Frivolous Units: Wider Networks Are Not Really *That* Wide

Stephen Casper,[⋆1,2] **Xavier Boix,**[⋆1,2,3]
Vanessa D'Amario,[3] **Ling Guo,**[4] **Martin Schrimpf,**[2,3] **Kasper Vinken,**[1,2] **Gabriel Kreiman**[1,2]

⋆ *Shared first authorship.*

[1]Boston Childrens Hospital, Harvard Medical School, USA
[2]Center for Brains, Minds, and Machines (CBMM)
[3]Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, USA
[4]Neuroscience Graduate Program, University of California San Francisco, USA

scasper@college.harvard.edu, xboix@mit.edu

Abstract

A remarkable characteristic of overparameterized deep neural networks (DNNs) is that their accuracy does not degrade when the network's width is increased. Recent evidence suggests that developing compressible representations is key for adjusting the complexity of large networks to the learning task at hand [3, 62, 52]. However, these compressible representations are poorly understood. A promising strand of research inspired from biology is understanding representations at the unit level as it offers a more granular and intuitive interpretation of the neural mechanisms. In order to better understand what facilitates increases in width without decreases in accuracy, we ask: Are there mechanisms at the unit level by which networks control their effective complexity as their width is increased? If so, how do these depend on the architecture, dataset, and training parameters? We identify two distinct types of "frivolous" units that proliferate when the network's width is increased: prunable units which can be dropped out of the network without significant change to the output and redundant units whose activities can be expressed as a linear combination of others. These units imply complexity constraints as the function the network represents could be expressed by a network without them. We also identify how the development of these units can be influenced by architecture and a number of training factors. Together, these results help to explain why the accuracy of DNNs does not degrade when width is increased and highlight the importance of frivolous units toward understanding implicit regularization in DNNs.

1 Introduction

A striking feature of deep neural networks (DNNs) is that wider networks with more units in each layer tend to generalize as well or better than thinner ones, even when trained without explicit regularization. In practice, these networks are typically overparameterized, *ie.* the number of free parameters is often several orders of magnitude greater than the number of training examples, yet wide versions still avoid overfitting relative to thinner ones [39, 40, 42, 45]. This phenomenon is demonstrated in Fig. 1, in which we plot the testing accuracy of common overparameterized architectures trained in ImageNet and CIFAR-10 while varying each network's width (see section 4 for details). We vary the number of units in fully connected layers and filters in convolutional layers by a size factor. For each architecture, the test errors of wider, more overparameterized, networks do not degrade relative to thinner ones, despite having a greater potential for overfitting.

Frankle and Carbin [14] find that in certain DNNs, the crucial computations are performed by weight-sparse subnetworks with initializations primed for the learning task, *ie.* such subnetworks have won the "initialization lottery." In doing so, they suggest that wide networks may perform as well as or better than thin ones because they "buy more lottery tickets" and more reliably contain these fortuitously initialized subnetworks. However, regarding subnetworks that are not part of a winning lottery ticket, it remains unclear why they do not have a harmful effect on testing performance. Some clues may come from recent works involving compression-based performance bounds [3, 62, 52], suggesting a link between compressibility and non-overfitting in wider networks. This leads to the central question addressed in this paper: what neural mechanisms facilitate increases in network width without decreases in accuracy?

A promising strand of research inspired from neuroscience has aimed to understand network representations at the individual unit level [58, 61]. Units have been referred to as the "building blocks of interpretability" [44], as analyzing them allows for simple interpretations of DNNs. In this paper, we investigate the ways in which unit-level representations evolve as the width of a network increases. Several signs point to unit-level compressibilities. One example is the success of dropout [50]. Other clues come from biological brains which are remarkably robust to neuronal death and have redundant patterns of activity, suggesting that many neurons are not necessary for short-term performance [51, 17].

In a series of experiments, we identify two types of *frivolous* units which emerge in greater quantity when the network's width is increased: *prunable units* which can be dropped out of the network without significant change to the output, and *redundant units* whose activities can be expressed as a linear combination of other units. These units are types of complexity constraints because the function the network represents could be expressed by a thinner network without them. We show that the rate at which these frivolous units

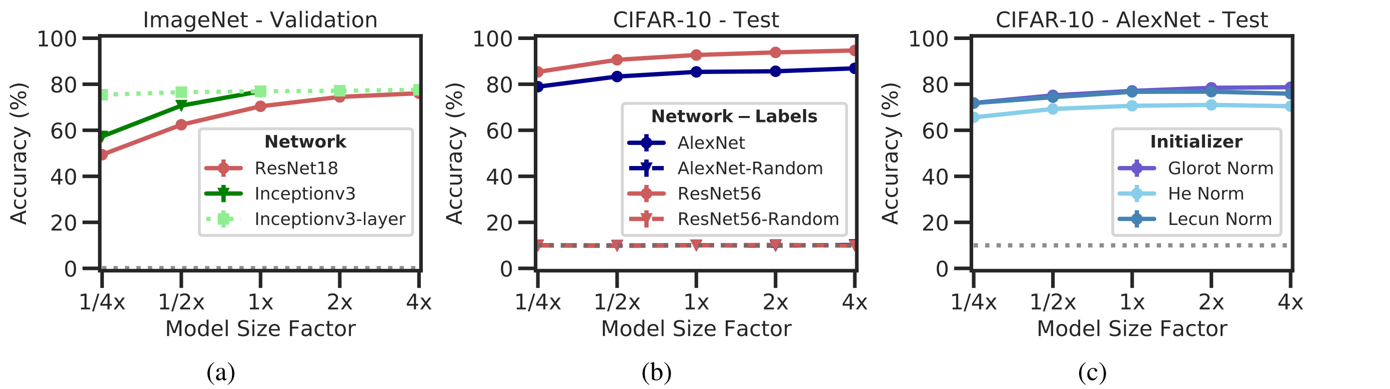


Figure 1: **Test accuracy does not degrade with model width in overparameterized deep networks.** Top-1 test accuracies across model sizes and datasets. (a) ResNet18s, Inception-v3s, and Inception-v3-layer (with a single layer's size varied) trained in ImageNet. (b) Regularized AlexNets and ResNet56s with and without training on random labels in CIFAR-10. (c) Unregularized Alexnet with Glorot, He, or LeCun initialization, in CIFAR-10.

appear consistently outpaces the growth of the network as a whole. This suggests that they play a major role in how DNNs constrain their effective complexity as their width is increased. Furthermore, our results add to the growing body of literature that investigate the *implicit regularization* mechanisms in DNNs by showing that frivolous units help to adjust the width of the network to the task at hand.

2 Related Works

This work lies at the intersection of three areas of research involving DNNs: understanding implicit regularization, interpreting units, and network compression.

Understanding Implicit Regularization. DNNs exhibit fascinating properties related to generalization including double descent [34, 38, 11] and the ability memorize entire datasets with random labels while still generalizing well when trained on uncorrupted data [59]. A number of works have aimed to uncover causal mechanisms that explain why, out of the many optima DNNs can reach, they tend toward simple and geometrically smooth solutions that generalize well [30, 39, 40, 41, 60, 45, 42]. In particular, some have shown that DNNs fit simple patterns more readily than complex ones [2, 4, 16, 12] and that deep ReLU networks tend to exhibit "surprisingly few" activations patterns [22, 56].

Instead of focusing on the broader question of why DNNs generalize well, in this work we focus on the more constrained problem of understanding how performance does not degrade with increases in model complexity. This problem follows in part from the lottery ticket hypothesis [14] as it remains unclear how the exceeding capacity of wider networks does not degrade the accuracy.

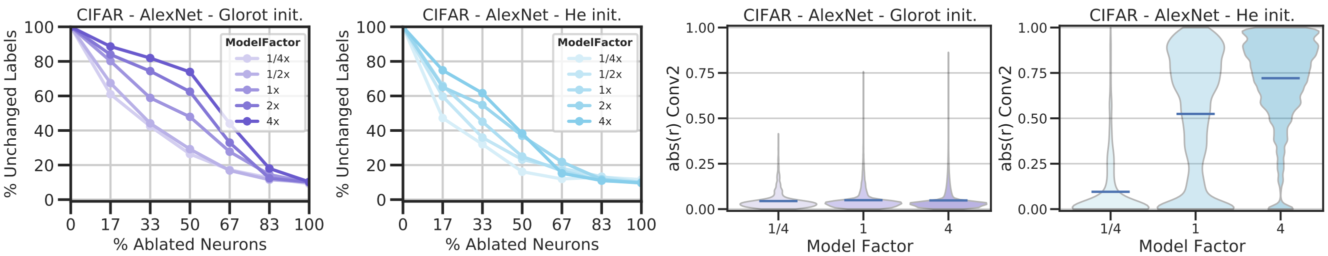
Understanding Networks at the Unit Level. A promising framework inspired from neuroscience is understanding representations at the unit level because it leads to elementary and intuitive interpretations of the neural mechanisms. In artificial DNNs, these methods tend to focus on analyzing how units respond to data [58, 36]. Much progress has been made in developing semantic understandings of units in networks via optimizing inputs to maximize the activations of units [43, 44], ablational analysis both in recognition networks and GANs [61, 6], and interpreting units via a dataset of visual semantic concepts (known as "dissection") [5, 37].

Network Compression. Recent evidence suggests that networks develop compressible representations to adjust the complexity of large networks to the learning task at hand [3, 62, 52]. Broadly speaking, existing compression algorithms fall into four distinct categories: quantization, knowledge distillation, parameter pruning, and low-rank factorization [8, 10]. Of these, we focus on pruning and low-rank factorization because they both allow for a simple mapping from an overparameterized network to a compressed one with fewer parameters. Several existing works have studied compression via the removal of low-valued weights and have shown that the number of parameters can often be compressed by an order of magnitude or more this way [15, 35, 54, 33]. However, weights are more difficult to interpret than units, and these methods prune weights based on magnitude alone without analyzing how the network behaves across a dataset. Contrastingly, the approaches we discuss in the following section are data-driven and operate on the unit level.

3 Frivolous Units

In this section, we introduce unit-level mechanisms that facilitate increases in network width without decreases in accuracy. We demonstrate that these lead to a function computed by the wider network that can be well-approximated by a thinner one, and hence help to explain how accuracy does not degrade with width. Let $\mathcal{N}$ and $\mathcal{W}$ denote respectively a narrow and a wide DNN with the same architecture which represent mappings from datapoints to labels denoted as $f_{\mathcal{N}}$ and $f_{\mathcal{W}}$. Thus, to explain how the network $\mathcal{W}$ does not utilize its full capacity, we show that $f_{\mathcal{W}}$ can be expressed with a thinner network, *ie.* $f_{\mathcal{W}} \approx f_{\mathcal{N}}$ with respect to a data distribution.

