CNNs are Myopic

Vamshi C. Madala  
University of California, Santa Barbara  
vamshichowdary@ucsb.edu

Shivkumar Chandrasekaran  
University of California, Santa Barbara  
shiv@ece.ucsb.edu

Abstract

We claim that Convolutional Neural Networks (CNNs) learn to classify images using only small seemingly unrecognizable tiles. We show experimentally that CNNs trained only using such tiles can match or even surpass the performance of CNNs trained on full images. Conversely, CNNs trained on full images show similar predictions on small tiles. We also propose the first a priori theoretical model for convolutional data sets that seems to explain this behavior. This gives additional support to the long standing suspicion that CNNs do not need to understand the global structure of images to achieve state-of-the-art accuracies. Surprisingly it also suggests that over-fitting is not needed either.

1 Introduction

One of the most significant obstacles to the deployment of Convolutional Neural Networks (CNNs) in safety critical applications is the lack of a prior error models that explain their success. Many theoretical papers have been written about how deep neural networks (DNNs) can overcome the curse of dimensionality (CoD) in different contexts, but there is no experimental evidence, as far as we know, to confirm their applicability in practice. We propose for the first time a simple theoretical model that seems to predict correctly the performance of CNNs on well-known image classification data sets.

The model predicts that the CoD is overcome by the simple mechanism of learning the noisy labels of all the small tiles in the image and averaging the tile-wise predictions. This simple model is easy to analyze and yields testable bounds. As a consequence we claim that the main reasons for being cautious in deploying CNNs can now be overcome.

So our main claim is that CNNs learn to classify images only using small tiles. In fact we observe experimentally that CNNs do not require full images to be used during training, but only randomly cropped tiles as inputs, to match (or even surpass) the performance of models trained using full images. Conversely, CNNs trained on full images have similar predictions on small tiles. The proposed simple theoretical model can explain this behavior surprisingly well. For example, on the CIFAR-10 data set, without any data augmentations or pre-trained weights, a ResNet18 model trained only on 16x16 tiles reaches ~91% accuracy on the test set compared to ~89% using the full 32x32 images. We also use this observation to generate heat maps for any image to explain a given CNN model’s (mis)classification of the image. We also discuss how this observation sheds light on the amazing generalizing performance of CNNs while providing a natural regularizing strategy for training CNNs and saving computational costs.

2 Related work

Even though there is much published literature explaining the expressive power of deep neural networks in general, to the best of our knowledge, we could not find any prior work that attempts to
explain the amazing generalization capabilities of convolutional neural networks in particular, even though the latter are much more successful in practice.

Poggio et al. [11] seems to be the closest to this paper theoretically. They show that deep CNNs can avoid the curse of dimensionality for hierarchically compositional spatially local functions. CNNs definitely fall into this category, but the convolutional nature of the data was not fully exploited in their analysis. In particular knowledge of the right compositional model is needed for their results to be applicable.

Very early work on unsupervised learning by Coates et al. [3] shows that “convolutionally extracted” features achieve good performance on unsupervised tasks because of the similarity between the extracted tiles/features making them easier to cluster using k-means. This correlates well with our theory that the predicted labels from convolutionally extracted features on small tiles gives the CNNs their advantage. However no theoretical models were proposed and this work was before CNNs were introduced.

Sokolic et al. [15] discuss the generalization error of invariant classifiers like CNNs under explicit input transformations and obtain an error bound, but they do not extend the discussion to tiles and how the invariance when applied to the tiles of input images affects the generalization performance of CNNs.

A very closely related paper is Brendel and Bethge [2] which proposes the BagNet architecture. In BagNet a modified ResNet backbone is applied to each \( q \times q \) tile of the image and the output is a class heat map. The average of these heat maps is then fed into a linear classifier. The heat maps highlight the tiles which contribute to the network’s decision in a straight-forward way because of the linear classifier, improving explainability but at the cost of some accuracy due of the modified architecture. The first main difference with our work is that, during training they consider the output of the linear classifier i.e. the weighted aggregate of the individual decisions on \( q \times q \) tiles to be the network’s prediction for the image and thus it is not directly evident what the network is considering as the labels for the individual tiles. On the other hand, we train the standard CNNs to classify each tile as belonging to its parent class making it very clear that our heat maps are the network’s predictions on individual tiles. We also show using the evidence of heat maps that the CNNs trained using standard training procedures also learn the labels on tiles implicitly, thus providing explainable decisions to traditional CNN architectures without any loss of accuracy. Finally, we provide a theoretical model which explains why such a seemingly handicapped strategy can be a critical element for the success of CNNs. The theory also provides a numerical estimate for the generalization performance which correlates well with the observed test accuracies on popular data sets.

