIX

## A.1 LABEL SMOOTHING

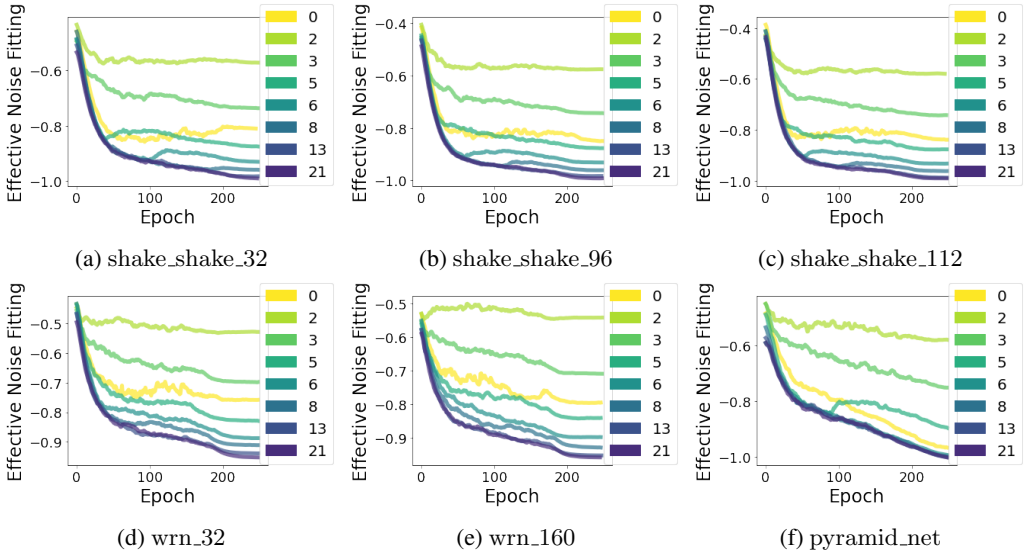



(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

Figure 8: **All six models we tested exhibit spectral bias.** Here we show effective noise fitting when training each model with different frequencies of radial wave label smoothing.

