ON EMERGENCE OF ACTIVATION SPARSITY IN TRAINED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper reveals a curious observation that modern large-scale machine learning models with Transformer architectures have *sparse* activation maps. By activation map we refer to the intermediate output of the multi-layer perceptrons (MLPs) after a ReLU activation function, and by "sparse" we mean that on average very few entries (e.g., 3.0% for T5-Base and 6.3% for ViT-B16) are nonzero for each input to MLP. Moreover, larger Transformers with more layers and wider MLP hidden dimensions are sparser as measured by the percentage of nonzero entries. Through extensive experiments we demonstrate that the emergence of sparsity is a prevalent phenomenon that occurs for both natural language processing and vision tasks, on both training and evaluation data, for Transformers of various configurations, at layers of all depth levels. We discuss how sparsity immediately implies a way to significantly reduce the FLOP count and improve efficiency for Transformers. Moreover, we demonstrate perhaps surprisingly that enforcing an even sparser activation via Top-k thresholding with a small value of k brings a collection of desired but missing properties for Transformers, namely less sensitivity to noisy training data, more robustness to input corruptions, and better calibration for their prediction confidence.

1 INTRODUCTION

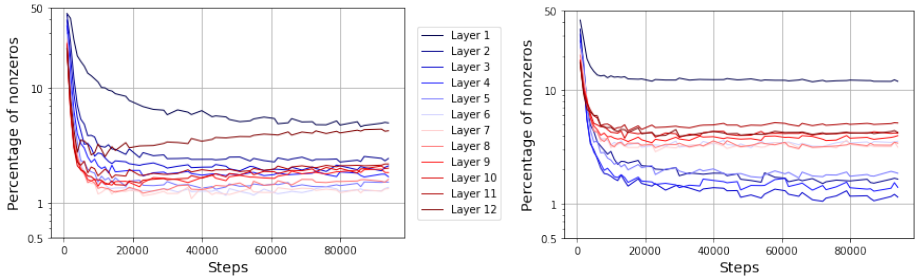
The great success of modern machine learning for tasks in computer vision, natural language processing, game playing and beyond is driven primarily by the computational model known as deep neural networks (DNNs) (LeCun et al., 2015). With inspirations drawn from biological intelligent systems, DNNs are massive systems of distributed computational nodes (a.k.a. neurons) with learned inter-connections, which possess the capacity of accomplishing complex real-world tasks.

Although originally motivated from biological brains, there are differences at very fundamental levels on how DNNs work compared to biological neural networks. One of such differences is in the sparsity of neural activities. Evidence from neuroscience suggests that neural activity in biological brains is *sparse*, namely, only a small percentage of all neurons fire at each time (Ahmed et al., 2020; Barth & Poulet, 2012; Kerr et al., 2005; Poo & Isaacson, 2009). Sparse firing suggests that despite having billions of neurons, only a small fraction of the brain participates in computation at each time, which may explain why brains can sustain at a very low energy cost. In contrast, learning and inference with DNNs rely primarily on dense computations where all neurons are involved for any input. In fact, modern computational hardware for deep neural networks, such as GPUs and TPUs, are designed to facilitate massive scale dense computations. Even with such dedicated hardware, DNNs are still notoriously resource-demanding to train and deploy. Aside from computation efficiency, DNNs also lag far behind biological brains in terms of robustness to input perturbation, error correction for erroneous training labels, confidence calibration for the predictions, etc.

1.1 AN INTRIGUING OBSERVATION: ACTIVATIONS ARE SPARSE IN TRAINED TRANSFORMERS

This paper reveals a surprising observation that despite performing dense computations, DNNs produce very *sparse* activation in their intermediate layers once trained[1]. Specifically, we study

[1]This implies, as we explain in detail later, that a lot of the computations are spent in vain with multiplying a value by zero.



(a) T5 Encoder (b) T5 Decoder

Figure 1: Percentage of nonzero entries (y-axis, log scale) in the activation map as a function of number of training steps (x-axis) for a T5-Base model trained with the span corruption objective on the C4 dataset. **Left:** layers (from shallow to deep) of the encoder. **Right:** layers of the decoder.

Transformer (Vaswani et al., 2017), a DNN model architecture that has become a workhorse for modern applications. Transformers are constructed by interleaving a self-attention module and a multi-layer perceptrons (MLPs) of depth 2, and the focus of this paper is on the activation map of the *first* MLP layer. Figure 1 shows the sparsity of the activation maps, measured by the percentage of nonzeros, in all MLP layers of a T5-Base model (Raffel et al., 2020). We see that the percentage of nonzero entries is around 50% at initialization, which is expected: randomly initialized weights produce roughly equal numbers of positive and negative entries in the pre-activation map, resulting in about 50 % non-zeros after the ReLU. However, at the end of training the percentage of nonzero entries reduces drastically: the average value across all encoder-decoder layers is 2.7% with the largest one being 12.0% and the smallest one being only 1.1%. The emergence of sparse activation in Transformers bears a similarity to the sparsity of neural activities in biological brains, revealing an interesting connection between artificial and biological networks. Moreover, unlike classical sparse methods where such a connection is established via *explicit* sparse regularization (Olshausen & Field, 1996), the sparsity observed in Transformers is emergent without any explicit design.

1.2 Prevalence, Benefits, and Causes of Sparsity

This paper studies the aforementioned phenomenon of sparse activation in trained Transformers, with a focus on the following two questions. First, is the phenomenon shown in Figure 1 a corner case or does it occur broadly? Second, why should we care about the sparsity in DNNs, other than the appeal of its similarity to biological brains? Our main results along these two lines are summarized below.

1. **Sparsity is a prevalent phenomenon.** We show in Section 2 that the emergence of sparse activation reported in Figure 1 is not an isolated and cherry-picked case. Rather, sparsity is prevalent, and occurs broadly in Transformer models: it emerges in all layers of a Transformer, for Transformers trained on both vision and natural language data, for Transformers of various configurations, and for activation maps computed on both train and test data, etc. Moreover, through controlled experiments on the width and depth of Transformers, we reveal that larger models are sparser, as measured by percentage of nonzero entries. We also show in the Appendix B.1 that sparsity emerges in MLP-Mixers, vanilla MLPs, and to some extent Residual Networks.

2. **Sparsity improves efficiency.** Sparsity of activation map implies that a lot of the computations being carried out in Transformer training and inference are wasted: they are doing nothing but multiplying values by zero. This immediately suggests that FLOPs can be reduced by avoiding all such computations, which we discuss in Section 3.1. Importantly, our method does not affect model output or their performance, in contrast to existing work that uses sparsity to improve efficiency which is built upon trial-and-error without ensuring their correctness (Hoefler et al., 2021). To explicitly control the sparsity level, we further introduce *Top-k Transformer* in Section 3.2, a simple modification of Transformers where a Top-k thresholding is applied to the activation maps[2]. We provide experimental evidence that reduction in FLOPs via sparsity brings wall time benefits for unbatched decoding on TPUv4 with Top-k T5 models with 11B parameters.

[2]The approach is previously adopted in ConvNets for improving model robustness (Ahmad & Scheinkman, 2019), and more recently in Gupta et al. (2021) for improving memory efficiency of Transformers.

3. **Sparsity improves robustness and calibration.** We further show in Section 3.3 that enforcing explicit sparsity via Top-k Transformers improves model performance in terms of less sensitivity to noisy training data, less sensitivity to input corruptions, and better confidence calibration.

In addition, we provide a study on the causes of sparsity in the Appendix D, showing that sparsity is likely not an artifact of the training data, and may be attributed to the training dynamics in the optimization process.

1.3 Experimental Setup

We study the sparsity in activation maps of Transformers with two commonly used Transformer models, namely Text-to-Text Transfer Transformer (i.e., T5) and Vision Transformer (i.e., ViT).

- **T5** is an encoder-decoder model for natural language processing tasks (Raffel et al., 2020). We train T5 on the Colossal Clean Crawled Corpus (C4) using the span corruption task.
- **ViT** is an encoder model for vision tasks (Dosovitskiy et al., 2021). Unless specified otherwise, we train ViT on ImageNet-21k (Deng et al., 2009), an image classification dataset with 14M images and 21k classes. For certain cases we also use ImageNet-1k which is a subset of ImageNet-21k with 1.3M images and 1k classes.

We measure the sparsity level at the intermediate output of the two-layer MLPs in a Transformer. Recall that an MLP performs the following mapping

$$f(\boldsymbol{x}; \boldsymbol{K}, \boldsymbol{V}) \doteq \sum_{i=1}^{d_{\text{ff}}} \Big(\sigma(\langle \boldsymbol{k}_i, \boldsymbol{x}\rangle) \cdot \boldsymbol{v}_i\Big), \quad \text{or equivalently,} \quad f(\boldsymbol{x}; \boldsymbol{K}, \boldsymbol{V}) \doteq \boldsymbol{V}\sigma(\boldsymbol{K}^\top \boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^{d_{\text{model}}}$ is the input, $\boldsymbol{K} = [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{d_{\text{ff}}}] \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d_{\text{ff}}}] \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ are learnable layer parameters, and $\sigma()$ is a nonlinear activation function. We use ReLU as the activation function $\sigma()$ for both T5 and ViT[3]. A two-layer MLP may be regarded as having d_{ff} neurons in the hidden layer, where the i-th neuron performs the computation $\sigma(\langle \boldsymbol{k}_i, \boldsymbol{x}\rangle) \cdot \boldsymbol{v}_i$, and the final layer output is the sum of the output of all neurons. Each neuron is called *activated* if $\sigma(\langle \boldsymbol{k}_i, \boldsymbol{x}\rangle)$ is strictly positive. Hence, the sparsity of neuron activation can be measured by the number of nonzero entries in the feature map

$$\boldsymbol{a} \doteq \sigma(\boldsymbol{K}^\top \boldsymbol{x}) \in \mathbb{R}^{d_{\text{ff}}}. \tag{2}$$

Both T5 and ViT come with several configurations for $d_{\text{model}}, d_{\text{ff}}$, number of layers, etc. Unless specified otherwise, we will use the Base models (i.e., T5-Base and ViT-B/16) which have $d_{\text{model}} = 768$, $d_{\text{ff}} = 3072$, and 12 layers (for ViT) and 12 encoder layers $+12$ decoder layers (for T5). Training details of T5 and ViT are provided in Appendix A.

2 Prevalence of Sparsity in Learned Transformers

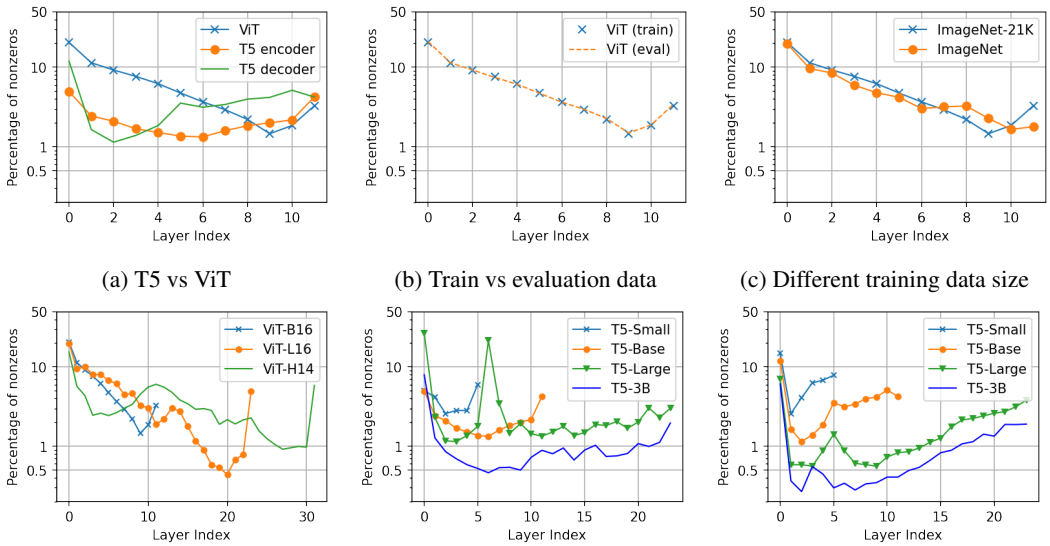
This section shows thorough experiments on commonly used Transformers that sparsity in activation maps is a prevalent phenomenon. We also show through some controlled experiments that deeper and wider Transformers tend to be sparser measured by percentage of nonzero entries in activation maps.