3 Experiments

3.1 Experimental setup

To only focus on the ability of CNNs in learning the labels of tiles, we fix the training hyper-parameters to be same across all training runs and use the following setup. In all our results, the accuracies are on average within \( \pm 1-2\% \) of the reported numbers. We also do not try to match/obtain state-of-the-art accuracy on any data set or model as it is not our intended goal, and hence do not use any data augmentations or strategies like CutMix [17] or Mixup [18].

CNN Models. ResNet9, ResNet18 and ResNet34 without any pre-trained weights. We choose the ResNet [5] family because of its ubiquity in practice and smaller number of parameters. To show that our results are robust, we provide similar experimental results on training with tiles using other CNN architectures in Appendix A.5.

Data sets. We use CIFAR-10 [6], CIFAR-100 [6], STL-10 [4], Tiny-Imagenet-200 [7] and Imagenet-1k [12] data sets, covering a wide range of variability in number of training samples, number of classes and image resolutions. CIFAR-10 has 10 classes with 5,000 \( 32 \times 32 \) resolution images per class, whereas CIFAR-100 has the same number of total images but among 100 classes, constraining the training set to only 500 \( 32 \times 32 \) images per class. STL-10 has 10 classes with 500 training and 800 test images per class which are labelled but these images are of higher resolution (96x96) than CIFAR. Tiny-Imagenet-200 has 200 classes with 100,000 labelled training images of \( 64 \times 64 \) resolution. Imagenet-1k has 1.2 million images with varying resolutions from 1,000 classes.
Figure 1: (Top) With each tile as an input to a Resnet9, all the tiles classified as truck are shown as a heat map with the model’s output for that class represented as the color of the heat map. (a) Sample image of truck is from CIFAR-10 test set. Some 8x8 tiles are shown in (b) and 12x12 tiles in (e). Resnet9 is trained separately with 8x8 and 12x12 tiles of CIFAR-10 respectively. (Bottom) Example heat maps generated using 64x64 tiles by a mobilenet_v2 pre-trained on Imagenet-1k 224x224 full images and without any additional training with tiles.

Computational resources We used a dual 40-core Xeon CPU with 192G RAM and an NVIDIA TITAN RTX 24G GPU and a 24-core Xeon CPU with 64G RAM and an NVIDIA 1080Ti 11G GPU in all our experiments.

Training hyper-parameters. We train all the models, except those on Imagenet-1k, for 400 epochs using the standard cross-entropy loss and Adam optimizer with a weight decay of 1e-4, a starting learning rate of 0.002, and a multi-step learning rate decay with gamma of 0.5 at milestones at 40, 80, 100, 150, 200, 250 and 300, epochs. We use batch sizes of 128 for CIFAR10 and STL10, and 512 for CIFAR100 and Tiny-Imagenet-200. For Imagenet-1k, due to resource constraints, we follow the training routine suggested by FFCV [8] for faster training. There we use ResNet18 for 100 epochs with a batch size of 1024, cross-entropy loss, SGD optimizer with momentum of 0.9, weight decay of 5e-5, cyclic learning rate with starting value 0.5. Test time resolution is fixed at 256 for inference on Imagenet-1k.

Image pre-processing. We only use random horizontal flips for augmentation and zero mean and zero standard deviation normalization as part of the image pre-processing pipeline and no other transformation or augmentation is used.

Generation of tiles. We produce tiles by random cropping of original image without any padding at train time (for example, by using RandomCrop transform of PyTorch [10] in the DataLoader with desired tile size and padding = 0 as the arguments). Here we take the opportunity to note that our setting of training a CNN model with just the tiles is completely different from common practice of deep learning practitioners using random cropping in their pre-processing pipeline. In the common practice, random cropping is explicitly used as a regularization trick while taking care in selecting padding size and crop size such that the cropped image would still be recognized by a human as belonging to its label class. But in our experiments, we deliberately train with a full range of crop sizes without padding and in the case of very small sizes, the tiles could not be attributed to any class by a human, for example, as shown in Figure[1] We also clarify here that in regards to the labels of tiles, we use the same label as the parent image from which a tile is drawn, even for small tiles that could not be attributed to be of the parent image’s class label. In other words, no label transformation is applied when training with tiles.
Table 1: Training and test accuracies with ResNet models for different tile sizes on different image classification data sets. The data sets differ in number of training samples (N), classes (K) and image resolution (HxW). “Final Train” corresponds to the training set accuracy observed at the end of 400 epochs; Train and Test correspond to (tiled) training set and (regular) test set accuracies observed when the test accuracy is highest. CIFAR-10 and CIFAR-100 are trained using ResNet9, STL-10 and Imagenet-1k using ResNet18 and Tiny-Imagenet-200 using ResNet34.