## A.2 SENSITIVITY TO NATURAL IMAGE DIRECTIONS



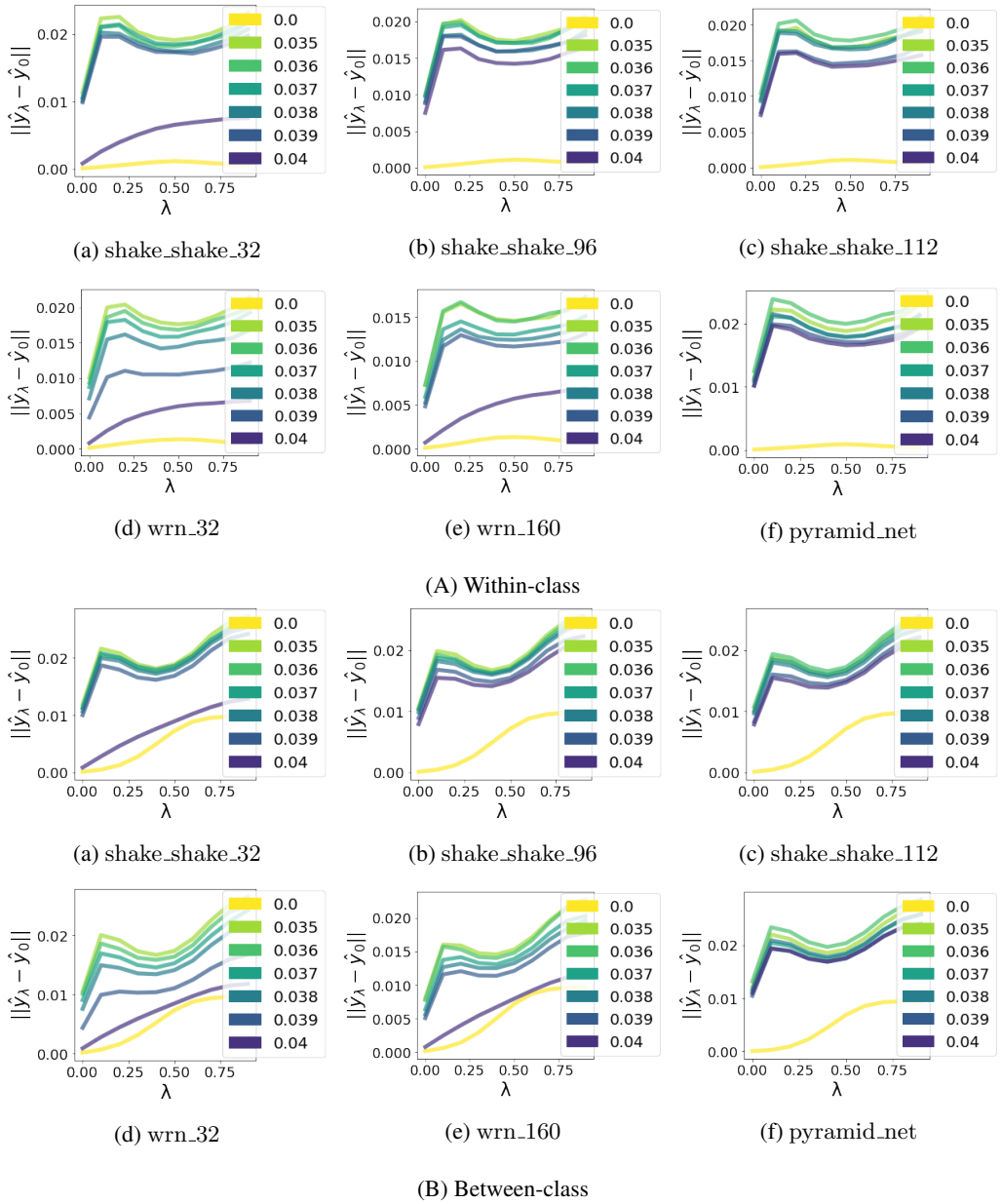

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

Figure 9: **All six models we tested exhibit sensitivity to variations of low (but nonzero) image frequency, which are dominant in natural images.** Here we show effective noise fitting when training each model with label smoothing of frequency 0.038 in various unit norm direction corresponding to Fourier basis images indexed by frequency $k$.

## A.3 Agreement Between Label Smoothing and Linear Interpolation

### A.3.1 Varying the frequency of the radial wave label smoothing




(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(A) Within-class

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(B) Between-class

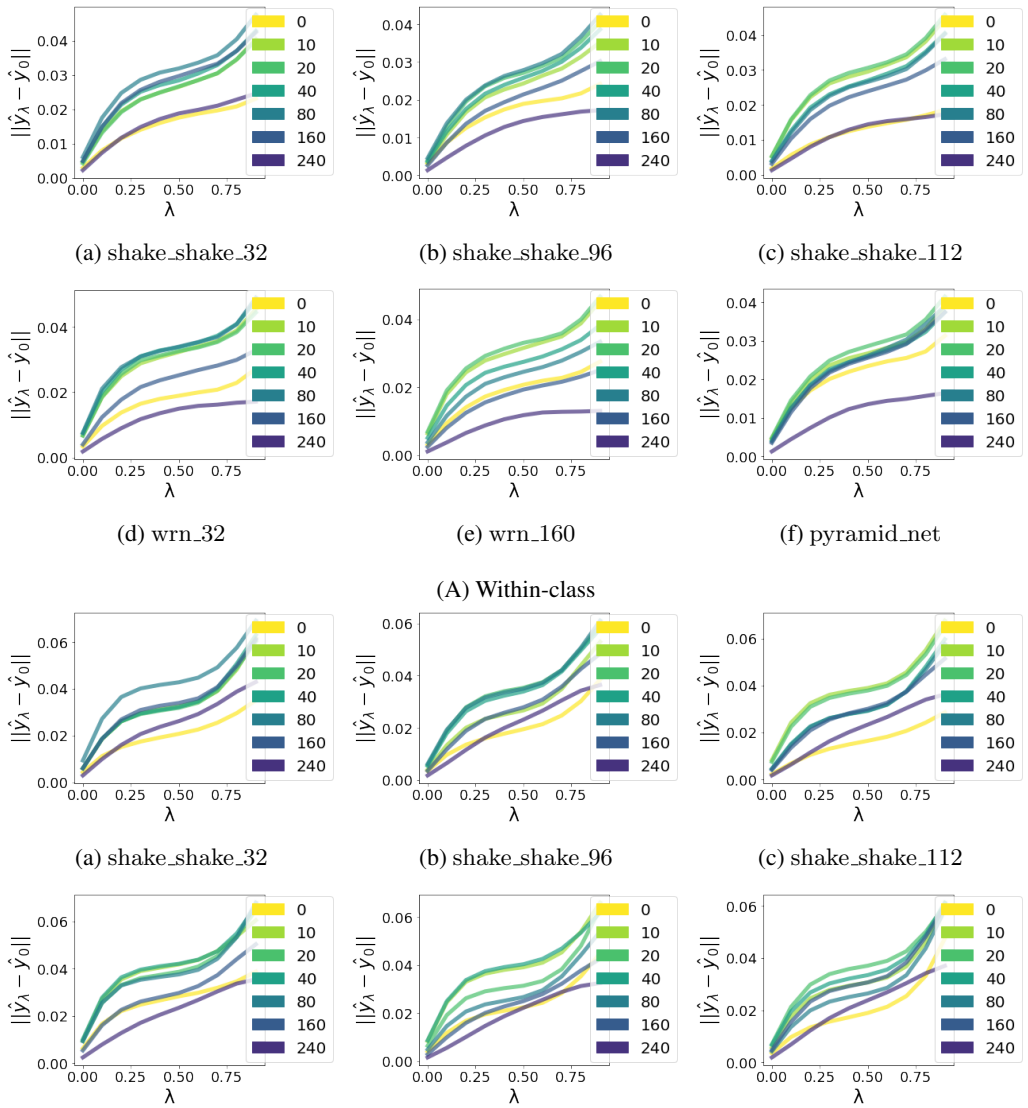
Figure 10: **Spectral bias is evident via interpolation when training with radial wave label smoothing at various frequencies.** Results here parallel those in figure 8 but using the linear interpolation methodology.

