# Using effective dimension to analyze feature transformations in deep neural networks

**Kavya Ravichandran** [1] **Ajay Jain** [1] **Alexander Rakhlin** [1]

## Abstract

In a typical deep learning approach to a computer vision task, Convolutional Neural Networks (CNNs) are used to extract features at varying levels of abstraction from an image and compress a high dimensional input into a lower dimensional decision space through a series of transformations. In this paper, we investigate how a class of input images is compressed over the course of these transformations. In particular, we use singular value decomposition to analyze the relevant variations in feature space. These variations are formalized as the effective dimension of the embedding. We consider how the effective dimension varies across layers within class. We show that across datasets and architectures, the effective dimension of a class increases before decreasing further into the network, suggesting an initial whitening transformation. Further, the decrease rate of the effective dimension deeper in the network directly correlates with training performance of the model.

## 1. Introduction

### 1.1. Background

Deep neural networks (DNNs) are a powerful class of function approximators due to their ability to learn non-linear mappings from inputs to outputs. However, the intervening transformations and resulting features are not well-characterized, making it tough to understand the class of functions DNNs approximate.

Previous research has explored visualization techniques for understanding feature spaces (1) (2). While this line of work identifies abstractive properties of neural networks, it does not formalize how learned transformations simplify or *compress* the high-dimensional input to the low-dimensional output.

Notions of dimensionality and compression have been explored on the *parameter space* of neural networks. In Li et al. (3), fully connected network and convolutional neural network (CNN) weights are constrained to a low dimensional subspace of the full parameter space. The minimum parameter subspace dimension needed to solve a given task is termed the *intrinsic dimension* of the optimization landscape. This is a compression- or pruning-related result, though only explored for parameter spaces. Further, Antognini and Sohl-Dickstein (4) explore low dimensional visualizations of a parameter space over the course of training and a random walk.

Comparatively, exploration of the dimensionality of *feature spaces* on a per-layer basis has been limited. Recently, Dittmer et al. (5) proposed a singular-value and Gaussian width based interpretation of the action of ReLU layers, intended to distinguish data that is correctly and incorrectly classified in intermediate layers.

### 1.2. Motivation

We seek to understand how learned transformations in CNNs hone in on relevant features, as measured by how high-dimensional datasets are mapped into lower-dimensional distributions during the process of inference. A network trained for classification must map an input in high dimensional image space into low dimensional class label space. Characterizing this set of transformations would elucidate whether and when task-specific learned networks (a) identify latent attributes of the data distribution that are most relevant to the task and (b) remove uninformative attributes from deep feature spaces. Formalism surrounding these transformations could aid theoretical analysis of sample complexity.

### 1.3. Contributions

We propose a notion of **effective dimensionality** (Section 2.3) of a feature space and investigate how the effective dimensionality of a class of input images changes over the

[*]Equal contribution [1]Massachusetts Institute of Technology, Cambridge, MA. Correspondence to: Kavya Ravichandran <rkavya@mit.edu>.

*Understanding and Identifying Phenomena in Deep Learning Workshop at the $36^{th}$ International Conference on Machine Learning*, Long Beach, California, 2019. Copyright 2019 by the author(s).

course of linear and nonlinear transformations found in neural networks. In particular, we use singular value decomposition (SVD) to analyze relevant intra- and inter-class variances.

We show that effective dimension of a class increases before decreasing across neural network layers, ending in increasingly eccentric feature spaces that allow for sharp decision boundaries. We analyze how the singular value spectra of activation matrices composed of a single class changes throughout a network experimentally.

## 2. Methods

In this section, we discuss the datasets and architectures we study. We also introduce the notion of the **effective dimension** of a class at a given layer.