### 3.2 Training with only tiles

Table 1 shows the training and test set accuracies of ResNet models when trained on tiles of different sizes and tested on test set images of full resolution of the corresponding datasets. We can see that a model trained with just 16x16 tiles on CIFAR-10 has a ~91% test accuracy whereas the model trained on full 32x32 images has ~90%. Highest test accuracy is observed for model trained with 24x24 tiles. Remarkably, model trained with only 8x8 tiles shows ~75% test accuracy.

Figure 1 gives a perspective on how 8x8 tiles may appear to humans taken from a sample CIFAR-10 testset image, so it is extremely surprising that a model trained only on such tiny tiles with very noisy labels, is exhibiting such a high accuracy. Here we remind the reader again that the labels provided for each of these randomly selected tiles is just the label of its parent image, irrespective of whether the tile covers the class subject in it or not. So it is highly likely that during training a lot of inputs are just tiles of the background but paired with labels of the parent image. And the CNNs seem like they do not have any trouble distinguishing between such noisy label tiles and tiles with correct labels (i.e. tiles which cover the class subject), which is evident from (i) the absence of over fitting observed when training with tiles and (ii) observing the heat maps of training set images after the CNNs are trained. We discuss the absence of over fitting aspect in Section 3.3 and the heat maps in Section 5.

Similar results are observed for CNNs trained on STL-10 and Tiny-Imagenet-200, each of which has different image resolutions, classes and number of training images. For example, on STL-10, the model trained on 32x32 tiles outperforms the one trained on 96x96 tiles by ~7%. However for CIFAR-100 the highest test accuracy is observed on full sized images, even though the ~61% accuracy of the model trained on 16x16 tiles is comparable to the ~68% for the one trained on full images, especially considering that CIFAR-100 has 100 classes of just 32x32 resolution images with only 500 samples per class available for training. And similarly on Imagenet-1k, ResNet18 trained on 168x168 tiles match the performance of models trained on 256x256 and 224x224 images and remarkably, a ResNet18 trained on 112x112 tiles, which is half the usual input size that Imagenet-1k models are generally trained at, almost matches the performance of models trained on larger images. Given this seemingly wide range of behaviour we show in Section 4 that a simple theoretical model can predict all of this behaviour of CNNs over what we call convolutional data sets.

### 3.3 Implicit regularization when training with tiles

When training with smaller tiles, we observe that the CNNs do not over fit on the input tiles during training and the training accuracy saturates at much less than 100%, as measured on the training tiles. This can be seen from the “Final Train” column of Table 1 for each data set. Figure 2 also shows that the models trained on smaller tiles are better regularized than the model with full images as inputs during training. Moreover, the accuracy of each model on full sized training set images, as reported in Table 2 shows that the models trained on smaller tiles have better generalization gap (difference between training error and test error). In fact, the generalization gap continuously decreases as the tile size is decreased, even showing negative values for models trained on 8x8 tiles of CIFAR-10, and 8x8 and 16x16 tiles of STL-10; i.e., a better performance on the test set than on the train set!
Figure 2: Training loss and Test accuracy measured during training with tiles of different sizes on CIFAR-10 and STL-10. Models trained on smaller tiles show implicit regularization.

| Tile Size | CIFAR-10 | STL-10 |
|-----------|----------|--------|
| 32        | 99       | 90     |
| 9         | 96       | 79     |
| 7         | 71       | 14     |

Table 2: Accuracies on full sized images of training and test sets of CIFAR-10 and STL-10 for models trained on different sizes of tiles. Models trained on smaller tiles have smaller (even negative) generalization gaps.

| Tile Size | CIFAR-10 | STL-10 |
|-----------|----------|--------|
| 32        | 90       | 89     |
| 24        | 91       | 89     |
| 16        | 91       | 89     |
| 12        | 91       | 85     |
| 8         | 75       | 60     |

Table 3: Test accuracy on CIFAR-10 and STL-10 for ResNet9 and ResNet18 models showing the effect of depth of CNNs when trained on tiles of different sizes.

When we look closely at the model’s predictions on tiles of training set images using the heat maps (see Section 5), we find that during training, CNNs learn which tiles correspond to the distinguishing regions for a given class, while tending to discard tiles that do not have any information regarding its class label. This crucial observation, combined with another important observation that the models trained on full images produce identical looking heat maps as the ones trained on tiles, we conjecture that the over fitting that is apparent in models trained on full images is only the over fitting of individual tile decisions into a single decision for an image, which is explicitly controlled when training with tiles, thus making them implicitly regularized.

