Emergence of Sparse Representations from Noise

Trenton Bricken [1 2] **Rylan Schaeffer** [3] **Bruno Olshausen** [2] **Gabriel Kreiman** [4]

Abstract

A hallmark of biological neural networks, which distinguishes them from their artificial counterparts, is the high degree of sparsity in their activations. This discrepancy raises three questions our work helps to answer: (i) Why are biological networks so sparse? (ii) What are the benefits of this sparsity? (iii) How can these benefits be utilized by deep learning models? Our answers to all of these questions center around training networks to handle random noise. Surprisingly, we discover that noisy training introduces three implicit loss terms that result in sparsely firing neurons specializing to high variance features of the dataset. When trained to reconstruct noisy-CIFAR10, neurons learn biological receptive fields. More broadly, noisy training presents a new approach to potentially increase model interpretability with additional benefits to robustness and computational efficiency.

1. Introduction

A striking difference between biological and artificial neural networks is *activation sparsity*. The brain is highly sparse, with an estimated 15% of neurons firing at any given time (Attwell & Laughlin, 2001), whereas deep learning models are often dense. A unifying understanding of this difference is elusive, since there are advantages, disadvantages and unclear implications of sparse representations (Olshausen & Field, 2004).

A popular reason for sparsity in the brain is metabolic efficiency, since action potentials consume ∼20% of the brain's energy (Sterling & Laughlin, 2015; Sengupta et al., 2010). This theory is supported by the field of sparse coding which enforces an L_1 penalty on neuron activations and results in neurons learning biological receptive fields when trained on a reconstruction task (Olshausen & Field, 1997).

In this work, we advance an alternative theory that the need to handle noise is a key driver behind activation sparsity.Removing the L_1 activation penalty and simply adding random isotropic Gaussian noise to the data also produces sparse activations and biological receptive fields.

Previous work found that by taking a first order Taylor approximation, noisy training added a single penalty to the loss function: minimize the Frobenius norm of the model's Jacobian. Formally, $\min_\theta ||\delta f_\theta(\mathbf{x})/\delta \mathbf{x}||_F^2$ where $\mathbf{x}$ is the input data and $f_\theta(\mathbf{x})$ is any non-linear function that maps the input to the output (Bishop, 1995; Camuto et al., 2020). In words, this means the model outputs should be robust with respect to small input perturbations.

However, this result leaves much unexplained. Empirically, training networks with this additional loss term fails to produce sparsity or biological receptive fields. Theoretically, this loss makes no predictions for how the network parameters should change and being a Taylor approximation it only holds in the small noise limit. Moreover, because the Frobenius norm is an L_2 penalty instead of L_1, this adds to the mystery of why noise results in sparse activations.

Motivated by this discrepancy, we avoid the Taylor approximation and disentangle more nuanced effects that noisy training implicitly has on the loss function. To summarize, noisy training adds three additional terms that the loss seeks to *maximize*:

$$\begin{aligned}\text{Noise Terms} = \text{Maximize}([&\text{Neuron Sparsity}] \\ &+ [\text{Neuron Activation Margin}] \\ &+ [\text{Specialized Model Weights}])\end{aligned} \tag{1}$$

The theoretical effects of Eq. 1 on what each neuron learns are summarized in Fig. 1. The ReLU activation function plays a key role. By default, each neuron should have a very negative pre-activation so that it is turned off and unperturbed by any noise. When a neuron must turn on to help in the reconstruction task, it should "jump" from being very negative to very positive in order to maximize the margin around the ReLU activation threshold of 0.

This threshold is where the noise has a non-linear effect on

[1]Systems, Synthetic and Quantitative Biology, Harvard University [2]Redwood Center for Theoretical Neuroscience, University of California, Berkeley [3]Computer Science, Stanford University [4]Programs in Biophysics and Neuroscience, Harvard Medical School. Correspondence to: Trenton Bricken <trentonbricken@g.harvard.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

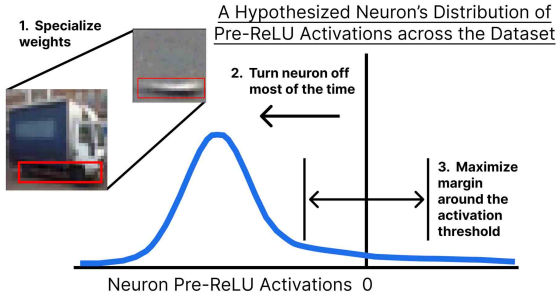


Figure 1: **The three loss objectives introduced by noisy training.** We show the distribution of activations across the dataset for an idealized, hypothetical neuron to clarify the loss terms. 1. The neuron should learn specialized weights, in this case we plot a real example of a Gabor filter that will fire strongest for edges like that along the truck bottom. 2. The neuron should be negative such that it spends a majority of its time off (blocking the influence of noise). 3. The neuron when it is activated should "jump" over the ReLU activation threshold to a large positive value, reducing the ability for noise to switch it off. This produces the long right hand tail.

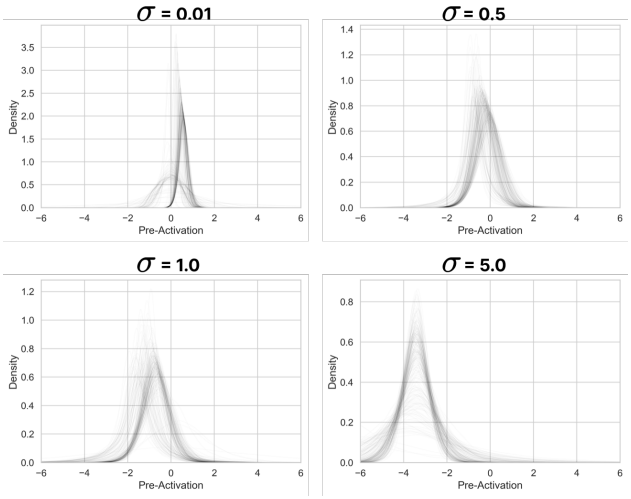


Figure 2: **Pre-Activation Distributions of Shallow Denoising Autoencoders on CIFAR-10 pixels.** Distribution of 250 units' pre-activation values, randomly sampled from 10,000 neurons. The bias for each neuron becomes more negative and tails grow longer in proportion to the noise.

the model output and is avoided by the neuron's receptive field only modelling high variance regions of the distribution. Finally, each neuron should learn receptive fields that decorrelate the effects of noise on the activations, resulting in specialization to non-redundant subsets of the high variance data features.

Empirically, we find that trained networks follow these theoretical predictions (e.g., Fig. 2). In proportion to the noise variance up to a cutoff, each neuron learns a negative bias so that it is off by default. Many of the neurons also learn biological receptive fields known to capture the high variance features of natural images and have low activation covariance. Finally, the bias terms and weight norms synchronize such that a sparse $\sim k$ neurons fire for any input datum.

This form of sparsity is distinct from sparsity created via an L_1 activation penalty where many of the neurons in the network are *dead* and never fire for any input. Dead neuron sparsity is misleading and equivalent to having a pruned, smaller network that is densely firing. We distinguish this alternative form of sparsity throughout our work. Note that we consider *activation* sparsity, not *weight* sparsity, which can contribute to activation sparsity but is fundamentally different.

We show that our findings generalize across a number of deep learning tasks, datasets, architectures, and can even be used as a pre-training task. Pretraining with noise then removing it still results in sparse activations and, aside from the highest noise variances, has no final performance cost.

Noisy training is appealing because of its simplicity and the inductive biases it introduces. Alternative methods to create activation sparsity are varied but all have in common explicitly penalizing or preventing dense activations and need to account for the issues this introduces (Bricken et al., 2023; Gregor & LeCun, 2010; Elhage et al., 2022a; Srivastava et al., 2014; Martins & Astudillo, 2016). For example, sparse coding with an L_1 penalty needs to be combined with constraints on weight norms to avoid the model "cheating" by shrinking its weights (Olshausen & Field, 1997).

Meanwhile, methods like Top-k that force exactly the k most active neurons to turn off must pre-determine their k value and need to slowly introduce this constraint (Bricken et al., 2023). Noisy training endogenously incentivizes sparsity, in addition to weight norm regularization and feature specialization. Noise also results in different receptive fields from sparse coding, resulting in a transition from Gabors to center-surround (Karklin & Simoncelli, 2011). This latter finding suggests that noisy training could potentially be a different approach to removing neuron polysemanticity (Elhage et al., 2022a;b).

2. Related Work

Injecting noise into training data has a rich history, dating back as early as Sietsma & Dow (1991). Noise injection has been interpreted as a form of model regularization to help avoid overfitting (Zur et al., 2009) and improve generalization (Sietsma & Dow, 1991; Matsuoka, 1992). Interest in training with noise rose recently due to de-noising autoen-

coders playing a critical role in modern diffusion models (Goodfellow et al., 2015; Song & Ermon, 2019).

Poole et al. (2014) discovered that noise injection results in sparse activations but focused on comparisons to Dropout (Srivastava et al., 2014) and noise injection at different model layers. Our work enriches their contributions by correcting erroneous equations[1], investigating the network sparsity mechanism, deriving more detailed implicit loss terms, exploring the generality of findings across model architectures, and finding the emergence of biological receptive fields (Olshausen & Field, 1997).

It was previously discovered that de-noising auto encoders, when trained on the MNIST hand written digits dataset (LeCun & Cortes, 2005), learn receptive fields that can be interpreted as Gabor-like (Vincent et al., 2008; Chen et al., 2014). However, because MNIST digits consist of strokes, it is easy to interpret the receptive fields as stroke detectors for this specific dataset, rather than generalized Gabor filters appearing in V1 (Sterling & Laughlin, 2015). To the best of our knowledge, our work is the first to find the emergence of biological receptive fields from MLPs trained on naturalistic whole CIFAR10 images. Regarding the formation of receptive fields found by other models such as AlexNet (Krizhevsky et al., 2012), what is striking about our receptive fields is not how quantitatively similar they look to true biological receptive fields but that they emerge at all just from noisy training.

Another interesting connection exists with the implicit loss terms found in the Kullback-Leibler Divergence (KLD) of variational autoencoders (VAEs). Intuitively, VAEs can be thought of as introducing noise through stochastically sampling from their latent space. This noise is enforced by the prior distribution over the latents, incentivizing the model to parameterize its latent distribution with non-zero variance. Kumar & Poole (2020) use the Jacobian approximation to make this relationship explicit while Chen et al. (2018) decompose the KLD loss term into three terms which are closely related to maximizing the neuron activation margin and de-correlating neuronal activations. The sparsity term is notably absent but should only appear with rectified non-linearities that are uncommon in VAE latent spaces.

Approaches to enforce activation sparsity include using Top-k activation functions (Ranzato et al., 2007; Makhzani & Frey, 2014; Ahmad & Scheinkman, 2019), novel regularization terms (Kurtz et al., 2020; Yang et al., 2020) and other approaches (Andriushchenko et al., 2022; Schwarz et al., 2021; Molchanov et al., 2017; Srivastava et al., 2014; Elhage et al., 2022a; Martins & Astudillo, 2016).

[1]The authors confirmed and kindly helped us to verify the limitations of Eqns. 4 & 7 via private correspondence (Appendix F has a corrected derivation).

We state that our model is more robust simply because it has been trained with random noise perturbations (Kireev et al., 2021). However, we do *not* extend this claim to include *adversarial* robustness due to both noise and sparsity causing "gradient masking" that confounds canonical gradient based adversarial attacks. For example, Xiao et al. (2020) used Top-k activation sparsity to boost adversarial robustness. However, this was overturned by Tramèr et al. (2020), showing this was the result of gradient masking and alternative attacks remained effective. Li et al. (2018) claim similar adversarial robustness from additive noise and acknowledge the gradient masking confounder, however, it is unclear if it is adequately accounted for.

Learning reconstructions of training data with sparse activations is the mainstay of sparse coding (for a formal introduction, see App. A). Prior work showed that combining the sparse coding L_1 penalty with noisy training produces biological receptive fields (Karklin & Simoncelli, 2011; Doi et al., 2012). Karklin & Simoncelli (2011) notably used a non-linear model that not only learned ReLU like activation functions but also negative activation thresholds just like the bias terms our networks learn. The model also shows, in agreement with our findings, that increasing noise causes the transition from Gabors to center-surround receptive fields. Our work brings these findings into the field of deep learning (e.g., using stochastic gradient descent, deeper models, full images) and simplifies the model assumptions by considering only training with noise applied to the input data rather than inside the model, and without additional activation penalties or weight norm constraints.

Finally, sparsely activated networks, when trained with noise, are closely related to the Sparse Distributed Memory (SDM) and Modern Hopfield Network associative memory models that activate a sparse subset of neurons and handle noisy queries (Kanerva, 1988; Krotov & Hopfield, 2016). SDM in particular can be viewed as a de-noising autoencoder and can also be written as an MLP using the Top-k activation function (Bricken et al., 2023; Keeler, 1988).

3. Results

Shallow Denoising Autoencoder - As a starting point, we train a single hidden layer ReLU autoencoder, with additive random noise ϵ sampled from a zero mean, isotropic Gaussian with variance σ^2. The corrupted datum $\tilde{x} = x + \epsilon$ is then fed through the encoder and decoder, and the training objective is to reconstruct the noiseless input. For the single hidden layer case of Eq. 2, the encoder and decoder weight matrices and bias terms are: $W_e \in \mathbb{R}^{m \times n}, \mathbf{b}_e \in \mathbb{R}^m, W_d \in \mathbb{R}^{o \times m}, \mathbf{b}_d \in \mathbb{R}^o$ where n is the input dimension, o is output dimension, and m is the number of neurons in the hidden layer. Our loss function uses the mean squared error between the original image and reconstruction across our full

dataset, X.

$$\begin{aligned}\tilde{\mathbf{x}} &= \mathbf{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(0, \sigma^2 I) \\ \mathbf{z} &= W_e \tilde{\mathbf{x}} + \mathbf{b}_e \\ \mathbf{h} &= \varphi(\mathbf{z}) \\ \tilde{\mathbf{y}} &= W_d\, \mathbf{h} + \mathbf{b}_d \\ \mathcal{L} &= \sum_{\mathbf{x} \in X} ||\mathbf{x} - \tilde{\mathbf{y}}||^2.\end{aligned} \tag{2}$$

where φ is the element-wise ReLU nonlinearity. We use Kaiming randomly initialized weights (He et al., 2015) and train until the fraction of active neurons converges.[2] We primarily use the CIFAR10 dataset of 50,000 images with 32x32x3 dimensions, training either on the raw pixels (flattening them into a 3,072 dimensional vector) or latent embeddings of 256 dimensions, produced by a ConvMixer pretrained on ImageNet (Trockman & Kolter, 2022; Russakovsky et al., 2015). All code and training parameters can be found at: https://github.com/TrentBrick/SparsityFromNoise.

To visually communicate the relation between noise variance and reconstruction quality, we visualize the reconstructions of shallow autoencoders trained directly on CIFAR10 pixels in Fig. 3. Unsurprisingly, the more noise that is applied, the worse reconstruction performance is. This is especially true for noise levels large enough to make a competing image closer to the noisy input than the ground truth input image.