To investigate what types of representations $\mathcal{W}$ develops to facilitate that $f_{\mathcal{W}} \approx f_{\mathcal{N}}$, we introduce prunable units which can be removed from a network by excision and redundant units which can be removed by factorizing their layers. Furthermore, we show that these units are distinct and can



(a) (b) (c) (d)

Figure 2: **Networks develop units which are prunable and units that are linearly dependent to varying degrees.** All networks were identically trained AlexNets except for initialization. No explicit regularization was used. Glorot (low-variance) initialized networks are in purple, and He (high-variance) in blue. (a-b) Prunable units: each point shows the proportion of labels that did not change when the percentage of units on the x-axis was randomly ablated. (c-d) Redundant units: violin plots with absolute-valued correlation coefficients between unit activations for the final convolutional layer across the testing set.

emerge independently of one another. Similar motifs have been targeted by previous compression works which focus on finding smaller networks that have similar accuracy to an uncompressed one. However, in contrast with these methods, we focus on showing that smaller networks exist which compute similar *functions* as a larger network, *ie.* $f_{\mathcal{W}} \approx f_{\mathcal{N}}$ across a data distribution. This is a more stringent condition than evaluating the accuracy, as preserving accuracy is a necessary condition for showing $f_{\mathcal{W}} \approx f_{\mathcal{N}}$ but not sufficient.

Prunable Units. Several network compression algorithms based on pruning nonessential units have been used to compress networks to a fraction of their original size while maintaining testing accuracy [20, 21, 26, 32]. The fact that DNNs are robust to the removal of certain prunable units suggests that these networks compute functions that have a lower effective complexity than their architectures are capable of.

Suppose that a narrow network, $\mathcal{N}$, is identical to a wide one, $\mathcal{W}$, but with a set of units removed and that $f_{\mathcal{W}} \approx f_{\mathcal{N}}$ across a data distribution. If so, then we refer to the units that were removed from $\mathcal{W}$ as *prunable*. Note that, due to interactions between the effects of multiple units, joint prunability and individual prunability are distinct. A variety of phenomena could result in prunability ranging from simple explanations such as units being sparsely activated among examples or their outgoing weights being small, to more complex ones such as subsequent layers discarding their activity.

To evaluate how well removing a set of unit preserves network function, we analyze the proportion of examples in a testing set whose labels do not change when units are ablated. Finding the largest set of units that can be removed from a network in a way that results in a given proportion of unchanged labels is NP-hard. Instead of searching for optimal prunable sets, to scalably measure how prunable units are on average, we analyze the proportion of unchanged labels when random ablations are applied to various numbers of units (see Section 4 for details). Fig. 2a-b shows these results. As more units are removed, more output labels change for all networks, however, they exhibit different trends. Fig. 2a shows a case in which the labels output by wider networks are more resistant to random ablations of units than thinner networks. This indicates that in this case, much of the wider networks' excess capacity is being used for the development of prunable units. Contrastingly, Fig. 2b shows a case in which there is significantly less increase in robustness in a set of networks that only differ from those in Fig. 2a by how they were initialized. However, both types of networks maintain their performance when width is increased as shown in Fig. 1, suggesting that the networks in Fig. 2b may be developing other types of capacity constraints. This motivates the search for a second mechanism by which complexity constraints can be understood at the unit level.

Redundant Units. In contrast to compression algorithms based on pruning are ones which focus on low-rank factorization. These methods in practice have successfully compressed networks to a fraction of their original size while maintaining testing accuracy [47, 13, 49]. Most previous works using these methods compress units by representing them in weight space, but units can similarly be represented in activation space across a dataset. We denote as *non-redundant* the largest set of units whose activations are linearly independent (though in experiments we relax this via PCA). The number of non-redundant units is equal to the rank of a matrix that represents the activations of the units across a dataset. The remaining *redundant* units help to regulate the complexity of the network because a thinner network can be constructed without them such that $f_{\mathcal{W}} \approx f_{\mathcal{N}}$ across the dataset. In the Appendix, we explain how the thinner network is obtained by removing the redundant units and refactoring the outgoing weights of the non-redundant ones.

As with prunability, networks sometimes develop very different levels of redundancy at different width factors. Fig. 2c-d depicts the distributions of absolute-valued unit-to-unit correlation coefficients for the final convolutional layers in two classes of networks which only differ in how they were initialized. Fig. 2c shows a case in which the level of unit-to-unit correlation increases only slightly as network width is increased. However, Fig. 2d shows a case in which much more correlation develops with wider networks. Correspondingly, these networks develop more redundant units (see Section 4 for further details). This implies that here, excess capacity is largely utilized to form these redundant units.

Dataset	Network	Initialization	Optimizer	Regularizers	L.Rate-B.Size
Uncorr. 10 dim	MLP	Normal⋆	Momentum	None	Best
Uncorr. 10k dim	MLP	Normal⋆	Momentum, SGD Adam ⋆	None	Best
CIFAR-10 (+ rand. labels⋆)	AlexNet ResNet56	Glorot/LeCun/He⋆ Glorot	Momentum Momentum	None, DA, DO, WD⋆ BN, DA, WD	Best⋆ Best⋆
ImageNet	ResNet18 Inception-v3	Glorot Normal	Momentum RMSProp	BN, DA, WD BN, DA, WD	Best⋆ Best

Table 1: **Network training and performance details:** "BN" refers to batch normalization, "DA" refers to data augmentation, "DO" refers to dropout, and "WD" refers to L2 weight decay. "Best" refers to learning rate/batch size combinations that achieved the highest accuracy. Stars (⋆) indicate factors for which we tested multiple hyperparameters/variants.

Relating Prunable and Redundant Units

As reflected in Fig. 2, we clarify here that prunability and redundancy are distinct and prove by construction that they can emerge either together or independently. Recall that $\mathcal{N}$ and $\mathcal{W}$ denote respectively a narrow and a wide DNN with the same architecture and that each represents a mapping from datapoints to labels denoted by $f_\mathcal{N}$ and $f_\mathcal{W}$. Let $\mathbf{u}$ refer to the activity of the layer before the output layer in $\mathcal{N}$ and $\boldsymbol{\theta}$ the incoming weights for the output layer of $\mathcal{N}$. Thus, the output of $\mathcal{N}$ is equal to $\boldsymbol{\theta}^T\mathbf{u}$. Examples can be constructed of a wide network $\mathcal{W}$, twice the size of $\mathcal{N}$, such that $f_\mathcal{N} = f_\mathcal{W}$ (not only approximately, but exactly) yet each $\mathcal{W}$ has different levels of prunability and redundancy. Note that there are other possible cases aside from the prototypes presented here.

More of Both Prunable and Redundant Units. We can build a $\mathcal{W}$ that computes the same function as $\mathcal{N}$ by first duplicating the final layer of $\mathcal{N}$ such that the layer before the output has activity $[\mathbf{u}, \mathbf{u}]$ and setting the output weights equal to $[\boldsymbol{\theta}, \mathbf{0}]$. Because $[\boldsymbol{\theta}, \mathbf{0}]^T[\mathbf{u}, \mathbf{u}] = \boldsymbol{\theta}^T\mathbf{u}$, the outputs will be identical to $\mathcal{N}$. All other layers of $\mathcal{W}$ can then be constructed analogously in the order of deeper to shallower layers. In this case, both prunable and redundant units would be greater in quantity in $\mathcal{W}$ because half of the units in the network are redundant with the other half and can be pruned due to having $\mathbf{0}$-valued outgoing weights. Another example would be constructing $\mathcal{W}$ with the units in $\mathcal{N}$ and with units whose activity is always equal to 0 because such unresponsive units are prunable and have no activity.

Only More Prunable Units. A $\mathcal{W}$ that is more prunable but not more redundant can be constructed by first making the units before the output layer in $\mathcal{W}$ equal to $[\mathbf{u}, \mathbf{v}]$, where $\mathbf{v}$ gives units with activities orthogonal to $\mathbf{u}$ and each other across the data distribution. Thus, the units are not redundant. To make $f_\mathcal{W}$ equivalent to $f_\mathcal{N}$, the outgoing weights of this layer can be set equal to $[\boldsymbol{\theta}, \mathbf{0}]$, and as in the previous case, the units that are multiplied by 0 (half of the layer) are prunable. Then constructing other layers analogously from deeper to shallower layers results in a $\mathcal{W}$ that is only more prunable.

Only More Redundant Units. We can duplicate the final layer of $\mathcal{N}$ so that $\mathcal{W}$'s final layer has activations equal to $[\mathbf{u}, \mathbf{u}]$, making half of the units redundant. Then we can obtain a network that computes the same function as $\mathcal{N}$ via "weight balancing" with weights leading into the output layer of $[\frac{\boldsymbol{\theta}}{2} + b, \frac{\boldsymbol{\theta}}{2} - b]$, where b is a large constant. Although $[\frac{\boldsymbol{\theta}}{2} + b, \frac{\boldsymbol{\theta}}{2} - b]^T[\mathbf{u}, \mathbf{u}] = \boldsymbol{\theta}^T\mathbf{u}$, when a unit is pruned, the layer's output changes by an offset of $(b - \frac{\theta_k}{2})u_k$, where u_k is the removed unit activation, making it not prunable when b is sufficiently large. Then $\mathcal{W}$ can be completed to be only more redundant by performing the same procedure on on the rest of the network from deeper to shallower layers.

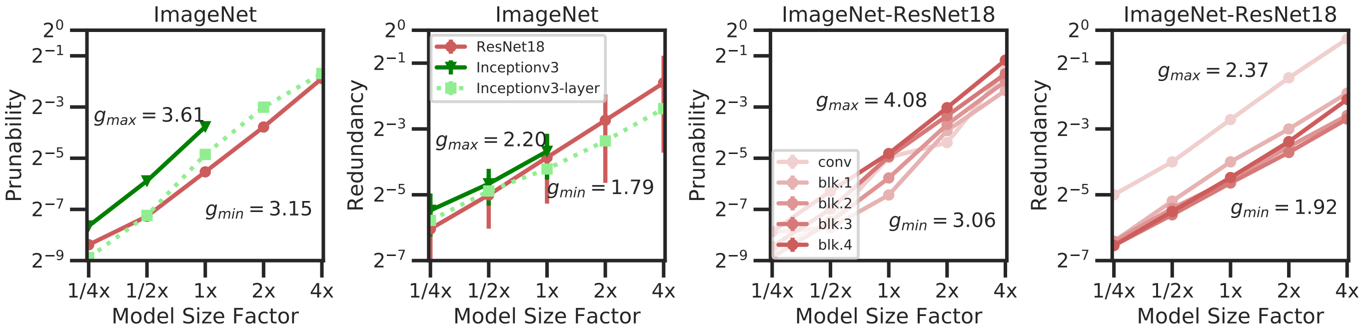
Upshot. Finally, it is helpful to note that $f_\mathcal{W} \approx f_\mathcal{N}$ does not necessarily imply that frivolous units must exist in $\mathcal{W}$ because capacity constraints need not emerge at the unit level, and circuits of different complexity can still compute similar functions. We provide clarifying examples in the Appendix. Also in the Appendix, we show that for randomly initialized networks, prunability and redundancy will, in expectation, be proportional to the network width. However, it remains unclear precisely how they emerge in trained networks. As we will show in the following sections, the growth of frivolous units tends to outpace the growth of a network as a whole.

4 Methods

Table 1 gives training details for networks. Features we tested multiple variants of are marked with a star (⋆). Further details for all networks are in the Appendix. To see how prunability and redundancy vary with width, we tested variants of each network in which the number of weights/filters in each layer/block/module were multiplied by factors of $1/4$, $1/2$, 1, 2, and 4.