3.4 Computational advantages of training with tiles

A single full pass of one 224x224 image through a ResNet18 model uses 1.81 GFlops, whereas a 168x168 image uses 1.11 GFlops, and 112x112 image uses only 0.48 GFlops. On the memory side, a full pass of one batch of 1024 224x224 images through ResNet18 needs 23.5 GB of GPU memory, whereas a batch of 1024 168x168 images requires 14.6 GB, and 112x112 images require only 7.1 GB of GPU memory. So the advantages of training on tiles is also pretty clear from the computational point of view especially if there is no drop in generalization performance. Moreover, using small tile sizes may even enable one to switch to a smaller CNN with reduced depth without significantly dropping the accuracy, as was the case for CIFAR10 and STL10 (see Table 3).

4 Generalization error bound for convolutional data sets

We now turn our attention to explaining this observed behavior of CNNs using a simple theoretical model. We argue that CNNs are able to generalize extremely well to unseen tiles owing to the mitigation of the curse of dimensionality (CoD) which often plagues the function approximations in high dimensions. Given our experimental observations that CNNs actually learn the labels of tiles of images, rather than those of whole images, we can now see that this brings down the effect of CoD in two ways – i) by reducing input dimensions as number of pixels in tiles is much smaller than that of full image, and ii) by increasing the number of samples available during training because for each image of size \((H, H)\), there are \((H - T_m + 1)^2\) tiles that are of size \((T_m, T_m)\), which grows
Figure 3: Predicted generalization error from our theoretical upper bound compared with the observed test set errors for different tile sizes on multiple data sets. Predicted error at each tile size is only considering that single tile size in its error estimate, whereas CNNs trained on tiles of size $T_m$ have access to all the tiles $\leq T_m$.

We say a data set is convolutional if the label for the image can be transferred to its tiles at fairly small sizes in a meaningful way. The degree to which this transfer can be achieved will depend on the data set and the associated labels. For example if the labels are counting the number of objects of a particular class in an image (one cow, two cows, three sheep, etc.), then it is unlikely that such data sets are very convolutional.

We plot the right hand side of the error bound for each of the image classification data sets as a function of input tile size $T_m$ along with our reported test accuracies from Section 3.2 in Figure 3. In all these plots it should be noted that the generalization error predicted by the error bound assumes CNN inputs as only tiles of single size, given by the corresponding x-axis value. CNN models in practice, however, have access to tiles of all sizes that are $\leq T_m$ when fed with the input tiles of size $T_m$. This is because of how the convolutional layers are constructed with 3x3 kernels of deeper layers stacked on top of those of shallow layers with each layer’s weights operating on tiles of sizes that correspond to the receptive-field of that layer. Due to this, a model trained with larger tiles, also enjoys the advantages of smaller tile training described earlier and shows a test error that is lower than that is predicted by the error bound. So in effect, we should actually be comparing the observed test errors with minimum of the right hand side of inequality across all tiles sizes smaller than $T_m$. However, we believe just plotting the per tile size upper bound is more meaningful as the “optimal” tile size is more readily apparent this way. So beware that the (near) monotonicity of the actual error curve versus the cup-shape of the theoretical curve is due to this choice and not a sign of a flaw in the model.

In all the plots, the error increases exponentially at tile sizes near zero, since the labels at that scale are completely random. And when the inputs are of full resolution i.e. towards far right of the plots,
The error increases again due to CoD. The true generalization power of CNNs is realized in those intermediate tile sizes as we have also seen from the results of Section 3.2. The data sets used in the plots span a wide range of parameters present in the error bound, thereby revealing the interplay between them for different combinations, while at the same time being familiar to most deep learning practitioners.

The first plot shows the error curves for CIFAR-10 and CIFAR-100 data sets (32x32 images), where we have chosen $S$ to be 1 and 2 respectively. Between these two data sets, the error model correctly estimates the difficulty in generalization when the number of classes is increased or the number of training samples available per class is decreased for a given resolution. STL-10 and Tiny-Imagenet-200, which have larger resolutions than CIFAR but smaller number of training samples per class, also follow the curve predicted by the bound very closely. We chose $S = 5$ for STL-10 and $S = 3$ for Tiny-Imagenet-200 which have input resolutions of 96 and 64 respectively. Finally, for Imagenet-1k which has 1.2M images of varying resolutions with 1000 classes, our error model predicts the generalization error that is remarkably close to the observed test error. For Imagenet-1k we chose $S = 4$. In the same plot we also plot our model’s predicted generalization error if the input images were of 512x512 resolution instead of 256x256. We did not have the compute resources to check the test error in that setting, but we welcome other experimentalists to verify our prediction/conjecture. In summary, our error model does an excellent job in predicting the generalization error of CNN models on various standard image classification data sets, and shows that simple classical approximation theory ideas (suitably extended) might be useful in understanding CNNs.