2.1 Sparsity is a Ubiquitous Phenomenon

We start by providing experimental evidence that the emergence of sparse activation in trained Transformers is a ubiquitous phenomenon. To this end, we plot the percentage of nonzero entries of activation maps in different Transformers, and present the results in Figure 2. These results demonstrate the following.

- *Sparsity emerges for both Vision and NLP tasks.* Figure 2a shows the percentage of nonzero entries of trained T5 and ViT models evaluated on their respective training datasets. We see that both encoder and decoder of T5, as well as the ViT, all exhibit sparsity.
- *Sparsity emerges on both training and evaluation data.* Figure 2b shows the percentage of nonzero entries in a trained T5 model measured on both the training data and the evaluation data. We see

[3] ViT originally uses GeLU as its activation function (Dosovitskiy et al., 2021). Here we switch to using ReLU as it allows us to more easily measure the sparsity level using the number of nonzero entries.



(a) T5 vs ViT (b) Train vs evaluation data (c) Different training data size

(d) Varying configuration (ViT) (e) Varying config. (T5 Encoder) (f) Varying config. (T5 Decoder)

Figure 2: Percentage of nonzero entries across different layers of trained Transformers (a) for both language data with T5 and vision data with ViT, (b) on both train and evaluation data, (c) for ViT trained on ImageNet of 21k vs 1k classes, (d) on ViT of varying configurations, and (e, f) on T5 of varying configurations. Note that the y-axis is in $\log$ scale. *Sparsity emerges in all cases.*

that the property of sparsity generalizes very well to evaluation data as the curves for training and evaluation data align very closely with each other.

- *Sparsity emerges on datasets of varying scale.* Figure 2c shows the percentage of nonzero entries in ViT trained on both ImageNet-21k and ImageNet-1k, where the former is a superset of the later with approximately $10\times$ more images and $21\times$ more classes. We see that the scale of data does not affect much of the sparsity level.
- *Sparsity emerges on Transformers of varying configurations.* Figure 2d shows the percentage of nonzero entries for ViT of varying configurations in model size. Figure 2e and 2f show the percentage of nonzero entries for encoder and decoder, respectively, of T5 with varying configurations in model size. We see that sparsity persists for all cases.
- *Sparsity emerges across all layers of a Transformer.* Finally, all plots in Figure 2 show that sparsity emerges in all layers of a Transformer. Moreover, in all cases the first few and last few layers tend to be denser than intermediate layers.

The presence of sparsity in activation maps does not rule out the possibility that a small percentage of the neurons are always activated for all inputs, whereas the rest of the neurons are never activated. To illustrate that this is not the case, we experiment with a pretrained T5 base model[4] to plot the percentage of layer inputs for which each of the d_{ff} neurons is activated when evaluated on 800 examples taken from C4 dataset with span corruption task. Note that there are $800 \times 512 = 409600$ samples as MLP activation is computed per token. The results are presented in Figure 3 with x-axis being indices of neurons in the first encoder layer of T5 sorted in descending order according to percentage of layer inputs on which they are activated. It can be seen that while a few neurons are activated for around 50% of the time, the vast majority of neurons (around 93.5%) are activated less than 10% of the time. Moreover, there are no dead neurons that are never activated, and the least activated neuron is activated for around 0.001% of the time, and 99% of neurons are activated over 1% of the time. Finally, while the

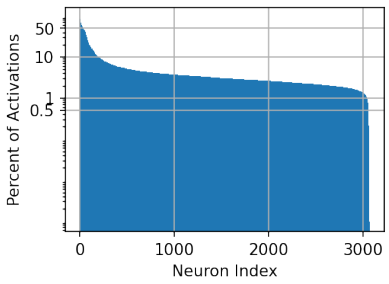
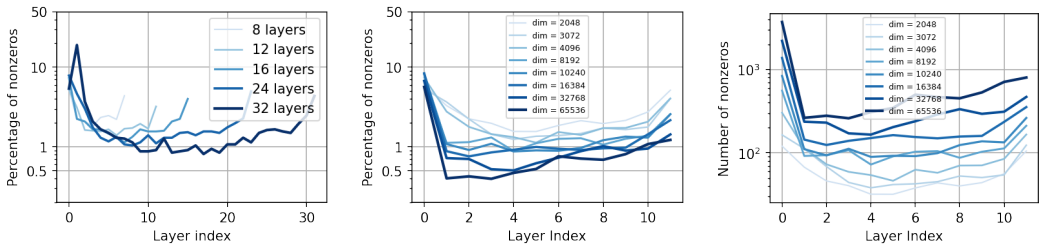


Figure 3: Percentage of times that each neuron in the first MLP layer of a trained T5 is activated on C4 dataset.

[4] https://github.com/google-research/t5x/blob/main/docs/models.md#t5-checkpoints



(a) Sparsity vs. depth (b) Sparsity (percentage) vs. width (c) Sparsity (count) vs. width

Figure 4: Activation sparsity across different encoder layers of trained T5 Transformers of (a) varying depth and (b, c) varying width. Since with varying width the dimension of activation maps also changes, we evaluate sparsity both in term of the percentage (as in (b)) and the count (as in (c)) of nonzeros. *Deeper and wider models are sparser in terms of percentage of activated neurons.*

results here are for neurons in the first MLP layer of a pretrained T5 base encoder, all other MLP layers show qualitatively similar behavior.

2.2 The Larger, the Sparser

We next examine the effect of model size on the sparsity level of activation maps. Note that Figure 2e and Figure 2f provide evidence with T5 of varying configuration that larger models tend to be sparser. Here we perform controlled experiments to examine the effect of model depth, measured by the number of Transformer layers, as well as the effect of model width, measured by the dimension of activation map of MLPs (i.e., d_{ff}), separately. Towards that, we take a standard T5 model and vary the depth and width, respectively while keeping the rest of the configuration fixed, and examine their sparsity level after training. The results are presented in Figure 4 for the encoder, whereas we omit the results for the decoder as they are qualitatively the same as those for encoder.

It can be seen from Figure 4a that deeper Transformers are arguably sparser. For example, many of the middle layers of the 32-layer model have less than 1% nonzero entries while all shallower models have more than 1% nonzero entries across all layers. For comparing networks of different widths, we measure the sparsity with the percentage and the count of nonzero entries in Figure 4b and Figure 4c, respectively. It can be seen that wider models have a lower percentage of nonzero entries, though a higher count of nonzero entries.

3 Efficient, Robust, and Calibrated: Sparsity is All You Need?

Exploiting sparsity is one of the most common strategies for improving the computational efficiency of Transformers. To introduce activation sparsity, previous work has tried using, e.g., an extra mixture-of-experts module (Fedus et al., 2022b), often with some nontrivial engineering design. The emergence of sparse activation and its prevalence discussed in Section 2 reveals that sparsity actually comes for free without the need of extra engineering for inducing sparsity at all.

In Section 3.1 we discuss how the free sparsity in Transformers brings us free computation efficiency in terms of FLOPs count. In Section 3.2 we introduce Top-k Transformers, a simple modification of Transformers where a top-k thresholding operation is applied to the activation maps in all MLPs. Top-k Transformer allows us to obtain sparsity for all input to MLPs and throughout training, hence enables the benefit of sparsity to be realized during training and for all data. Finally, in Section 3.3 we show that enforcing sparser activation with smaller values of k in Top-k Transformer (without any other hacks, tweaks and hyperparameter tuning) bestows Transformers several desired properties, namely, robustness of training with erroneous annotations, less sensitivity to input noise/perturbation, and better confidence calibration of the predictions.

3.1 Efficiency for Free

Given an embedding dimension d_{model} and an MLP intermediate dimension d_{ff}, the computational complexity of a Transformer for an input sequence of length N is $\mathcal{O}(Nd_{\text{model}}^2 + N^2 d_{\text{model}} + N d_{\text{model}} d_{\text{ff}})$, where the first term comes from computing the key, query, and value matrices, the second term

comes from computing the self-attention matrix, and the third term comes from the MLP. For a fixed sequence length N, and considering the fact that d_{ff} is often much larger than d_{model}, it is arguable that MLP poses the computational bottleneck in large Transformers. In the following, we explain how sparsity in activation map of MLP can be leveraged to significantly reduce its computational cost, without affecting the model performance.

Efficiency for the Second MLP Layer. The sparse activation immediately suggests that a lot of the computation for inference with Transformers is not needed at all. That is, while doing dense matrix-matrix multiplications, much of it is about multiplying a vector by a value of zero, which can be avoided to save computation.

Specifically, we consider the second layer of the MLP in (1) which performs the computation

$$\boldsymbol{V}\boldsymbol{a}, \tag{3}$$

where $\boldsymbol{a} \in I\!R^{d_{\text{ff}}}$ is the intermediate activation map of MLP (see (2)) and $\boldsymbol{V} \in I\!R^{d_{\text{model}} \times d_{\text{ff}}}$ is the layer parameter. Eq. (3) involves a simple matrix-vector multiplication which has a FLOP count of $2d_{\text{model}} \times d_{\text{ff}}$. However, if $\boldsymbol{a}$ is sparse with, say s nonzero entries, then the FLOP count for (3) reduces to $2d_{\text{model}} \times s$. Hence,

FLOP in the second MLP layer is reduced by a factor of $1 - \frac{s}{d_{\text{ff}}}$.

Note that $\frac{s}{d_{\text{ff}}}$ is exactly the percentage of nonzeros plotted in the y-axis of e.g. Figure 1, which is 2.7% averaged across all layers. Hence, the computational cost of the second MLP layer can be reduced by a significant amount. More excitingly, the reduction factor $1 - \frac{s}{d_{\text{ff}}}$ is likely to be even bigger for larger Transformer models (see Figures 4a and 4b), pointing to a greater reduction in computation.

Efficiency for the First MLP Layer. The sparsity in the intermediate activation map of MLP does not immediately suggest a reduction in computation for the first MLP layer. Nonetheless, it is possible to significantly reduce the computation in the first MLP layer by leveraging approximate nearest neighbor search, which we explain next.

Recall from (1) that the computation in the first MLP layer is given by

$$\sigma(\boldsymbol{K}^\top \boldsymbol{x}), \tag{4}$$

with $\boldsymbol{K} = [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_{d_{\text{ff}}}] \in I\!R^{d_{\text{model}} \times d_{\text{ff}}}$ being the layer parameter and $\boldsymbol{x}$ being the layer input. If the output is sparse with k nonzero entries, then the calculation in (4) may be formulated as finding k points from the set $\{\boldsymbol{k}_i\}_{i=1}^{d_{\text{ff}}}$ that are "closest" to the input $\boldsymbol{x}$ measured by values of inner product. Such a problem is well-known as the nearest neighbor search (NNS) problem or the maximum inner product search problem. While naive solution of the NNS problem has linear complexity in d_{ff}, there exists *approximate* algorithms (Guo et al., 2020; Johnson et al., 2019; Shrivastava & Li, 2014) that are of sublinear complexity, and using them in Transformers means that

FLOP in the first MLP layer may be reduced to have sublinear complexity in d_{ff}.