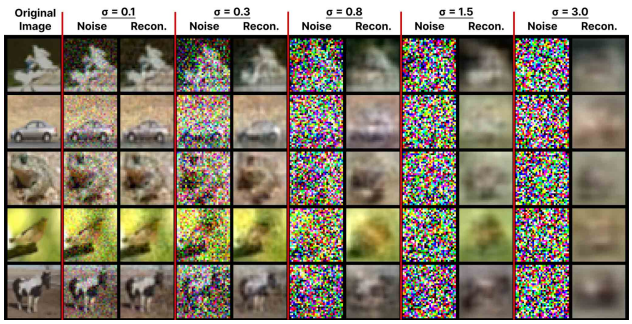


Figure 3: **Example CIFAR10 reconstructions.** Example reconstructions obtained at different noise levels for five randomly selected images from the test data. The network at $\sigma \geq 0.8$ qualitatively transitions from fuzzy reconstructions to more general image details.

We summarize our derivation of the three implicit loss terms introduced by noisy training where additional steps can be found in Appendix E. We then present empirical results and investigations confirming that noisily trained networks agree with our analysis.

[2]This happens after the train and test losses plateau

Theory - The key to our result is taking an expectation over the noise (Bishop, 1995). Working with scalars for clarity:

$$\begin{aligned}\text{Loss} &= \frac{1}{D} \sum_{x \in X} \mathbb{E}_\varepsilon \left[\sum_i^o (x_i - \tilde{y}_i)^2 \right] \\ &= \frac{1}{D} \sum_{x \in X} \sum_i^o r_i^2 - 2 r_i \mathbb{E}_\varepsilon[\xi_i] + \mathbb{E}_\varepsilon[\xi_i^2]\end{aligned} \tag{3}$$

where D is the size of the dataset, $r_i = x_i - \bar{y}_i$, and $\xi_i = \tilde{y}_i - \bar{y}_i$ is the difference between $\tilde{y}_i$, the output produced by the input with noise ε, and the hypothetical output without noise $\bar{y}_i$. Keep in mind that because the input noise is independent of the data, driving the error from noise $\xi_i \to 0$ will maximize the quality of the reconstruction.

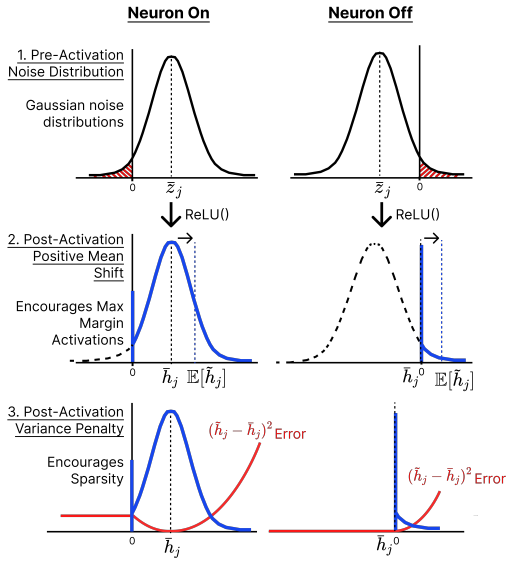


Figure 4: **Intuition for why noise results in max margin activations and sparsity.** The columns reference the noise free neuron being on (left, $\bar{z}_j > 0$) or off (right, $\bar{z}_j \leq 0$). Going row-wise: #1. Shows the noise distribution around $\bar{z}_j$. #2. Post-ReLU, the truncated tail results in a positive mean shift $\mathbb{E}_\varepsilon[\tilde{h}_j] \geq \bar{h}_j$ (black vs blue vertical dotted lines). #3. The variance $\mathbb{E}_\varepsilon[\eta_j]$ depends on how much of the distribution is positive.

Defining $\xi_i = \sum_j^m W_{d_{i,j}} \eta_j$, where $\eta_j = \tilde{h}_j - \bar{h}_j$, we can write $\mathbb{E}_\varepsilon[\xi_i] = \sum_j^m W_{d_{i,j}} \mathbb{E}_\varepsilon[\eta_j]$. The noise penalty ultimately results from the terms $\mathbb{E}_\varepsilon[\eta_j]$, to maximize the activation margin, and $\mathbb{E}_\varepsilon[\eta_j^2]$, to both sparsify activations and de-correlate the encoder weights. We show in Appendix E that $\tilde{h}$ is a rectified Gaussian distribution (Salimans, 2016) and $\mathbb{E}_\varepsilon[\eta_j] \geq 0$. Intuitively, this is because, as shown in Fig. 4 going from the top row to the second row, the noise creates a Gaussian distribution around the pre-ReLU neuron activation $\bar{z}_j$ which gets truncated where it crosses 0. This destroys symmetry and results in three parts to the Gaussian,

the two tails, one of which is now set to 0, and the center which is still symmetric. Taking an expectation, the center cancels with itself but only one of the tails on the positive right hand side is non-zero resulting in the expectation becoming more positive. This positive shift is proportional to how close $\bar{z}_j$ is to the 0 activation threshold and given by the integral:

$$\mathbb{E}_\varepsilon[\eta_j] = \int_{t=|\bar{z}_j|}^{\infty} N(x; 0, ||\mathbf{w}_{e_j}||\sigma)(x-t)dx \geq 0, \quad (4)$$

where $N()$ is the PDF of a normal distribution. Turning to our other term $\mathbb{E}_\varepsilon[\xi_i^2]$, we can expand it as:

$$\mathbb{E}_\varepsilon[(\sum_j^m W_{d_{i,j}}\eta_j)^2] = \mathbb{E}_\varepsilon[\sum_j^m W_{d_{i,j}}^2\eta_j^2 + \sum_{k\neq j}^m W_{d_{i,j}}W_{d_{i,k}}\eta_j\eta_k] \quad (5)$$

This results in integrals over the error terms summarized by the bottom row of Fig. 4 and given formally in Appendix E. There are three observations to be made from these terms: 1. The penalty when the neuron is on ($\bar{h}_j > 0$) is strictly greater than when it is off. 2. When the neuron is off, $\bar{z}_j$ should be as negative as possible to avoid the noise flipping it positive. 3. The cross term $W_{d_{i,j}}W_{d_{i,k}}\mathbb{E}_\varepsilon[\eta_j\eta_k]$ incentivizes both minimizing the sum of the decoder weights outer product $||\mathbf{w}_{d_i}\mathbf{w}_{d_i}^T||$ and de-correlating the encoder weights. This is because the term is only present when both neurons are turned on and the probability of this is reduced as $\bar{z}_j \to -\infty$.

Because $\bar{z}_j$ and $\bar{z}_k$ receive the same noise sample they are not independent and their correlation is a function of the cosine similarity between their encoder weights $\mathbf{w}_{e_j}$ and $\mathbf{w}_{e_k}$. Therefore, a way to utilize both neurons in the model (increasing model capacity) but avoid this cross term is by making their encoder weights specialize to unique, localized features of the data.

It is important to note that the encoder weights cannot increase their activation margin by simply increasing their L_2 norm. This is because, increasing their scale will linearly increase noise as much as it does the activation. Therefore, the way to both reduce noise and maximize the activation margin is by having the weights specialize to distinct dataset features, ignoring noise across most of the input while being very activated by unique features. Combining this with the neuron's bias term being strongly negative is a good way to ensure the neuron is not otherwise activated. These desiderata predict that neurons should have activation distributions across the data distribution like in Fig. 1 where it has a negative mean and a long right hand tail (shown empirically in Fig. 2). Ideally, the tail would in fact be a another mode resulting in a bimodal distribution that straddles the activation threshold, however, this depends upon the distribution of the particular dataset features the neuron specializes to. We now support the theoretical predictions from our derivation with empirical results from training neural networks with noise.

Symmetric Mean Zero Noise Induces Sparse Coding Networks - Our headline Fig. 5 shows that a single hidden layer ReLU network with 10,000 neurons learns to become a sparse coding network with a smaller k number of neurons activated by any input as the amount of noise increases. This network was trained on the CIFAR10 embeddings to mimick training within a deeper network.

While Fig. 5 only shows the mean number of neurons that are active across all 50,000 CIFAR10 inputs per epoch, Fig. 6 shows the fraction of active neurons for every training batch. This reveals that as noise increases, the variance for how many neurons are active shrinks, making the mean an accurate summary statistic. Note that there are no dead neurons. Fig. 6 also shows the bias terms for each neuron and the L_2 norm of their encoder weights W_e. It is clear that these values all synchronize to a value decided on by the network and are responsible for the sparse k neurons remaining active after the ReLU nonlinearity is applied.

Figure 5 also shows what looks like a delayed phase transition for the higher noise training to sparsify (e.g. $\sigma = 10$, brown line). This delay arises from the network waiting for the L_2 norms of its encoder weights W_e to become sufficiently small and bias terms sufficiently negative to silence more than 50% of the neurons (bottom row of Fig. 6). Higher noise levels increase the L_2 norm of the data, which the weights must counteract to keep the neuron activations small enough so that bias term can have an effect (each bias is initialized at 0 and can only become negative so quickly). Appendix B confirms this explanation by initializing weights with smaller L_2 norms that result in faster sparsification.

Shared Bias Terms and Inhibitory Interneurons - In biological neural networks, sparsity is achieved via inhibitory interneurons that suppress all but the most active neurons from firing (Haider et al., 2010). Theoretical and empirical results support their involvement in both silencing noise and separating neural representations. Examples include horizontal interneurons in the retina (Sterling & Laughlin, 2015) and Golgi interneurons in cerebellar-like structures (Fleming et al., 2022; Lin et al., 2014; Xie et al., 2022).

To test the observed transition into a sparse coding network, we modify the encoder bias from $\mathbf{b}_e$ to $b_e\mathbf{1}$, where b_e is a (shared) scalar and $\mathbf{1}$ is the all ones vector. This results in near identical model performance and sparsity.

It is interesting that the bias terms of all the neurons converge to a particular negative scalar and L_2 norms of the weights also converge to a different scalar value (Fig. 6). This results in any given data point, producing activations

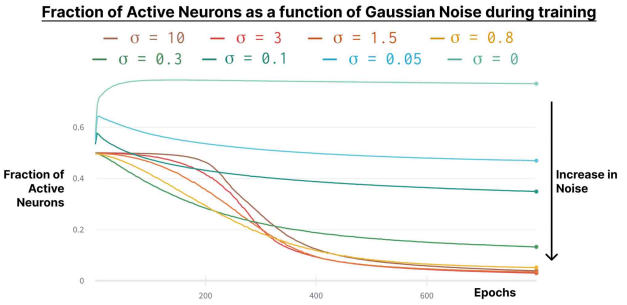


Figure 5: **The positive relationship between activation sparsity and noise.** The average fraction of neurons active during each latent CIFAR10 training batch over 800 epochs. Each line corresponds to training randomly initialized networks with different noise levels σ denoted by different colors. We show the average of three different random seeds and their standard error of the mean (not visible as variance is so low). The higher noise levels take longer to sparsify because the noise results in larger magnitude activation values that require more parameter updates to counteract.

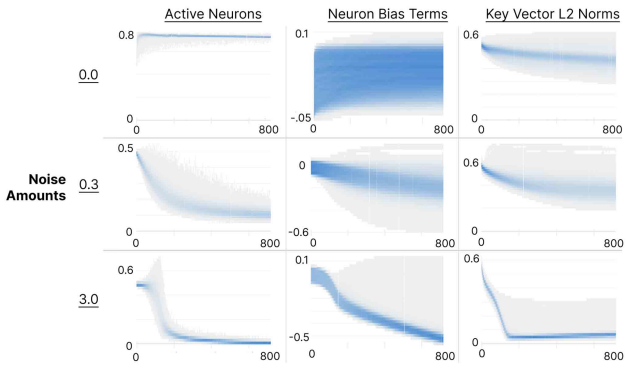


Figure 6: **Noise induces the formation of sparse coding networks.** Shown column-wise are the fraction of neurons that are active for each training batch (Left), the values of each of the 10,000 bias terms (Middle) and the L_2 norms of the neuron encoder weights W_e (Right), all as a function of training epochs. Each is shown for three different representative noise levels (rows). These plots show each input to the network and use the density of blue to represent how many times a particular value occurs.

that all fall within a particular range of values that is calibrated with the negative bias term such that only a sparse k neurons are active for any given input.

This is analogous to the operation of an inhibitory interneuron implementing a Top-k activation function. This dynamic transition during learning into a Top-k network also avoids complications associated with an explicit Top-k implementation (App. D) (Bricken et al., 2023; Makhzani & Frey, 2014; Ahmad & Scheinkman, 2019).

Biological Receptive Fields - Looking closer at the sparse solutions found by our ReLU networks, we discovered that for $\sigma \geq 0.3$, neurons begin to form receptive fields reminiscent of V1 Gabor filters and center-surround retinal ganglion cells (Sterling & Laughlin, 2015; Olshausen & Field, 1997). Figure 7 shows the receptive fields of the 125 neurons most activated (sorted from left-to-right and top-to-bottom) by a noisy image of a car when $\sigma = 0.8$. The longer training progressed, the more the receptive fields resembled Gabor-like and center-surround functions. See Appendix C for more receptive fields at varying noise levels and over the course of training.

Relations to Sparse Coding - Training with the L_1 penalty and enforcing a unitary L_2 norm on all weights also resulted in biological receptive fields (Fig. 14). The receptive fields were all Gabors or textures with the latter likely existing due to training on full CIFAR10 images instead of the typical smaller 8x8 patches (Olshausen & Field, 1997; Karklin & Simoncelli, 2011). The L_2 weight normalization is crucial to obtaining biological receptive fields; otherwise, weights can change to avoid the activation penalty.[3] Karklin & Simoncelli (2011) showed, in agreement with our results, that an increase in noise causes a transition from Gabors to center-surround receptive fields. Both Gabors and center-surround fields capture aspects of natural image statistics. However, center-surround covers a smaller portion of the image and we hypothesize that they are thus less sensitive to noise, resulting in their emergence with noisy training.

Another difference between noisy training and L_1 is how sparsity emerges. Noisy training uses the synchronization between its weight norms and negative bias terms without killing off neurons while L_1 uses negative neuron weights to kill off most of its neurons. However, for reasons outlined in Appendix I.2, we find that eventually our noisy training will also kill off many neurons when using the Adam optimizer. This means a different trajectory on the optimization landscape is followed but that it is still possible to kill many neurons without affecting performance. Finally, the use of a shared bias term combined with the ReLU activation function can be related to the lambda coefficient that scales the sparse coding L_1 activation penalty (see Appendix G).

Noisy Pre-Training Retains Model Sparsity - Investigating our noise-trained sparse networks, we find that even after removing noise, the networks remain sparse. We train our models in the same way as in Fig. 5 but at epoch 800 linearly anneal the noise to zero over the next 800 epochs. Figure 8

[3]Interestingly, the L_1 penalty is not completely avoided and the network still sparsifies, however, it does not learn biological receptive fields.

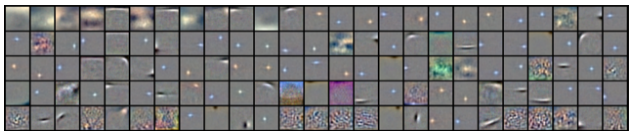


Figure 7: **Biological receptive fields form with noise.** Networks trained on CIFAR10 pixels with $\sigma = 0.8$ Gaussian noise. The most active 125 neuron receptive fields are reshaped into 32x32x3 and re-scaled so their values span the pixel range $[0, 255]$. Neurons are sorted by activity levels (top left is most active, going across rows then down columns) for a randomly chosen car image. Starting at $\sigma \geq 0.1$ and particularly for $\sigma = 0.8$ (shown here) we observe both Gabor filters and center-surround receptive fields.