5 Heat maps

The process of generating heat maps is very straightforward. We send every $T \times T$ patch of the input image with a stride of 1 and appropriate padding, as input to a trained model and note the predicted class and model’s output value for that predicted class. To get the heatmap for a given class, we then superimpose the output values on the original image such that each pixel of the original image is nearly at the center of the corresponding tile patch. Padding is done with zeros on image edges to generate the same number of tiles as the number of pixels. Thus we can generate heat maps on images of any resolution using tiles of any size (with appropriate padding) and compare their quality across different models. To avoid confusion, we denote the tile size that is used to generate the heat maps by $T_g$, and the tile size that is used to train a model by $T_m$ i.e. the random crop size applied during the training preprocessing to generate training tiles.

In our experiments, we compare the heat maps generated using different tile sizes ($T_g$) by models trained on different tile sizes ($T_m$) and also models which have been trained normally using com-
mon training practices i.e. pretrained and available to download via packages like PyTorch [10] and TensorFlow [11] without any additional training. Figure 4 shows the heat maps of ResNet18 models that were trained on tiles of sizes 16x16, 56x56 and on full images of 96x96 of STL-10 dataset, generated using tiles of sizes 16, 24, 32, 48, 56 and 64 on a sample image. (Figure 10 in Appendix A.3 shows the heatmap on a randomly downloaded image generated with 64x64 tiles using a mobilenet_v2 pretrained on Imagenet-1k model downloaded using PyTorch’s model zoo and without any additional retraining.) These heat maps show that CNNs learn to classify on small tiles whether they were explicitly trained on tiles or not, leading us to theorize that CNNs are only learning the labels of tiles and in general do not care about the global structure of the image, aided by the mitigation of curse of dimensionality, as discussed in Section 4. For a more comprehensive comparison of heat maps of Figure 4 see the Appendix.

5.1 Explaining model predictions

Heat maps also help us understand the predictions of CNNs. From our discussion so far, if all that CNNs do is just predict labels on tiles, we should expect that similar tiles have similar labels across all images and we indeed observe the same. Figure 5 shows heat maps for two such examples taken from STL-10 data set. For the plane class, tiles from the sky background form the activating tiles instead of the planes themselves. This makes sense for obvious reasons that in most of the plane images, sky is the common attribute whereas the orientations, colors and sizes of planes vary. Similarly, for the car class, wheels form the primary attribute and are accordingly highlighted in the heat maps.

This characteristic of CNNs to only focus on tiles can backfire if foreground tiles of one class are present as background tiles in an image belonging to another class. Figure 6 shows exactly that for sample bird images from STL-10. Here instead of a heat map, we color code each 16x16 tile based on its predicted class and assign it to the corresponding center pixel in the full image. For the first two samples, even though the bird tiles are correctly classified, the overall prediction on the image is negatively influenced by the background sky tiles which are classified as plane. Conversely, in the third sample, the green background tiles are classified as bird but the tiles belonging to bird are falsely classified as cat, ultimately influencing its final prediction on the whole image.

These heat maps can thus help to understand CNN models before deploying them to real world applications. If the heat maps generated by the model on sample images do not conform to human visual understanding of the image, the model is probably not going to be reliable in practice. This can be especially useful in medical applications where now the decisions by CNNs can be cross-checked by trained medical professionals. The heat maps can also suggest remedies for appropriate new training images with appropriate combinations of background and foreground tiles.
5.2 Effect of image resolution

In Figure 7 we show the heat maps generated by models that were trained with different resolution images from the same dataset, STL-10. For this we downsampled all the images in the dataset from 96x96 to 32x32 and trained ResNet9 models with tiles of sizes 8x8, 16x16 and 32x32 (regular training with full images) generated from these downsampled images. We compare these models with another set of ResNet9 models that are trained (using different tile sizes) on original 96x96 images, by looking at their corresponding full image test accuracies, at respective resolutions and at each of the tile sizes and also using the predicted generalization error for such a change in resolution. As our model predicts, the test error just shifts towards smaller tile sizes without much change in the test performance. This is also supported by the heat maps in Figure 7 which show that models trained on lower resolution and using tiles of smaller sizes, produces identical heat maps irrespective of the input resolution. For more examples of heat maps between images of different resolutions, refer the Appendix.