Datasets. For larger scale experiments, we used the ImageNet [46] and CIFAR-10 [28] datasets. For small-scale experiments, we used training and testing datasets of 1,000 binary-labeled examples generated by $1/4$x-sized randomly-initialized multilayer perceptrons (MLPs) from uncorrelated Gaussian inputs. We verified these teacher MLPs to output each label for between 40% and 60% of inputs. We display only results for the test set, but they were generally indistinguishable from the train set.

Networks. For ImageNet, we used ResNet18s from He et al. [24] and Inception-v3 networks from Szegedy et al. [53]. Due to hardware restrictions (we used a dgx1 with 8x NVIDIA V100 GPUs 32GB), we were not able to train any 2x or 4x Inception-v3s and instead experimented with versions where a single layer's size varied from $1/4$x to 4x (denoted as *Inception-v3-layer* in plots). For experiments with CIFAR-10, we used AlexNet models based on Zhang et al. [59] and ResNet56s from He et al. [24]. For smaller-scale experiments, we used simple MLPs with 1 hidden layer of 128 units for the 1x size. For all networks tested, increasing model size



(a) (b) (c) (d)

Figure 3: **Prunability strongly outpaces the growth of the network as a whole and redundancy increases at rates similar to the network in ResNet18s and Inception-v3s.** (a) Prunability. (b) Redundancy. (c) Prunability layerwise for ResNet18s. (d) Redundancy layerwise for ResNet18s. The gain, g, represents the increase in prunability or redundancy when the network size is doubled. Max and min g values are given for each plot.

results in equal or improved performance as demonstrated in Fig. 1. In the Appendix, we also plot the number of trainable parameters for each network.

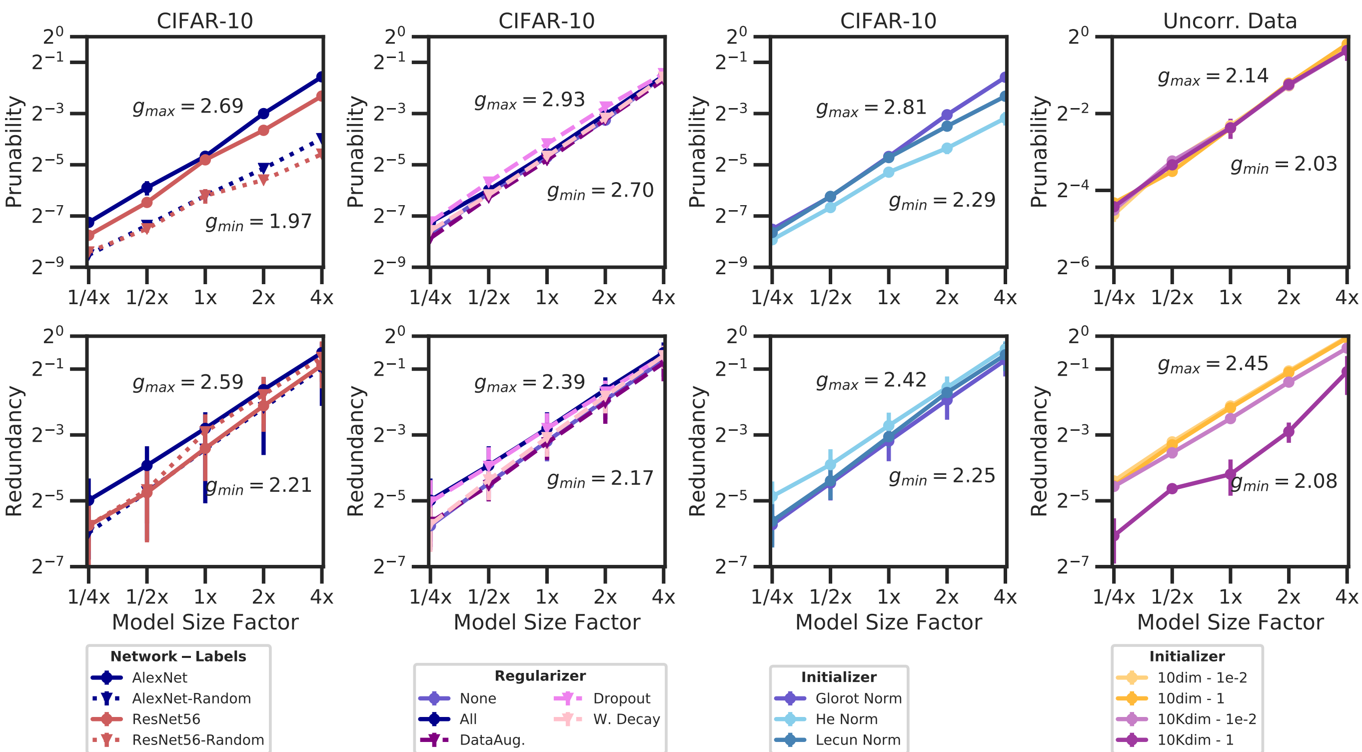
Measuring Prunability and Redundancy

We measure frivolous units via ablation and linear analysis of activations which allows us to experiment with hundreds of networks including convolutional nets at the ImageNet scale.
Prunability. To scalably measure the extent to which the units of a network are prunable on average, we analyze robustness to random ablation. We first find what proportion of labels for the test set do not change when varying proportions of units are ablated. These are applied to fully connected layers and feature map outputs in convolutional layers such that each spatial location is treated as a different unit. After obtaining the corresponding ablation curves as shown in Fig. 2, we use linear interpolation to estimate the proportion of units which can be randomly ablated on average with a tolerance for label corruption of 0.2 (though trends were similar for different tolerance levels which we tested up to 0.5). This procedure is repeated three times with different sets of ablated units and the set that yields the largest number of prunable units is selected. Finally, we divide by the number of units in the 4x model to normalize the results.
Redundancy. In order to scalably quantify the extent to which units in a DNN are redundant, we collect each layer's activation matrix across the test set. For convolutional layers, we treat each feature map output as a unit and consider each spatial location as an example for the same unit. We use principal component analysis on the activations to calculate the number of units that are redundant with a tolerance of 0.05 for the proportion of unexplained variance (though trends were similar for different tolerance levels which we tested up to 0.15). We divide by the size of that layer in the 4x model to normalize the results and report the average across all layers of a network weighting each equally regardless of size.

5 Results

Here, we introduce how prunability and redundancy vary as a function of model width across size factors from 1/4x to 4x. We use a $\log$ scale for both axes in all plots and include maximum and minimum gain values for the curves in each plot represented as $g_{\min}$ and $g_{\max}$. This gain value gives the average increase in the number of prunable and redundant units as a network width is doubled. A gain $g > 2$ indicates that one type of unit more-than-doubles on average when the network width is doubled. Excluding Figs. 3a,c,d, points are averages across three trials. Error bars giving standard deviations are provided, but do not always appear at the given scale. See the Appendix for further details.
Prunability and/or redundancy increase at a rate greater than units of the network overall. Fig. 3a-b and Fig. 4a-d show that both types of units consistently emerge across experiments. For all but two cases (redundancy in Inception-v3s and prunability in ResNet56s trained on random labels), $g_{\min} > 2$ meaning that the growth of frivolous units is greater than that of the overall network. We also plot trends for the individual layers of ResNet18s in Fig. 3c-d and Inception-v3s in the Appendix. While frivolous units consistently emerge in individual layers when their width is increased, they do so to varying extents, and we find no consistent relationship between depth and frivolity.

The fact that frivolous units tend to more-than-double is mirrored by the fact that their complements tend to less-than-double which is shown in the Appendix. We also find that although non-prunable and non-redundant units increase at a lower rate, they increase nonetheless. This suggests avenues for future work toward understanding non-prunable and non-redundant units using more sophisticated mechanisms than analysis of prunable and redundant units separately. One promising possibility is analysing whether thinner equivalent networks can be constructed by removing both prunable and redundant units. Also note that in some networks, particularly Inception-v3s and ResNet18s in Fig. 3a-b, trends in prunability are quantitatively different from trends in redundancy which demonstrates that these measurements respond to distinct sets of units.
Prunability and redundancy develop in networks trained on randomly-labeled data. To better understand the relationship between frivolous units and generalization, we ana-



(a) (b) (c) (d)

Figure 4: **Prunability and/or redundancy outpace the growth of the overall network for different architectures, regularizers, initializers and datasets.** (a) AlexNets and ResNet56s trained with and without random labels (CIFAR-10). (b) AlexNets trained with and without regularization (CIFAR-10). (c) AlexNets trained with various initializations (CIFAR-10). (d) MLPs with different trainsets and initializations (synthetic data). Max and min gain factors are given for each plot.

lyze networks trained to memorize randomly labeled images. Fig. 4a compares the results with AlexNet and ResNet56 models trained with and without random labels. With random labels, all networks of the 1x size or greater were able to fit the training set with at least 99% accuracy. Even when fitting noise, these models increase the quantity of frivolous units and even increase the number of redundant ones with $g_{\min} > 2$. This demonstrates that although the emergence of frivolous units with $g > 2$ implies implicit regularization, it does not imply generalization.

Trends are similar under explicit regularization. To compare the effects of implicit regularization with explicit regularization, we ask how explicit regularizers influence the emergence of frivolous units. In Fig. 4b, we show that data augmentation, dropout, weight decay, and all three together have only modest effects on the trends in frivolous units. This suggests that implicit regularization may operate at the unit level in significantly different ways than explicit regularization.

Initialization influences prunability and redundancy. Some recent works have suggested that network initialization has a large influence over generalization behavior [14, 9, 55]. To see what effects it may have on frivolous units, we test several common methods of weight initialization. Fig. 4c presents results for AlexNets trained with Glorot [18], He [23], and Lecun [31] initializations which each initialize weights using Gaussian distributions with variances depending on the layer widths. In these networks, as the initializations change, there is a tradeoff between the rates at which prunable and removable units increase. We also display the same curves for uniform-distributed versions of these initializations in the Appendix and find their results to be similar, suggesting that the initialization distribution matters little compared to the variance.

Datasets influence prunability and redundancy. To see if frivolous units result from structure in the input data, we train MLPs on uncorrelated data with labels from randomly initialized teacher networks. Fig. 4d, shows the results of altering initialization variance in MLPs trained on these datasets with examples of 10 and 10,000 dimensions. Results from the full hyperparameter searches are in the Appendix. In these networks, despite the uncorrelated training data, prunability and redundancy emerge nonetheless with $g > 2$, though not as large as for other experiments. For the case with high initialization variance and high input dimension, these MLPs have similar g values for prunability to the other networks but have a much lower number of prunable units. This indicates an interaction between initialization and dataset in the emergence of prunable units.