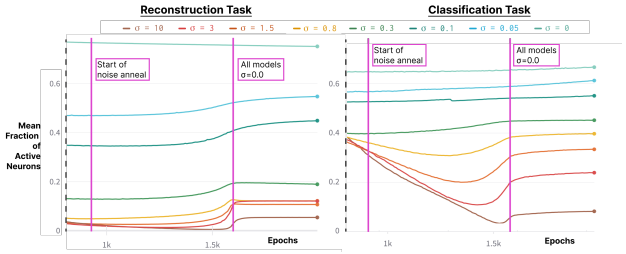


Figure 8: **Noisy pretraining retains network sparsity.** The vertical purple lines denote the start and end of noise annealing (epochs 800 and 1,600) where each noise level is linearly decayed to $\sigma = 0$. For both reconstruction (left) and classification (right) the networks remain highly sparse even after noise is removed. Moreover, there is no noticeable performance difference between any of the networks and moderate noise models even perform slightly better than the noise free baseline. We truncate the x-axis to start at epoch 600 (dotted vertical gray line) for the sake of clarity. The classification networks continue to sparsify even when noise annealing has started only because, unlike in the reconstruction task, the sparsity level did not converge within the first 800 epochs.

shows that the sparsity levels remain largely unperturbed.

For classification (Fig. 8, right), it is clear that while the correlation between noise and sparsity holds, it is not quite as strong. This is likely due to the fact that classification only needs to cluster the data into the ten CIFAR10 labels, meaning interference between neural representations is less of an issue and denser representations can be used.

There is also no reduction in reconstruction or classification performance for this single MLP setting.[4] In fact, for classification both $\sigma \in \{0.1, 0.3\}$ get 94.3% validation accuracy instead of the baseline 93.4%, showing slightly better generalization while also being more sparse (66% for baseline versus 55% for $\sigma = 0.1$ and 45% for $\sigma = 0.3$, see Appendix H.1). Note that networks with $\sigma \geq 0.8$ for reconstruction and $\sigma \geq 3.0$ for classification, annealing noise results in dead neurons, presumably because the volume of the data manifold shrinks, leaving some neurons behind. However, these dead neurons neither harm task performance nor entirely explain the sparsity levels. See Appendix I.3 for data on the fractions of dead neurons.

Model Ablations and Generalization - We investigate sparsity across experimental conditions in the single hidden layer setting and find that they all reproduce our results (Appendix H.1): (i) datasets: CIFAR10 pixels, CIFAR10 latent embeddings, MNIST, CIFAR100, and SVHN; (ii) two noise distributions: Gaussian and Laplace; (iii) four numbers of neurons: 100, 1,000, 10,000 and 100,000; (iv) two training tasks: reconstruction and classification.

Additionally, we test multiple nonlinearities: sigmoid, GELU, ELU, and Tanh activation functions (Hendrycks & Gimpel, 2016). We find that the asymmetric ReLU, GELU, and ELU activation functions generally produce sparsity across datasets, albeit to slightly different extents while the symmetric sigmoid and Tanh do not (Appendix H.1). This is consistent with our mathematical derivations where noise is minimized by driving pre-activations to near-zero gradient regions of the nonlinearity's domain that correspond to being "off" for the asymmetric ReLU, GELU, and ELU but being either very "on" or very "off" for the symmetric sigmoid or Tanh. On the CIFAR10 pixel dataset, none of the activation functions become as sparse as ReLU but the GELU networks do implement the same sparse coding convergence shown in Fig. 6 and form biologically plausible receptive fields like those of Fig. 7. It is important to note that ReLU is the only truly sparse activation function with the others requiring arbitrary activation thresholds close to zero that are used to label neurons as "off".

Optimizers and Dead Neurons - Exploring the effects of different optimizers, batch sizes, and learning rates produced a number of interesting findings. First, SGD – the only optimizer without an adaptive learning rate – was found to be capable of learning a non-sparse coding solution on the latent CIFAR10 dataset with ∼50% neuron activity and no updating of bias terms. This solution is only found on the easier latent CIFAR10 dataset whereas the CIFAR10 pixels dataset requires all models to sparsify. However, it is nevertheless interesting as it shows how the inductive biases of noise can be counteracted by a combination of a low learning rate and large batch size. We investigate this finding further in Appendix H.2.

Second, Adam was found to produce dead neurons if we continued training beyond the discovery of the sparse coding solution without any effect on training or validation loss.

[4] Deeper MLPs or Transformers also don't show performance costs. Only AlexNet does (Appendix J.3).

The final number of dead neurons created corresponded to the complexity of the dataset with many more for the latent CIFAR10 dataset compared to the raw pixels. We hypothesize that neurons that are on less often are gradually killed off due to the "Stale Momentum" phenomenon. Bricken et al. (2023) discovered that adaptive optimizers like Adam not only update inactive neurons using an out-of-date moving average but also create large gradient spikes when they are re-activated after periods of quiescence (see Appendix D of Bricken et al. (2023) for an investigation of this phenomenon in greater detail).

Crucially, for our single-layer MLP results, the dead neurons appear after the sparse coding solution is found and not beforehand, which would be a misleading source of sparsity. However, this was not the case for some of the deeper models outlined in the next section where dead neurons appear early in training. Changing optimizer and increasing batch size removed dead neurons but still had no effect on training or validation loss (Appendix I.2).

Deeper Models - While our primary focus is on single-layer MLPs, we extend the work to Transformers, AlexNet, and three-layer MLPs.

Transformers - We train small GPT2 models (Radford et al., 2019) with three blocks of interleaved Attention and Feed-forward MLP layers on the WikiText-103 dataset for next token prediction.[5] In front of each MLP, we inject one of the noise levels $\sigma \in \{0.0, 0.05, 0.1, 0.8, 1.5, 3.0, 8.0\}$ to sparsify each of them. At training step 200k, we remove noise to evaluate the effects of noisy pretraining.

Sparsity increased with noise and is retained even after noise is removed. Furthermore, there are no dead neurons and aside from the highest noise $\sigma = 8$, pre-training has no significant effect on model performance (Appendix J.1). The $\sigma = 3$ noisy pretraining results in layers that are $\sim$3x more sparse in the first layer (17% for the baseline vs 6%), $\sim$9x for the second layer (18% vs 2%), and $\sim$3x (30% vs 13%) for the third layer.

AlexNet - Alexnet uses five layers of convolutions followed by three MLP layers. We trained it on raw CIFAR10 image classification for 3.5k epochs with noise $\sigma \in \{0.0, 1.5, 3.0, 5.0, 10.0\}$ applied for the first 1,000 epochs and linearly annealed to 0 over the next 500 epochs, leaving 2,000 epochs of noise-free training. Unlike the Transformer, where we apply noise in front of every MLP layer, we only apply noise to the input pixels.

Interestingly, upon removing the noise, the models recovered perfect training accuracy like the baseline but failed to recover validation accuracy, seeing $\sim$20% reductions (82% baseline vs $\sim$62%). The noisy models remained more sparse but this was misleadingly caused by dead neurons. We first hypothesized that these dead neurons harmed model capacity resulting in poor validation accuracy. However, taking steps to reduce dead neurons I.2 still failed to improve validation accuracy (Appendix J.3). We suspect that the convolutional layers are responsible and incompatible with noisy training but leave further investigation to future work.

Three-Layer MLP - Using 200 neurons per layer, we train for 2,000 epochs on classifying the latent 256 dimensional CIFAR10 embeddings with noise $\sigma \in \{0.0, 0.1, 0.5, 1.5, 3.0, 5.0, 10.0\}$. Noise annealing starts at epoch 500 and reaches $\sigma = 0$ by epoch 1,000. As in AlexNet, we only apply noise to the input layer. The networks showed sparsity in proportion to noise in the second and third layers but not the first.

Unlike with AlexNet but in line with the single-layer MLP and Transformer, once noise was removed, every noisy model saw equal or slightly improved (up to +1%) validation accuracy compared to the baseline (see Appendix J.2). However, like with AlexNet, the sparsity was created by dead neurons. Using SGD or SparseAdam as an optimizer solved this problem and allowed for neurons to implement the sparse coding solution.

Additional experiments injected noise only deeper inside the model, just in front of the second or third layers to see how this may affect the sparse coding solution. Interestingly, we found that this allowed the model to "cheat" the noise injection by making the L_2 norm of its weights and bias terms in the layers preceding the noise very large. This produced very large activation magnitudes that reduced the perturbative effect of noise.

4. Discussion

Our noise-free baselines show that neural networks do not typically choose to become sparse. So why does noisy training induce our network to become a biologically plausible sparse coding network? Fundamentally, the network should want to use as many neurons as necessary to give the best reconstruction of the input data as possible. However, it must trade-off the increase in accuracy from pooling more neurons with increased noise interference which favours sparsity. An increase in noise will create more erroneous low-level activations in neurons that must be ignored to maximize signal-to-noise ratio. Therefore, during noisy training, the model must learn weight vectors to encode information about the data distribution that can be separated from the noise using only ReLU functions. In natural images, for example, certain projections (e.g., with Gabor filters) have high kurtosis which allows them to be easily distinguished from noise via thresholding or "coring" (Simoncelli & Adel-

[5]These Transformers use the HuggingFace implementation that includes LayerNorm and Residual connections.

son, 1996). Similarly, the encoding layer in our network is learning such projections, so that thresholding performs noise rejection while also producing sparse activations.

It is interesting that the number of dead neurons does not have an effect on training or validation accuracy and appears only after the model performance and sparsity have fully converged. Bricken et al. (2023) found Stale Momentum harmed continual learning but it may be a feature instead of a bug when training within a single data distribution by acting as a way to gradually prune less active, unnecessary neurons.

The positive correlation between noise and sparsity is opposite to Transformer Attention and associative memory models such as SDM and Hopfield Networks (Bricken & Pehlevan, 2021; Krotov & Hopfield, 2016). Because these models don't store patterns in a distributed fashion across neurons, they utilize a form of nearest neighbour lookup. In the low noise regime, where the target pattern is nearby, they use sparse activations to retrieve just the target. With more noise, they resort to averaging over many more neurons and patterns, losing accuracy but giving a solution in the correct neighbourhood (see App. K).

Limitations - Regarding the biological plausibility of training with random noise, we acknowledge that some noise in biological systems is dynamic and correlated with neural activity. However, there is also uncorrelated random noise in any biological process. For example, Brownian noise that influences the diffusion of neurotransmitters (Sterling & Laughlin, 2015; Doi et al., 2012). Moreover, for the inputs trained on directly on image pixels, Gaussian and Poisson noise are realistic for modelling photon noise on photoreceptors (Sterling & Laughlin, 2015). We also only provide qualitative comparisons to biological receptive fields rather than formally quantifying these, however, we restate that it is impressive these receptive fields emerge from noisy training alone.

Conclusion - We have shown that simple Gaussian noise results in three implicit loss terms that produce sparse and specialized receptive fields to capture high variance features in the data distribution. Empirical results support our theory and introduce a new approach to sparsify deep neural networks. The fact noise alone results in sparse coding without an explicit sparsity penalty in the loss function suggests that the primary driver behind sparse coding may be the handling of noise with metabolic efficiency as an added benefit, rather than the other way around. Combining noisy training with other approaches to induce sparsity, including taking additional ideas from sparse coding, may result in even higher degrees of sparsity with potential downstream benefits to robustness, continual learning, interpretability, and computational efficiency. More broadly, this work builds a new bridge between artificial and biological neural networks by showing how noise can make them more similar.

Acknowledgements

Thanks to Dr. Beren Millidge, Dr. Felix Petersen, Dr. Fritz Sommer, and Jamie Simon for providing invaluable inspiration, discussions, and feedback. Cluster time for the Transformer and Deep Model experiments was provided by Hofvarpnir Studios. T.B. was supported by the NSF Graduate Research Fellowship Program. R.S. was supported via Stanford's Google Graduate Fellowship in Computer Science Fund. B.O. was supported by NSF grants 1718991 and 2147640. G.K. was supported by NSF grant 1231216 and NIH grant R01EY026025.

Author Contributions

- Trenton Bricken identified the research direction, implemented and conducted most of the experiments and analyses, mathematically derived the theoretical results and wrote the paper with suggestions from co-authors.
- Rylan Schaeffer helped implement and conduct experiments, helped mathematically develop the theoretical results, and helped write the paper.
- Bruno Olshausen advised on experimental results and theory, particularly concerning sparse coding.
- Gabriel Kreiman supervised the project, providing guidance throughout on theory and experiments.

References

Ahmad, S. and Scheinkman, L. How can we be so dense? the benefits of using highly sparse representations. *ArXiv*, abs/1903.11257, 2019.

Ainsworth, S. K., Hayase, J., and Srinivasa, S. S. Git rebasin: Merging models modulo permutation symmetries. *ArXiv*, abs/2209.04836, 2022.

Andriushchenko, M., Varre, A., Pillaud-Vivien, L., and Flammarion, N. Sgd with large step sizes learns sparse features. *ArXiv*, abs/2210.05337, 2022.

Attwell, D. and Laughlin, S. B. An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow & Metabolism*, 21:1133 – 1145, 2001.

Ba, D. E. Deeply-sparse signal representations (ds2p). *IEEE Transactions on Signal Processing*, 68:4727–4742, 2020.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016.

Bishop, C. M. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7:108–116, 1995.

Blitzstein, J. K. and Hwang, J. *Introduction to probability*. CRC Press, Taylor & Francis Group, 2019.

Bricken, T. and Pehlevan, C. Attention approximates sparse distributed memory. *NeurIPS*, 2021.

Bricken, T., Davies, A., Singh, D., Krotov, D., and Kreiman, G. Sparse distributed memory is a continual learner. *ICLR*, 2023.

Camuto, A., Willetts, M., Simsekli, U., Roberts, S. J., and Holmes, C. C. Explicit regularisation in gaussian noise injections. *ArXiv*, abs/2007.07368, 2020.

Chen, M., Weinberger, K. Q., Sha, F., and Bengio, Y. Marginalized denoising auto-encoders for nonlinear representations. In *ICML*, 2014.

Chen, R. T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in vaes. 2018.

Doi, E., Gauthier, J. L., Field, G. D., Shlens, J., Sher, A., Greschner, M., Machado, T. A., Jepson, L. H., Mathieson, K., Gunning, D. E., Litke, A. M., Paninski, L., Chichilnisky, E. J., and Simoncelli, E. P. Efficient coding of spatial information in the primate retina. *The Journal of Neuroscience*, 32:16256 – 16264, 2012.

Elhage, N., Hume, T., Olsson, C., Neel, N., Tom, H., Scott, J., Sheer, E., Nicholas, J., Nova, D., Ben, M., Danny, H., Amanda, A., Kamal, N., Jones, , Dawn, D., Anna, C., Yuntao, B., Deep, G., Liane, L., Zac, H.-D., Jackson, K., Tom, C., Shauna, K., Stanislav, F., Saurav, K., Josh, J., Eli, T.-J., Jared, K., Jack, C., Tom, B., Sam, M., Dario, A., and Christopher, O. Softmax linear units. *Transformer Circuits Thread*, 2022a. URL https://transformer-circuits.pub/2022/solu/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022b. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Fleming, E., Tadross, M. R., and Hull, C. Local synaptic inhibition mediates cerebellar pattern separation necessary for learned sensorimotor associations. *bioRxiv*, 2022.

Goodfellow, I. J., Bengio, Y., and Courville, A. C. Deep learning. *Nature*, 521:436–444, 2015.