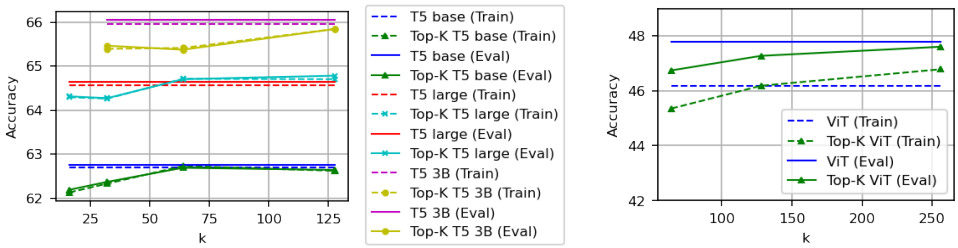
There are of course the questions of whether such approximate NNS algorithms could hurt Transformer performance, which we leave for future study.

Finally, while the gain in efficiency discussed above is measured by FLOPs, the methods therein require computation with *unstructured* sparsity and data-dependent sparsity patterns, which are usually not well supported on computation hardwares such as TPUs and GPUs. As a result, they do not necessarily translate to wall time reductions with naive implementations. In the next section, we present a Top-k Transformer with sparse computation realized by the implementation in Chern et al. (2022), which we show to bring wall time reduction for certain cases. We leave a study of more general implementation of sparse computation for obtaining wall time reduction to future work.

3.2 Controlling Sparsity with Top-k Transformers

The benefit of efficiency from sparsity in Section 3.1 comes with caveats. First, while the activation maps are sparse on average, there is the possibility that some of the activation maps for certain inputs are denser hence cannot benefit from sparse computation. Second, sparsity occurs only in trained Transformers while the computation is dense during and particularly at the beginning of training.

Here we present Top-k Transformer, a simple modification to Transformer architecture that allows us to control sparsity level for all model inputs, and throughout training. Top-k Transformer is built



(a) T5 (b) ViT

Figure 5: Training and evaluation accuracy of Top-k T5 for three different sizes: base, large and 3B (left) and Top-k ViT (right) with varying k. *Top-k Transformer is on par with regular Transformer for a large enough k.* e.g. for T5 3B with $k = 128$, and ViT with $k = 256$, the drop is around 0.3%.

upon a regular Transformer with the only modification being the MLP layers, where at the output of the activation function $\sigma()$ (see (1)) we add a Top-k thresholding operator. That is, the MLPs of Top-k Transformers perform the following computation

$$f(\boldsymbol{x}; \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V} \cdot \text{Top}_k\Big(\sigma(\boldsymbol{K}^T \boldsymbol{x})\Big), \tag{5}$$

where $\text{Top}_k(\cdot)$ performs a thresholding that all entries other than those of the largest k values are set to zero with k being a hyper-parameter subject to design choices. Note that Top-k Transformer reduces to a regular Transformer if we set $k = d_{\text{ff}}$. By using a small value of k, the benefit of efficiency in terms of reduction in FLOP as discussed in Section 3.1 applies to Transformer training as well.

The immediate question for Top-k Transformer is whether it offers controllable sparsity at the cost of a reduced performance. Here we conduct experiments with Top-k T5 and Top-k ViT, and evaluate their performance measured by prediction accuracy for C4 span corruption and ImageNet-21k classification tasks, respectively. The results are provided in Figure 5. We see that with the Top-k T5-{Base, Large, 3B} (resp., Top-k ViT) Transformer, taking k to be 128 (resp., 256) is sufficient for closely matching the test performance of the vanilla T5-{Base, Large, 3B} (resp., ViT). Note that this is achieved without any other hyper-parameter tuning for the Top-k Transformers upon those used for a regular Transformer, and other hyper-parameter choices may further improve the performance of Top-k Transformers.

Finally, we evaluate the benefit of Top-k in terms of reducing inference time latency. In our experiment, we add a Top-k thresholding to T5X (Roberts et al., 2022) with the implementation of `jax.lax.approx_max_k` (Chern et al., 2022) using a recall target of 0.95. Then, we evaluate the decoder per-token wall time for unbatched greedy decoding during inference on a single TPUv4 chip. The results with varying model sizes and varying values of k are presented in Figure 6. We observe that larger models have more wall time reduction, due to the fact that they have larger d_{ff} hence more FLOPs reduction by our discussion in Section 3.1. In particular, for T5-11B we observe around 10% wall time reduction with $k \leq 128$, though this amount becomes smaller with a larger $k = 256$.

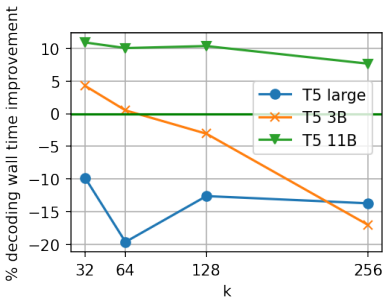


Figure 6: Latency reduction for unbatched greedy decoding in decoder of Top-k Transformers on TPUv4.

3.3 Bonus! Improved Robustness and Calibration

Despite not being explicitly designed for such purposes, inducing sparse activation via Top-k Transformer has the benefits of improving model robustness[5] and confidence calibration. We demonstrate this using the image classification task with the ImageNet-1k dataset, and present the results in Table 1. All results for Top-k ViT are obtained without any model and training hyper-parameter tuning upon those for ViT. Contexts and details are presented below. More results are presented in Appendix C.

[5]This is previously demonstrated in Ahmad & Scheinkman (2019) for ConvNets.

Table 1: Evaluation of Top-128 ViT for ImageNet-1k classification in terms of 1) natural accuracy with ImageNet-1k evaluation set, 2) robust accuracy with {40%, 80%} corrupted training labels, 3) robust accuracy under input perturbation with additive {Gaussian, Impulse, Shot} noise on evaluation images, and 4) calibration error on evaluation data measured by ECE. *Top-128 ViT is on par with ViT for natural accuracy while is significantly better for model robustness and calibration.*

Methods	Natural Accuracy	Accuracy w/ Train Label Noise		Accuracy under Input Perturbation			Expected Calibration Error (ECE)
		40%	80%	Gaussian	Impulse	Shot	
ViT	74.85%	59.44%	25.35%	39.54%	37.37%	38.56%	8.42%
Top-128 ViT	74.83%	62.13%	30.80%	42.29%	40.07%	40.68%	7.48%

Robustness to Label Noise. An important challenge for DNNs is that they are highly susceptible to label noise, the problem where a certain percentage of training labels are corrupted or erroneously generated. This may be attributed to the fact that DNNs are often over-parameterized, hence too "capable" that they tend to overfit, or "memorize" the noisy labels without generalizing to test data. While many dedicated techniques exist (see e.g., Algan & Ulusoy (2021); Song et al. (2022) for a review), here we show that a simple Top-k Transformer can effectively address the label noise issue.

We conduct experiments using the ImageNet-1k dataset for which we replace $p\%$ of the labels in the training set with a random label drawn uniformly from the set of all possible labels. The evaluation performance under $p \in \{40\%, 80\%\}$ label noise is presented in Table 1. It shows that Top-k offers a consistent performance gain with label noise.

Confidence Calibration. Aside from label noise, another symptom of over-parameterization of DNNs is that they tend to be overly confident in their predictions. In the context of classification problems, they tend to assign a high (i.e., close to 1) probability to the class of its prediction, while it is more desirable that they produce a probability that is commensurate with its confidence level (Guo et al., 2017). A commonly used metric for confidence calibration is the expected calibration error (ECE) (Naeini et al., 2015), which is the discrepancy between the probability to the class of a model's prediction and the probability that its prediction is actually correct.

Here we measure the calibration of Top-k ViT via ECE and report the results in Table 1. It shows that Top-k enables the Transformer to be more calibrated when compared to a vanilla Transformer.

Robustness to Input Perturbation. Another important challenge with DNNs is that their outputs tend to be sensitive to naturally occurring image corruptions, which limits their application to mission critical tasks (Bhojanapalli et al., 2021). Here we evaluate the robustness of Top-k ViT to three types of additive noises, namely Gaussian noise, impulse noise, and shot noise. For that purpose, we train Top-k ViT on standard ImageNet-1k training data and report their classification accuracy on ImageNet-C (Hendrycks & Dietterich, 2019), a benchmark that contains algorithmically generated Gaussian, impulse, and shot noise (among many others types) applied to the ImageNet-1k test dataset. For each noise type, there are five severity levels. We report the averaged performance over all severity levels of each corruption type in Table 1.

4 Related Work

Prior efforts on introducing sparsity in deep neural networks abound, though often with diverse motivations and objectives. Here we provide a brief overview of several popular lines of work.

Sparsity for Efficiency. Sparsity in either model weights or activation maps is often used for improving training and inference efficiency (see e.g. Hoefler et al. (2021) for a review). For activation sparsity in particular, sparsity for efficiency is explored perhaps first in ConvNets (Georgiadis, 2019; Kurtz et al., 2020; Rhu et al., 2018) before subsequently becoming a key design component in many of the largest Transformer based language and vision models (Du et al., 2022; Fedus et al., 2022a;b; Rajbhandari et al., 2022). The Top-k thresholding that we use in Top-k Transformer has also been previously used in Gupta et al. (2021) to improve memory efficiency of Transformers. However, it has been unclear *a priori* whether sparsity hurts model performance, hence the practice often relies on wishful design, trial-and-error, and post-hot justification (Baykal et al., 2022). Our discovery that

Transformers naturally produce sparse activation maps, and that larger models are even sparser, may provide principled perspectives towards efficiently training future large models.

Sparsity for Robustness. Many work find that smaller and sparser networks obtained by model compression are more robust to adversarial perturbation (Chen et al., 2022; Guo et al., 2018; Jordao & Pedrini, 2021) and label noise (Xue et al., 2022). Another line of work that uses sparsity for robustness leverages the property that practical data corruption is often sparse (Ghosh et al., 2017; Liu et al., 2022; You et al., 2020). None of the work mentioned above is based on sparsity in activation maps. More closely related to ours is the work of Ahmad & Scheinkman (2019) where sparsity in activation map of convolutional DNNs is shown to improve robustness to input perturbation, and Muthukumar & Sulam (2022) that leverages sparse activation to derive robust generalization error bounds.

Sparsity for Explainability. Work on leveraging sparsity for interpreting deep learning models long exist but often in a post-hoc fashion for examining the semantic meanings encoded by a neuron of a trained model (Dalvi et al., 2019). For Transformers, evidence suggests that the learned knowledge is encoded mainly in its MLPs with individual neurons expressing specific factual knowledge (Dai et al., 2022). Moreover, enforcing neuron activation sparsity in MLPs helps to improve the percentage of neurons that are interpretable (Elhage et al., 2022). Hence, our discovery may point to new directions towards developing more interpretable DNNs (Cuadros et al., 2022; Sajjad et al., 2021).

Sparsity for Data Modeling. Following the seminal work of Olshausen & Field (1996), there are a lot of interests in sparsity as an effective modeling of natural signals (Mairal et al., 2014). With the close resemblance of the computational structure of ReLU networks and sparse encoding algorithms (Gregor & LeCun, 2010), it became natural to study a DNN as a multi-layer sparse modeling of the data (Papyan et al., 2018). Along with substantial theoretical understanding of such a modeling are obtained (Papyan et al., 2017; Sulam et al., 2018), there are also experimental results on their practical benefits (Sun et al., 2018) though less often on modern large-scale data.