When the frequency of the radial wave used for label smoothing increases, models take more time to fit the smoothing noise. Figure 3 *(Left)* and figure 8 show this using label smoothing (relative dip position); the corresponding interpolation experiment is shown in figure 10. The target frequencies between 0.035 and 0.04 are close enough that, were the model to fit each perfectly, the interpolation curves would be nearly visually indistinguishable, as we can tell from figure 2. However, the model is actually smoother when trying to fit higher frequency label smoothing noise, because it fits this noise less well (in addition to fitting it later in training). We can see this effect, for instance, by

noting that wrn_32, wrn_160, and shake_shake_32 fail to fit frequency 0.04, both in figure 8 and figure 10.

It is also worth noting that, in this experiment and repeatedly across our interpolation experiments, the learned function is smoother (has less variation between examples) within-class and less smooth between-class, as evidenced by the relative scaling of the top and bottom sub-figures of figure 10.

### A.3.2 Learning low frequencies first




(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(A) Within-class

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(B) Between-class

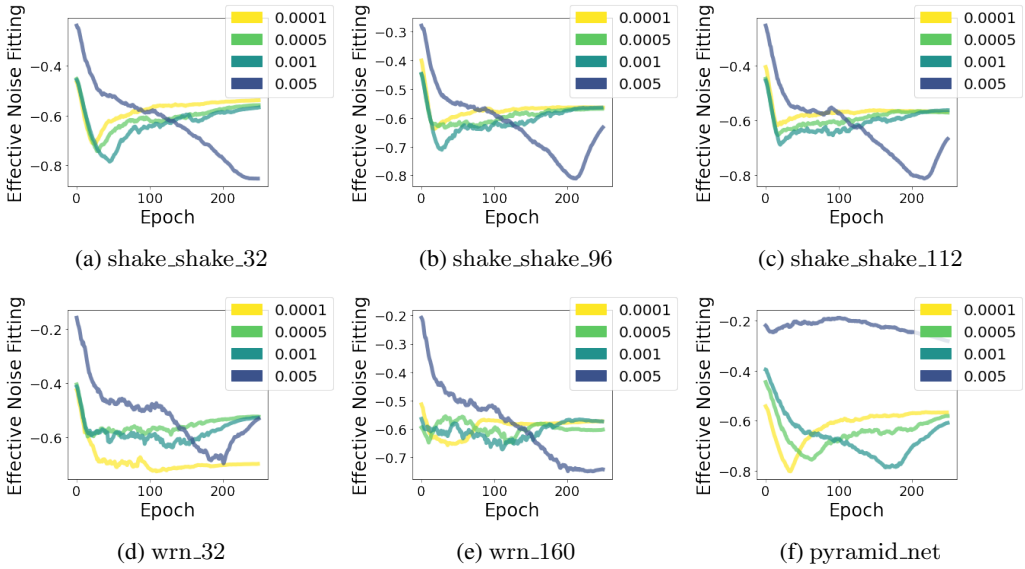
Figure 11: **Spectral bias is evident via interpolation when training as usual and comparing checkpoints at different epochs.**