Gregor, K. and LeCun, Y. Learning fast approximations of sparse coding. In *ICML*, 2010.

Haider, B., Krause, M. R., Duque, A., Yu, Y., Touryan, J., Mazer, J. A., and McCormick, D. A. Synaptic and network mechanisms of sparse and reliable visual cortical activity during nonclassical receptive field stimulation. *Neuron*, 65(1):107–121, 2010.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

Kanerva, P. *Sparse Distributed Memory*. MIT Pr., 1988.

Karklin, Y. and Simoncelli, E. P. Efficient coding of natural images with a population of noisy linear-nonlinear neurons. *Advances in neural information processing systems*, 24:999–1007, 2011.

Keeler, J. D. Comparison between kanerva's sdm and hopfield-type neural networks. *Cognitive Science*, 12(3):299 – 329, 1988. ISSN 0364-0213. doi: https://doi.org/10.1016/0364-0213(88)90026-2. URL http://www.sciencedirect.com/science/article/pii/0364021388900262.

Kireev, K., Andriushchenko, M., and Flammarion, N. On the effectiveness of adversarial training against common corruptions. In *Conference on Uncertainty in Artificial Intelligence*, 2021.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

Krotov, D. and Hopfield, J. J. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Kumar, A. and Poole, B. On implicit regularization in β-vaes. In *International Conference on Machine Learning*, 2020.

Kurtz, M., Kopinsky, J., Gelashvili, R., Matveev, A., Carr, J., Goin, M., Leiserson, W. M., Nell, B., Shavit, N., and Alistarh, D. Inducing and exploiting activation sparsity for fast neural network inference. 2020.

LeCun, Y. and Cortes, C. The mnist database of handwritten digits. 2005.

Li, B., Chen, C., Wang, W., and Carin, L. Certified adversarial robustness with additive noise. In *Neural Information Processing Systems*, 2018.

Lin, A. C., Bygrave, A. M., de Calignon, A., Lee, T., and Miesenbock, G. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature neuroscience*, 17:559 – 568, 2014.

Makhzani, A. and Frey, B. J. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2014.

Martins, A. F. T. and Astudillo, R. F. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, 2016.

Matsuoka, K. Noise injection into inputs in back-propagation learning. *IEEE Trans. Syst. Man Cybern.*, 22:436–440, 1992.

Molchanov, D., Ashukha, A., and Vetrov, D. P. Variational dropout sparsifies deep neural networks. In *ICML*, 2017.

Olshausen, B. A. and Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

Olshausen, B. A. and Field, D. J. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004.

Poole, B., Sohl-Dickstein, J. N., and Ganguli, S. Analyzing noise in autoencoders and deep networks. *ArXiv*, abs/1406.1831, 2014.

PyTorch. Sparse adam. https://pytorch.org/docs/stable/generated/torch.optim.SparseAdam.html, 2022. Accessed: 2022-05-18.

Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., and Sutskever, I. Better language models and their implications. *OpenAI Blog https://openai.com/blog/better-language-models*, 2019.

Ranzato, M., Boureau, Y.-L., Chopra, S., and LeCun, Y. A unified energy-based framework for unsupervised learning. In *AISTATS*, 2007.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 2011.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

Salimans, T. A structured variational auto-encoder for learning deep hierarchies of sparse features. *ArXiv*, abs/1602.08734, 2016.

Schwarz, J., Jayakumar, S. M., Pascanu, R., Latham, P. E., and Teh, Y. W. Powerpropagation: A sparsity inducing weight reparameterisation. In *NeurIPS*, 2021.

Sengupta, B., Stemmler, M. B., Laughlin, S. B., and Niven, J. E. Action potential energy efficiency varies among neuron types in vertebrates and invertebrates. *PLoS Computational Biology*, 6, 2010.

Sietsma, J. and Dow, R. J. F. Creating artificial neural networks that generalize. *Neural Networks*, 4:67–79, 1991.

Simoncelli, E. P. and Adelson, E. H. Noise removal via bayesian wavelet coring. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 1, pp. 379–382. IEEE, 1996.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *ArXiv*, abs/1907.05600, 2019.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

Sterling, P. and Laughlin, S. Principles of neural design. 2015.

Tramèr, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *ArXiv*, abs/2002.08347, 2020.

Trockman, A. and Kolter, J. Z. Patches are all you need? *ArXiv*, abs/2201.09792, 2022.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *ICML '08*, 2008.

Xiao, C., Zhong, P., and Zheng, C. Enhancing adversarial defense by k-winners-take-all. *arXiv: Learning*, 2020.

Xie, M., Muscinelli, S. P., Harris, K. D., and Litwin-Kumar, A. Task-dependent optimal representations for cerebellar learning. *bioRxiv*, 2022.

Yang, H., Wen, W., and Li, H. H. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. *ArXiv*, abs/1908.09979, 2020.

Zur, R. M., Jiang, Y., Pesce, L. L., and Drukker, K. Noise injection for training artificial neural networks: a comparison with weight decay and early stopping. *Medical physics*, 36 10:4810–8, 2009.

Appendix

A. Sparse Coding Overview

Sparse coding uses an overcomplete basis with an L_1 activation penalty. The (flattened) image reconstruction can be written as $\hat{\mathbf{x}} = \sum_i^N a_i \mathbf{w}_i$, where $\mathbf{w}_i$ is a column of weight matrix W, corresponding to the output weights of a neuron and a_i is the neuron's activity. The objective function aims to minimize the reconstruction error while having sparse neuron activity: $\arg\min_{W,\mathbf{a}} (\mathbf{x} - \hat{\mathbf{x}})^2 + \lambda \sum_i^N |a_i|$. This sparse coding task is an example of a statistical model converging with neuroscience because, in addition to producing sparse representations, the neuron weights learn to become Gabor filters (Olshausen & Field, 1997).

B. Varying Weight Initialization Scale

Here we confirm the hypothesis that the reason higher noise models take longer to become sparse is because they need to significantly shrink the weight norms of their encoder weights W_e. We test this by initializing W_e with weights of varying scales and using $\sigma = 3.0$ noise. Figure 9 shows how networks with smaller weight initializations sparsify faster. In addition, their bias terms become larger negative values and L_2 norms fall until they reach values where the bias terms can effectively implement sparsity.

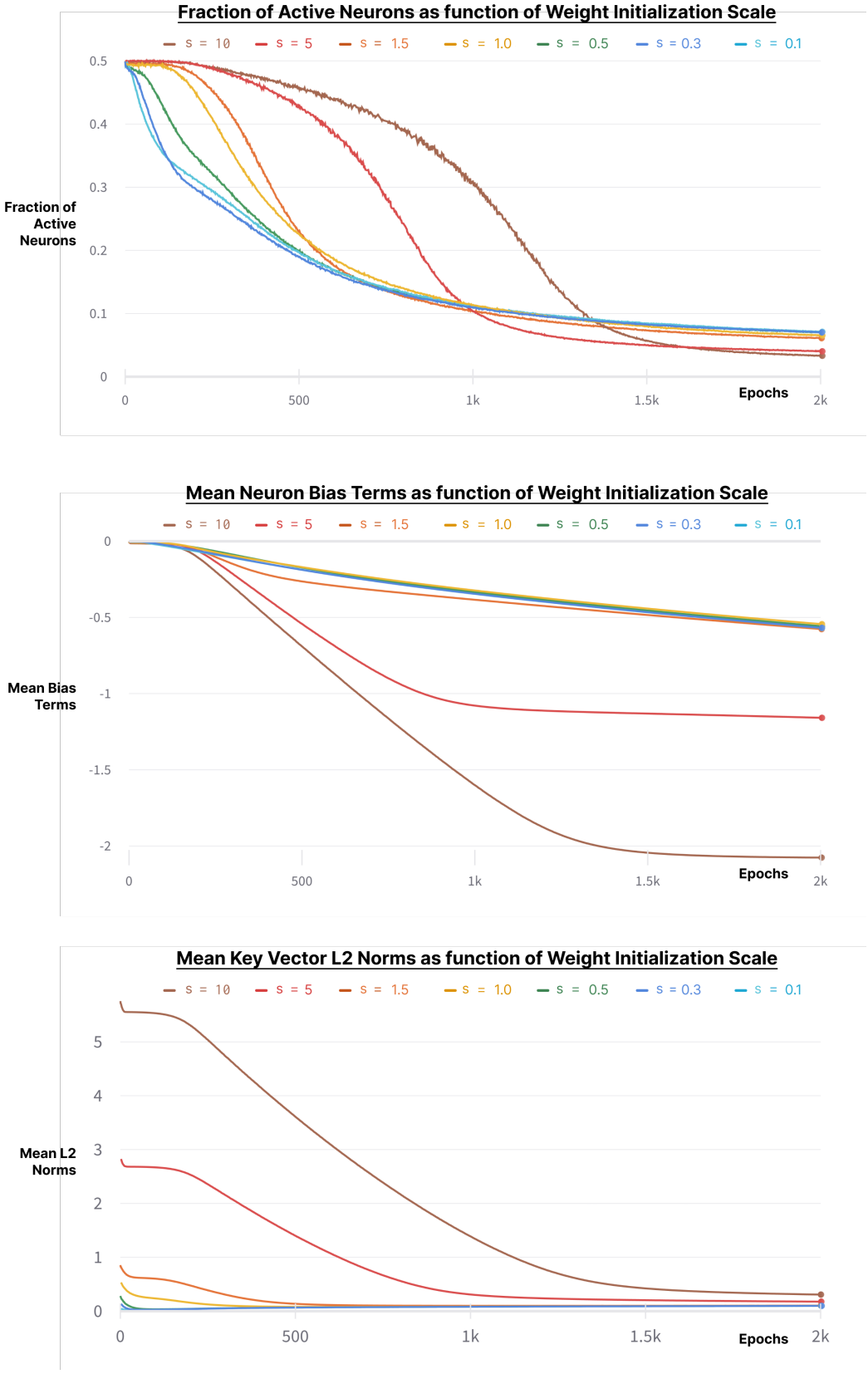


Figure 9: **Varying Weight Initialization Scale.** We use s to denote the amount we multiply our initialized weights by before training. $s = 1.0$ (yellow) is the default Kaiming initialization (He et al., 2015).

C. Receptive Fields

Here we show additional images of the receptive fields for networks trained with different amounts of noise (Fig. 11), the queries used to activate these neurons (Fig. 10) and how the receptive fields look more biological as the network is trained for longer (Fig. 12).

Figure 10: **Queries used to activate the receptive fields.** These queries were randomly selected. We use the cat (left) and truck images (middle) with the relevant noise levels to activate the neurons shown in Fig. 11. We use the car image (right) for the main text Fig. 7 and receptive field changes across epochs in Fig. 12.

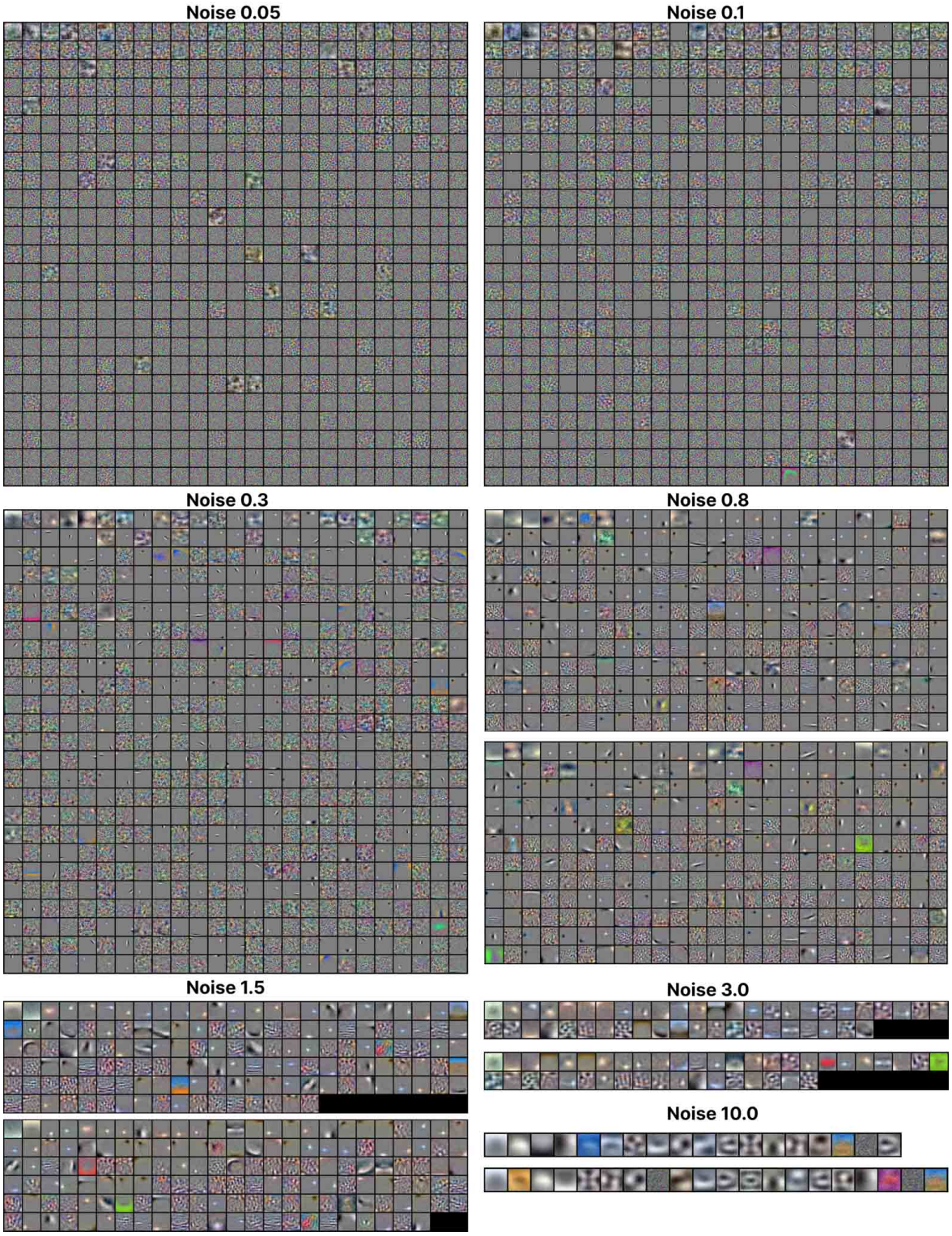


Figure 11: **Receptive fields across noise levels.** We show across all noise levels the ≤ 625 most active neurons for the truck query (middle of Fig. 10). For $0.8 \geq \sigma$ which activates fewer neurons, we use the extra space to also show the cat query. Biological receptive fields start to appear when $\sigma = 0.1$ and can be seen most clearly when $\sigma \in \{0.3, 0.8, 1.5\}$.