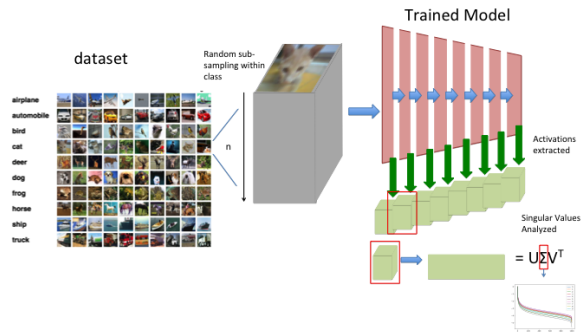


*Figure 1.* Experimental procedure. On the CIFAR-10 dataset, a subset of training images from a single class is used for inference. Our principal results subsample 1000 images from a single class. Activations are saved, then flattened into a matrix per layer $\Phi^{(l)} = U\Sigma V^T$ to compute a singular value spectrum per layer.

### 2.1. Datasets

For experiments, models are evaluated on the CIFAR-10 dataset training set (6) and Tiny ImageNet (7). CIFAR-10 consists of 10 classes of objects and animals, with 10,000 training images of size $32 \times 32$ pixels per class. To accelerate inference and SVD computation, we subsample 1,000 images from each class, using the same subsample for all experiments. Images are centered and rescaled to unit variance based on the mean and standard deviation pixel intensity. See Figure 1 for data sampling and inference procedure. In this work, we only consider training images, since we seek to characterize the transformations learned on the training data.

The computational intensity of considering large numbers of these high-dimensional embeddings and computing statistics on these matrices made CIFAR-10, a small but visually-diverse dataset, a better choice for initial experiments than datasets like ImageNet. However, in the interest of understanding whether trends we found were specific to CIFAR-10 or whether they also occurred in other datasets, we considered Tiny ImageNet. Images were preprocessed similarly but are also resized to match the input size of the pretrained model available in Keras. In this case, due to computational restrictions, 10 classes were randomly sampled from the 200 classes and 100 representative images are used per class.

*Table 1.* Accuracy of representative models on 10,000 image stratified sample of the CIFAR-10 training set.

| Model | Dataset | Sample accuracy |
|---|---|---|
| MLP12 | CIFAR-10 | 77.5% |
| VGG-16 | CIFAR-10 | 99.9% |
| VGG-19 | CIFAR-10 | 99.1% |
| VGG-16 | Tiny ImageNet | 4.14% (11.2% @ Top 5) [2] |

### 2.2. Representative models: MLP and CNN

We study three trained architectures: a 12-hidden layer multi-layer perceptron (MLP), a convolutional neural network (CNN) similar in architecture to VGG-16 (8), and a CNN similar in architecture to VGG-19 (8) [1]. All these architectures have non-increasing numbers of dimensions through the network. The MLP has 1000 hidden units throughout the network, such that compressive behaviors can be studied independently of decreasing feature dimensions. We train the MLP until convergence on the training set (366 epochs) with stochastic gradient descent with Nesterov accelerated gradients, initial learning rate 0.01, and learning rate decay every 20 epochs. The VGG-16 weights for CIFAR-10 are used from (9). For VGG-16 on Tiny ImageNet, we use the default implementation in Keras, which is taken from (10). This has low performance, in line with (but worse than) publicly-released performance on the dataset (7). For VGG-19 on CIFAR-10, the model is trained with similar parameters to before. For all models, data is augmented via shifts, rotation, and horizontal flipping.

Table 1 presents the accuracy of these models. Classification accuracy is relevant because lower accuracy indicates poor class separation in the final hidden layer, implying suboptimal geometric transformations by the network.

### 2.3. Analysis and Statistics

In the interest of understanding whether neural networks compress the feature space within a class, we consider activation matrices $\Phi^{(l)}$ where a row corresponds to the flattening of the activation matrix for each input $x_i$ into the vector $\phi_i^{(l)}$.

[1] The differences between our implementations and the original were mainly in adapting the input and output shapes to the datasets.