Sparsity for Theory of Over-parameterized Models. Because of its simplicity and well-develped theory in classical machine learning (Candès & Wakin, 2008; Vidal et al., 2015; Wright & Ma, 2022), sparse modeling is often used to provide theoretical understanding of modern large and over-parameterized models. This include works on implicit regularization (Chou et al., 2021; Nacson et al., 2022; Vaskevicius et al., 2019; Woodworth et al., 2020; Zhao et al., 2019), nonconvex optimization (Buhai et al., 2020; Sulam et al., 2022), noise interpolators (Chinot et al., 2022; Donhauser et al., 2022; Koehler et al., 2021), etc. However, the aforementioned work uses sparsity as a testbed or toy model to gain insights, without implication of existence of sparsity in DNNs.

5 Discussion

This work demonstrates the natural emergence of sparse activation in commonly used Transformer models (Section 2). The notion of sparsity pertains to the law of parsimony, a.k.a. *Occam's razor*, where among all possible explanations of observed data, the *simplest* ones are preferred. It is a fundamental scientific principle broadly used in various scientific and engineering subjects (Domingos, 1999; Epstein, 1984), including classical machine learning (Tibshirani, 1996). Hence, our discovery may be suggesting that the law of parsimony is playing a role in Transformers even though they are not explicitly designed so, resonating with recent view on the role of sparsity for intelligence systems (LeCun, 2022; Ma et al., 2022; Roberts, 2021; Vasudevan et al., 2021). More importantly, we back such a perspective by providing evidence of improved robustness and calibration via enforcing sparsity using Top-k thresholding (Section 3), which indicates that sparsity is indeed a pertinent prior for good generalization. We hope that our work may motivate future effort on introducing sparsity in deep learning models in a more principled way for obtaining more efficient, robust, and calibrated models. Finally, while our motivation of studying sparse activation in Transformers comes (partly) from study of biological brains, establishing such a connection may reciprocally benefits efforts on applying artificial intelligence to the study of biology and neuroscience (Richards et al., 2022).

REFERENCES

Subutai Ahmad and Luiz Scheinkman. How can we be so dense? the benefits of using highly sparse representations. *arXiv preprint arXiv:1903.11257*, 2019.

Mohsin S Ahmed, James B Priestley, Angel Castro, Fabio Stefanini, Ana Sofia Solis Canales, Elizabeth M Balough, Erin Lavoie, Luca Mazzucato, Stefano Fusi, and Attila Losonczy. Hippocampal network reorganization underlies the formation of a temporal association memory. *Neuron*, 107(2): 283–291, 2020.

Görkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 215:106771, 2021.

Alison L Barth and James FA Poulet. Experimental evidence for sparse firing in the neocortex. *Trends in neurosciences*, 35(6):345–355, 2012.

Cenk Baykal, Nishanth Dikkala, Rina Panigrahy, Cyrus Rashtchian, and Xin Wang. A theoretical view on sparsely activated networks. *arXiv preprint arXiv:2208.04461*, 2022.

Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10231–10241, 2021.

Rares-Darius Buhai, Yoni Halpern, Yoon Kim, Andrej Risteski, and David Sontag. Empirical study of the benefits of overparameterization in learning latent variable models. In *International Conference on Machine Learning*, pp. 1211–1219. PMLR, 2020.

Emmanuel J Candès and Michael B Wakin. An introduction to compressive sampling. *IEEE signal processing magazine*, 25(2):21–30, 2008.

Tianlong Chen, Zhenyu Zhang, Santosh Balachandra, Haoyu Ma, Zehao Wang, Zhangyang Wang, et al. Sparsity winning twice: Better robust generalization from more efficient training. In *International Conference on Learning Representations*, 2022.

Felix Chern, Blake Hechtman, Andy Davis, Ruiqi Guo, David Majnemer, and Sanjiv Kumar. Tpu-knn: K nearest neighbor search at peak flop/s. *arXiv preprint arXiv:2206.14286*, 2022.

Geoffrey Chinot, Matthias Löffler, and Sara van de Geer. On the robustness of minimum norm interpolators and regularized empirical risk minimizers. *The Annals of Statistics*, 50(4):2306–2333, 2022.

Hung-Hsu Chou, Johannes Maly, and Holger Rauhut. More is less: Inducing sparsity via overparameterization. *arXiv preprint arXiv:2112.11027*, 2021.

Xavier Suau Cuadros, Luca Zappella, and Nicholas Apostoloff. Self-conditioning pre-trained language models. In *International Conference on Machine Learning*, pp. 4455–4473. PMLR, 2022.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6309–6317, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

Pedro Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999.

Konstantin Donhauser, Nicolo Ruggeri, Stefan Stojanovic, and Fanny Yang. Fast rates for noisy interpolation require rethinking the effects of inductive bias. *arXiv preprint arXiv:2203.03597*, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Jones, , Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/solu/index.html.

Robert Epstein. The principle of parsimony and some applications in psychology. *The Journal of Mind and Behavior*, pp. 119–130, 1984.

William Fedus, Jeff Dean, and Barret Zoph. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022a.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022b.

Georgios Georgiadis. Accelerating convolutional neural networks via activation map compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7085–7095, 2019.

Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pp. 399–406, 2010.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pp. 3887–3896. PMLR, 2020.

Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. Sparse dnns with improved adversarial robustness. *Advances in neural information processing systems*, 31, 2018.

Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pp. 39–52, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241):1–124, 2021.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

Artur Jordao and Hélio Pedrini. On the effect of pruning on adversarial robustness. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1–11, 2021.

Jason ND Kerr, David Greenberg, and Fritjof Helmchen. Imaging input and output of neocortical networks in vivo. *Proceedings of the National Academy of Sciences*, 102(39):14063–14068, 2005.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Frederic Koehler, Lijia Zhou, Danica J Sutherland, and Nathan Srebro. Uniform convergence of interpolators: Gaussian width, norm bounds and benign overfitting. In *Advances in Neural Information Processing Systems*, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *International Conference on Machine Learning*, pp. 5533–5543. PMLR, 2020.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. 2022.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by over-parameterization. 2022.

Yi Ma, Doris Tsao, and Heung-Yeung Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, pp. 1–26, 2022.

Julien Mairal, Francis Bach, Jean Ponce, et al. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

Ramchandran Muthukumar and Jeremias Sulam. Adversarial robustness of sparse local lipschitz predictors. *arXiv preprint arXiv:2202.13216*, 2022.

Mor Shpigel Nacson, Kavya Ravichandran, Nathan Srebro, and Daniel Soudry. Implicit bias of the step size in linear diagonal neural networks. In *International Conference on Machine Learning*, pp. 16270–16295. PMLR, 2022.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *The Journal of Machine Learning Research*, 18(1):2887–2938, 2017.

Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad. Theoretical foundations of deep learning via sparse representations: A multilayer sparse model and its connection to convolutional neural networks. *IEEE Signal Processing Magazine*, 35(4):72–89, 2018.

Cindy Poo and Jeffry S Isaacson. Odor representations in olfactory cortex:"sparse" coding, global inhibition, and oscillations. *Neuron*, 62(6):850–861, 2009.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *arXiv preprint arXiv:2201.05596*, 2022.

Minsoo Rhu, Mike O'Connor, Niladrish Chatterjee, Jeff Pool, Youngeun Kwon, and Stephen W Keckler. Compressing dma engine: Leveraging activation sparsity for training deep neural networks. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 78–91. IEEE, 2018.

Blake Richards, Doris Tsao, and Anthony Zador. The application of artificial intelligence to biology and neuroscience. *Cell*, 185(15):2640–2643, 2022.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*, 2022. URL https://arxiv.org/abs/2203.17189.

Daniel A. Roberts. Why is ai hard and physics simple?, 2021.

Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. Neuron-level interpretation of deep nlp models: A survey. *arXiv preprint arXiv:2108.13138*, 2021.

Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in neural information processing systems*, 27, 2014.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Jeremias Sulam, Vardan Papyan, Yaniv Romano, and Michael Elad. Multilayer convolutional sparse modeling: Pursuit and dictionary learning. *IEEE Transactions on Signal Processing*, 66(15): 4090–4104, 2018.

Jeremias Sulam, Chong You, and Zhihui Zhu. Recovery and generalization in over-realized dictionary learning. *Journal of Machine Learning Research*, 23(135):1–23, 2022.

Xiaoxia Sun, Nasser M Nasrabadi, and Trac D Tran. Supervised deep sparse coding networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 346–350. IEEE, 2018.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34: 24261–24272, 2021.

Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. *Advances in Neural Information Processing Systems*, 32, 2019.

Rama K Vasudevan, Maxim Ziatdinov, Lukas Vlcek, and Sergei V Kalinin. Off-the-shelf deep learning is not enough, and requires parsimony, bayesianity, and causality. *npj Computational Materials*, 7(1):1–6, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Rene Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis. *Interdisciplinary Applied Mathematics*, 43:22–23, 2015.

Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pp. 3635–3673. PMLR, 2020.

John Wright and Yi Ma. *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications*. Cambridge University Press, 2022.

Yihao Xue, Kyle Whitecross, and Baharan Mirzasoleiman. Superior generalization of smaller models in the presence of significant label noise. *arXiv preprint arXiv:2208.08003*, 2022.

Chong You, Zhihui Zhu, Qing Qu, and Yi Ma. Robust recovery via implicit bias of discrepant learning rates for double over-parameterization. *Advances in Neural Information Processing Systems*, 33: 17733–17744, 2020.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Peng Zhao, Yun Yang, and Qiao-Chu He. Implicit regularization via hadamard product over-parametrization in high-dimensional linear regression. *arXiv preprint arXiv:1903.09367*, 2019.

Appendices

The appendices are organized as follows. In Section A we provide the implementation details for experiments conducted in this paper. In Section B we demonstrate the emergence of sparse activation in other architectures and with other optimizers than those used in Section 2. In Section C we provide additional experiments upon those in Section 3 to demonstrate the benefits of sparsity. In Section D we explore the potential causes of sparsity, with a focus on the effect of training data. In Section E we present a derivation to show that during early training, the final MLP layer's intermediate activation tends to get sparse. Finally in Section F we present insights on the emergence of activation sparsity from experiments on 2-layer MLP models.

A Implementation Details

A.1 T5

For most of the experiments, except the Top-k transformer, we used vanilla T5 architecture (Raffel et al., 2020). We trained model with Adafactor optimizer, an inverse square root learning rate schedule, and no dropout. For the first 10,000 steps we also use a fixed learning rate of 0.01 as warm-up. The training task is span corruption without any mixture, and unless specified otherwise, we train the model for 100,000 steps with batch size of 256 to save compute and time, as the sparsity or accuracy trend is already clear by then. We used 512 tokens on the encoder side and 114 tokens on the decoder side.

A.2 ViT

Following Dosovitskiy et al. (2021), we train ViT using ADAM (Kingma & Ba, 2015) as the optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$. Other training details such as weight decay, dropout rate, and learning rate all follow the description in (Dosovitskiy et al., 2021, Section B.1) except that we train for 180 epochs (as opposed to 300) on ImageNet-1k.

B Additional Results On Prevalence of Sparsity

B.1 Sparsity and Network Architecture

We evaluate the sparsity level of activation map in several commonly used network architectures beyond T5 and ViT. This includes BERT which is also a Transformer based architecture, as well as non-Transformer based architectures such as MLP-Mixer and ConvNets. We also examine whether residual connection accounts for the emergence of sparsity.

BERT. We evaluate the sparsity level of BERT models (Devlin et al., 2019). We specifically consider BERT Base (12 layers) and BERT Large (24 layers) Transformer models, with ReLU activation in the MLP layers. We follow the same training receipe as Devlin et al. (2019) and pre-train these models on Wikipedia and Books dataset using Masked Language Modelling (MLM) objective. We train for 450000 steps with a batch size of 1024 using AdamW optimizer with $1e-4$ learning rate.