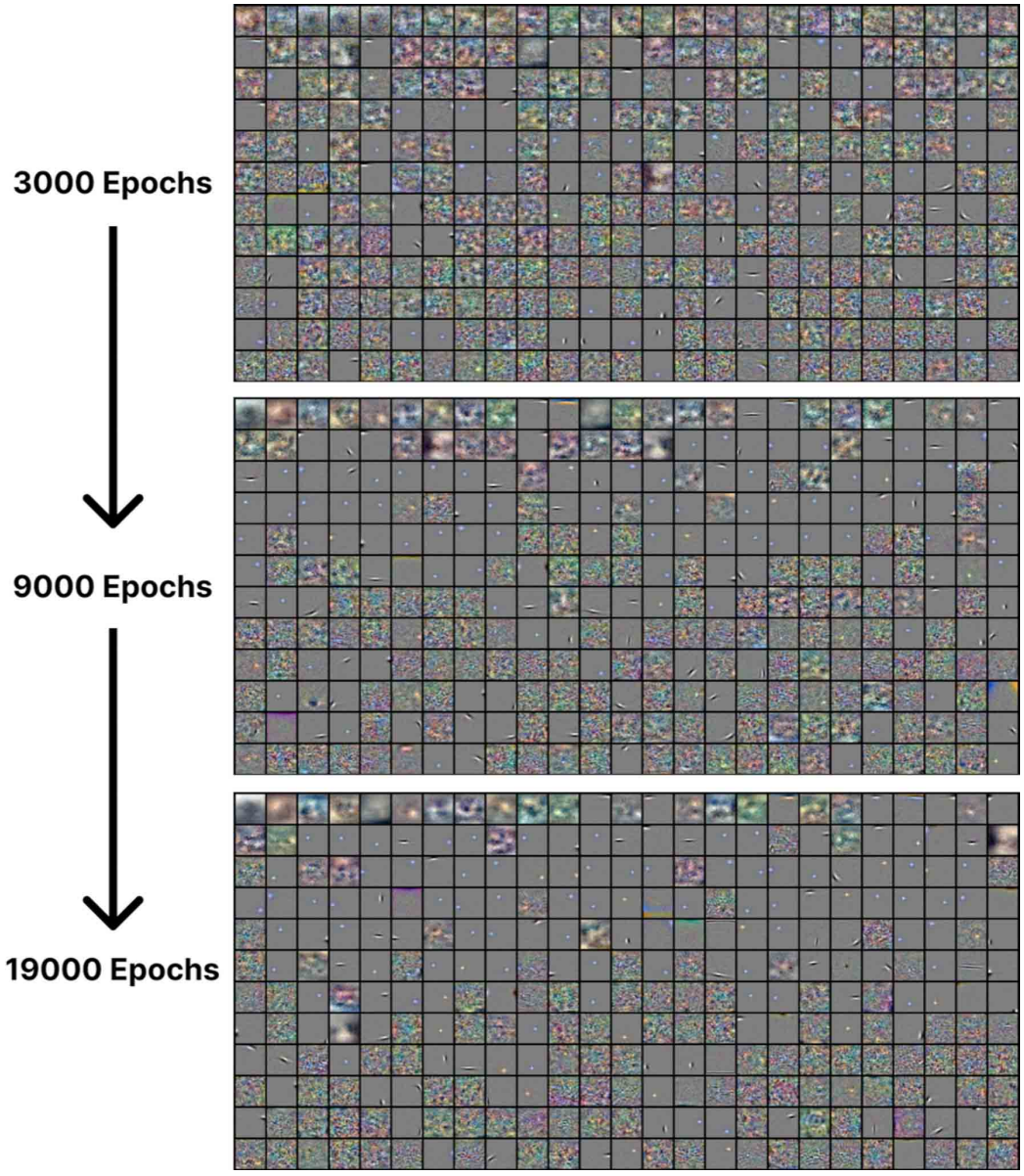


Figure 12: **Receptive fields become more biological across epochs.** We show how the most active receptive fields for the car image (rightmost in Fig. 10) with $\sigma = 0.3$ evolves over the course of training to become more biologically similar. The specific number of epochs used (3,000, 9,000 and 19,000) were chosen simply because of when the models were checkpointed and re-loaded for additional training.

D. L_1 and Top-k compared to Noise

To test the performance, sparsity, and receptive fields of L_1 and Top-k in comparison to noisy training, we trained 10,000 neuron models using Adam for 2,000 epochs to reconstruct CIFAR10 pixels. Figure 13 shows the relationships between validation loss, number of dead neurons, and sparsity.

With the noisy models, we anneal noise from epochs 1000 to 1500 to give a more accurate final performance comparison. This resulted in dead neuron like those shown in Appendix I.3 but in agreement with the reconstruction task of Fig. 8, did not significantly change sparsity. We test noise levels $\sigma \in \{0.05, 0.1, 0.3, 0.8, 1.5, 3.0, 10.0\}$, $L_1 \in \{1e-04, 1e-05, 1e-06, 1e-07, 1e-08\}$ ($\geq 1e-04$ killed all neurons) and Top-$k \in \{3, 10, 30, 100, 300, 1000, 3000\}$. For Top-$k$ we linearly annealed the k value from 10,000 down to its final value within the first 500 epochs.

Across methods there is a clear Pareto frontier between sparsity and validation loss. For at least the CIFAR10 raw pixel dataset and the Adam optimizer, there is also a relationship between sparsity and the number of dead neurons. However, caution should be applied when interpreting the number of dead neurons, due sparse models being sensitive to choice of optimizer, learning rate, and batch size (App. I.2) (Bricken et al., 2023). For Noise and Top-k, it is possible that more careful annealing schedules would also prevent more dead neurons.

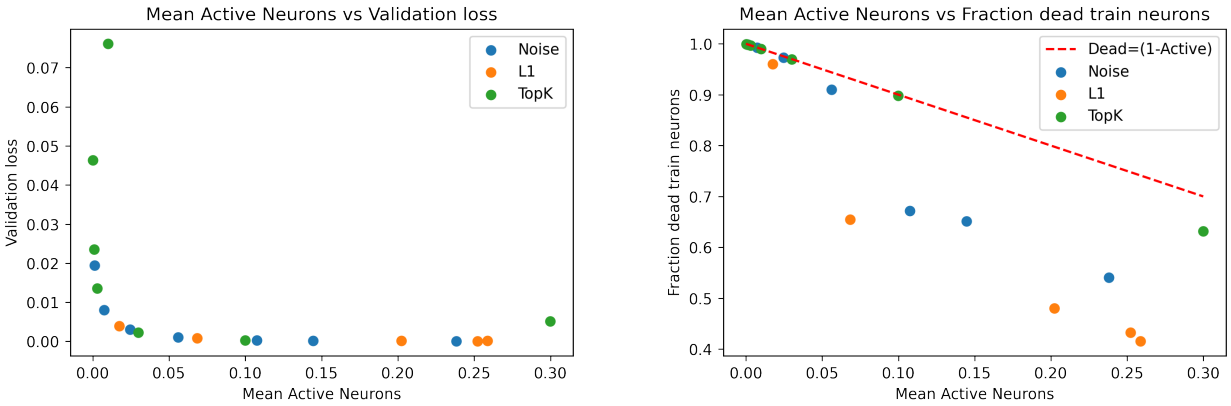


Figure 13: **Relationships between validation reconstruction loss, sparsity, and dead neurons for Noise, L_1 and Top-k.** There are a few observations here: (i) There is a Pareto frontier between sparsity and validation loss (left). (ii) The sparser the activity, the more neurons die (right), however, Noise and L_1 sparsity is not explained exclusively by neuron dead which is largely true for Top-k.

Figure 14 shows the receptive fields for the different L_1 coefficients and the formation of Gabor filters for the sparsest (highest L_1) models. In agreement with Karklin & Simoncelli (2011) there are no center-surround receptive fields that form absent noise.

We tried to visualize the most active receptive fields for the Top-k model however, the most active neurons are completely un-interpretable because they have very large and uniform magnitudes that when normalized look all white (some examples in Fig. 15).[6] This result, combined with the high number of dead neurons (Fig. 13) suggests a problem documented in Bricken et al. (2023) where there are a few "greedy" neurons with much larger weight norms with activations always in the top k. That work used L_2 normalization of weights to ensure all neurons participated democratically. This need for weight normalization is an issue with explicit Top-k algorithms that noisy training shows an ability to regulate on its own (e.g. Fig. 6). Instead of showing these all white receptive fields, we display 625 random receptive fields in Fig. 15. Interestingly, in further agreement with Bricken et al. (2023), a number of these neurons learn to memorize specific training examples yet this is with a reconstruction task instead of classification.

[6]Note that the decoder weights did exhibit meaningful structure such as spots of color and for larger values of k could reconstruct the image (as supported by the competitive validation losses in Fig. 13). Therefore, we assume there are imperceptible differences in the weights that still ensure a different subset of neurons fire for each data point.

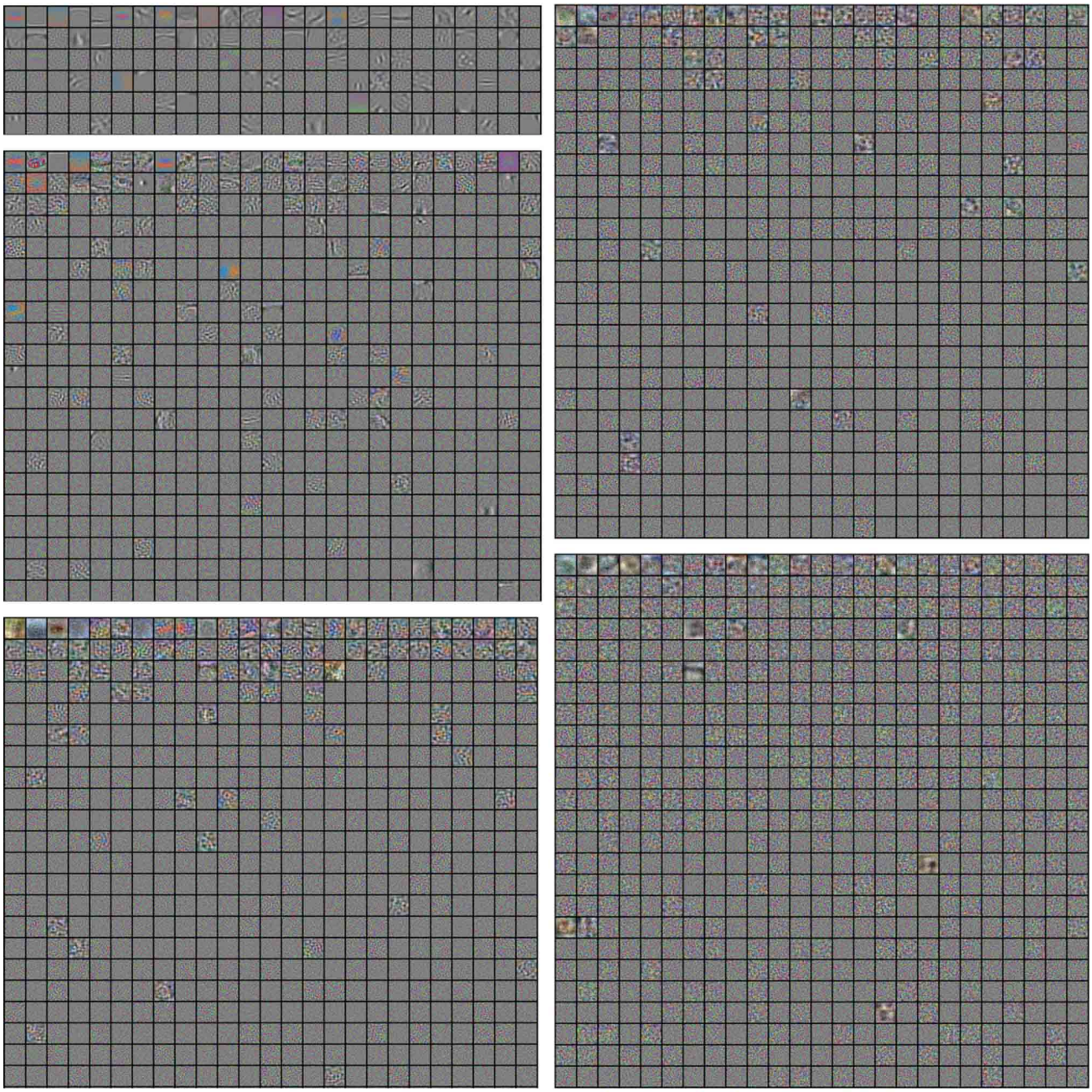


Figure 14: **Receptive fields across L_1 coefficients.** We show the ≤ 625 most active neurons for a red car query (not shown). The L_1 coefficients go from highest to lowest, top to bottom, left to right 1e-04, 1e-05,1e-06, 1e-07, 1e-08. Gabor filters are visible for the highest L_1 coefficients of 1e-4 and 1e-5 (left column top and middle figures). However, no center surround receptive fields are visible.

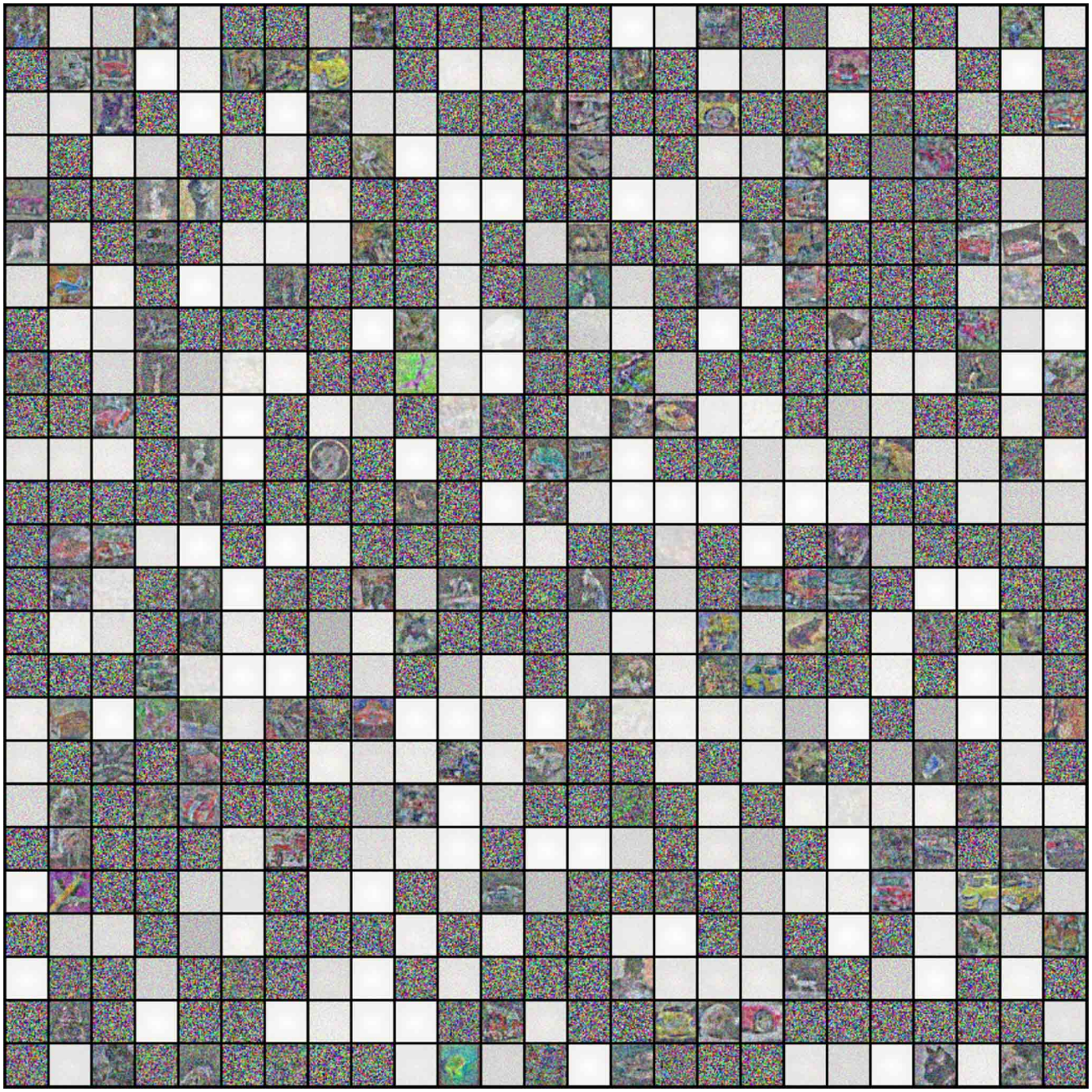


Figure 15: **Receptive fields across Top-k.** We show a random subset of 625 neurons. All white receptive fields that have large uniform weights which dominate in the Top-k are shown along with neurons that have memorized specific CIFAR10 datapoints. Examples of cars, planes, dogs, and frogs are all visible.