The dip experiment presented in figure 3 *(Left)* and figure 8 shows that low frequency target functions are learned earlier than higher frequency targets; we can also confirm this finding using interpolation with model checkpoints saved at different epochs throughout training (without any label smoothing). We see that indeed in the early stages of training low frequencies predominate, with higher frequen-

cies entering as training proceeds. Surprisingly, we also see that towards the end of training low frequencies again predominate; the reason for this is an interesting question for further research.

### A.3.3 WEIGHT DECAY

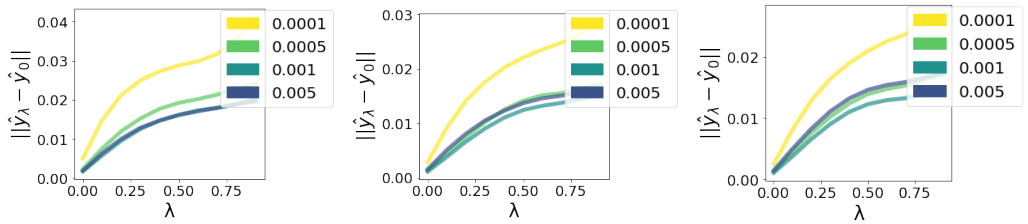
Figure 5 shows on shake_shake_32 via both label noise and linear interpolation that increasing the weight decay results in a smoother (lower frequency) learned function. Figure 12 shows the same result across all six models we tested.



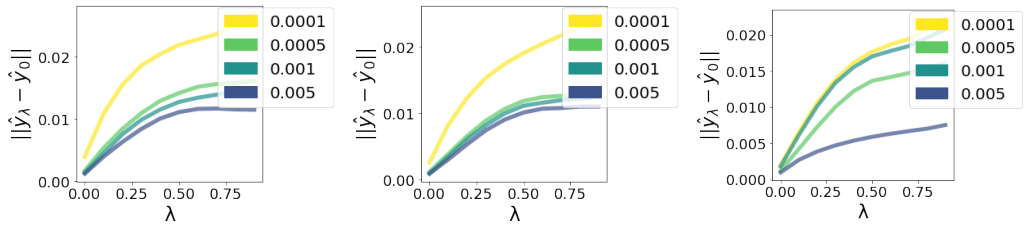

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(A) Radial wave label smoothing