Such an activation matrix is non-square in general, decomposable via the singular value decomposition (SVD), $\Phi^{(l)} = U\Sigma V^T$. Understanding subtleties of dimensionality (a geometric property) requires understanding how many *important* singular values there are, not necessarily absolute magnitudes. Hence, we normalize the computed singular values by the largest singular value of the activation matrix, yielding statistic $\Sigma'(\Phi) = \frac{1}{\sigma_{\max}}\Sigma(\Phi)$.

**Effective Dimension** Based on the aforementioned criteria, we propose the following metric for dimensionality of a collection of feature vectors:

$$d_{\text{eff}}(\Phi) = |\Sigma'(\Phi)|_2 = \frac{1}{\sigma^2_{\text{max}}(\Phi)} tr(\Sigma(\Phi\Phi^T)) \qquad (1)$$

where the last equality holds since the elements of $\Sigma^2$ are non-negative. This is equivalent to the trace of the covariance matrix, a quantity which Liang and Rakhlin show is important to generalization bounds (11).

By definition, $d_{\text{eff}} \in [1, \sqrt{d_l}]$, where $d_l$ is the length of the feature vector and thus the number of singular values. For a perfect classifier of $L$ layers, $d_{\text{eff}}(\Phi_c^{(L)}) = 1$ for any class $c$. Similarly, $d_{\text{eff}}(\Phi^{(l)}) = \sqrt{d_l}$ for rank-$d_l$ scaled identity matrix $\Phi^{(l)} = (\alpha I_{d_l}; 0)^T$. However, in Section 3, we demonstrate that the upper bound $d_l$ is loose in practice because of the low-rank nature of activation matrices.

The effective dimension of a matrix captures the number of significant directions of variation between its rows. Srebro and Shraibman use the trace of the singular matrix as a measure of the complexity of a matrix, e.g. in matrix completion tasks (12). While the notion of measuring complexity of a feature embedding using the profile of the spectrum is a natural one, our work is, to our knowledge, the first to formalize it and use it to study transformations carried out by neural networks.

In order to understand the effective dimensionality of the data within a class, we computed the singular values of the activation matrix at each of the layers and evaluated the effective dimension. The final plots present the average over all classes. Plots of the singular values directly are presented in the Appendix.

# 3. Results

## 3.1. Effective dimension increases prior to decreasing

Within a class, the effective dimension of the inputs increases prior to decreasing for all tested architectures (VGG-16, VGG-19, MLP12) and datasets (CIFAR-10 and Tiny ImageNet) (Figures 2, 3, 4, 5). Concretely, this means that the number of directions of variation that are important increases prior to decreasing. As the number of important directions of variation increase, input data points are spherized. Then, as the number of important directions of variation decrease, the data are more eccentric and so more elliptical. We posit potential explanations in Section 4.1.

## 3.2. Performance of model and rate of dropoff appear to be correlated

In high-performing models (VGG-16 on CIFAR-10 and VGG-19 on CIFAR-10, Figures 2, 3), we see a sharp increase in effective dimension followed by a sharp decrease. In the MLP12 model with lower performance, the decay in $d_{\text{eff}}(\Phi^{(l)})$ with respect to $l$ is more gradual, and in the VGG-16 model tested on Tiny ImageNet, this dropoff is very noisy and slow. This suggests a correlation between performance and effective dimension dropoff, regarding which we speculate in Section 4.2. Indeed, the effective dimension of the final post-softmax activation matrix is smallest for the well performing models according to the tight lower bound $d_{\text{eff}} \geq 1$ for perfect classifiers presented in Section 2.3 (empirically 1.09 in VGG-16 on CIFAR-10).

# 4. Discussion

In this section, we analyze and discuss the implications of our findings. Further, we propose complementary analyses that would bolster our findings.