Results are consistent across optimizers, learning rates,

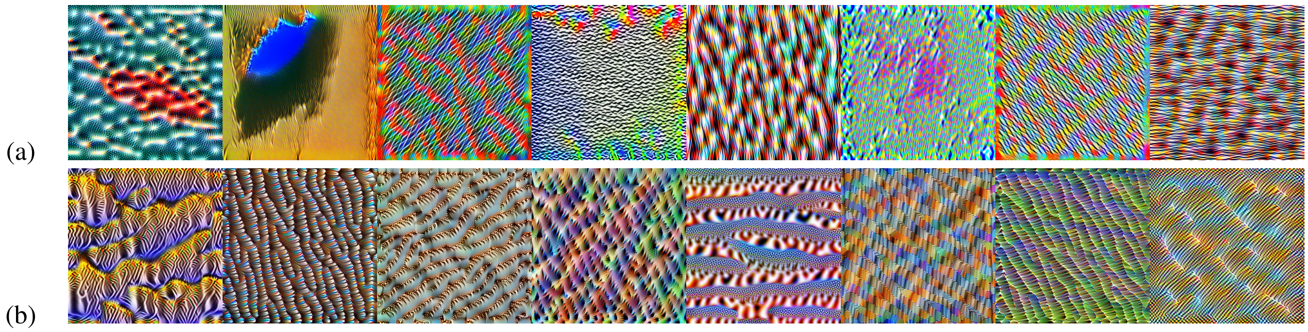


Figure 5: **Example visualizations of 8 units from the first block of the 1/4x (a) and 4x (b) ResNet-18s.** The total variations of these images and their Fourier transforms were used for hypothesis testing via a rank-based permutation test.

batch sizes, and number of training epochs. Results from additional experiments are shown in the Appendix.

Network width influences interpretability but with no consistent trend across layers. Given that prunable units do not seem to represent any features essential for the task at hand and that redundant units may be representable as a combination of multiple feature directions, we ask whether these units pose difficulties for interpreting units in wide networks. To test this, we use Lucid [43] to generate images optimized to maximize the activation of units in 1/4x and 4x ResNet18s and 1/4x and 1x Inception-v3s. Examples from the first block of ResNet-18s are shown in Fig. 5 (see Appendix for more visualizations). As a proxy measure for interpretability, we calculate the *total variation* of these visualizations which measures how different the values of adjacent pixels are. The total variations were analyzed both for the raw visualizations and their Fourier transforms (lower total variation means better interpretability). To compare results between thin and wide networks for each block/module, we use a rank-based permutation test. These tests provide evidence that there are differences between the thin and wide networks across layers, but not that the thin ones are consistently more interpretable. Thorough details including a table of p values and effect sizes are in the Appendix.

6 Discussion and Conclusions

We have analyzed the emergence of prunable and redundant units in relation to the fact that the generalization ability of DNNs does not tend to decrease as network width increases. Our results show that the number of prunable and/or redundant units increases at a rate which outpaces that of the network overall which suggests that complexity-constraining features in deep networks emerge largely at the unit level. Thus, we offer the following hypothesis: consider a narrow deep network $\mathcal{N}$ and a wide one $\mathcal{W}$, both with the same architecture, trained on the same data with the same regularization and initialization schema, each with a tuned set of hyperparameters. Alongside $\mathcal{W}$ generalizing as well or better than $\mathcal{N}$, $\mathcal{W}$ will develop an increased proportion of prunable and/or redundant units relative to $\mathcal{N}$ due to implicit regularization. We hope that future work extending or challenging this hypothesis will lead to a richer understanding of the emergent properties of DNNs.

Despite a great deal of recent progress, to our knowledge, ours is the first work to date that has quantitatively studied the connections between prunability and redundancy together in context of implicit regularization. These methods which target multiple compressible features could be used to improve upon compression based bounds [3] including for approaches such as that of [62] which only considers prunability but not redundancy. We also reveal specific architectures, initializations, and training methods that can be used to control what types of compressible features networks develop. Nonetheless, the fact that both of these types of units consistently proliferate highlights a need for hybrid compression methods which target both.

The framework used here can also be useful for understanding questions about complexity, robustness, and redundancy in neuroscience. Given that brains are often surprisingly capable of recovering from damage and neuronal death, it is understood that they are overparameterized networks which are both highly prunable and have built-in redundancies [51, 17]. In fact, Glassman [17] estimates that at least half of the neurons in the human brain fit under our definition of frivolous. The neural activity of DNNs for object recognition has been shown to resemble neural activity in parts of the brain [57], and several works have aimed to understand redundancy and robustness in brains using artificial networks as models [48, 1, 19]. Thus, the framework presented in this paper could be helpful to understand prunability and redundancy in brains and how biological networks implicitly regularize.

While we have shown that frivolous units are necessary to understand implicit regularization, we do not find that they are sufficient. Although non-frivolous units increase at a smaller rate, they increase nonetheless. Future work should expand on understanding additional types of compressible motifs derived from other techniques such as network distillation [25], subnetwork analysis, or kernel-inspired analysis [27]. Nonetheless, frivolous units play a key role in how networks constrain their effective complexity and offer a milestone toward understanding them at the unit level.

Acknowledgments

We are grateful to Tomaso Poggio for helpful feedback and discussions. This work is supported by the Center for Brains, Minds and Machines (funded by NSF STC award CCF-

1231216), the Harvard office for Undergraduate Research and Fellowships, Fujitsu Laboratories (Contract No. 40008401 and 40008819) and the MIT-Sensetime Alliance on Artificial Intelligence.

Ethics Statement

This work may contribute to consequential progress in three areas: (1) *Designing robust networks:* understanding how prunability and redundancy emerge allows for more control over how they implicitly regularize and what types of representations they develop. This may lead to insights on improving network design and training. (2) *Interpretability:* knowledge of frivolous units is helpful for interpreting networks at the unit level which can be valuable for developing a more basic understanding and verifying properties of learning systems regarding performance or robustness. We expect this to be primarily beneficial, especially for systems in safety-critical settings. (3) *Compression:* our findings suggest directions for work in advancing compression techniques which could improve space and time efficiency in deep learning models. This is of particular interest to mobile and web development and could make these systems significantly more accessible. However, this may come with concerns involving the reliability of these systems or their proliferation outpacing effective oversight.

We join with others in the research community calling for the safe and judicious use of AI in ways that benefit all of humanity.

References

[1] Aerts, H.; Fias, W.; Caeyenberghs, K.; and Marinazzo, D. 2016. Brain networks under attack: robustness properties and the impact of lesions. *Brain* 139(12): 3063–3083.

[2] Ansuini, A.; Laio, A.; Macke, J. H.; and Zoccolan, D. 2019. Intrinsic dimension of data representations in deep neural networks. In *NeurIPS*.

[3] Arora, S.; Ge, R.; Neyshabur, B.; and Zhang, Y. 2018. Stronger generalization bounds for deep nets via a compression approach. In *ICML*.

[4] Arpit, D.; Jastrzebski, S.; Ballas, N.; Krueger, D.; Bengio, E.; Kanwal, M. S.; Maharaj, T.; Fischer, A.; Courville, A.; Bengio, Y.; et al. 2017. A closer look at memorization in deep networks. In *ICML*.

[5] Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*.

[6] Bau, D.; Zhu, J.-Y.; Strobelt, H.; Zhou, B.; Tenenbaum, J. B.; Freeman, W. T.; and Torralba, A. 2018. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597* .

[7] Bengio, Y.; LeCun, Y.; et al. 2007. Scaling learning algorithms towards AI. *Large-scale kernel machines* 34(5): 1–41.

[8] Cheng, Y.; Wang, D.; Zhou, P.; and Zhang, T. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* .

[9] Chizat, L.; Oyallon, E.; and Bach, F. 2019. On Lazy Training in Differentiable Programming. In *NeurIPS*.

[10] Choudhary, T.; Mishra, V.; Goswami, A.; and Sarangapani, J. 2020. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review* 1–43.

[11] d'Ascoli, S.; Refinetti, M.; Biroli, G.; and Krzakala, F. 2020. Double Trouble in Double Descent: Bias and Variance (s) in the Lazy Regime. *arXiv preprint arXiv:2003.01054* .

[12] De Palma, G.; Kiani, B.; and Lloyd, S. 2019. Random deep neural networks are biased towards simple functions. In *NeurIPS*.

[13] Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*.

[14] Frankle, J.; and Carbin, M. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*.

[15] Fukuyama, F. 2014. *State-building: governance and world order in the 21st century*. Cornell University Press.

[16] Gidel, G.; Bach, F.; and Lacoste-Julien, S. 2019. Implicit regularization of discrete gradient dynamics in deep linear neural networks. In *NeurIPS*.

[17] Glassman, R. B. 1987. An hypothesis about redundancy and reliability in the brains of higher species: Analogies with genes, internal organs, and engineering systems. *Neuroscience & Biobehavioral Reviews* 11(3): 275–285.

[18] Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

[19] Hammelman, J.; Lobo, D.; and Levin, M. 2016. Artificial neural networks as models of robustness in development and regeneration: stability of memory during morphological remodeling. In *Artificial Neural Network Modelling*, 45–65. Springer.

[20] Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*.

[21] Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *NIPS*.

[22] Hanin, B.; and Rolnick, D. 2019. Deep relu networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems*, 361–370.

[23] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

[24] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

[25] Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

[26] Hu, H.; Peng, R.; Tai, Y.-W.; and Tang, C.-K. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* .

[27] Jacot, A.; Gabriel, F.; and Hongler, C. 2018. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*.

[28] Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

[29] Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

[30] Kubo, M.; Banno, R.; Manabe, H.; and Minoji, M. 2019. Implicit regularization in over-parameterized neural networks. *arXiv preprint arXiv:1903.01997* .

[31] LeCun, Y. A.; Bottou, L.; Orr, G. B.; and Müller, K.-R. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*, 9–48. Springer.

[32] Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*.

[33] Ma, X.; Yuan, G.; Lin, S.; Li, Z.; Sun, H.; and Wang, Y. 2019. Resnet can be pruned 60x: Introducing network purification and unused path removal (p-rm) after weight pruning. *arXiv preprint arXiv:1905.00136* .

[34] Mei, S.; and Montanari, A. 2019. The generalization error of random features regression: Precise asymptotics and double descent curve. *arXiv preprint arXiv:1908.05355* .

[35] Molchanov, D.; Ashukha, A.; and Vetrov, D. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2498–2507. JMLR. org.

[36] Montavon, G.; Samek, W.; and Müller, K.-R. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* 73: 1–15.

[37] Mu, J.; and Andreas, J. 2020. Compositional Explanations of Neurons. *arXiv preprint arXiv:2006.14032* .

[38] Nakkiran, P.; Kaplun, G.; Bansal, Y.; Yang, T.; Barak, B.; and Sutskever, I. 2019. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292* .

[39] Neyshabur, B.; Bhojanapalli, S.; McAllester, D.; and Srebro, N. 2017. Exploring generalization in deep learning. In *NIPS*.

[40] Neyshabur, B.; Li, Z.; Bhojanapalli, S.; LeCun, Y.; and Srebro, N. 2019. The role of over-parametrization in generalization of neural networks. In *ICLR*.

[41] Neyshabur, B.; Tomioka, R.; and Srebro, N. 2014. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614* .

[42] Novak, R.; Bahri, Y.; Abolafia, D. A.; Pennington, J.; and Sohl-Dickstein, J. 2018. Sensitivity and Generalization in Neural Networks: an Empirical Study. In *ICLR*.

[43] Olah, C.; Mordvintsev, A.; and Schubert, L. 2017. Feature visualization. *Distill* 2(11): e7.