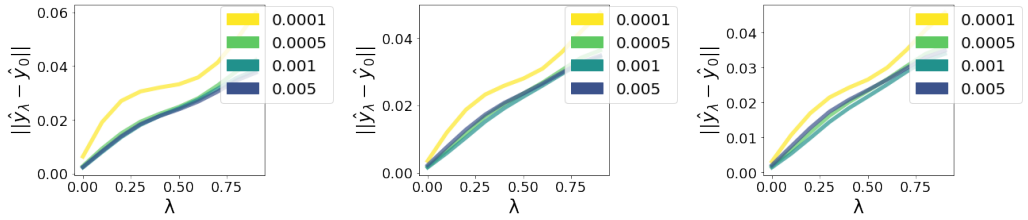



(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

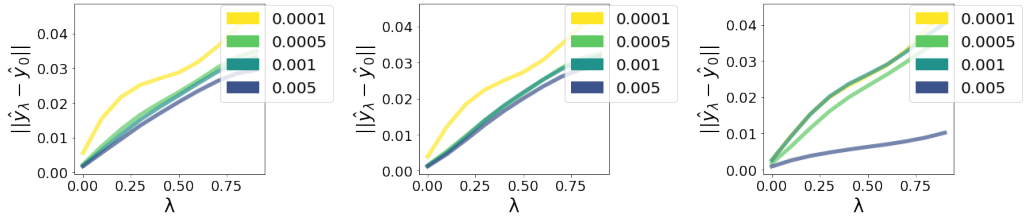



(d) wrn_32 (e) wrn_160 (f) pyramid_net

(B) Within-class linear interpolation




(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112



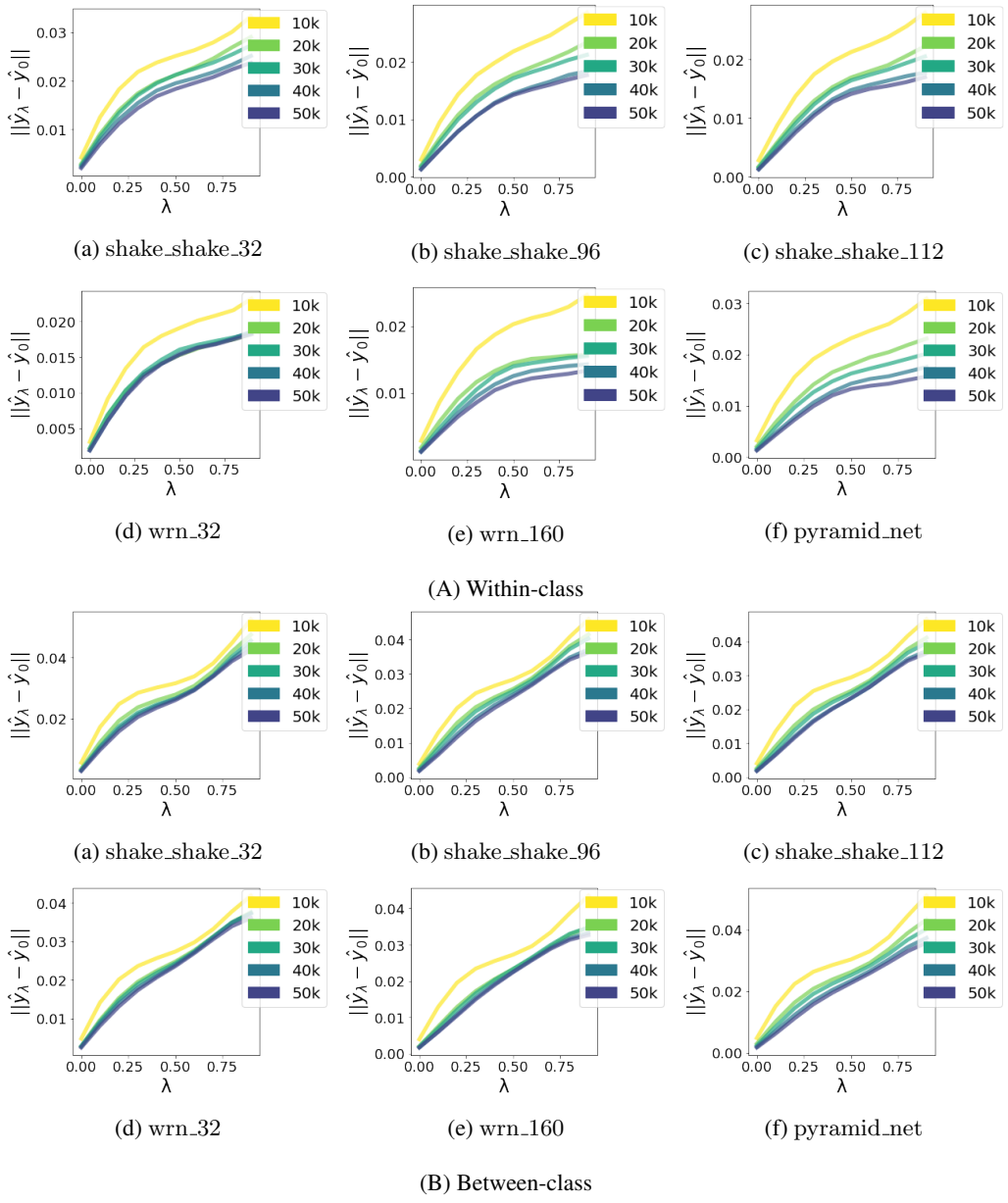

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(C) Between-class linear interpolation

Figure 12: **Weight decay increases spectral bias, producing a lower-frequency learned function.**

### A.4 TRAINING SET SIZE

Figure 6 *(Left)* shows the regularization (frequency reduction) effect of increasing dataset size for wrn_160. Figure 13 shows the same effect on all six models we tested.