In Figure B.2 we plot the sparsity levels of both BERT models for all the intermediate MLP layers (plots a and b). We observe that both these models exhibit high levels of sparsity (< 10%) as other Transformer models. We further visualize the pre-activation values of the MLP layers as histograms in plots c and d. We observe that while they have mean 0 at initialization, the mean quickly becomes negative as training progresses, resulting in high sparsity levels.

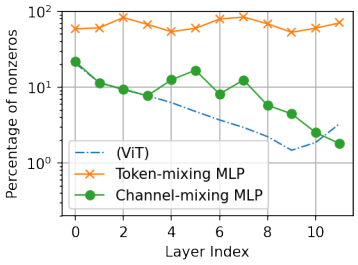
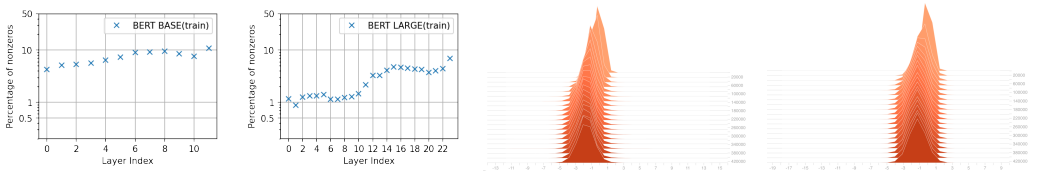


Figure B.1: Percentage of nonzero entries in activation maps of MLP-Mixer trained on ImageNet-21k. Results for token-mixing and channel-mixing MLPs are plotted in separate curves.



(a) BERT Base (b) BERT Large (c) Pre-activations: Layer 1 (d) Pre-activations: Layer 12

Figure B.2: Plots a, b: Percentage of nonzero entries in activation maps of BERT Base and Large models (Devlin et al., 2019) trained on Wikipedia dataset. We observe high levels of sparsity (<10%) similar to other Transformer models. Plots c, d: Histograms of pre-activation values for layers 1 and 12 of a Bert Base model. We notice that while at initialization the activations are distributed with mean 0, the mean quickly shifts negative as the training progresses, resulting in high levels of sparse activation values.

MLP-Mixer. We evaluate the sparsity level of the MLP-Mixer (Tolstikhin et al., 2021), an all-MLP architecture constructed from cascading token-mixing and channel-mixing MLPs. Specifically, we use Mixer-B16 as the architecture, ADAM with $\beta_1 = 0.9, \beta_2 = 0.999$ as the optimizer, and train on ImageNet-21k for 300 epochs. While Tolstikhin et al. (2021) sweeps over a product set of hyper-parameters, here for simplicity we use a fixed set of hyper-parameters with weight decay of 0.03, gradient norm clipping at 1.0, base learning rate of 0.003, RandAugment magnitude of 10, no mixup, no stochastic depth, and no dropout.

Figure B.1 shows the sparsity level at the intermediate layer of both token mixing and channel mixing MLPs of Mixer-B16. We also plot the sparsity level of ViT (i.e., the plot in Figure 2a) to Figure B.1 for a comparison. It can be seen that the first four layers of channel mixing MLP and ViT have almost identical sparsity levels, while the rest of the layers (other than the last one) of channel mixing MLP are denser than the corresponding layer of ViT. On the other hand, the token mixing MLPs produce dense activation maps with more than 50% nonzero entries, probably because the dimension of the activation maps (384) is too small.

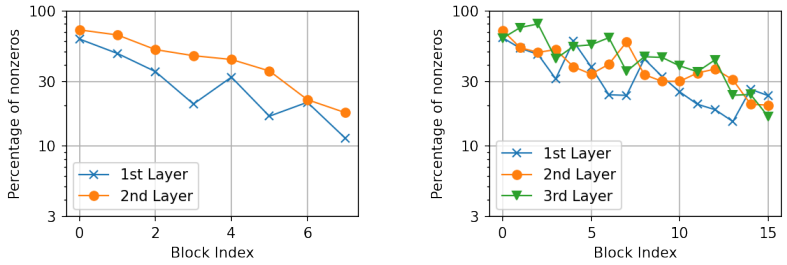
Convolutional Neural Network (ConvNet). Sparsity in activation maps has been studied for ConvNets such as the AlexNet (Krizhevsky et al., 2017) at least as early as in the work of Rhu et al. (2018). There are also follow-up work (Georgiadis, 2019; Kurtz et al., 2020) on how enforcing sparse activation maps can help to gain computation efficiency. For completeness, we evaluate and present results for the sparsity level of residual networks (ResNets) (He et al., 2016), which is one of the most commonly used ConvNets, trained on ImageNet-1k. In particular, we focus on ResNet-18 and ResNet-50 which are constructed from stacking 8 standard residual blocks and 16 "bottleneck" residual blocks, respectively, where each block has two and three convolutional and ReLU layers, respectively. We examine the sparsity of activation maps after each of the ReLU layers in each residual block.

The results for ResNet-18 and ResNet-50 are reported in Figure B.3a and Figure B.3b, respectively. Here, the x-axis is the index of the residual block, and the sparsity of different layers in the residual blocks are plotted with separated curves in each figure. It can be observed that

- Layers near the network output tend to produce sparser activation maps than layers near the network input. This is aligned with the observation with ViT trained on ImageNet-1k (see Figure 2b).
- For each residual block, the intermediate layers (i.e., the 1st layer for ResNet-18 and the 1st & 2nd layers for ResNet-50) produce sparser activation maps than the output layer (i.e., 2nd layer for ResNet-18 and 3rd layer for ResNet-50).

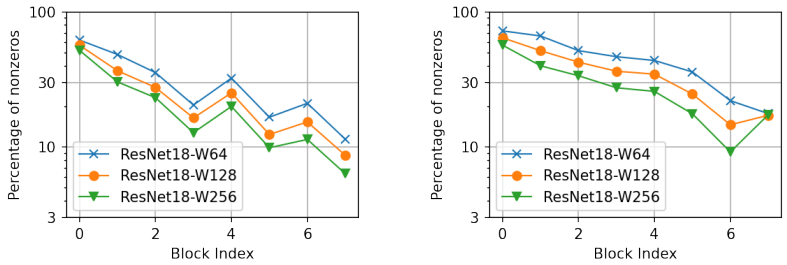
In addition, all residual blocks are divided into four stages that have different output feature map sizes. For ResNet-50, the four stages are composed of blocks 0 - 2, 3 - 6, 7 - 12, and 13 - 15. Figure B.3b shows that there are patterns on how sparsity level varies within each stage and across the boundary of the stages.

- For the 1st layers, percentage of nonzeros decreases within each stage, and jumps up from the last layer of each stage to the first layer of next stage.
- For the 2nd layers, percentage of nonzeros decreases quickly at the beginning of each stage then becomes stable.



(a) ResNet-18 (b) ResNet-50

Figure B.3: Percentage of nonzero entries in activation maps of ResNet-18 and ResNet-50 trained on ImageNet-1k. Results for the two (resp., three) layers in each residual block (resp., bottleneck residual block) of ResNet-18 (resp., ResNet-50) are plotted in separate curves.



(a) 1st Layers (b) 2nd Layers

Figure B.4: Effect of network width $\in \{64, 128, 256\}$ on sparsity level across layers of ResNet-18.

- For the 3rd layers, percentage of nonzeros tend to increase slightly within each stage, and jumps down from the last layer of each stage to the first layer of next stage.

Such observations may help to understand the role of each stage in ResNets.

Comparing the percentage of nonzero entries in ResNets (shown in Figure B.3a and Figure B.3b) and for Transformers (shown in Figure 2b), both of which are trained on ImageNet-1k, we see that ResNets produce much denser activation maps with more than 10% nonzero entries in all layers. Meanwhile, the trend that larger models are sparser for Transformers (see Section 2.2) holds for ResNets as well, as seen in Figure B.4. Here, we vary the width of ResNet-18 by multiplying the number of output channels of each convolutional layer by a factor of 1 (for width = 64), 2 (for width = 128), and 4 (for width = 256). The two subfigures show the effect of width on sparsity of the first and second layers in each residual block, respectively. In both cases, wider models have smaller percentage of nonzero entries across all layers, except for the very last layer (i.e., the 2nd layer in block #7 shown in Figure B.4b).

Sparsity and Residual Learning. We provide a study on the effect of residual connections on activation sparsity using T5. Each Transformer block contains two types of residual connections: the one that is in parallel with the attention blocks, and the one that is in parallel with the MLP blocks. We focus on the residual connection parallel to the MLP blocks. Towards that, we train two T5-Large models, one using the vanilla Transformer block and the other with residual connection removed for the Transformer block on `encoder layer 6` (i.e., the 7th encoder layer, as we count from 0). There is a 1.6% evaluation accuracy drop with the latter model compared to the former model.

The percentage of nonzero entries of these two Transformers are presented in Figure B.5 for the encoder layers and in Figure B.6 for the decoder layers. It can be seen that in `encoder layer 6` for which the residual connection is removed, the sparsity has a very different trend during training compared to the corresponding layer of the vanilla Transformer. Moreover, the sparsity level at all other layers also changes, though to a much smaller extend.

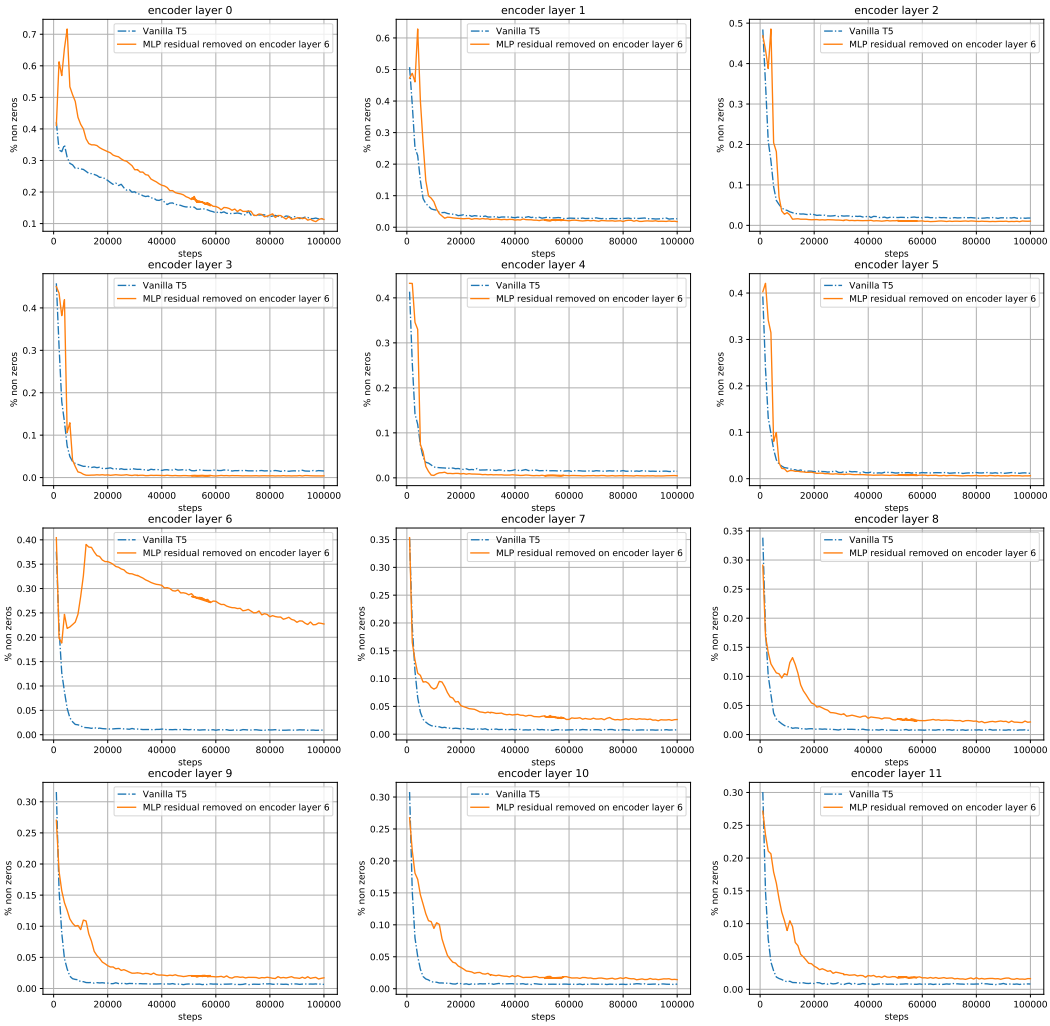


Figure B.5: Percentage of nonzero entries in activation maps in vanilla T5-Large and in a T5-Large with the residual connection parallel to MLP removed in the 7th encoder layer (i.e., `encoder layer 6`). Different subplots correspond to different *encoder* layers (see Figure B.6 for results on *decoder* layers). The `encoder layer 6`, which has its residual connection removed, shows a significant difference in both sparsity and the trend of sparsity during training. Sparsity level in other layers changes from vanilla T5-Large as well, though to a smaller extent.