[44] Olah, C.; Satyanarayan, A.; Johnson, I.; Carter, S.; Schubert, L.; Ye, K.; and Mordvintsev, A. 2018. The building blocks of interpretability. *Distill* 3(3): e10.

[45] Poggio, T.; Kawaguchi, K.; Liao, Q.; Miranda, B.; Rosasco, L.; Boix, X.; Hidary, J.; and Mhaskar, H. 2017. Theory of Deep Learning III: explaining the non-overfitting puzzle. *arXiv preprint arXiv:1801.00173* .

[46] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV* .

[47] Sainath, T. N.; Kingsbury, B.; Sindhwani, V.; Arisoy, E.; and Ramabhadran, B. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6655–6659. IEEE.

[48] Schuster, A. 2008. Robust artificial neural network architectures. *International Journal of Computational Intelligence* 4(2): 88–94.

[49] Srinivas, S.; and Babu, R. V. 2015. Data-free parameter pruning for deep neural networks. In *BMVC*.

[50] Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* .

[51] Strehler, B. L.; and Freeman, M. R. 1980. Randomness, redundancy and repair: roles and relevance to biological aging. *Mechanisms of Ageing and Development* 14(1-2): 15–38.

[52] Suzuki, T.; Abe, H.; and Nishimura, T. 2020. Compression based bound for non-compressed network: unified generalization error analysis of large compressible deep neural network. In *ICLR*.

[53] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

[54] Tung, F.; and Mori, G. 2018. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7873–7882.

[55] Woodworth, B.; Gunasekar, S.; Lee, J.; Soudry, D.; and Srebro, N. 2019. Kernel and Deep Regimes in Overparametrized Models. *arXiv preprint arXiv:1906.05827* .

[56] Xiong, H.; Huang, L.; Yu, M.; Liu, L.; Zhu, F.; and Shao, L. 2020. On the Number of Linear Regions of Convolutional Neural Networks. *arXiv preprint arXiv:2006.00978* .

[57] Yamins, D. L.; and DiCarlo, J. J. 2016. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience* 19(3): 356–365.

[58] Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.

[59] Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*.

[60] Zhang, C.; Liao, Q.; Rakhlin, A.; Sridharan, K.; Miranda, B.; Golowich, N.; and Poggio, T. 2017. Theory of Deep Learning III: Generalization Properties of SGD. *CBMM Memo* .

[61] Zhou, B.; Sun, Y.; Bau, D.; and Torralba, A. 2018. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891* .

[62] Zhou, W.; Veitch, V.; Austern, M.; Adams, R. P.; and Orbanz, P. 2019. Non-vacuous generalization bounds at the imagenet scale: a PAC-bayesian compression approach. In *ICLR*.

A A Wider Network without Frivolous Units can Compute the Same Function as a Thin Network (Section 3)

We demonstrate in section 5 that as we increase the width of a network, frivolous units tend to grow at a rate which outpaces the growth of the network overall. One may ask whether this is a surprising result or if frivolous units must necessarily emerge if a wider network computes a similar function to a thin one. As we show here, a wider network need not have frivolous units in order to compute a similar function to a thinner one.

For a wider network, there can be different ways to compute the same function as a thinner network which require a different minimum number of units. For example, consider a dataset of $N+1$ dimensional datapoints in which for each, the parity of the last N dimensions is equal to the value of the first dimension. Then suppose that each point were associated with a label that was equal to the identity of the first component or equivalently, the parity of the final N. If so, this labeling function could be computed equally well by evaluating the identity of the first dimension or the parity of the last N dimensions. For networks with a single hidden layer, the identity method of computing this function would require only one unit while the parity method could be computed with 2^N hidden units (each representing a miniterm that detects one of the possible inputs, as shown in [7]). Thus, a wide network with 2^N units could correctly label the data with all units being equally non-frivolous, but also with a single unit capturing the first dimension and the rest of the units being frivolous.

Another reason why a wider network computing a similar function to a thin network may not develop disporoportionately more frivolous units is because of capacity constraints at the weight level as opposed to the unit level. The success of weight-pruning as a common regularization and compression method (*e.g.* [15, 35, 14, 54, 33]) suggests that complexity constraints can and do appear at this level in addition to the unit level.

B Prunability and Redundancy in Untrained Networks (Section 3)

Consider an untrained network in which the weights within each layer are initialized i.i.d. from some random distribution. We show here that for large datasets, as the model width is increased, the number of prunable units and redundant units will each increase proportionally to the width in expectation over the initialization. In other words, the expectation of g as defined in section 5 will be 2 for untrained networks.

Prunability: Given any fixed data distribution, the output units of a network will have a distribution of input values determined by the network's weights. By symmetry, in expectation over the initialization of the network, each unit will contribute equally the the variance of each output unit's activation. Then also by the symmetry and and the fact that each layer influences the output layer only by feed-forward action, for a layer of n units, each will in expectation be responsible for $1/n$'th of the variance for each output unit. Therefore, randomly ablating a proportion p of the units in any layer will, in expectation, reduce the variance of each output unit by a factor of p regardless of the layer's size. So robustness to the random ablation of a given *proportion* of units will be constant regardless of width for untrained networks (*ie.* $\mathbb{E}(g) = 2$).

Redundancy: If a large dataset of points is used, because the activations of each unit will be uncorrelated due to the randomness of the network's initialization. So in expectation, redundancy will be proportional to a network's size (*ie.* $\mathbb{E}(g) = 2$).

C Activational Low-Rank Factorization (Section 3)

Consider an $m \times n$ weight matrix W for a fully-connected layer. For a dataset of d examples, let A be the $d \times m$ matrix whose rows give the activations of layer L_i for each example. If so, then the inputs to layer L_{i+1} will be given by the matrix product AW.

The goal of finding a low rank refactorization of L_i based on activations is to utilize a basis of $m' < m$ units and a refactored $m' \times n$ weight matrix W' such that if A' is the $d \times m'$ matrix giving the activation of the basis, then $A'W' \approx AW$. For any basis of m' units in L_i, in order to achieve $A'W' \approx AW$, both sides can be multiplied by the left inverse of A' denoted as $A_L^{'-1}$ to find the optimal $W' = A_L^{'-1}AW$. In order to approximate the number of units needed for such as basis, we use principal component analysis on A. By analyzing the eigenvalues of the covariance matrix of A, we calculate the minimal number of components needed to reconstruct A from A' with a given error tolerance on the $L2$ distance.

D Methodology (Section 4)

Network Implementations

ResNet18s (ImageNet): We use off-the-shelf models and the established training procedure from He et al. [24]. They consisted of an initial convolution and batch normalization followed by four building blocks (v1) layers, each with two blocks and a fully connected layer leading to a softmax output. All kernel sizes in the initial layers and block layers were of size 7×7 and stride 2. All activations were ReLU. In the 1x-sized model, the convolutions in the initial and block layers used 64, 64, 128, and 256 filters respectively. After Glorot initialization [18], we trained them for 90 epochs with a default batch size of 256 and an initial default learning rate of 1 which decayed by a factor of 10 at epochs 30, 60, and 80. Optimization was done with stochastic gradient descent using 0.9 momentum. We used batch normalization, 0.0001 weight decay, and data augmentation with random cropping and horizontal flipping. Results were generated using the ImageNet validation set of 50,000 images.

Inception-v3s (ImageNet): We used off-the-shelf models and the established training procedure from Szegedy et al. [53] following the established training procedure. For the sake of brevity, we will omit architectural details here. After using a truncated normal initialization with $\sigma = 0.1$, we trained these networks with a default batch size of 256 and initial default learning rate of 1 with an exponential decay of 4% every 8 epochs. Training was run for 90 epochs on ImageNet using the RMSProp optimizer. We used a weight decay of 0.00004, batch normalization using 0.9997 decay on the mean and an epsilon of 0.001 to avoid dividing by zero, and augmentation using random cropping and horizontal flipping. Due to hardware constraints, we were not able to train 2x and 4x variants of the network (we used a dgx1 with 8x NVIDIA V100 GPUs 32GB). Instead, we trained the 1/4x-1x sizes along with versions of the network with 1/4x-4x sizes for the "mixed 2: 35 x 35 x 288" layer only. We generate results using the ImageNet validation set of 50,000 images.

AlexNet (CIFAR-10): We use a scaled-down version of the network developed by Krizhevsky et al. [29] similar to the one used by Zhang et al. [59] for CIFAR-10. The network consisted of 4 hidden layers: two convolutional layers with 96 and 256 kernels respectively and two dense layers with 384, and 192 units in the 1x model size. In each convolutional layer, 5×5 filters with stride 1 were applied, followed by max-pooling with a 3×3 kernel and stride 2. Local response normalization with a radius of 2, alpha $= 2 * 10^{-5}$, beta $= 0.75$ and bias $= 1.0$ was applied after each pooling. Each layer contained bias terms, and all activations were ReLU. We used Glorot initialization [18] by default and trained these networks with early stopping based on maximum performance on the $5,000$ image CIFAR-10 validation set. Weights were optimized with stochastic gradient descent using 0.9 momentum with an initial learning rate of 0.01, exponentially decaying by 5% every epoch. By default, we used a batch size of 128, and no explicit regularizers. We generate results using the 10,000 image testing set.

ResNet56 (CIFAR-10): These networks were used off-the-shelf from He et al. [24]. They consisted of an initial convolution and batch normalization followed by three building block (v1) layers, each with nine blocks, and a fully connected layer leading to a softmax output. In the 1x-sized model, the convolutions in the initial and block layers used 16, 16, 32, 64, and 128 kernels respectively. Kernels in the initial layers and block layers were of size 3×3 and stride 1. All activations were ReLU. After Glorot initialization [18], we trained them for 182 epochs with a default batch size of 128 and an initial default learning rate of 1 which decayed by a factor of 10 at epochs 91 and 136. Optimization was done with stochastic gradient descent using 0.9 momentum. We used batch normalization, 0.0002 weight decay, and data augmentation with random cropping and flipping (except for our variants trained on randomly labeled data). We generate results using the 10,000 image testing set.

MLPs (synthetic uncorrelated data): We use simple multilayer perceptrons with either 10 or $10,000$ dimensional inputs and binary output. They contained a single hidden layer with 128 units for the 1x model size and a bias unit. All hidden units were ReLU activated. Weights were initialized using a Gaussian distribution with default standard deviation of 0.01. Each was trained by default using stochastic gradient descent with momentum of 0.9 for 50 epochs on $1,000$ examples produced by a 1/4x sized teacher network with the same architecture which was verified to produce each output for between 40% and 60% of random inputs. Results are generated using a $1,000$ image testing set.

Number of Parameters

In Fig. E1, we show the number of trainable parameters for each network, showing that they increase exponentially with the model size factor.

Samplings and Replicates

Due to the number of units in the models and the size of the datasets, analyzing all activations for convolutional filters was intractable in experiments involving redundancy. Instead of an exhaustive sampling, we based our measures on a sampling of spatial locations for each filter capped at 50,000 across the testing set. We ran three independent samplings using this method and found that the variance between them is negligible for all networks.