6 Conclusion

We have shown experimental evidence for our claim that CNNs only learn labels of tiles when applied to image classification problems, and provided a simple theoretical model that can explain their generalization performance by mitigating the curse of dimensionality and obtained an upper bound for the generalization error. We also showed that this estimated generalization error closely matches the observed generalization performance of CNNs on various standard image classification data sets. We also provided a diagnostic tool in the form of heat maps, which is a natural application of our theory, that can be of practical use to deep learning practitioners in analysing their models and data sets. Moreover, as discussed in Section 5.1, our theory can help in designing better training sets, verifying models for deployment, as well as in new architecture design. Our theory also provides new insights into various unexplained behaviors of deep neural networks such as unsupervised learning, transfer learning, adversarial robustness and object detection. These and other matters will be presented elsewhere.
Acknowledgments and Disclosure of Funding

We would like to thank Lucas Beyer (Twitter user @giffmana), Andrew Brown, Jason Bunk, Aidan Chandrasekaran, Hirish Chandrasekaran, Jason Dunne, B.S. Manjunath, Hrushikesh Mhaskar and Abhejit Rajagopal, for useful discussions and valuable feedback.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[2] W. Brendel and M. Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. International Conference on Learning Representations, 2019. URL https://openreview.net/pdf?id=SkfMWhAqYQ.

[3] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In G. Gordon, D. Dunson, and M. Dudík, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15 of Proceedings of Machine Learning Research, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/coates11a.html.

[4] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In G. Gordon, D. Dunson, and M. Dudík, editors, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15 of Proceedings of Machine Learning Research, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/coates11a.html.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[6] A. Krizhevsky. Learning multiple layers of features from tiny images. University of Toronto, 05 2012.

[7] Y. Le and X. S. Yang. Tiny imagenet visual recognition challenge, 2015.

[8] G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry. ffcv. https://github.com/libffcv/ffcv/, 2022. commit e97289fdacb4b049de8dfefeb250cc35abb6550.

[9] H. N. Mhaskar and T. Poggio. Function approximation by deep networks. Communications on Pure and Applied Analysis, 19(8):4085–4095, 2020. URL https://arxiv.org/abs/1905.12882.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[11] T. A. Poggio, H. N. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep - but not shallow - networks avoid the curse of dimensionality: a review. CoRR, abs/1611.00740, 2016. URL http://arxiv.org/abs/1611.00740.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[13] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision, 128(2):336–359, oct 2019. doi: 10.1007/s11263-019-01228-7. URL https://doi.org/10.1007%2Fs11263-019-01228-7.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL https://arxiv.org/abs/1409.1556.
[15] J. Sokolic, R. Giryes, G. Sapiro, and M. Rodrigues. Generalization Error of Invariant Classifiers. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1094–1103. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/sokolic17a.html.

[16] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2019. URL https://arxiv.org/abs/1905.11946.

[17] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019. URL https://arxiv.org/abs/1905.04899.

[18] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization, 2017. URL https://arxiv.org/abs/1710.09412.
We introduce two new parameters $S$ and $\alpha$ without shrinking this upper bound; $S = (S_H, S_W)$ controls the effective number of total tiles in the data set and acts similar to the *stride* parameter.
that is popular in convolutional kernels and this parameter is there to account for the fact that the nearby tiles are correlated and are unlikely to contribute to an improvement in generalization error. So the total effective number of samples becomes $N_{T_{\text{eff}}} = N \left( \frac{H_T W_T}{S_H} + 1 \right) \left( \frac{W_T}{S_W} + 1 \right)$, where we denote the number of uncorrelated $H_T \times W_T$ tiles in $z$ with $T_{\text{eff}}$. We let $\alpha$ control the effective dimension of the tiles as it is likely that the different channels in the image are also correlated, and in our experiments we use $\alpha = 3$. So the mesh norm is re-parameterized as:

$$
\mu(z_T) \leq c_3 \left( \frac{1}{N_{T_{\text{eff}}}} \right)^{\frac{1}{\alpha}}. 
$$

We assume that the new approximation $F_T$ trained on tiles is rougher than $F$ and consequently assume that the gradient norm in the inequality (2) depends on the tile size and dimensions via a power law and that it is much larger than the gradient norm of the true function $f_T$, whose properties are unknown, i.e.

$$
\|f_T^\prime\| + \|F_T^\prime\| \approx \|F_T^\prime\| \lesssim m_1^\prime \left( \left( \frac{H_T W_T}{H W} \right) \right)^{\frac{1}{\alpha}},
$$

where $m_1^\prime$ is a monotonically decreasing function. In our results we use $m_1^\prime(\theta) = 1/\theta$. So at full image size the upper bound on the derivative is an known constant, but at smaller tiles sizes this norm grows. In higher dimensions this bound grows slower because there is more room to accommodate the roughness in the classification function.