## 4.1. Geometric Interpretation

Geometrically, early DNN layers increase the "sphericalness" of the data, following which extraneous dimensions are compressed. The early network sphericalization could correspond to the first several layers abstracting features common across classes. Indeed, work in feature visualization (1) finds that early layers extract features corresponding to local filters for patterns common across image classes, such as gradient and edge detectors. Later layers project into spaces where these features are highlighted; subsequently, as points separate by class, there remain fewer degrees of variation within each class. In models with poorer performance in classification, the model likely prunes uninformative directions of variation more poorly.

## 4.2. Apparent correlation between performance and dropoff and Implications

The trend in VGG-16 trained on ImageNet and tested on Tiny ImageNet is less drastic than the trend in the other three models. We posit this is due to poorer performance by that model, since poor grouping within class and separation between classes would lead to less-dramatic compressions in effective dimension. The 11% performance in our experiments is better than random guessing, suggesting that

[2]Note that Tiny ImageNet has 200 classes, so top 1 accuracy above 0.5% is better than random guessing.

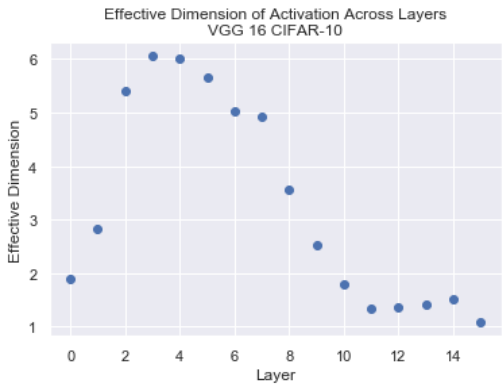


*Figure 2.* This model had close to 100% training accuracy, and a sharp increase followed by decrease in effective dimension is seen. Layer 0 is the input.

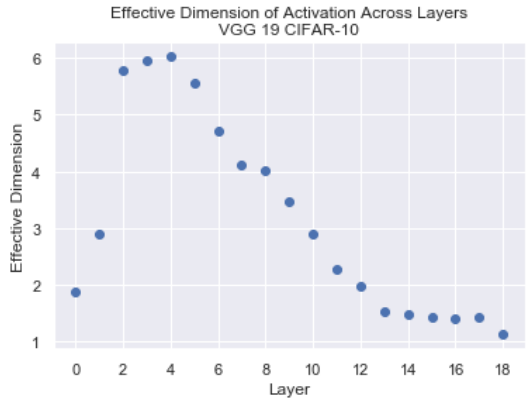


*Figure 3.* This model has close to 90% training accuracy; the sharp increase followed by decrease in effective dimension is comparable to that seen in Figure 2. Layer 0 is the input.

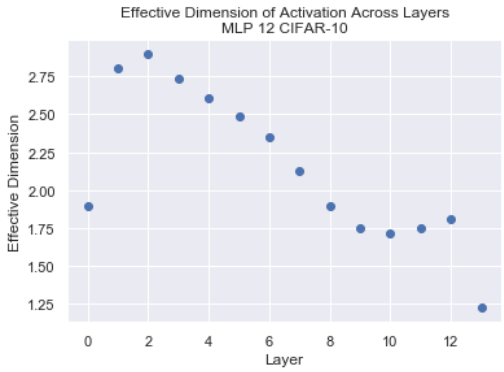


*Figure 4.* This MLP model had around 77% training accuracy; we see a similar increase but a slower decline in effective dimension. We posit that this is related to training accuracy.

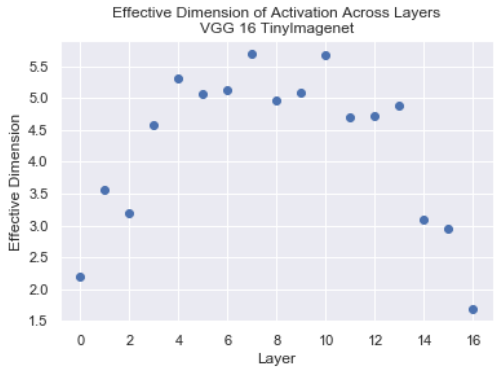


*Figure 5.* This VGG-16 model was trained/tested on ImageNet. Poor performance (11% top 5) appears to be related to the poor dropoff in the later layers. This is further discussed in Section 4.2.

the model does learn some valuable features but does not learn the best weights and therefore has not pared away unimportant dimensions.