E. Noise Analytical Derivation

Some proofs have used first order Taylor series approximations to show that noisy training minimizes the Frobenius norm of the model's Jacobian. Here, we avoid this approximation and are able to disentangle more nuanced effects that noisy training implicitly has on the loss.

$$\text{Loss} = \frac{1}{D}\sum_{x\in X}\mathbb{E}_{\varepsilon}\left[\sum_{i}^{o}(x_i-\tilde{y}_i)^2\right] \tag{6}$$

$$(x_i-\tilde{y}_i)^2 = (x_i-(\bar{y}_i+\underbrace{\tilde{y}_i-\bar{y}_i}_{\xi_i}))^2 = (\underbrace{x_i-\bar{y}_i}_{r_i}-\xi_i)^2 \tag{7}$$

$$= (r_i-\xi_i)^2 \tag{8}$$

$$\text{Loss} = \frac{1}{D}\sum_{x\in X}\sum_{i}^{o} r_i^2 - 2r_i\mathbb{E}_{\varepsilon}[\xi_i] + \mathbb{E}_{\varepsilon}[\xi_i^2] \tag{9}$$

where $r_i = x_i - \bar{y}_i$ and $\xi_i = \tilde{y}_i - \bar{y}_i$ is the difference between $\tilde{y}_i$, the output produced by the input with noise ε, and the output without noise $\bar{y}_i$. Keep in mind that because the input noise is independent of the data, minimizing $\xi_i \to 0$ will maximize the quality of the reconstruction.

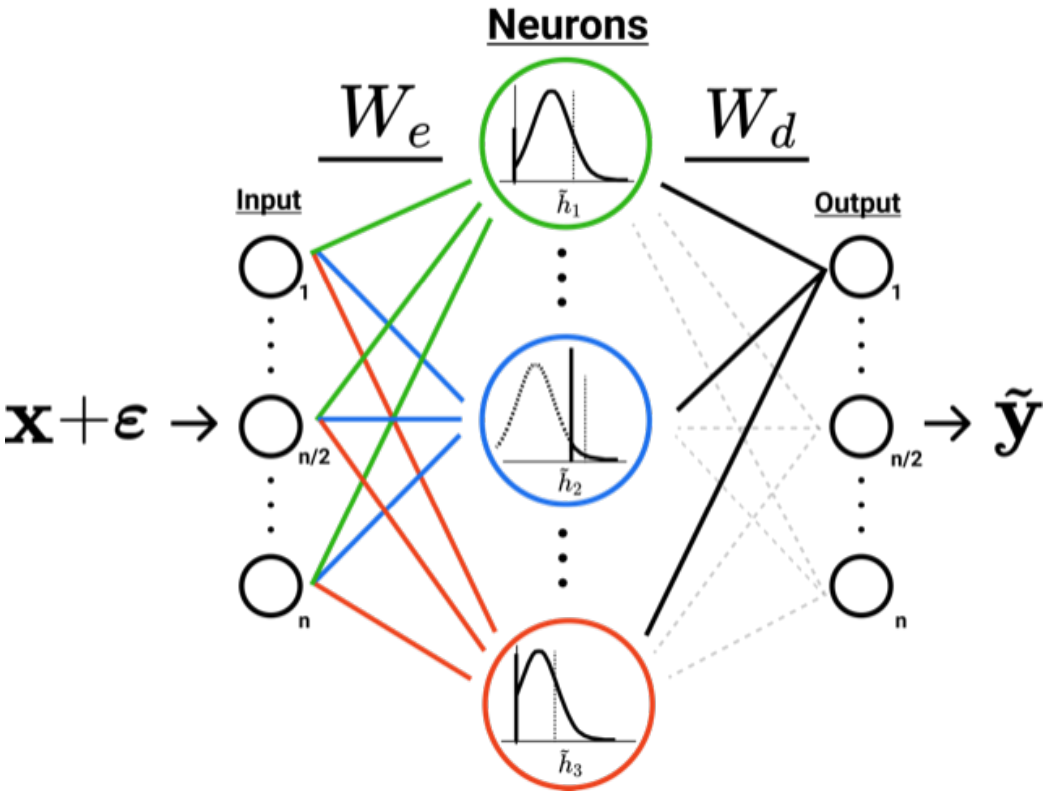


Figure 16: **Graphical depiction of how each neuron in the hidden layer is affected by noise.** Through our derivation we focus on one output unit and trace the effects of noise back through the network.

To understand the loss, we expand the operations performed on the noisy input for a single ξ_i:

$$\xi_i = \sum_j^m W_{d_{i,j}} \eta_j \tag{10}$$

$$= \sum_j^m W_{d_{i,j}} (\tilde{h}_j - \bar{h}_j) \tag{11}$$

$$= \sum_j^m W_{d_{i,j}} \left(\left[\mathbf{w}_{e_j} (\mathbf{x} + \boldsymbol{\varepsilon}) + b_j \right]_+ - \left[\mathbf{w}_{e_j} \mathbf{x} + b_j \right]_+ \right) \tag{12}$$

$$= \sum_j^m W_{d_{i,j}} \left(\left[\sum_k^o W_{e_{j,k}} (x_k + \varepsilon_k) + b_j \right]_+ - \left[\sum_k^o W_{ej,k} x_k + b_j \right]_+ \right) \tag{13}$$

Plugging this back into Eq. 9 full loss gives:

$$-2r_i \mathbb{E}_\varepsilon[\xi_i] + \mathbb{E}_\varepsilon[\xi_i^2] = -2r_i \left(\sum_j^m W_{d_{i,j}} \mathbb{E}_\varepsilon[\eta_j] \right) + \mathbb{E}_\varepsilon \left[(\sum_j^m W_{d_{i,j}} \eta_j)^2 \right] \tag{14}$$

$$\tag{15}$$

Expanding out the second term:

$$\mathbb{E}_\varepsilon \left[(\sum_j^m W_{d_{i,j}} \eta_j)^2 \right] = \mathbb{E}_\varepsilon \left[\sum_j^m W_{d_{i,j}}^2 \eta_j^2 + \sum_{k \neq j}^m W_{d_{i,j}} W_{d_{i,k}} \eta_j \eta_k \right] \tag{16}$$

And plugging these back into Eq. 9 again makes our full loss:

$$\text{Loss} = \frac{1}{D} \sum_{x \in X} \sum_i^o r_i^2 - 2r_i \sum_j^m W_{d_{i,j}} \underbrace{\mathbb{E}_\varepsilon[\eta_j]}_{\text{\#1 Max Margin}} + \sum_j^m W_{d_{i,j}}^2 \underbrace{\mathbb{E}_\varepsilon\left[\eta_j^2\right]}_{\text{\#2 Sparsity}} + \sum_j^m \sum_{k \neq j}^m W_{d_{i,j}} W_{d_{i,k}} \underbrace{\mathbb{E}_\varepsilon\left[\eta_j \eta_k\right]}_{\text{\#3 Specialization}} \tag{17}$$

The implicit loss terms of noisy training manifest in the three terms outlined by underbraces. Each of these terms should be minimized for the network to remove the independent random noise and optimize its reconstruction.

#1 Max Margin - We first show that $\mathbb{E}_\varepsilon[\eta_j] \geq 0$ because $\tilde{h}$ is a rectified Gaussian meaning that $\mathbb{E}_\varepsilon[\tilde{h}_j] \geq \bar{h}_j$. Before applying the ReLU:

$$\tilde{z}_j = \sum_k^o W_{e_{j,k}} (x_k + \varepsilon_k) + b_j \tag{18}$$

$$= \sum_k^o W_{e_{j,k}} x_k + W_{e_{j,k}} N(0, \sigma^2) + b_j \tag{19}$$

$$\sim N(\bar{z}, \sigma^2 ||\mathbf{w}_{e_j}||^2) \tag{20}$$

Giving us a Gaussian centered on the noise free point. With CDF$(-|\bar{z}|/\sigma^2 ||\mathbf{w}_{e_j}||^2)$ the noise is sufficient to flip a neuron from being on to off or vice versa. The mean of the rectified Gaussian will be $\bar{z} + \int_{t=|\bar{z}|}^{\infty} p(\varepsilon)(\varepsilon - t) d\varepsilon$ where the integral defines the Gaussian region where symmetry is not applied (these positive values are not cancelled out by the opposite values but instead by the threshold constant, $t = |\bar{z}|$). See Fig. 17 for visual intuition on why this is the case. For curious readers, the solution to this integral is:

$$\mathbb{E}_\varepsilon[\tilde{h}_j] = \bar{h}_j + \frac{||\mathbf{w}_{e_j}|| \sigma}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{|\bar{z}|}{||\mathbf{w}_{e_j}|| \sigma} \right)^2 \right) + \frac{|\bar{z}|}{2} \left(\text{erf}\left(\frac{|\bar{z}|}{||\mathbf{w}_{e_j}|| \sigma \sqrt{2}} \right) - 1 \right) \tag{21}$$

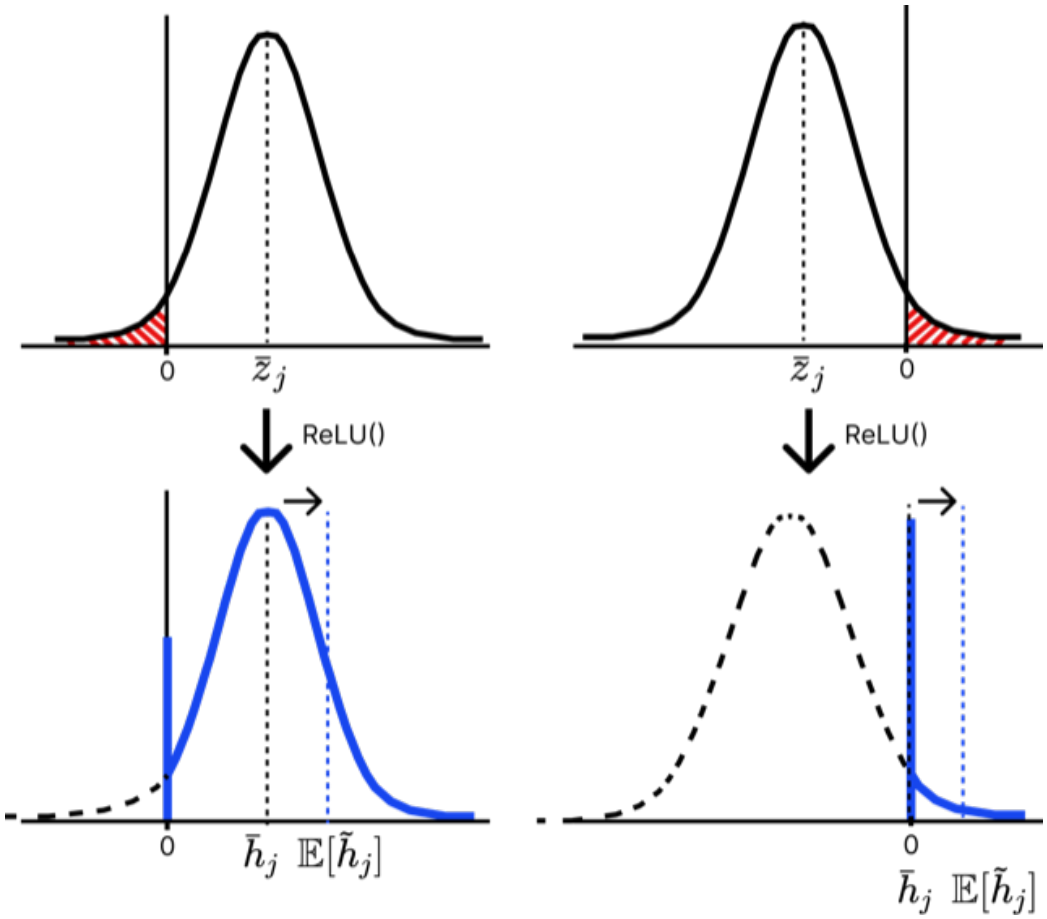


Figure 17: **ReLU truncates the Gaussian causing a positive shift in the mean.** We provide intuition for why only the positive right hand side of the tail remains resulting in the mean of the post activation noisy neuron being higher than it would otherwise be. This increase in the mean is a function of how much noise causes the neuron to cross the activation threshold.

The best way to minimize this error term in the loss is by setting $\bar{h}_j$ to be maximally far from the ReLU 0 activation threshold. It is also possible to shrink the weight norms of either $||\mathbf{w}_{e_j}||$ to reduce noise variance or $W_{d_{i,j}}$ to reduce the weighting of this loss term (refer back to Eq. 17) however, these are both in conflict with maximizing the reconstruction.

#2 Sparsity - $\mathbb{E}_\varepsilon\left[\eta_j^2\right] = \mathbb{E}_\varepsilon\left[(\tilde{h}_j - \bar{h}_j)^2\right]$ is the deviation between the noisy and noise free activation, squared which depends if $\bar{z}_j > 0$. Providing Fig. 18 for visual intuition, if $\bar{z}_j > 0$:

$$\mathbb{E}_\varepsilon\left[(\tilde{h}_j - \bar{h}_j)^2\right] = \bar{z}_j^2 \int_{-\infty}^{-\bar{z}_j} p(\varepsilon)d\varepsilon + \int_{-\bar{z}_j}^{\infty} p(\varepsilon)\varepsilon^2 d\varepsilon \tag{22}$$

$$= \bar{z}_j^2 \text{CDF}\left(\frac{-\bar{z}}{\sigma^2||\mathbf{w}_{e_j}||^2}\right) + \int_{-|\bar{z}_j|}^{\infty} p(\varepsilon)\varepsilon^2 d\varepsilon \tag{23}$$

Else if $\bar{z}_j \leq 0$:

$$\mathbb{E}_\varepsilon \left[(\tilde{h}_j - \bar{h}_j)^2 \right] = 0 * \int_{-\infty}^{|\bar{z}_j|} p(\varepsilon) d\varepsilon + \int_{|\bar{z}_j|}^{\infty} p(\varepsilon) \varepsilon^2 d\varepsilon \tag{24}$$

$$= \int_{|\bar{z}_j|}^{\infty} p(\varepsilon) \varepsilon^2 d\varepsilon \tag{25}$$

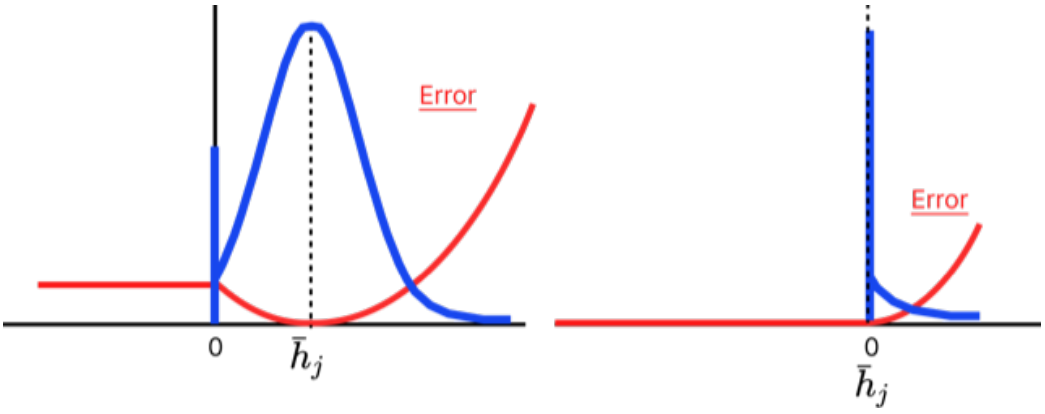


Figure 18: **The variance of the noise and its contribution to the error.** Graphical intuition behind Eqs. 23 & 25 .This term incentivizes neurons to be as negative as possible to avoid noise turning the neuron on.