For all non-ImageNet networks, we conduct three trials with independently trained networks, and plot error bars giving the standard deviations. For Redundancy in ImageNet, error bars reflect standard deviations between samplings of units. For prunability in ImageNet, points reflect single trials.

E Additional Results (Section 5)

Non-prunable and non-redundant units increase in quantity but with $g < 2$

In Fig. E2 and Fig. E3, we display plots analogous to those presented in the main paper in Fig. 3 and Fig. 4 but plot trends in non-prunable and non-redundant units. As is mirrored by the fact that frivolous units tend to more-than-double, their complements tend to less-than-double when model width doubles. While the gain values are very small for some MLPs, these non-frivolous units increase in quantity for all networks. A compelling direction for future work will be to analyze prunability and redundancy together or alternative types of capacity constraints to see how little the number of non-frivolous components of deep networks can be shown to increase.

Layerwise analysis for Inception-v3 models

Fig. E4 and Fig. E5 add to the analysis presented in the main paper in Fig. 3c-d. We show that individual layers in our Inception-v3s trained in ImageNet display unique trends and that even when frivolity increases with $g > 2$ for a network as a whole, it does not imply that it does so for all layers. We find that when a single layer is varied keeping others constant, frivolity only increases for that individual layer.

Weight initialization and input dimensionality experiments

In addition to using Gaussian initializations, we also test AlexNets with Uniform Glorot, He, and LeCun initializations to better understand the role of initialization. Glorot initialization [18] assigns weights with variance $\sigma^2 = 2/(\text{fan\_in} + \text{fan\_out})$, LeCun initialization with $\sigma^2 = 1/\text{fan\_in}$ [31], and He initialization with $\sigma^2 = 2/\text{fan\_in}$ [23]. Fig. E6 shows that their results are very similar to those of the networks with Gaussian distributed initial weights in Fig. 4c, suggesting that the initialization distribution matters little compared to the initialization variance.

In Fig. E8 and Fig. E7, we show the results of altering variance for Gaussian initializations in the MLPs trained on uncorrelated data. For the datasets with 10-dimensional inputs, results were fairly consistent under alternative initializations. For 10,000 dimensional inputs, however, the amount of redundancy developed was sensitive to initialization and was the highest for the smallest initializations. This demonstrates an interactive effect between data and initialization regarding redundancy.

Additional experiments with optimizers, learning rates, batch sizes, and number of training epochs

Optimizers. We test the effect of different optimizers for the MLPs fitting $10,000$ dimensional datapoints. Fig. E9 shows that while prunability is fairly stable under alternate choices of the optimizer, redundancy develops to varying degrees with momentumless stochastic gradient descent resulting in the most.

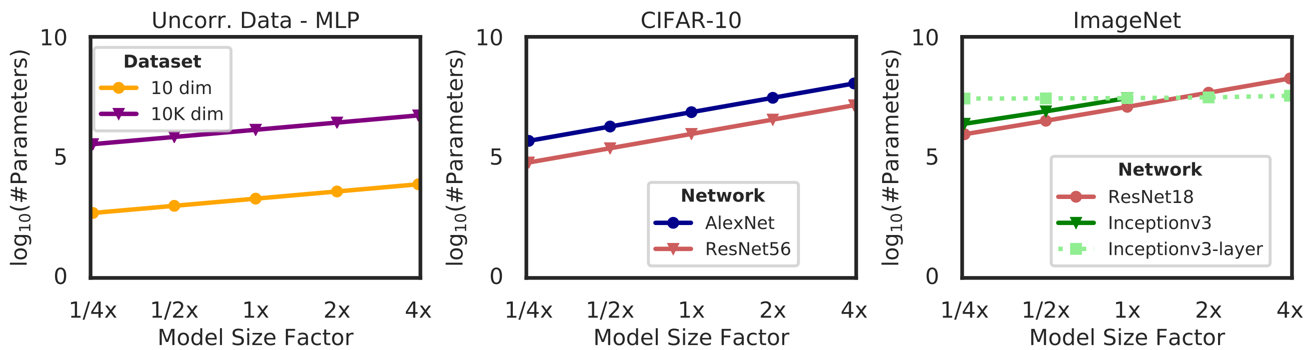
Batch Size and Learning Rate. Batch size and learning rate are very commonly varied in practice when training DNNs. To investigate its effects, we vary them by a constant factor, which we denote as k. In Fig. E10, we report the results for ResNet18 ImageNet and in Fig. E11 for ResNet56 and AlexNet in CIFAR-10. In all tested cases, the batch size and learning rate had no discernible effects on the trends in prunability and redundancy.

Number of Training Epochs. We find that throughout experiments, trends were robust to changes in the number of training epochs. In Fig. E12, we show that trends for prunable and redundant units are invariant to the amount of training time after convergence in ResNet18s.

Interpretability Experiments with Lucid

As described in the main paper, we visualize units in 1/4x and 4x ResNet18s and 1/4x and 1x Inception-v3s to test the hypothesis that the visualizations of units in the thinner networks would be more interpretable because of fewer frivolous units.

The images were created using Lucid [43] and were made to optimize images in order to maximize the post-ReLU activation of a set of 8 randomly chosen units at the end of each ResNet18 block and Inception-v3 module. Using Lucid, we parameterized



(a) (b) (c)

Figure E1: **Parameters:** (a) Multilayer perceptrons, (b) AlexNets and ResNet56s, (c) ResNet18s, Inception-v3s, and Inception-v3s with a single layer varied. The log number of trainable parameters at each model size.

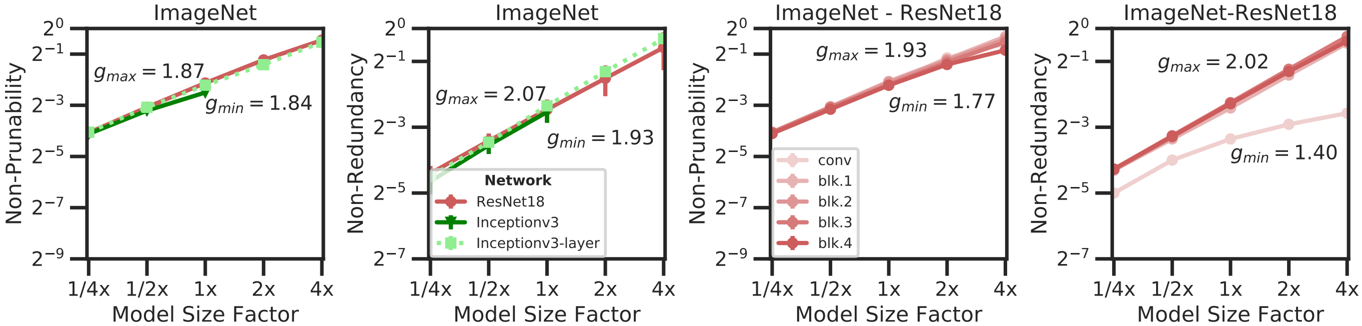
each image in decorrelated Fourier space, and used random padding, jittering, scaling, and rotation during optimization over 256 steps. Example images and their Fourier transforms from the first and final blocks of 1/4x and 4x ResNet18s are shown in Fig.E13.

For the images (both in pixel space and frequency space), our measure of interpretability was an image's total variation which measures the total amount of difference between adjacent pixels across the image. We calculated it by adding together the sum of the absolute valued vertical and horizontal gradients for each channel of the image:

$$\text{tvar}(V) = \sum_{c=1}^{3} \left| \sum_{ij} \nabla_{vert}(V_c) \right| + \left| \sum_{ij} \nabla_{horiz}(V_c) \right|, \quad (1)$$

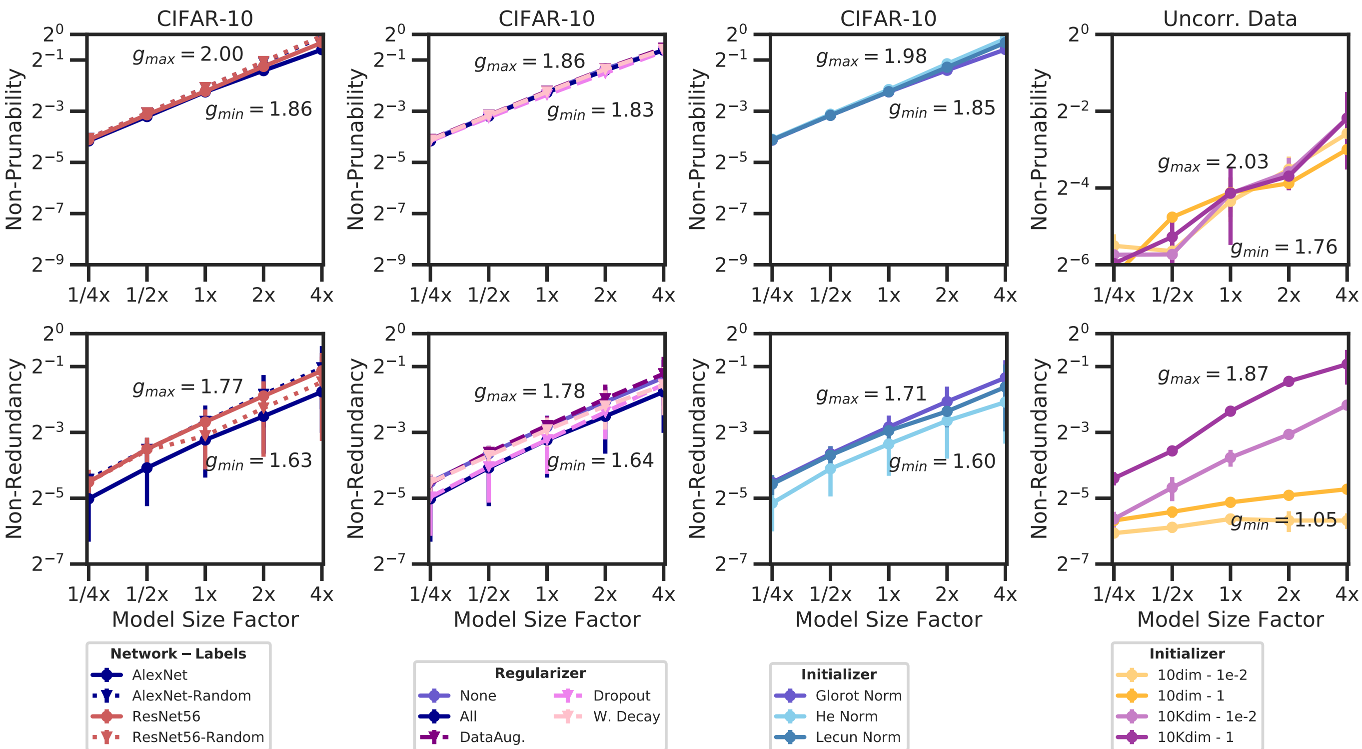
where $| \cdot |$ represents the absolute value.

When comparing the filter visualizations either in pixel or frequency space from homologous layers in the thin and wide networks, we used a one-sided rank-based permutation test. We calculated the ranks for the total variations of the visualization images from the thin network with respect to the images from both the thin and wide networks and used their sum as the test statistic. Over $100,000$ random shufflings of the ranks, we reported the one-sided p value as the proportion of samplings whose sum was less than or equal to this test statistic. See Table 2 for these and effect sizes. Note that a small p value indicates that the visualizations of units in the thinner network tend to have a lower total variation (*ie.* they were more interpretable). While these results show more evidence for the hypothesis that thinner networks are more interpretable than against it, results vary across layers, and there are no consistent trends.