It appears (preliminarily) that a sharp decline in effective dimensionality further in the network corresponds with higher accuracy. This seems plausible given that removing extraneous degrees of variation ought to correspond with better decision boundaries in classification. In practice, if this correlation is strengthened, we could factor this into an additional loss term that would incentivize compression of embeddings into lower effective dimension.

### 4.3. Complementary Analyses

An interesting qualitative analysis of separation of classes throughout layers of the network would entail computing a t-distributed stochastic neighbors embedding (t-SNE) of the vector activations at each layer for all the classes and considering how they change over the course of the network. This would provide qualitative insight regarding where separation begins and might provide evidence for or against the hypothesis that the first several layers act primarily as feature extractors while the last several layers act to project these features into spaces where they can be separated.

Currently, we are analyzing effective dimension in layers of the network during training to further understand how performance correlates with the rise and dropoff of effective dimension.

## 5. Conclusions

In studied examples, neural networks initially spherize embeddings and then collapse dimensionality. The compression of the dimensionality of feature spaces via transformations on inputs is more dramatic in better-performing networks.

## References


[1] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. 2009.

[2] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable Basis Decomposition for Visual Explanation. page 16.

[3] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

[4] Joseph Antognini and Jascha Sohl-Dickstein. Pca of high dimensional random walks with comparison to neural network training. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10328–10337. Curran Associates, Inc., 2018.

[5] Sören Dittmer, Emily J King, and Peter Maass. Singular values for relu layers. *arXiv preprint arXiv:1812.02566*, 2018.

[6] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[7] Tiny imagenet visual recognition challenge. https://tiny-imagenet.herokuapp.com/.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] Yonatan Geifman. cifar-vgg. https://github.com/geifmany/cifar-vgg, 2018.

[10] Very deep CNNS for large-scale visual recognition.

[11] Tengyuan Liang and Alexander Rakhlin. Just interpolate: Kernel "ridgeless" regression can generalize.

[12] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In Peter Auer and Ron Meir, editors, *Learning Theory*, volume 3559, pages 545–560. Springer Berlin Heidelberg.


## 6. Appendix

### 6.1. The sensitivity of trace-norm to scale

A Schatten norm such as the nuclear norm (trace-norm) is sensitive to the scale of the activation matrix $\Phi^{(l)} \in R^{n \times d_l}$ for intermediate layer $1 < l < L$ as it can be rescaled arbitrarily by scaling the weights of the parametric classifier without changing the classifier's decision boundaries. For instance, in a deep linear network (DLN), if feature $\phi^{(l)}$ is scaled by factor $\alpha$, the spectral norm of $\alpha * \Phi^{(l)}$ is $\alpha * \sigma_{\max}(\Phi^{(l)})$. This is also true in a ReLU network with $\alpha > 0$. Such a scaling is achieved while preserving $\phi^{(l+1)}$:

$$\phi^{(l+1)} = W^{(l+1)} W^{(l)} \phi^{(l-1)} = \alpha^{-1} W^{(l+1)} \alpha W_k^{(l)} \phi^{(l-1)} \tag{2}$$

While Srebro et al. (12) directly apply the trace-norm to bound the complexity of a completed matrix, we apply spectral normalization in Equation 1 to correct for this scale sensitivity. Hence, a small effective dimension corresponds to an eccentric feature space regardless of magnitude.

### 6.2. Individual singular values show breakdown of effective dimension

When considering the second, third, tenth, and one-hundredth singular values, we see the same initial trends as in the overall effective dimension. Later in the network, the smaller singular values strictly decay in high-performing networks, while the earlier ones sometimes increase or stagnate. The decay of the effective dimension, then, is dominated by the strict decay in later layers of the $\sigma_i$ for large $i$.