This reveals that the penalty for a neuron being on ($\bar{h}_j > 0$) is strictly greater than being off. Not only is there an additional positive term but also the shared ε^2 integral is over a $2|\bar{z}_j|$ larger interval spanning $[-|\bar{z}_j|, \infty]$ instead of $[|\bar{z}_j|, \infty]$. This means that neurons noise free activation ($\bar{z}_j$) should be as negative as possible.

Note that When the neuron is on, there will be a tradeoff between the max margin $\mathbb{E}_\varepsilon[\eta]$ term that wants the neuron to be as far away from the ReLU threshold at zero as possible and the sparsifying $\mathbb{E}_\varepsilon[\eta^2]$, which wants to be as close to zero as possible.

#3 Specialization - The cross term $\mathbb{E}_\varepsilon \left[\eta_j \eta_k\right]$ is closely related to covariance:

$$\mathbb{E}_\varepsilon \left[\eta_j \eta_k\right] = \mathbb{E}_\varepsilon \left[(\tilde{h}_j - \bar{h}_j)(\tilde{h}_k - \bar{h}_k) \right].$$

Where it is not a true covariance due to the non-linearity that means $\bar{h}$ is not the mean of $\tilde{h}$. In seeking to minimize covariance, the model is incentivized to both sparsify activations and specialize its weights. Sparsity comes from the fact that this term is only present when both neurons are turned on and the probability of this occurring is reduced the more negative $\bar{z}_j$ is. Specialization comes from $\bar{h}_j$ and $\bar{h}_k$ not being independent because they receive the same noise input. This means their covariance is a function of receptive field similarity. Therefore, one way to utilize both neurons in the model (increasing model capacity for the reconstruction task) but minimize the covariance term is by having weights that respond to different parts of the input space where there are independent noise values (e.g. Fig. 11).

We confirm that our networks minimize noisy activation covariance by taking five different CIFAR10 images, applying 100 different noise patterns to them at a range of noise levels and storing their activations to compute $\mathbb{E}_\varepsilon \left[\eta_j \eta_k\right]$ where $\bar{h}$ is the noise free image. The results are shown in Figure 19.

We found it difficult to quantify our notion of weight specialization which is qualitatively very distinct. For example, comparing the random weights of $\sigma = 0$ to the highly structured biological receptive fields of $\sigma = 0.8$. However, computing pairwise cosine similarity, L_1 or L_∞ norms all give higher values for the random weights displayed by $\sigma = 0$. While two Gabor filters may only fire for very specific features with the remainder of their weights all close to zero, these zero weights still result in less orthogonality than random weights covering the entire space of the image. Future work should develop

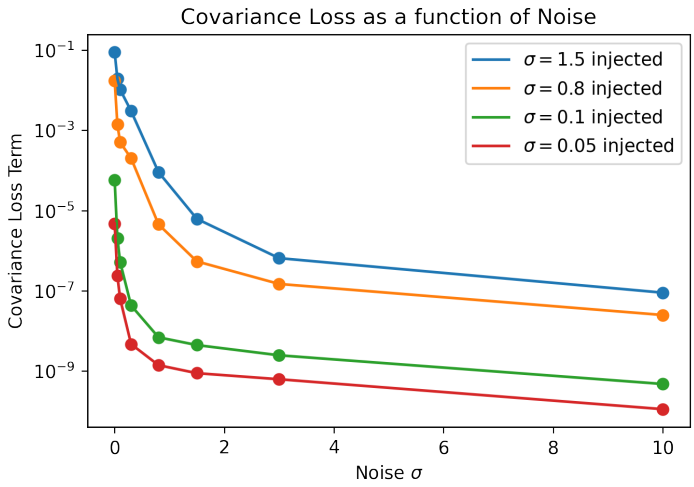


Figure 19: **Noise Trained models minimize neuron activation covariance.** We inject noise at different scales (shown by each line) into networks trained at different noise levels defined by the x-axis. We compute $\mathbb{E}_\varepsilon\left[\eta_j \eta_k\right]$ for 100 different noise patterns and average over ten queries. Independent of the noise level injected, neuron activation covariance is inversely proportional to noise. The y-axis is on a log scale. Networks are trained on CIFAR10 pixels.

some notion of structured orthogonality to better quantify the interpretable specialization weights that networks trained with noise display.

Loss Term Effects - The way our networks empirically (Fig. 2) decide to achieve all three of these objectives is by setting neuron activations to be very negative by default with only a sparse few jumping to very large activations that all for the reconstruction task to be solved. This is implemented by having large negative bias terms and weights that specialize to distinct dataset features. This maximizes the signal to noise ratio by ignoring noise across most of the input while being very activated by unique features, therefore not firing all the time and firing strongly when it does. In other words, the weights focus on capturing high variance regions of the data distribution while being unique from other neurons.

Relation to Sparse Coding's L_1 Penalty - Because we want to minimize $\mathbb{E}_\varepsilon[\xi_i^2]$ and have shown $\mathbb{E}_\varepsilon[\eta_i^2] \geq 0$ we can loosely relate this loss to the L_1 activation penalty of sparse coding by seeking to minimize: $\sum_j^m W_{d_{i,j}}^2 |\mathbb{E}_\varepsilon[\eta_j^2]|$. This linear sum of neuron activations has the same flavour as an L_1 penalty.

F. Taylor Series Approximation Derivation

Here we trace the reasoning of Poole et al. (2014), correcting incorrect results and furthering the analysis, to show how a similar conclusion can be reached via a Taylor series approximation. For convenience, we repeat the nonlinear autoencoder network equations (Eqn. 2):

$$
\begin{aligned}
\tilde{\mathbf{x}} &= \mathbf{x} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim N(0, \sigma^2 I) \\
\mathbf{z} &= W_e \tilde{\mathbf{x}} + \mathbf{b}_e \\
\mathbf{h} &= \varphi(\mathbf{z}) \\
\tilde{\mathbf{y}} &= W_d\, \mathbf{h} + \mathbf{b}_d \\
\mathcal{L} &= \sum_{\mathbf{x} \in X} ||\mathbf{x} - \tilde{\mathbf{y}}||^2.
\end{aligned}
$$

For small noise $\boldsymbol{\epsilon}$ with mean 0 and variance σ^2, the first-order Taylor series expansion is $\tilde{\mathbf{x}} \approx \mathbf{x} + J(\mathbf{x})\boldsymbol{\epsilon}$, where $J(\mathbf{x}) \stackrel{\text{def}}{=} \nabla_{\mathbf{x}} \bar{\mathbf{y}}$ is the input-output Jacobian at $\mathbf{x}$. The expected loss over dataset X (averaging over the noise) can then be written as (Bishop,

1995; Rifai et al., 2011):

$$\mathbb{E}_\epsilon[\mathcal{L}] \approx \sum_{\mathbf{x}\in X} ||\mathbf{x} - \bar{\mathbf{y}}||_2^2 + \sigma^2 \sum_{\mathbf{x}\in X} ||J(\mathbf{x})||_F^2 \tag{26}$$

The noise regularizes the Jacobian of the network, proportional to the noise's variance. In our toy autoencoder model, the Jacobian is:

$$\begin{aligned} J(\mathbf{x}) &\stackrel{\text{def}}{=} \nabla_{\mathbf{x}} \bar{\mathbf{y}} \\ &= \nabla_{\mathbf{x}} \Big(W_d \, \varphi(W_e \, \mathbf{x} + \mathbf{b}_e) + \mathbf{b}_d \Big) \\ &= \nabla_{\mathbf{x}} \, \varphi(W_e \, \mathbf{x} + \mathbf{b}_e) \, W_d^T \\ &= \Big(\nabla_{\mathbf{x}} W_e \mathbf{x} \Big) \, \text{diag}(\varphi'(W_e \mathbf{x} + \mathbf{b}_e)) \, W_d^T \\ &= W_e^T \, \text{diag}(\varphi'(W_e \mathbf{x} + \mathbf{b}_e)) \, W_d^T \end{aligned}$$

Pausing for a moment, Poole et al. (2014) reach a slightly different conclusion (their Equation 7). Two hints that their Equation 7 cannot be correct is (1) that it multiplies two incompatible matrices together, with shapes hidden-by-hidden and input-by-input, and (2) that the noise's variance appears squared even though the noise itself only appears squared (recalling that a variable squared produces its variance, not its variance squared). Continuing along, for commonly used non-linearities e.g. ReLU, GeLU, the nonlinearity's domain can be split in two: regions with non-zero gradients, and regions with (near) 0 gradients. We can split the Jacobian's squared Frobenius norm into two terms based on these gradient regions:

$$\begin{aligned} ||J(\mathbf{x})||_F^2 &= \sum_{ij} [J(\mathbf{x})]_{ij}^2 \\ &= \sum_{ij:\varphi'_h(\mathbf{z})\neq 0} \Big([W_e^T]_{ih} [\text{diag}(\varphi'(\mathbf{z}))]_{hh} [W_d^T]_{hj} \Big)^2 + \underbrace{\sum_{ijh:\varphi'_h(\mathbf{z})=0} \Big([W_e^T]_{ih} \underbrace{[\text{diag}(\varphi'(\mathbf{z}))]_{hh}}_{=0} [W_d^T]_{hj} \Big)^2}_{=0} \end{aligned} \tag{27}$$

Minimizing the Jacobian's squared Frobenius norm can be accomplished in one of two non-mutually-exclusive ways: (1) minimise the (gradient-scaled) squared dot product of the encoder and decoder weights, and/or (2) drive the pre-activations $\mathbf{z}$ to regions where the nonlinearity's gradient is 0. Because the first term is a squared error, it is likely to contribute at least some error, but the second term contributes exactly zero error, meaning pushing all pre-activations $\mathbf{z}$ into the zero-gradient region is highly advantageous. However, the reconstruction loss creates an opposing pressure by incentivizing the network to faithfully recapitulate the uncorrupted data. To balance these competing demands, a compromise is struck: keep enough units "on" to reconstruct the data, then for the weights in use, regularize them, and for the weights not in use, drive the corresponding activations to the regions where the nonlinearity gradient is 0.

We note that while the math here describes a denoising autoencoder with a single hidden layer, the same reasoning holds generally for feedforward networks on generic losses: the loss forces the network to propagate useful information in the data, while the noise implicitly regularizes the network's Jacobian's squared Frobenius norm, and since the Jacobian will always have the form of the pre-activation times derivative of nonlinearity times gradient of network output with respect to post-activation, moving the pre-activation to regions where the nonlinearity's derivative is zero will help minimize the loss. Because most non-linearities' zero-gradient regions coincide with where the nonlinearity is "inactive" (e.g., < 0 for ReLU), the network's activations become sparse.

This understanding yields two predictions: First, pre-activations of different non-linearities should increasingly concentrate in zero-gradient regions of the nonlinearity as noise variance σ^2 increases, for both standard non-linearities (e.g., ReLU, GELU) and non-standard non-linearities (e.g. horizontally reflected ReLU, $\text{sat}(x) \stackrel{\text{def}}{=} max(min(x, 0.5), -0.5)$). We test whether increasing noise drives pre-activation values towards zero-gradient regions of different non-linearities (Fig. 20).

Our second prediction is that, the same noise-inducing-sparsity result should hold qualitatively in nontrivial deep feedforward neural networks. We train nontrivial deep feedforward, convolutional and transformer architectures on standard benchmark tasks, validating that increasing noise drives increasing sparsity (App. J).

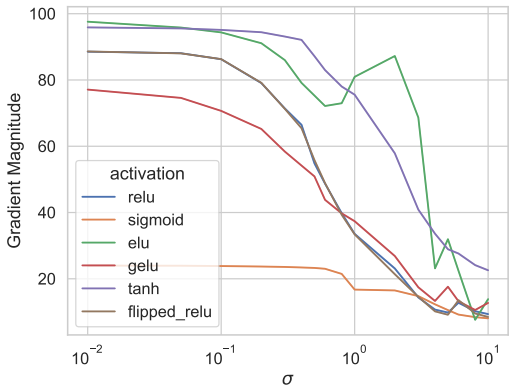


Figure 20: **Noise drives neural activations to near-zero gradient regions of the nonlinearity's domain.**

G. Relation between λ coefficient and noise bias terms

The use of a shared bias term combined with the ReLU activation function can be related to the lambda coefficient that scales the sparse coding L_1 activation penalty. Assuming that sparse coding learns orthogonal weights, this results in the optimal solution for neuron activity a_i when non-negative to be $\text{ReLU}(\mathbf{w}_i^T\mathbf{x} - \lambda)$, where $\mathbf{w}_i^T\mathbf{x}$ is the value obtained by projecting the data onto the neuron output weights $\mathbf{w}_i$ (Ba, 2020; Gregor & LeCun, 2010). Intuitively, if the values of our target image $\mathbf{x}$ reconstructed by $a_i\mathbf{w}_i^T$ are too small, it is better for $a_i = 0$ than to predict $\mathbf{x}$ with the threshold for this tradeoff given by λ. This $-\lambda$ term behaves as an activation threshold in an analogous fashion to the bias terms of our noisily trained network converging to a single shared negative value.

H. Ablations

H.1. Single MLP Layer Ablations

Figure 21 shows that the relationship between sparsity and noise holds across numbers of neurons, datasets, and noise distributions.

For CIFAR10, we train each model for 3,000 epochs and present the mean number of active neurons for GELU (Fig. 22) and sigmoid (Fig. 23). We don't show the latent CIFAR10 results as sigmoid and GELU are fully dense with no sparsity when the 0.0001 activation threshold is used. Neither of these networks are as sparse as the ReLU network. Also neither network creates sparsity through "dead" neurons.

While we just show the mean active neurons here, both solutions look like those of Fig. 6 with the neurons for all inputs converging to a Top-k solution where the bias terms converge to a single negative scalar and calibrate with the encoder L_2 weight norms.