B.2 Sparsity and Optimizer

Transformers are usually trained using ADAM or its variants as the optimizer (Kingma & Ba, 2015). It may be curious to ask whether the emergence of sparsity is specific to such optimizers and whether other optimizers, such as stochastic gradient descent (SGD), also leads to sparse activation maps. However, we find that SGD cannot effectively train Transformer architectures such as T5 and ViT. Hence, we study the effect of optimizer on activation sparsity by looking at ResNet trained on ImageNet-1k following the setup in Section B.1, since both SGD and ADAM can effectively train the network. To train ResNet with ADAM, we use the same hyper-parameters as those used in SGD, with the only difference being that the optimizer is ADAM with $\beta_1 = 0.9, \beta_2 = 0.999$. To make the comparison with SGD fair, we tune the base learning rate for ADAM and select $3e-3$, which is the one that gives the highest training accuracy among the set of $\{1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$. The training accuracy obtained by ADAM with base learning rate $3e-3$ is similar to that obtained by SGD, namely, 67.8% by ADAM vs 69.3% by SGD with ResNet-18, and 75.0% by ADAM vs 78.5% by SGD with ResNet-50.

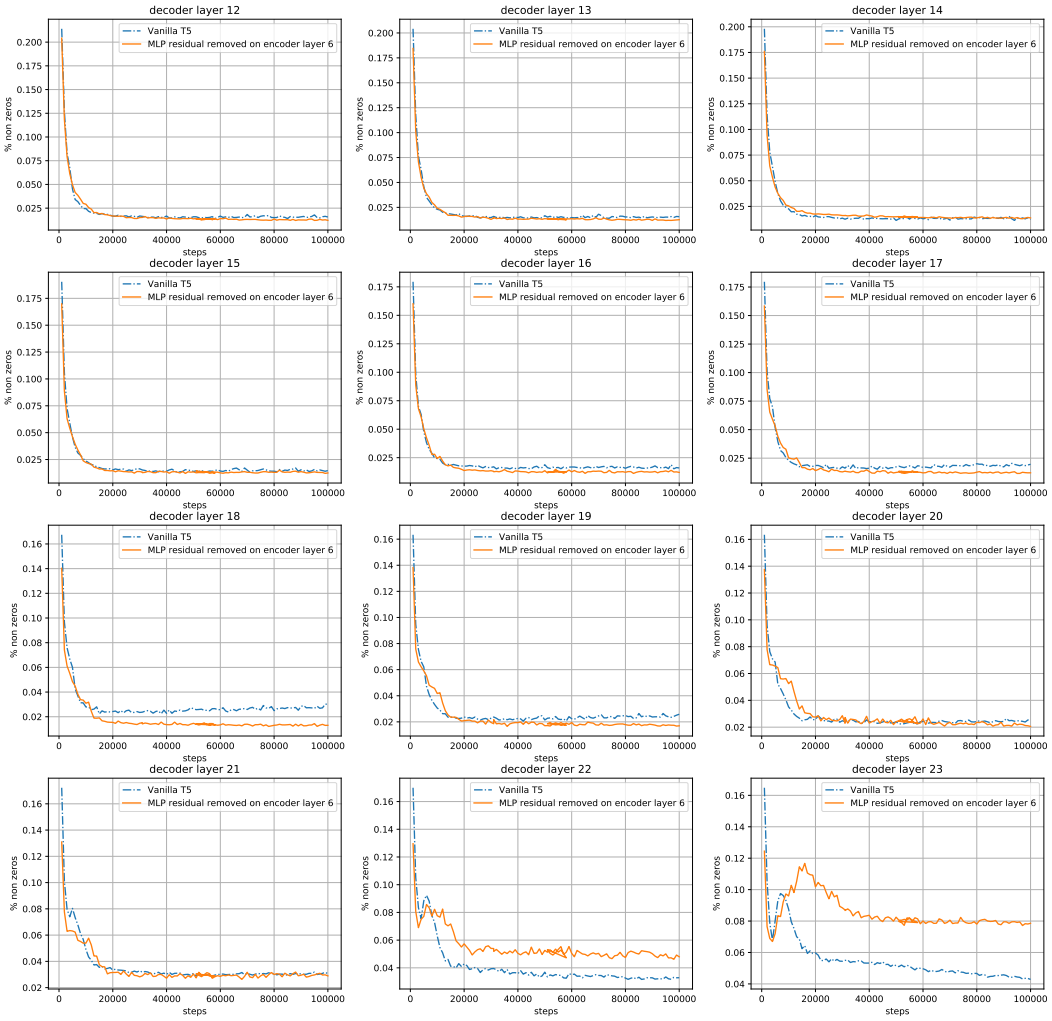
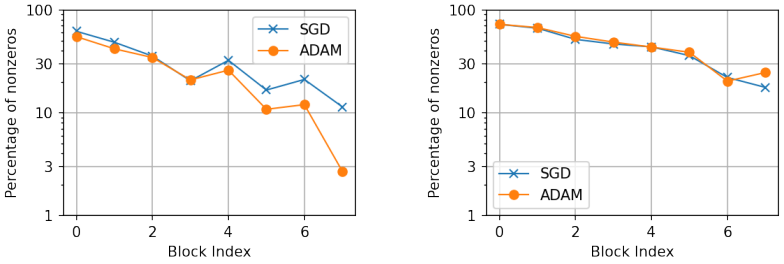


Figure B.6: Same setup as Figure B.5, but showing the results for the last 12 layers of the decoder.

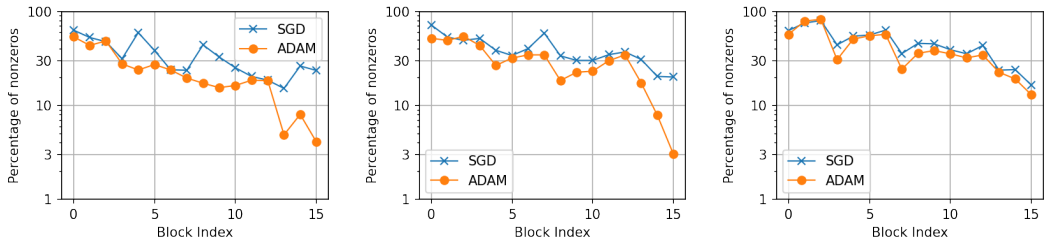


(a) 1st Layers

(b) 2nd Layers

Figure B.7: Effect of optimizer on sparsity level across layers of ResNet-18.

The results for ResNet-18 and ResNet-50 are presented in Figure B.7 and Figure B.8, respectively. For ResNet-18, we see that ADAM leads to a smaller percentage of nonzero entries particularly towards the output of the network for the first layers of each residual block. In contrast, ADAM and SGD have very similar sparsity level at the second layers of each residual block. Similar observation holds for ResNet-50, where the percentage of nonzero entries is smaller with ADAM for the first and second layers of each residual block, while for the third layer the sparsity level does not change much.



(a) 1st Layers (b) 2nd Layers (c) 3rd Layers

Figure B.8: Effect of optimizer on sparsity level across layers of ResNet-50.

C Additional Results on Benefits of Sparsity

C.1 Benefits of Sparsity Persists with ℓ_1-Norm Induced Sparsity

While Top-k thresholding is used in Section 3.3 to demonstrate the benefit of sparsity, we show that other means of obtaining sparsity, such as an explicit ℓ_1 norm regularization, also provides such benefits.

We experiment with ViT for ImageNet-1k classification under the same setup as in Section 3.3. Here instead of the Top-k ViT, we train a regular ViT but with an additional loss term, which is the sum of the ℓ_1 norm of all activation maps of ViT across all layers. We refer to the method as L1-ViT. We vary the weight λ on the ℓ_1 loss in the set $\lambda \in \{0.1, 0.5, 1.0\}$ to control the strength of the regularization, and denote the corresponding methods as L1-ViT-$\{0.1, 0.5, 1.0\}$.

The sparsity level, natural accuracy, robust accuracy under input perturbation, and ECE of L1-ViT are reported in Table C.1. We see that with $\lambda = 0.1$ or 0.5, the averaged percentage of nonzero entries do not change much, but already demonstrates performance gain in terms of accuracy under input perturbation and calibration without hurting the natural accuracy. Using a $\lambda = 1.0$ drastically reduces the percentage of nonzero entries but it offers a better performance in terms of accuracy under input perturbation and calibration compared to ViT.

C.2 Additional Results for Calibration

Under the same setup as in Section 3.3, we report the calibration of Top-k ViT during training and with varying values of k, and report the results in Figure C.1. At the beginning of training the model has a low ECE because the output probabilities are mostly uniformly distributed across all classes hence the model is not confident, and that its prediction is purely random hence wrong with high probability. The model tends to become overly confident as the training progresses, hence the ECE increases particularly towards the end of training. What we can observe is that Top-k enables the Transformer to be more calibrated when compared to a vanilla Transformer, particularly for small values of k. The results with $k = 128$ and its comparison with the vanilla ViT is also presented in Table 1.

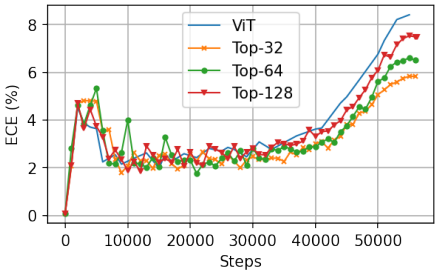
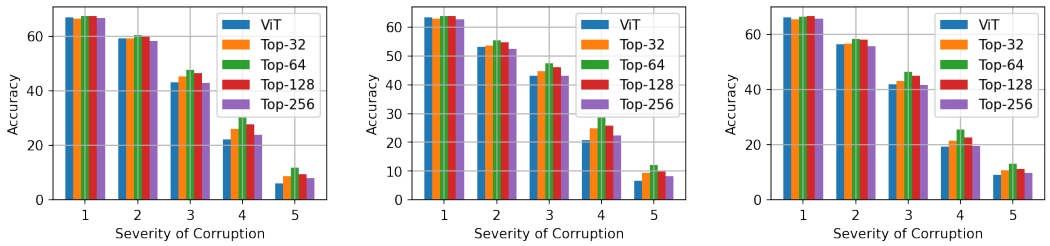


Figure C.1: Confidence calibration of Top-k ViT for ImageNet-1k classification.