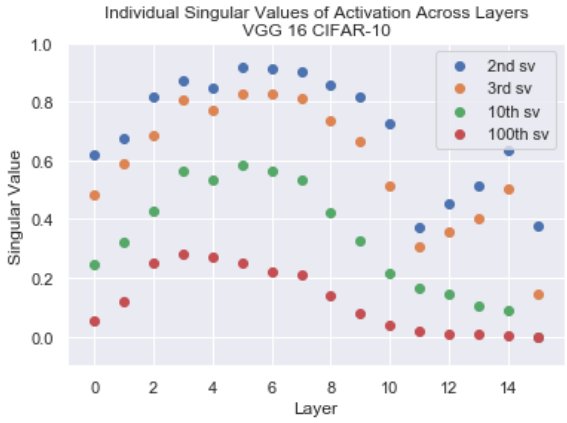


*Figure 6.* VGG-16 CIFAR-10 singular values

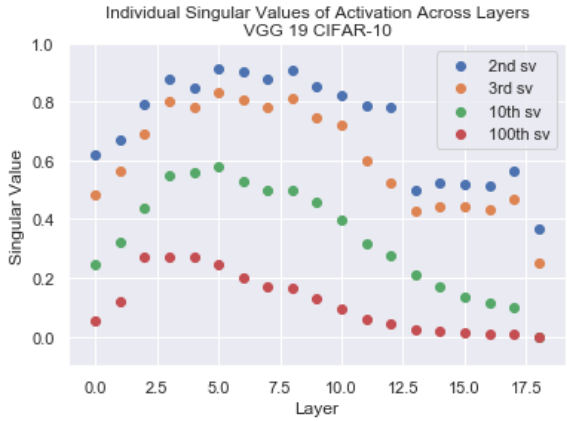


*Figure 7.* VGG-19 CIFAR-10 singular values

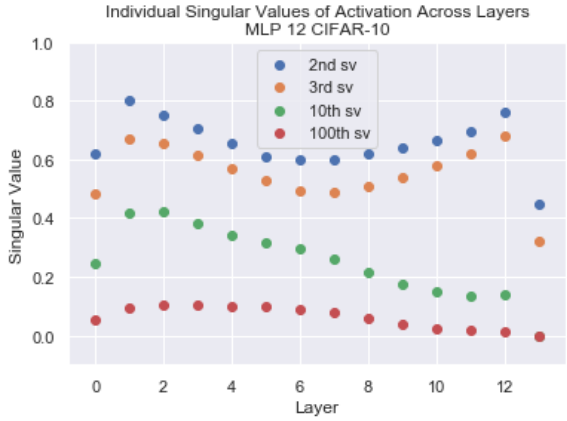


*Figure 8.* MLP12 CIFAR-10 singular values

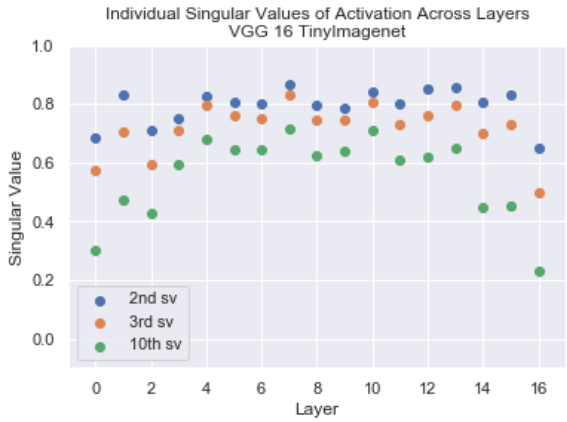


*Figure 9.* VGG-16 Tiny ImageNet shows much more noise/variation, likely corroborating with poor performance.