(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(A) Within-class

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

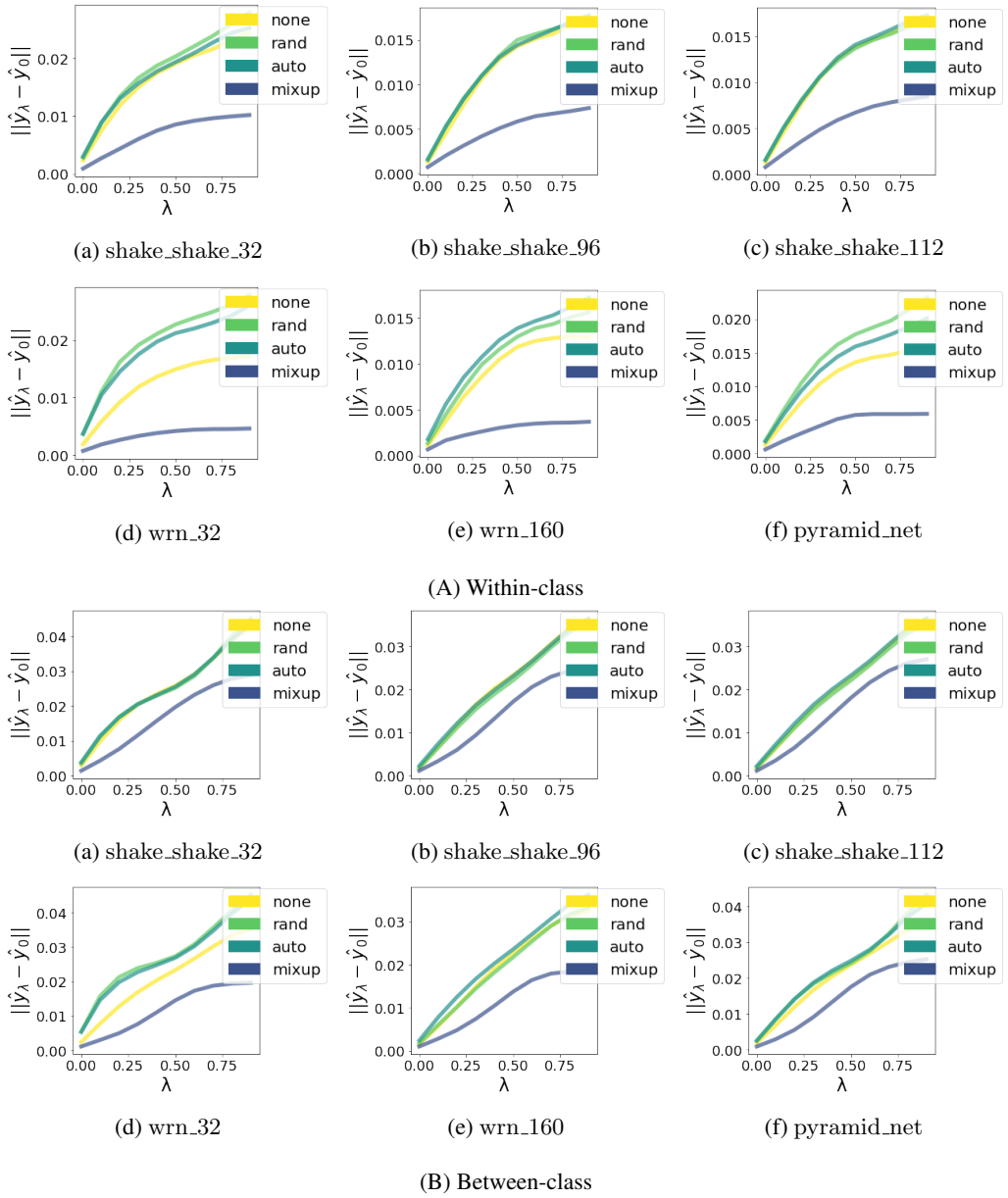
(d) wrn_32 (e) wrn_160 (f) pyramid_net

(B) Between-class

Figure 13: **Increasing the number of training examples reduces the frequency of the learned function.**

### A.5 Data Augmentation

Figure 6 *(Center)* uses linear interpolation to study the effect of common data augmentation procedures on the learned frequencies for wrn_160. Figure 14 shows the same experiment on all six models we tested.