C.3 Additional Results for Robustness to Input Perturbation

Under the same setup as in Section 3.3, we report the performance of Top-k ViT with varying value of k in Figure C.2. We can see that Top-k ViT offers a performance gain over the vanilla ViT, particular with $k = 64$ or $k = 128$ and the severity level is high. The averaged performance over all severity levels of each corruption type is reported in Table 1.



(a) Gaussian Noise (b) Impulse Noise (c) Shot Noise

Figure C.2: Performance of Top-k ViT on corrupted ImageNet-1k test data with Gaussian noise (left), impulse noise (middle), and shot noise (right), each under five severity levels. Top-k improves robustness for all noise types and on all corruption levels with a suitable choice of k.

Table C.1: Evaluation of ViT with a varying weight $\in \{0.1, 0.5, 1.0\}$ on a ℓ_1 regularization upon activation maps for ImageNet-1k classification in terms of 1) averaged percentage of nonzero entries in activation maps across all layers, 2) natural accuracy (i.e., on ImageNet-1k evaluation set), 3) robust accuracy under input perturbation with additive {Gaussian, Impulse, Shot} noise, and 4) calibration error measured by ECE.

Methods	Avg. Perc. of Nonzeros	Natural Accuracy	Accuracy under Input Perturbation			Expected Calibration Error (ECE)
			Gaussian	Impulse	Shot	
ViT	5.67%	74.85%	39.54%	37.37%	38.56%	8.42%
L1-ViT-0.1	5.66%	75.02%	40.70%	38.52%	39.56%	8.37%
L1-ViT-0.5	5.76%	74.83%	42.46%	40.64%	40.96%	8.22%
L1-ViT-1.0	1.60%	73.21%	40.26%	38.01%	38.95%	6.34%

D SPARSITY FROM TRAINING DYNAMIC?

In this section we study the causes of sparsity in activation maps of trained Transformers. Towards that, in Section D.1, D.2, and D.3 , we present a set of hypotheses and design corresponding experiments to validate or disprove them. We discuss the observation from the experiments and draw a tentative conclusion in Section D.4.

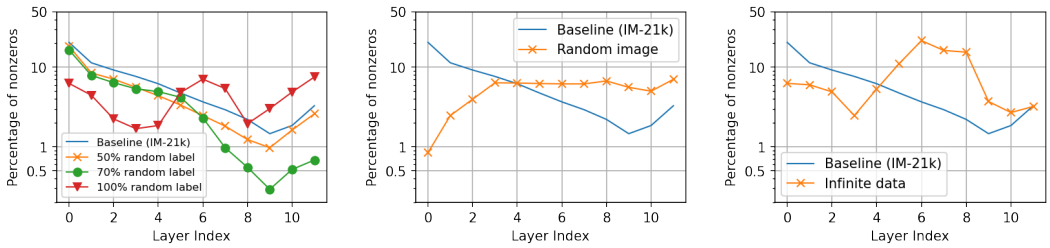
D.1 SPARSITY FROM LABELS?

Transformers are usually trained via supervised learning (e.g., using the ImageNet dataset for ViT) or self-supervised learning (e.g., using the span corruption task for T5). In both cases, the training labels provide a pertinent and meaningful description of the corresponding training data (e.g., image for ViT and text for T5). Sparsity may arise because the label set provides a structured organization of the massive training data, hence the training dataset admits a compact representation. This motivates us to formulate the following hypothesis.

Hypothesis D.1 (Sparsity from labels)**.** *Sparsity in trained Transformers arises from the labels for Transformer training, e.g., human annotations in supervised training or generated labels from data itself in self-supervised learning.*

We use a *random label* experiment with ViT for image classification to test Hypothesis D.1. Specifically, we generate a new training dataset by replacing $p\%$ of the labels in the ImageNet-21k dataset with random labels drawn uniformly at random from the set of all possible labels, where p is varied to examine the effects. With such a dataset, the labels for a certain percentage of images do not provide a meaningful description for the content of the image. Hence, if Hypothesis D.1 is valid, then the activation map will become dense.

The sparsity level of ViT trained on the random label datasets is shown in Figure D.1a. It can be seen that the percentage of activated neurons decreases with an increasing percentage of label noise up to



(a) Random label (b) Random image (c) Infinite data

Figure D.1: Percentage of nonzero entries in ViT trained on ImageNet-21k (IM-21K) with (a) *random labels* where $p\%$ labels are replaced by labels drawn from a uniform distribution with $p \in \{50\%, 70\%, 100\%\}$, (b) *random images* where each image is replaced by one where the pixels are drawn from i.i.d. uniform distribution in $[-1, 1]$, and (c) *infinite data* where sufficient training data is generated by drawing random image and random label pairs so that the model is never trained on the same pair twice.

70%. An even higher label noise level at 100% changes the sparsity level across layers as the shallow layers (i.e., layers 0 - 4) becomes sparser, while the deep layers (i.e., layers 5 - 11) becomes denser. Nonetheless, even with 100% label noise, all layers have $< 10\%$ activated neurons.

D.2 Sparsity from Data?

While modern image and text data are often of high-dimensional, their intrinsic degree of freedom is much smaller, i.e., they are low-dimensional and admit compact representations (Vidal et al., 2015; Wright & Ma, 2022). Hence, even if the labels do not provide meaningful descriptions of the data, it may still be possible that Transformers extract low-dimensional structures from data and produce compact representations in the form of sparse activation maps. This motivates the following hypothesis.

Hypothesis D.2 (Sparsity from natural data)**.** *Sparsity in trained Transformers arises from natural training data (e.g., images for ViT and texts for T5).*

We use a *random image* experiment to test Hypothesis D.2. With the ImageNet-21k dataset, we replace each image with a random image generated by drawing pixel values from an i.i.d. Uniform distribution in the range of [0, 255], and use these images (instead of the original images in ImageNet-21k) for model training. Such random images do not contain any low-dimensional structures nor compact representations.

The percentage of nonzero entries of a ViT trained on random image dataset is shown in Figure D.1b. It can be seen that the first four layers become sparser while the last few layers become relatively denser compared to training with natural images in ImageNet-21k. Nonetheless, all layers have $< 10\%$ activated neurons.

D.3 Sparsity from Data-fitting?

Modern deep neural networks are often *over-parameterized*, with sufficient capacity to fit practical training datasets and obtain close-to-zero training error. There is evidence suggesting that this result holds true even if the data and label are generated in random (Zhang et al., 2021). Hence, there is the possibility that sparsity arises because the training data, even if generated in random, is scarce relative to the scale of modern over-paremeterized models.

Hypothesis D.3 (Sparsity from data-fitting)**.** *Sparsity in trained Transformers arises from the fact that models have more than sufficient capacity to fit training data of practical scale.*

To test Hypothesis D.3, we design an *infinite data* experiment where the amount of training data is infinitely large so that any practical Transformer becomes under-parameterized relative to the data and cannot fit the data. The way we generate infinite training data is to sample images with random pixels as in the random image experiment, and for each image we sample a random label as in the random label experiment. Moreover, we generate sufficient amount of such training data to make

sure that the model never sees the same data point twice during the training. The number of training iterations in the infinite data experiment is kept the same as that of the random image and random label experiments.

The result of this experiment is presented in Figure D.1c. It can be seen that the first four layers produce sparser activation maps, while middle layers with index 4 - 7 are considerably denser compared to the baseline with near 10% to 20% nonzero entries.

D.4 Discussion

The results of random label, random image, and infinite data experiments in Figure D.1 show that labels, data, and data-fitting as conjectured in Hypothesis D.1, D.2, and D.3, respectively, all affect the sparsity level of the activation map. Nonetheless, none of them fully explains the emergence of sparsity since for all results in Figure D.1, the percentage of nonzero entries is considerably smaller than at the initialization (i.e., 50%).

Our results point to the possibility that sparsity comes from the training dynamic. Namely, at early training stage with any data, the descending direction of the gradient on the Transformer parameters tends to point to a regime where their MLPs produce sparse activation maps. This is because at the start of training, any non-zero neuron activity in the MLP is more likely to cause the cross entropy loss to go up rather than go down. We demonstrate this negative gradient effect at initialization on the last MLP layer in Appendix E, as well as an analysis on two-layer MLP models in Appendix F. Empirically, we find in Figure B.2 that the suppression of activations indeed happens at the early stages of training in Bert. We leave it for the future work to fully investigate the effect of training dynamics on sparsity for Transformer models.

E Negative gradient on activations at initialization

In this section, we provide theoretical insights into the emergence of sparsity in MLP hidden layers. In particular, we consider a classification model with a final MLP layer that maps a feature layer of neurons to the classification logits. Under the following assumptions:

- The loss function is the standard cross entropy loss.
- The final MLP layer has no bias terms.
- The feature layer uses the ReLU activation function.
- Each element of the weight matrix of the final MLP layer follows an i.i.d. random uniform initialization: $W_{ki} \sim U(-1, 1)$. (This can be easily relaxed to any mean 0, symmetric distribution.)

We show that the expectation value of the gradient on the feature layer activations, when averaged over different initializations of the last layer MLP weights, are negative at initialization:

$$\langle -\partial_{A_{ni}} \mathcal{L} \rangle < 0 \tag{E.1}$$

Where A_{ni} is the i-th feature layer activation resulting from the n-th datapoint in the training set.

Consider a neural classifier with its final layer being an output of a MLP:

$$y_{nk} = \sum_{i=1}^{N_e} W_{ki} R(A_{ni}) \tag{E.2}$$

where y_{nk} are classification logits of the n-th training example on the k-th class, $k \in \{1 \dots N_c\}$. W_{ki} are the final layer weights, R denotes ReLU activations and $A_i(x)$ with $i \in 1 \dots N_e$ are the pre-activations. N_e is the embedding width. The pre-activations can be computed by an arbitrarily complex network.

The cross entropy loss is given by:

$$\mathcal{L} = -\sum_{n=1}^{N_d} \sum_{k=1}^{N_c} t_{nk} \log p_{nk} \tag{E.3}$$

where $n \in 1 \dots N_d$ denotes different training examples and t_{nk} is the ground truth label distribution, p_{nk} is the predicted distribution computed by the model:

$$p_{nk} = \frac{e^{y_{nk}}}{\sum_{j=1}^{N_c} e^{y_{nj}}} \tag{E.4}$$

in particular,

$$\sum_k t_{nk} = \sum_k p_{nk} = 1 \tag{E.5}$$

We compute the gradient on the pre-activations of the i-th neuron:

$$-\partial_{A_{ni}} \mathcal{L} = \begin{cases} 0, & \text{if } A_{ni} \leq 0 \\ \sum_{k=1}^{N_c} W_{ki} \left(t_{nk} - p_{nk}\right), & \text{if } A_{ni} > 0 \end{cases} \tag{E.6}$$

The key observation of this derivation is that at initialization, there is no statistical correlation between the true label distribution and the model's randomly initialized weights, therefore the first term is zero on average. But because p_{nk} is computed using W_{ki}, there are some correlation between them, making the second term positive. As a result, the preactivations A_{ni} will receive a negative gradient due to mismatch between the predictions p_{nk} and the true distribution t_{nk}.