We are using the labels of a training image as the label for its tiles no matter how small the tile is or where it is drawn from. So these labels are intrinsically noisy. We therefore model the training error on the tile data set $\varepsilon_{N_T}$, to be bounded by a quantity that is inversely proportional to the area of the tile. So for some constant $c_4$:

$$
\varepsilon_{N_T} \leq c_4 m_2^\prime(K) m_3^\prime \left( \frac{H_T W_T}{H W} \right),
$$

where $m_2^\prime$ is a monotonically increasing and $m_3^\prime$ a monotonically decreasing function. In our results we use $m_2^\prime(K) = \sqrt{K}$ and $m_3^\prime(\theta) = -\log(\theta)$, arguing that for $H_T W_T = H W$ i.e. when full images are used, labels are noiseless. It should be noted here that our choices for $m_1^\prime$, $m_2^\prime$ and $m_3^\prime$ are very preliminary but our experimental results show that these simple approximations are able to satisfactorily model the generalization error across different image classification data sets and CNN architectures. We also make a note that in our approximation of training error, we do not take into account the effect of number of parameters in the CNN. In fact, as $N \to \infty$ and if so does the number of parameters in CNN at a suitable rate, one would expect the training error to go to zero, which is not reflected by the equation (9). In future work, we plan to further refine these approximations.

Finally the new bound on the generalization error for the model $F_T$ trained on the dataset $t_{N_T}$ and measured on samples $z_T$ drawn randomly from the tiled subspace $X_T \times Y_T$ where $f_T(z_T) \in Y_T$ can be rewritten as:

$$
\|f_T(z_T) - F_T(z_T)\| \leq c_5 \mu(z_T) \|F_T^\prime\| + \varepsilon_{N_T}
$$

$$
\leq c_5 \mu(z_T) m_1^\prime \left( \left( \frac{H_T W_T}{H W} \right) \right)^{\frac{1}{\alpha}} + c_4 m_2^\prime(K) m_3^\prime \left( \frac{H_T W_T}{H W} \right). 
$$

Inequality (10) estimates the upper bound on the generalization error for a model trained on tiles and tested on tiles. However, what we are actually interested in is the generalization performance of the model on unseen full sized images. We make the simple assumption that:

$$
f(z) \approx \frac{1}{\text{(number of tiles in } z)} \sum_{z_T \in z} f_T(z_T) 
$$

and accordingly,

$$
F(z) \approx \frac{1}{\text{(number of tiles in } z)} \sum_{z_T \in z} F_T(z_T) 
$$
We now assume that there are $T_{eff}$ number of uncorrelated $H_T \times W_T$ tiles in $z$ on average. Then one can expect, that on average the expected upper bound on the generalization error would be:

$$\|f(z) - F(z)\| \lesssim \frac{1}{\sqrt{T_{eff}}} \max_{z_T \in z} \|f_T(z_T) - F_T(z_T)\|$$

(13)

where $T_{eff}$ is the effective number of tiles in an image. And now from (2), (7) and (10) we finally get

$$\|f(z) - F(z)\| \lesssim \frac{1}{\sqrt{T_{eff}}} \left( c_6 \left( \frac{1}{NT_{eff}} \right) \frac{1}{m_1^T} m_2^T \left( \frac{H_T W_T}{HW} \right) \right)^{\frac{1}{2}} + c_4 m_2^T (K m_3^T \left( \frac{H_T W_T}{HW} \right))^{\frac{1}{2}}$$

(14)

Since the CNNs we use in our experiments typically use all tiles sizes from $3 \times 3$ onwards up to the maximum size $H_T \times W_T$, we assume that the minimum of the right-hand side of inequality (14) over the considered tile sizes should be taken as the generalization error for CNNs. In our experiments we use $c_6 = 1$ and $c_4 = 0.5$.

It is very surprising that this crude model does a good job of predicting the performance of CNN models on many standard image data sets. A more detailed analysis will be presented elsewhere.

### A.2 Heat maps

Figure 8 shows a more detailed version of Figure 4 comparing the heat maps generated using tiles of different sizes $(T_g)$ of ResNet18 models trained using different tile sizes $(T_m)$.

In Figure 9, we compare the heat maps of first 10 samples of STL-10 data set belonging to different classes and generated from models trained on different resolutions, as an extended version Figure 7 described in Section 5.2.

### A.3 Comparison of heat maps with GradCAM

We compare our heat maps with GradCAM [13] which is a popular tool to analyse CNN’s predictions on images in Figure 10. In order to compute GradCAM, gradients are calculated at an intermediate layer and the activated pixels back-projected to the input image, thereby losing some resolution. Our approach of generating heat maps do not require any computation of gradients, and heat maps of any desired resolution can be obtained just by changing the tile size appropriately. In Figure 10 both GradCAM’s as well as our heat map is generated using the same model mobilenet_v2 that was pre-trained on Imagenet-1k and downloaded using PyTorch’s model zoo. We used 64x64 tiles to generate the heat map.