(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(A) Within-class

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

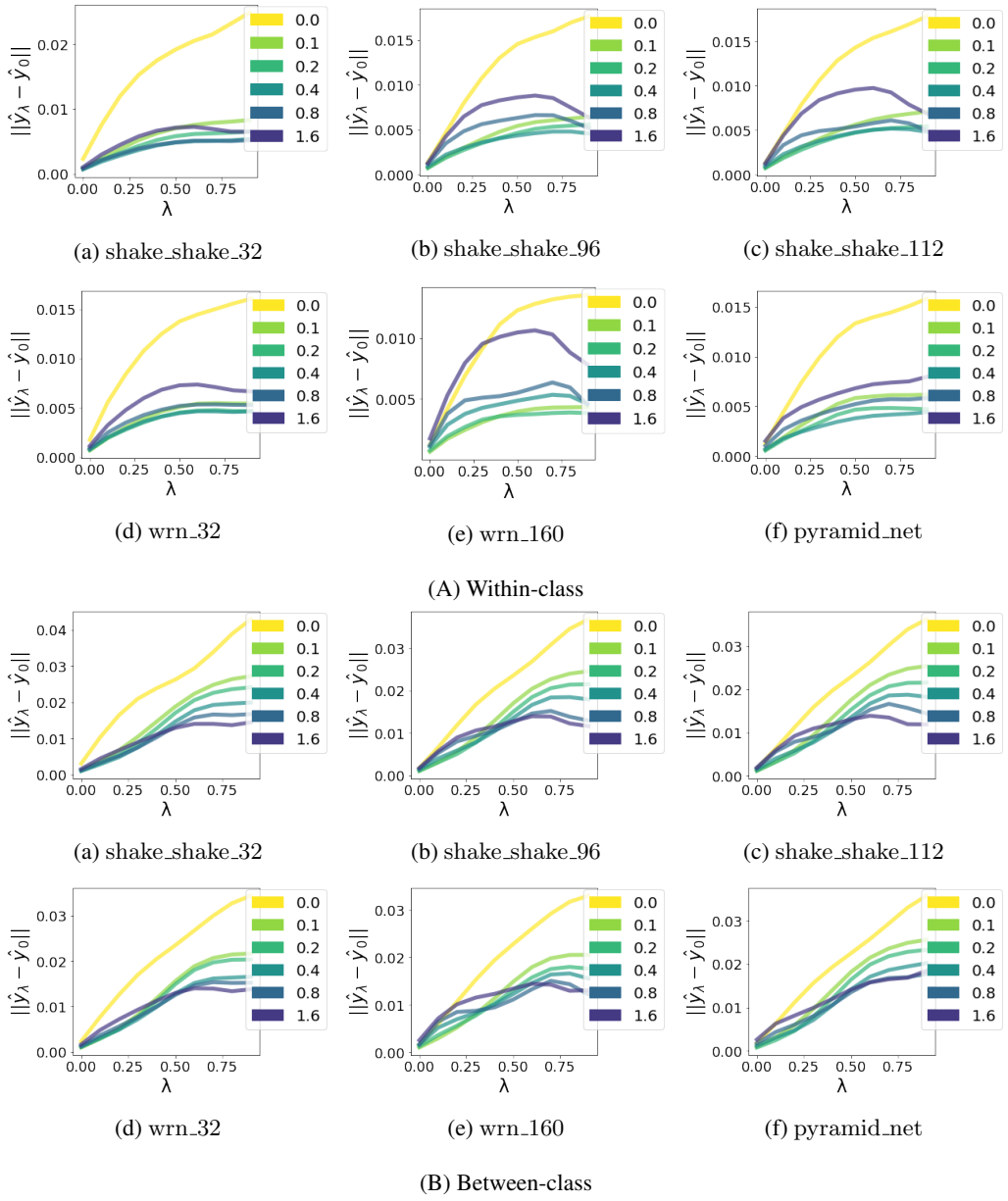
(d) wrn_32 (e) wrn_160 (f) pyramid_net

(B) Between-class

Figure 14: **Mixup augmentation produces a lower-frequency learned function on all models tested. On some models, RandAugment and AutoAugment produce a learned function with higher frequencies between examples from the same class.**

### A.5.1 MIXUP STRENGTH

Figure 6 *(Right)* uses linear interpolation to show that training with stronger Mixup data augmentation (Zhang et al., 2018) causes wrn_160 to learn a lower-frequency function. Figure 15 repeats the experiment on all six models we tested. For some models sufficiently strong Mixup augmentation produces a higher-frequency learned function between examples from the same class. The reason for this is an interesting direction for future work.