To see this effect more clearly, we compute the average gradient with respect to all random initializations of the last layer weights. Consider the cases where A_{ni} is positive so that the gradient is non-trivial:

$$\langle -\partial_{A_{ni}} \mathcal{L} \rangle \equiv \int dW P(W) \sum_k^{N_c} W_{ki} \left(t_{nk} - p_{nk}\right) \tag{E.7}$$

Assuming W_{ki} follows a uniform random distribution from say $(-1, 1)$, then

$$\langle -\partial_{A_{ni}} \mathcal{L} \rangle = 0 - \int dW \sum_k^{N_c} W_{ki} p_{nk} \tag{E.8}$$

$$= -\sum_k^{N_c} \int_{-1}^{1} \prod_{k',i'} dW_{k'i'} W_{ki} p_{nk} \tag{E.9}$$

$$\equiv -\sum_k^{N_c} \int_{-1}^{1} \prod_{(k',i') \neq (k,i)} e^{W_{k'i'} R(A_{ni'})} dW_{k'i'} I_{n,k} \tag{E.10}$$

$$\tag{E.11}$$

where,

$$I_{n,k} \equiv \int_{-1}^{1} dW_{ki} W_{ki} \frac{e^{W_{ki} A_{ni}}}{\sum_{j=1}^{N_c} e^{y_{nj}}} \tag{E.12}$$

$$\equiv \int_{-1}^{1} dW_{ki} W_{ki} f(W_{ki}) \tag{E.13}$$

where,

$$f(W_{ki}) \equiv \frac{1}{C_{ki} e^{-W_{ki} A_{ni}} + B_{ki}} \tag{E.14}$$

Note that $A_{ni} > 0$ from E.6. In addition, both C_{ki} and B_{ki} are positive constants independent of W_{ki}:

$$C_{ki} = \sum_{j \neq k}^{N_c} e^{\sum_{\ell=1}^{N_e} W_{j\ell} R(A_{n\ell})} > 0 \tag{E.15}$$

$$B_{ki} = e^{\sum_{\ell \neq i}^{N_e} W_{j\ell} R(A_{n\ell})} > 0 \tag{E.16}$$

With these positive constants, it is easy to see that

$$I_{n,k} = \int_0^1 dW_{ki} W_{ki} (f(W_{ki}) - f(-W_{ki})) > 0 \tag{E.17}$$

because for each positive value of $W_{ki} \in (0, 1)$ we have $f(W_{ki}) > f(-W_{ki})$. Therefore:

$$\boxed{\langle -\partial_{A_{ni}} \mathcal{L} \rangle < 0} \tag{E.18}$$

We have shown that the gradient at initialization on the pre-activations of the feature layer neurons are negative, assuming cross entropy loss and that the weights follow a uniform distribution at initialization.

Even though we only show this rigorously for the feature layer neurons, the intuition we are establishing here could be more general: if a neuron's activation causes an increase in the loss, then it will receive a negative gradient. We have shown that, at the beginning of the training, on average non-zero activations of randomly initialized neural networks are more likely to cause the loss function to go up rather than down. Therefore all activations receive a downward pressure from the gradients on average. This provides some hints to why sparsity may arise across different data types, including realistic data, random input, random labels as well as infinite data.

More generally, we conjecture that for most losses, the entropy of larger losses is always bigger compared to smaller losses. This is perhaps made clear for cross entropy loss in the derivation above. To provide another example, at any given value of a L2 loss, the volume of the space with larger loss is always infinite while the volume of the space with smaller loss is finite (the volume of a d-dimensional ball). Thus at initialization a non-zero neuron activation is more likely push the result towards the higher entropy regime, resulting in higher loss. This in turn causes gradient descent to push the neuron activations to negative values at early stages of training.

For each training example, we speculate that a small subset of neurons' activations happen to reduce the loss, then this neuron will be strengthened by this gradient and is more likely to survive the general downward pressure demonstrated above, thus causing the emergence of sparse activations in neural networks.

F Insights from Sparsity in MLPs

We study the sparsity of one of the simplest neural networks: 2-layer MLPs. As with transformers, we train the network with different types of data and study their effect on sparsity. We uncover interesting interplay between over/under-fitting and sparsity. In particular, we show that 1) in the extreme under-fitting limit, the model becomes completely sparse. This is consistent with our analysis in Section E; 2) as the model capacity increase and the model can fit more data, the sparsity level decreases; 3) the sparsity level peaks roughly when the model is just large enough to memorize the data; 4) in the overfitting regime, larger model leads to more sparsity.

We setup the experiment using 3 different datasets, the standard MNIST dataset, the random dataset where each pixel of MNIST is replaced by an i.i.d. uniform random number and each label is replaced by a random class amongst 10. However the input-label correlation is fixed for each training data entry so a large enough model trained long enough can memorize this correlation. We also have the infinite data regime where the random inputs and labels are generated on-the-fly, represeting a random dataset of infinite size.

We train MLPs with different hidden sizes: 32, 128, 512, 2048, 8192, 32768 and 131072 using three different optimizers: SGD, SGD with momentum and Adam for 200 epoches (as measured by the number of training data iterations in the MNIST and random data case), which is enough time for most of our metrics to converge. We report training accuracy and non-zero rate in the hidden dimension at the end of the trainings in F.1.

We first observe that all models becomes completely sparse in the infinite data case. During training we observe a smooth and monotonic decay of non-zero rates as training goes on. This phenomenon is consistent with the intuition developed by our theoretical analysis in Section E, where discrepancy between the model's prediction and the labels will put negative gradients on activations and drive sparsity.

At the other end of the spectrum, after the model enters the overfitting regime, for both MNIST and random data, increasing widths further will lead to more sparsity. This is consistent with our findings in transformer models.

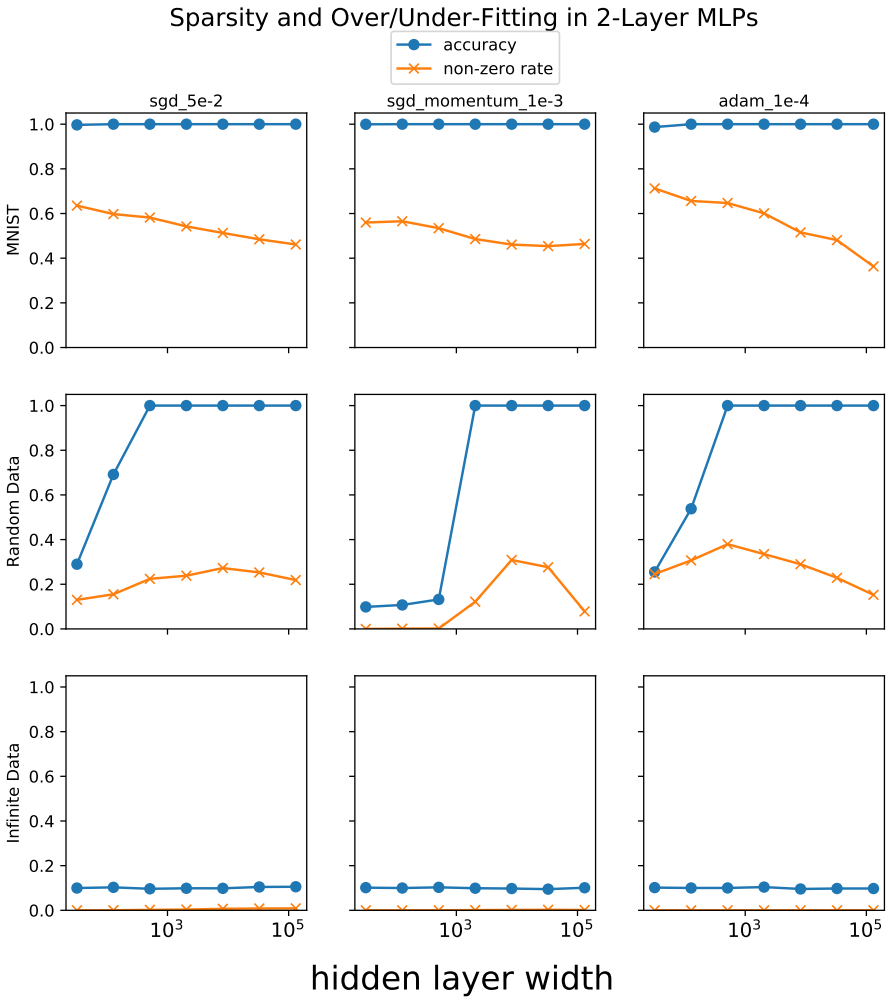


Figure F.1: The interplay between model width, training accuracy and sparsity in 2 layer MLPs of different widths, comparing different data types and optimizers. We report results after 200 epoches of training where convergence has mostly been achieved in these metrics. Random data refers to input images where each pixel follows an i.i.d. uniform random distribution while the labels are randomly selected from 10 classes. But the input-label relation is fixed for each image of the dataset. Infinite data refers to cases where the input random image and output random labels are randomly generated on-the-fly.

These two observations indicate an interesting transition at the regime where the model is just able to memorize the training data. From this point, increasing the model size leads to more sparsity, descreasing the model size leads to more underfitting, which also leads to more sparsity, therefore the non-zero rate has to peak around this regime. This is exactly what we found empirically with random data in Figure F.1.

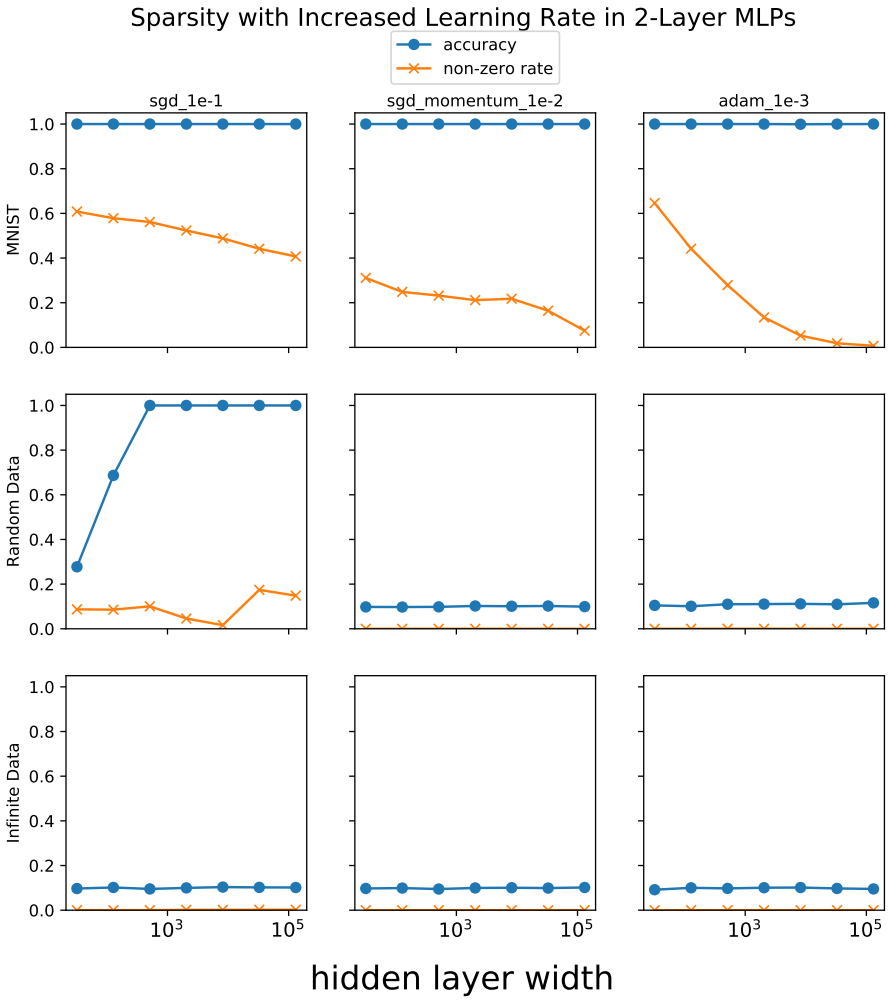


Figure F.2: Comparing to Figure F.1, we show that with increased learning rate, on same data types models tend to become more sparse. For random data this effect is so strong that it causes the model to collapse to completely sparse before it could memorize the data.