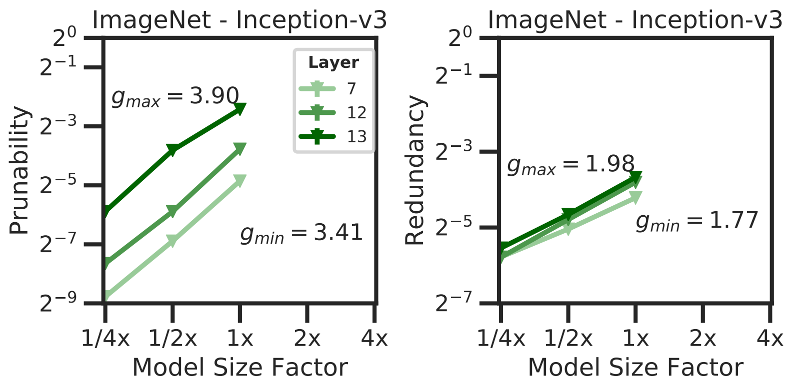
(a) (b) (c) (d)

Figure E2: **Non-frivolous units emerge in ResNet18s and Inception-v3s with smaller gains than frivolous ones.** (a) Non-prunability. (b) Non-redundancy. (c) Non-prunability layerwise for ResNet18s. (d) Non-redundancy layerwise for ResNet18s. The gain, g, gives the increase when the network size is doubled. Max and min g values are given.



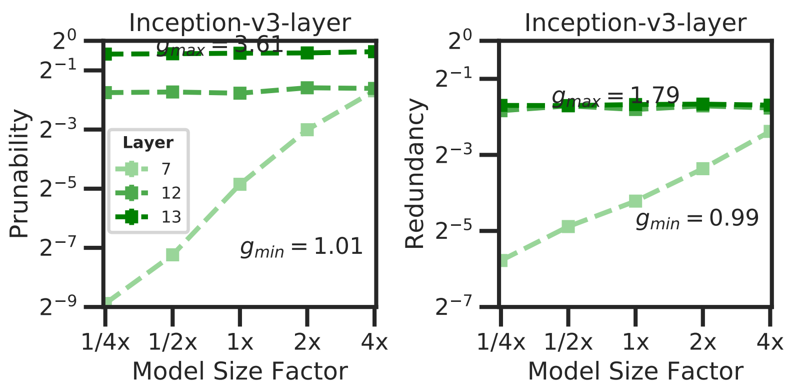
(a) (b) (c) (d)

Figure E3: **Non-frivolous units emerge with smaller gains than frivolous ones for different architectures, regularizers, initializers and datasets.** (a) AlexNets and ResNet56s trained with and without random labels (CIFAR-10). (b) AlexNets trained with and without regularization (CIFAR-10). (c) AlexNets trained with various initializations (CIFAR-10). (d) MLPs with different input sizes and initializations (synthetic data). Max and min gain factors are given for each plot.



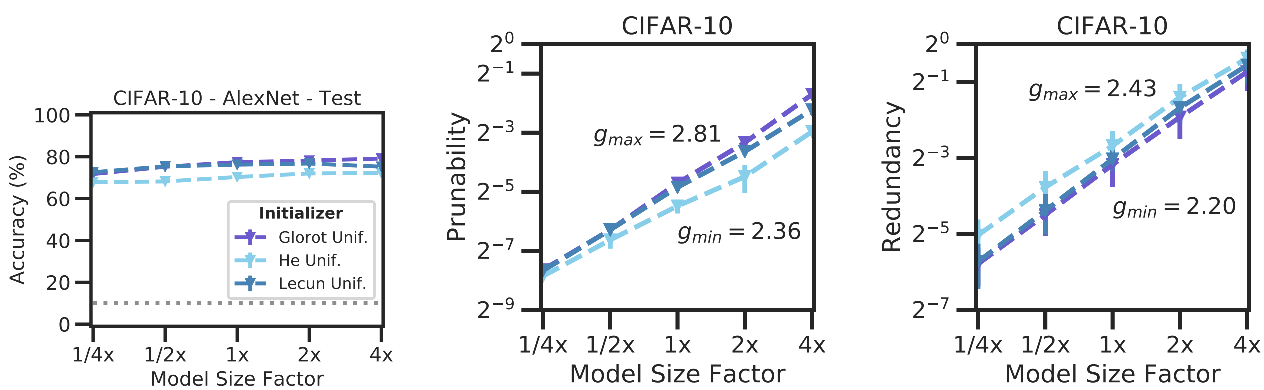
(a) (b)

Figure E4: **Inception-v3 prunability and redundancy layerwise (ImageNet):** Trends in (a) prunability and (b) redundancy among the final 35×35 (layer 7), 17×17 (layer 12), and 8×8 (layer 13) blocks within Inception-v3 in ImageNet. Gains for the individual layers are similar to the network as a whole.



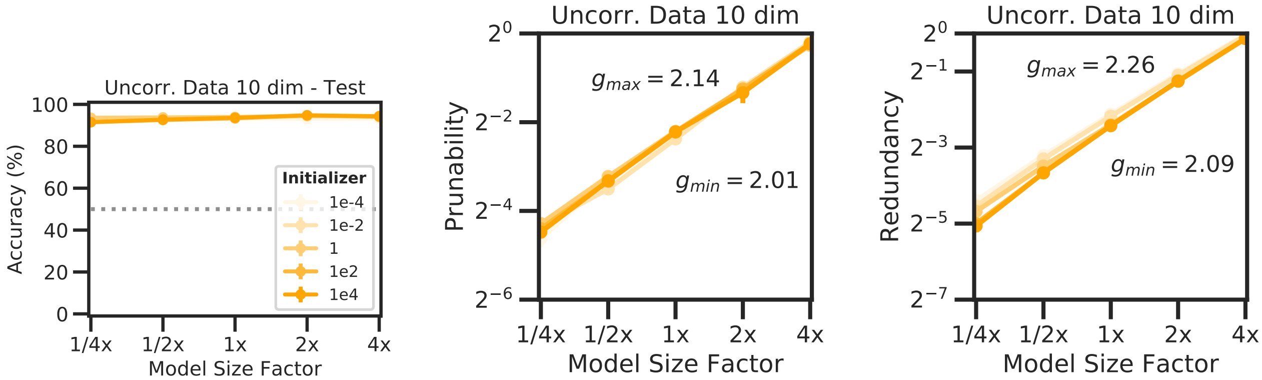
(a) (b)

Figure E5: **Inception-v3 single layer prunability and redundancy layerwise (ImageNet):** Trends in (a) prunability, and (b) redundancy among the final 35×35 (layer 7), 17×17 (layer 12), and 8×8 (layer 13) blocks within Inception-v3s in ImageNet as only the final 35×35 layer is varied in size. Varying the size of a single layer has no effect on the unit prunability and redundancy of other layers.



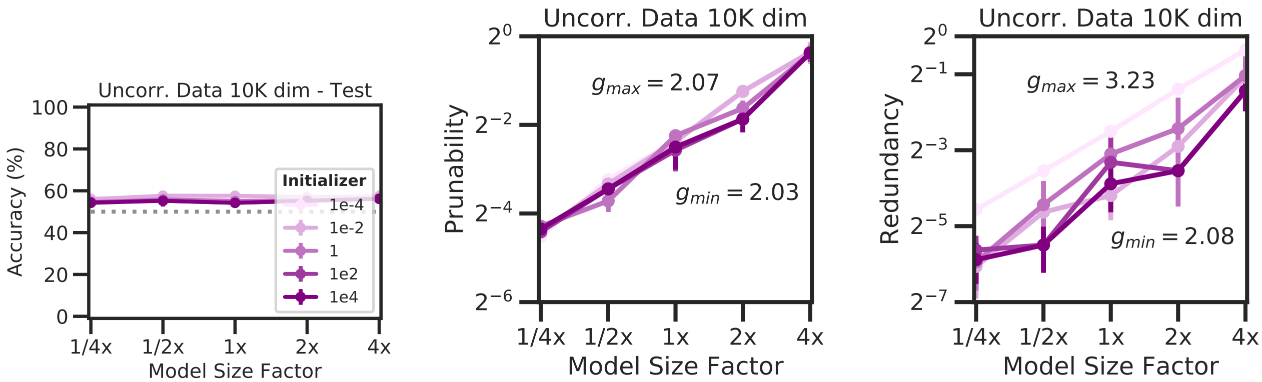
(a) (b) (c)

Figure E6: **AlexNet prunability and redundancy trends with uniform initializations (CIFAR-10).** Trends in (a) accuracy, (b) prunability, and (c) redundancy with Glorot, He, and LeCun uniform initializations across size factors for AlexNets. Trends resemble those for Gaussian initializations.



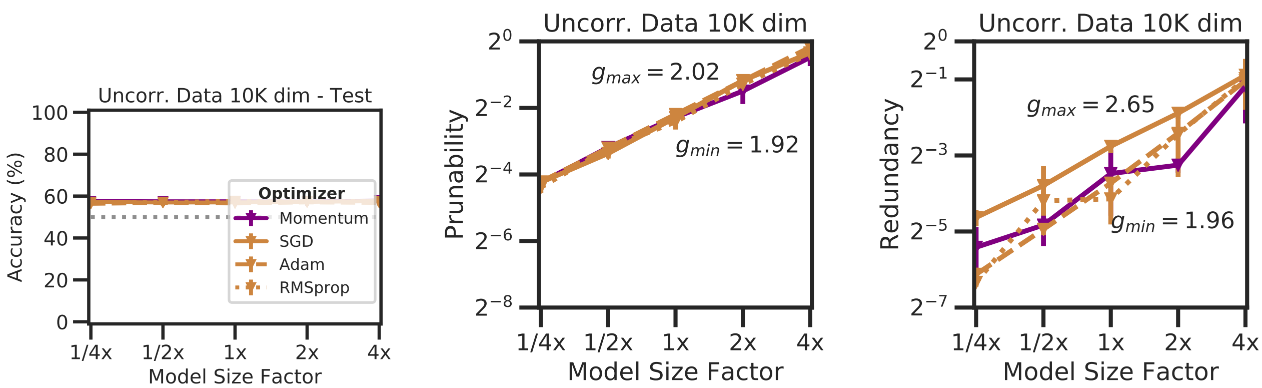
(a) (b) (c)

Figure E7: **Frivolous units are not sensitive to initialization variance in MLPs trained on 10 dimensional data.** Trends in (a) accuracy, (b) prunability, and (c) redundancy with multiple initialization variances across size factors for MLPs trained on synthetic uncorrelated data. The legend gives standard deviations for the Gaussian initialization.