(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(A) Within-class

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

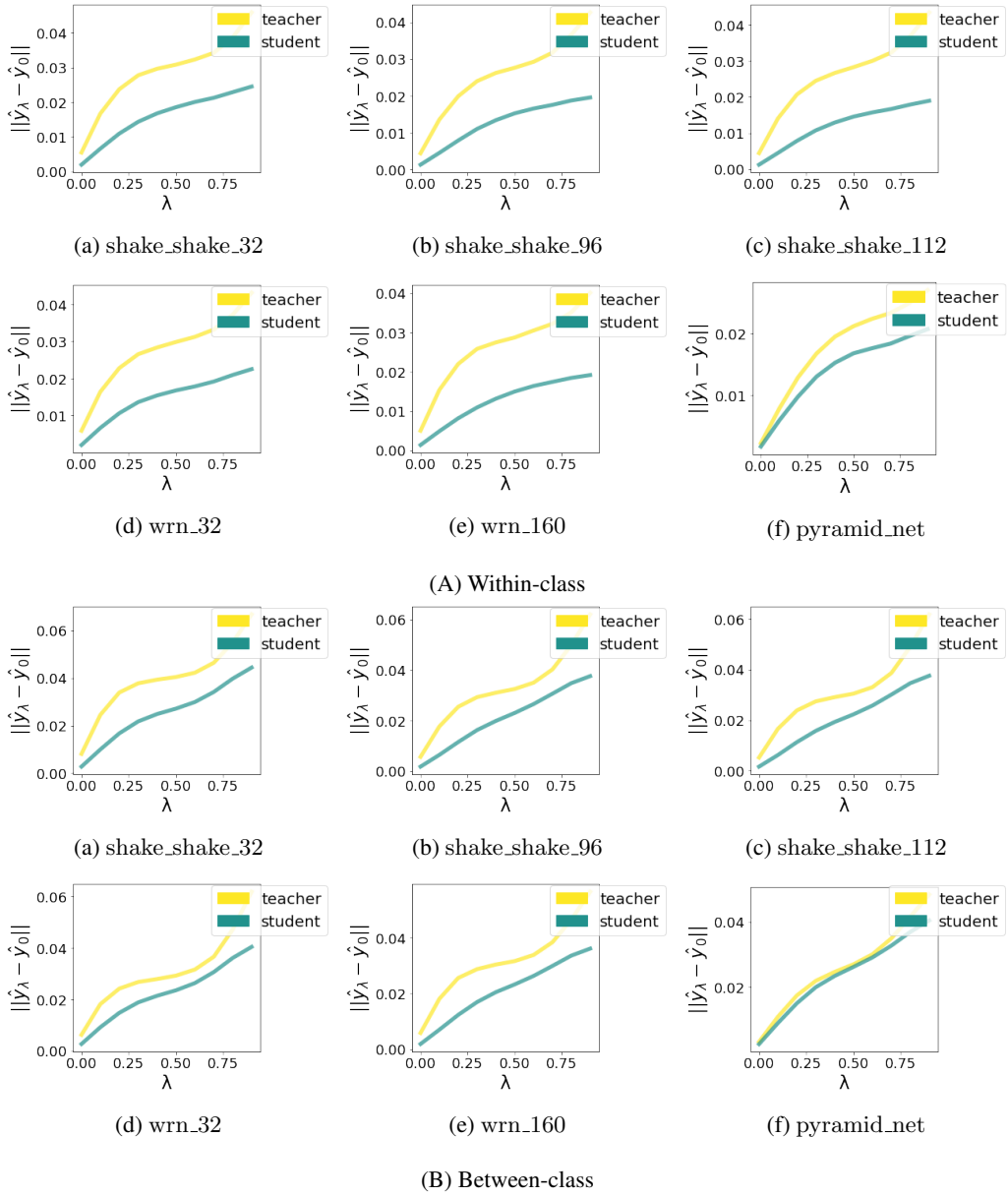
(d) wrn_32 (e) wrn_160 (f) pyramid_net

(B) Between-class

Figure 15: **Training with stronger Mixup augmentation produces a lower-frequency learned function, particularly between examples from different classes. Within-class, Mixup that is too strong can induce higher frequencies.**

### A.6 SELF-DISTILLATION

Figure 7 uses linear interpolation to show that self-distillation with shake_shake_96 produces a student model that is lower-frequency than its teacher, even though both are trained to the same training loss and the student has higher validation accuracy than the teacher. Figure 16 shows the same result across all six models we tested.




(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(A) Within-class

(a) shake_shake_32 (b) shake_shake_96 (c) shake_shake_112

(d) wrn_32 (e) wrn_160 (f) pyramid_net

(B) Between-class

Figure 16: **Self-distillation produces a lower-frequency student compared to its teacher.**