### A.4 Code for the experiments

#### A.4.1 Training the models with tiles

For all the experiments except for Imagenet-1k, we used the following template for training and testing CNN models using PyTorch’s torchvision package with image preprocessing to generate tiles $T_m$ during training by changing the variable tile_size. All the rest of training procedure is standard that can be found in any beginner’s tutorial of training a CNN classifier on image data sets, for example see PyTorch’s tutorial, with the training hyper-parameters described in Section 3.1.

```python
from torchvision import datasets, transforms
train_tile_transform = transforms.Compose([transforms.RandomCrop(tile_size, padding=0), transforms.RandomHorizontalFlip(), transforms.ToTensor(), transforms.Normalize(dataset_mean, dataset_std)])
trainset = datasets.ImageFolder(dataset_train_folder, transform=train_tile_transform())

test_transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize(dataset_mean, dataset_std)])
testset = datasets.ImageFolder(dataset_test_folder, transform=test_transform())
```

For Imagenet-1k, we used FFCV [8] to train the model faster and the training procedure is described in their webpage.
Figure 8: Heat maps for a sample dog image from STL-10 with ResNet18 models that were trained on different tile sizes $T_m$ on y-axis, generated using tiles of sizes $T_g$ on x-axis.
Figure 9: Comparison of heat maps of models trained with tiles generated from different resolutions of STL-10 data set for first 10 images from 10 classes. Heat maps titled "label" correspond to the heat map of label class and those titled "predicted" correspond to the heat map of predicted class.

Figure 10: Comparison between GradCAM and our heat maps.
Table 4: Train and Test accuracies of different types of CNN architectures trained on tiles of different sizes.

| Tile Size | EfficientNet B0 | VGG-16 like | VGG-11 like | 3 Layer CNN |
|-----------|-----------------|-------------|-------------|------------|
| 32        | 99 80           | 100 90      | 100 88      | 99 85      |
| 24        | 88 73           | 99 92       | 99 90       | 96 85      |
| 18        | 73 63           | 69 73       | 89 86       | 86 83      |
| 16        | 66 55           | 78 84       | 82 83       | 81 83      |
| 12        | 52 42           | 62 49       | 64 73       | 66 80      |
| 8         | 37 25           | 44 28       | 44 40       | 48 72      |

Table 5: Architecture configurations of CNNs used in Table 4.

| VGG-16 like | VGG-11 like | 3-Layer CNN |
|-------------|-------------|-------------|
| Conv3-64    | Conv3-128   | Conv3-64    |
| maxpool     | maxpool     | maxpool     |
| Conv3-128   | Conv3-256   | Conv3-128   |
| maxpool     | maxpool     | maxpool     |
| Conv3-256   | Conv3-512   | Conv3-256   |
| maxpool     | Conv3-512   | Conv3-512   |
| Adaptive AvgPool | Adaptive AvgPool | PC-1024 |

A.5 Ablation study

We show the train and test accuracies when trained using tiles for CNN models other than ResNet type, on CIFAR-10 dataset, in Table 4. Like in Table 1, here too the “Train” accuracy corresponds to the tiled training set accuracy, whereas “Test” accuracy corresponds to the accuracy on the regular full sized test set images. For EfficientNet B0 [16] models, we used the architecture provided by TorchVision package without any modification. For VGG [14] like models, we modified the architecture as shown in Table 5 to be able use them on very small input sizes like 8x8 tiles: we removed two max-pool layers of kernel size 2 which would each shrink the feature's spatial size in half and removed the final set of convolutional layers that are usually sandwiched between those two max-pool layers from the original VGG architecture. We also replaced the average pooling layer by the adaptive average pooling layer that enables us to use inputs of different spatial sizes on the same model. And finally, we also trained models using a simple 3-layer architecture that consists of 3 convolutional layers and a single fully-connected layer separated by a max-pool and an adaptive average pooling layer.

For the simple 3-layer CNN models, we can see that training only on 12x12 tiles can have comparable generalization accuracy (~80%) as training on full 32x32 images of CIFAR-10 (~85%). VGG models also show a similar story. However for EfficientNet B0 and VGG-like models, we notice that deeper CNNs perform poorly when trained on very small tile sizes. This is also observed from Table 3 where ResNet9 which has smaller depth shows better performance than ResNet18 when trained on 8x8 tiles. Thus the test accuracies in Table 4 show that our results are not confined to a single type of CNN architecture and are applicable to all types of CNNs in general. However, specific choices of CNN architectures like depth and convolutional filter sizes can impact the generalization performance.