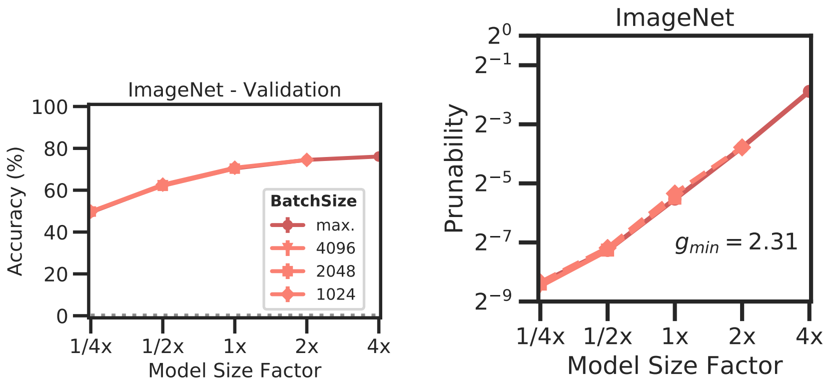
(a) (b) (c)

Figure E8: **Redundancy is sensitive to initialization variance in MLPs trained on 10,000 dimensional data.** Trends in (a) accuracy, (b) prunability, and (c) redundancy with multiple initialization variances across size factors for MLPs trained on synthetic uncorrelated data. The legend gives standard deviations for the Gaussian initialization. Redundancy is sensitive to initialization variance with the smallest variances resulting in the greatest amounts of redundancy.



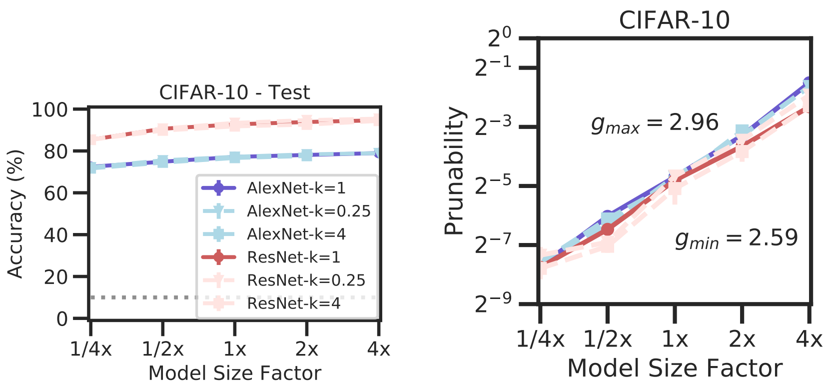
(a) (b) (c)

Figure E9: **Optimizers influence redundancy in MLPs trained on 10,000 dimensional uncorrelated data.** Trends in (a) accuracy, (b) prunability, and (c) redundancy with momentum, stochastic gradient descent, and Adam optimizers across size factors. Momentum, marked in purple, was used for all other experiments with these MLPs.



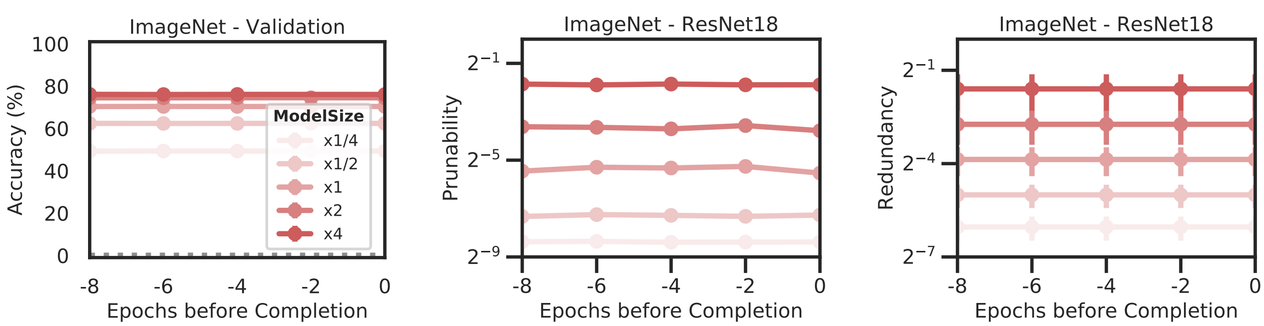
(a) (b) (c)

Figure E10: **Trends in accuracy and frivolous units do not depend on learning rate and batch size factor in ResNet18s trained in ImageNet.** Trends in (a) accuracy, (b) prunability, and (c) redundancy across model sizes. We vary a constant factor k from $1/4$ to 4 as a multiplier for the batch sizes and learning rates. "Max" refers to the maximum batch size that could be used for training a model given available hardware (dgx1 with 8x NVIDIA V100 GPUs 32GB).



(a) (b) (c)

Figure E11: **Trends in accuracy and frivolous units do not depend on learning rate and batch size in networks trained on CIFAR-10.** Trends in (a) accuracy, (b) prunability, and (c) redundancy across model sizes for AlexNets and ResNet56s. We vary a constant factor k from $1/4$ to 4 as a multiplier for the batch sizes and learning rates. 4x AlexNets with $k = 4$ were not trained due to hardware restrictions.



(a) (b) (c)

Figure E12: **Prunability and redundancy are stable in ResNet18s under different numbers of training epochs past convergence.** Trends in (a) accuracy, (b) prunability, and (c) redundancy across training epochs after convergence.

(a)

(b)

(c)

(d)

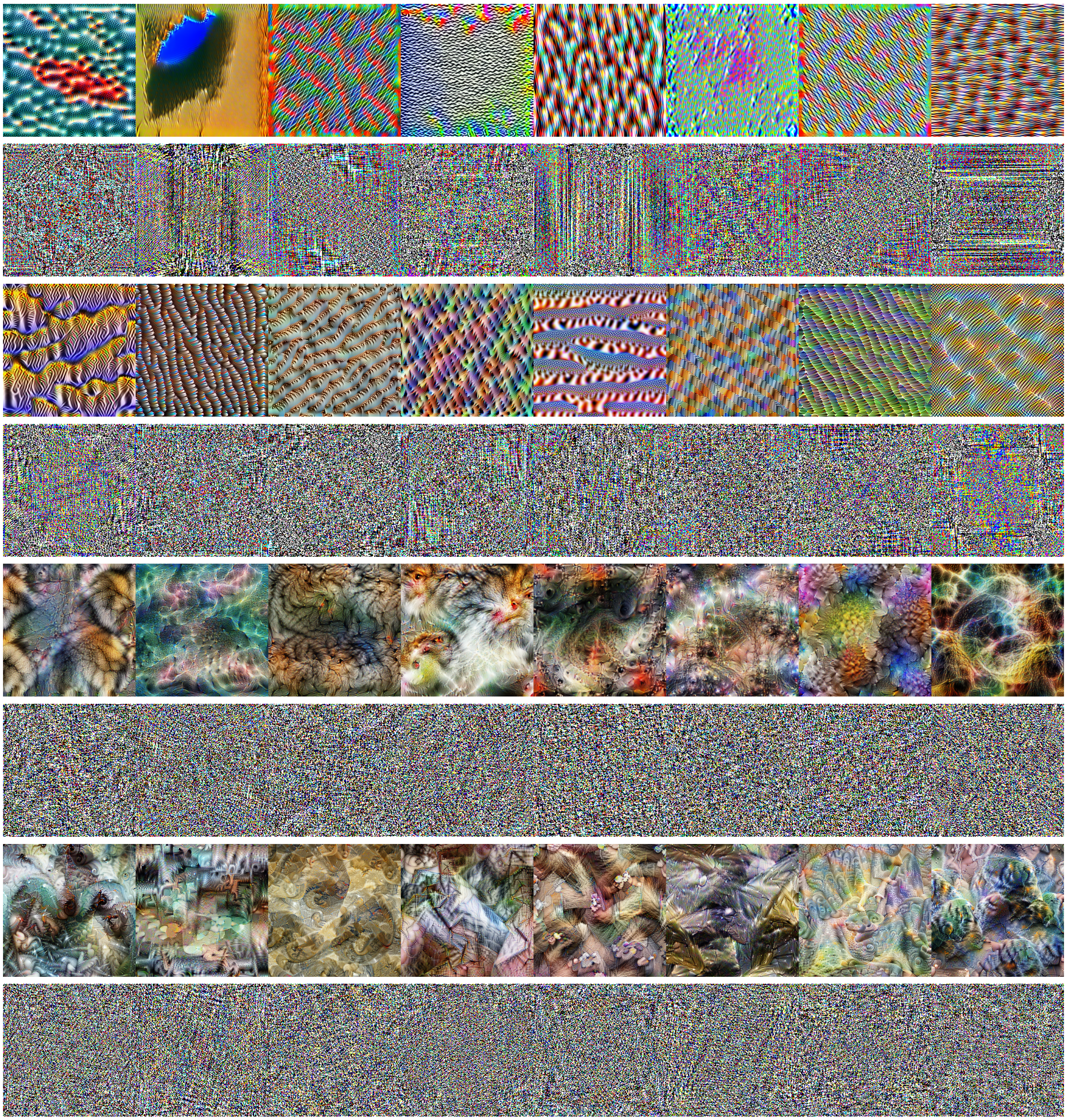


Figure E13: **Visualizations in pixel and Fourier space of 8 units from each of the first block (a,b) and final block (c,d) of the 1/4x (a,c) and 4x (b,d) ResNet18s.** The total variations of these images were used for hypothesis testing via a rank-based permutation test..

ResNet18	Block 1	Block 2	Block 3	Block 4
Total Var p (Pixel)	0.751	0.835	0.563	**0.001**
Mean Effect (Pixel)	1.027	1.171	0.987	0.883
Total Var p (Freq)	0.361	0.750	0.25	**0.001**
Mean Effect (Freq)	0.970	1.127	0.967	0.863

Inception-v3	Conv4	35x35x256a	35x35x288a	35x35x288b	17x17x768a	17x17x768b
Total Var p (Pixel)	**0.982**	0.566	0.080	**0.018**	**0.990**	0.140
Mean Effect (Pixel)	1.237	1.016	0.820	0.856	1.275	0.883
Total Var p (Freq)	**0.989**	0.263	**0.019**	0.065	**0.998**	0.118
Mean Effect (Freq)	1.406	0.983	0.831	0.875	1.324	0.920
	17x17x768c	17x17x768d	17x17x768e	17x17x1280a	8x8x2048a	8x8x2048b
Total Var p (Pixel)	0.097	0.116	0.520	**0.010**	**0.005**	**0.004**
Mean Effect (Pixel)	0.900	0.898	0.977	0.858	0.884	0.814
Total Var p (Freq)	0.194	0.096	0.714	0.443	0.288	0.138
Mean Effect (Freq)	0.943	0.926	1.018	0.943	0.973	0.917

Table 2: **One-sided p values and mean effect sizes testing for low total variation among visualizations of thin networks compared to wide ones.** Each p value is based on a rank-based permutation test on 8 samples from each network and 100,000 samples. Results are shown both for these images in pixel space and in frequency space. Here, a small p value indicates that the visualizations of units in the thinner network has a lower total variation (i.e. they were more interpretable). P values below 0.025 or above 0.975 are shown in bold. Each effect size gives the mean total variation for the thin networks divided by the same for the wide networks. A small mean effect means that the thin network's visualizations had a smaller total variation (i.e. they were more interpretable) on average.