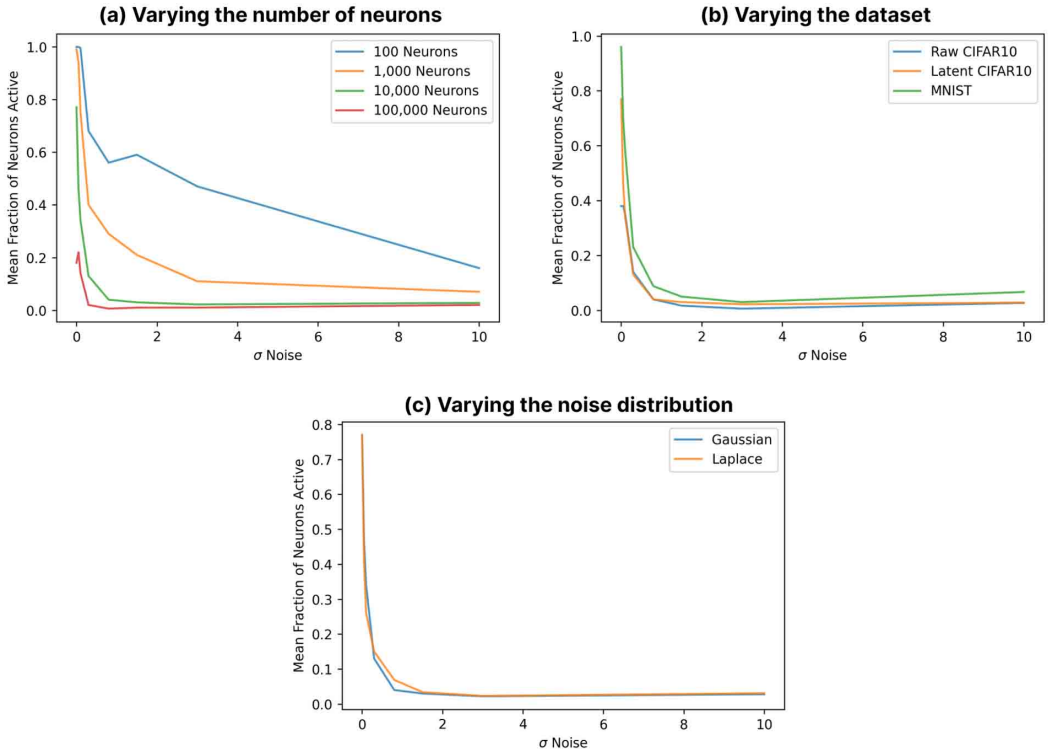


Figure 21: **Training ablations.** The positive correlation between noise and sparsity is robust to **(a)** neuron counts, **(b)** datasets, **(c)** noise distributions.

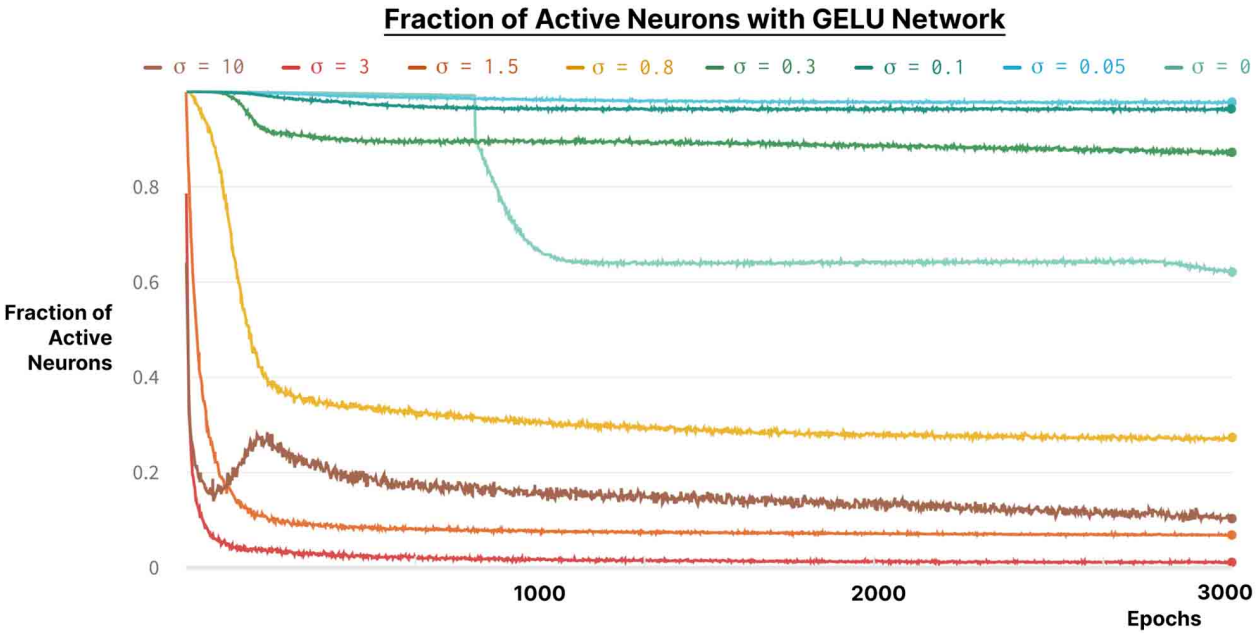


Figure 22: **The GELU network sparsifies with noise when trained on CIFAR10.** The GELU network closely resembles the ReLU network by sparsifying (but not to the same degree). It also forms biological receptive fields. We use the arbitrary 0.0001 absolute activation threshold to label if a neuron is on or off.

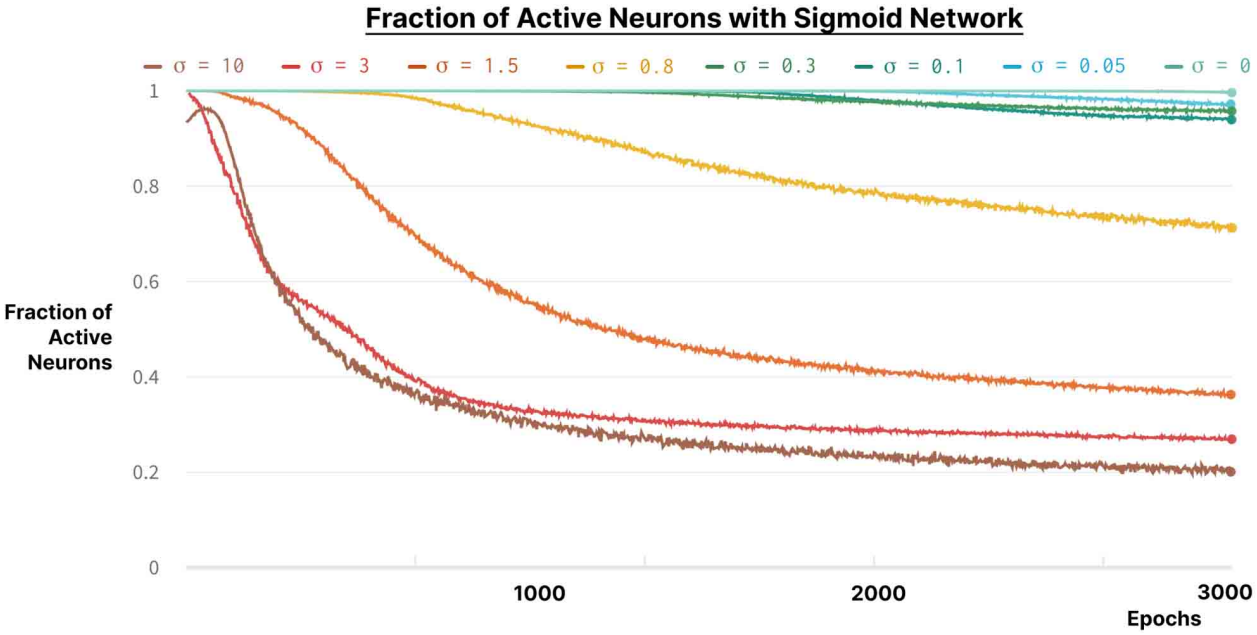


Figure 23: **The sigmoid network also sparsifies with noise for CIFAR10.** While sigmoid sparsifies in proportion to noise, it does not form biological receptive fields and remains less sparse than GELU or ReLU. This is likely explained by there being a zero gradient positive region (see Fig. 20).

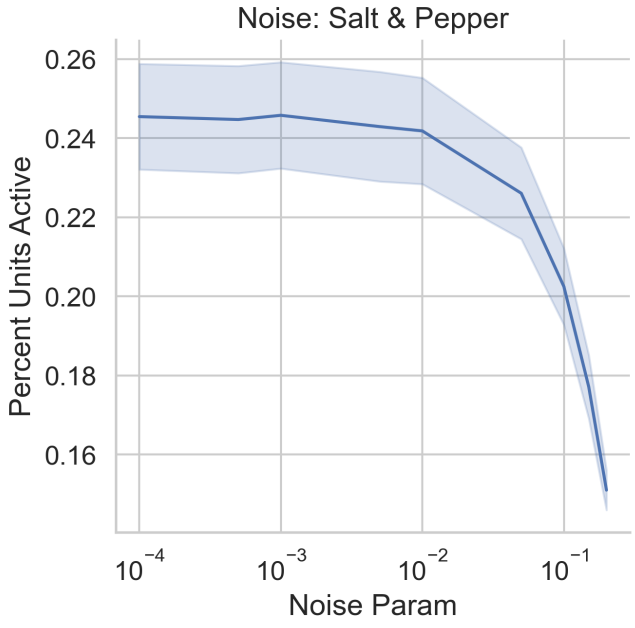


Figure 24: **Alternative noise distributions produce the same increasing sparsity with increasing noise variance.** Here, multiplicative "Salt and Pepper" noise causes the same increasing sparsity with increasing noise probability.

H.2. SGD Learning Rate and Batch Size

We discover that SGD is able to avoid the sparsifying effects of noise when either its batch size is large enough or its learning rate is low enough as shown in Figure 25. We hypothesize that both of these observations are explained by the random noise being cancelled out by the central limit theorem. Formally, let g^* be the true gradient update on the full dataset and $\hat{g}$ be the minibatch gradient that approximates g^* with some error ϵ_b caused by our minibatch: $g^* = \hat{g} + \epsilon_b$. In the presence of random noise, we can de-compose our minibatch gradient further into $\hat{g} = \hat{g}_b + \hat{g}_n$ to represent the gradient from the data and the gradient from the noise. This gives us:

$$g^* = \hat{g}_b + \hat{g}_n + \epsilon_b.$$

Because $\hat{g}_n$ comes from independent random variables with mean 0 and finite variance, by the Lyapunov Central Limit Theorem, as our batch size $b \to \infty$ the variance of $\hat{g}_n$ will shrink by $\sigma/\sqrt{b}$ to zero (Blitzstein & Hwang, 2019). This means that as our batch size increases, the contribution of noise to the gradient converges to zero; the same reasoning can also be applied to the noise ϵ_b from the batch size.

Reducing the learning rate approximates increasing the batch size by allowing for the noise terms of the gradient step aggregate across multiple steps. This can only occur if each gradient step is sufficiently small and the optimization landscape sufficiently smooth such that each intermediate gradient step does not cause the model to deviate from its noise free optimization trajectory.

As an additional ablation we tried giving our bias terms a separate, higher learning rate, hypothesizing that without a parameter specific adaptive optimizer the learning rate may be too small for the bias terms that have larger values than the weights. However, even with a large parameter sweep we found that having a different bias term learning rate did not have an effect.

When we test SGD without any noise, we find that neither modifying the batch size nor learning rate have any sparsity effect. This supports our hypothesis that increasing batch size or reducing learning rate helps to reduce the random noise injection, rather than having a smaller batch size introducing a new source of noise. However, there is evidence that with a very high learning rate and a sustained plateau in training loss, the minibatch can be a source of label noise that induces sparsity as analyzed by Andriushchenko et al. (2022).

Therefore, using SGD with a low enough learning rate or large enough batch size removes the effects of noising training. This removes the implicit noise loss terms and results in a model that is not sparse.[7]

The fact a non-sparse solution to the de-noising problem exists is on the surface problematic for our statements about sparse coding being better at modelling the noisy data. However, we found that this result did not generalize to the more challenging raw CIFAR10 pixels dataset. Even using the full batch of 50,000 images and a low learning rate resulted in a sparse coding solution that corresponds to better train and validation loss.

More broadly, the fact that SGD with low learning rate and large batch size can have different inductive biases may generalize to other problems and findings. For example, we are aware of SGD with a low learning rate failing to produce networks with weights that could be interpolated between using Git Rebasin (Ainsworth et al., 2022).

[7]One caveat is that the models retain their initial 50% sparsity rather than becoming fully dense as they do when there is no noise present.

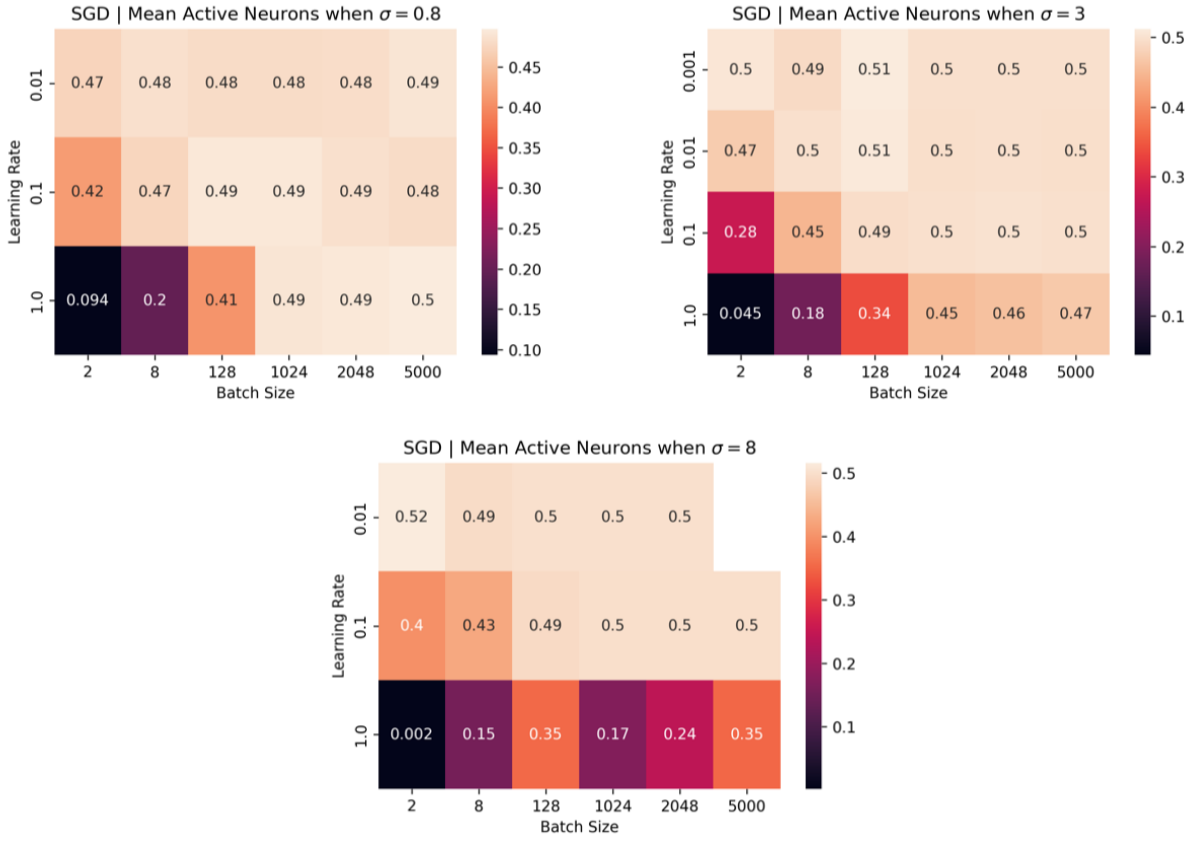


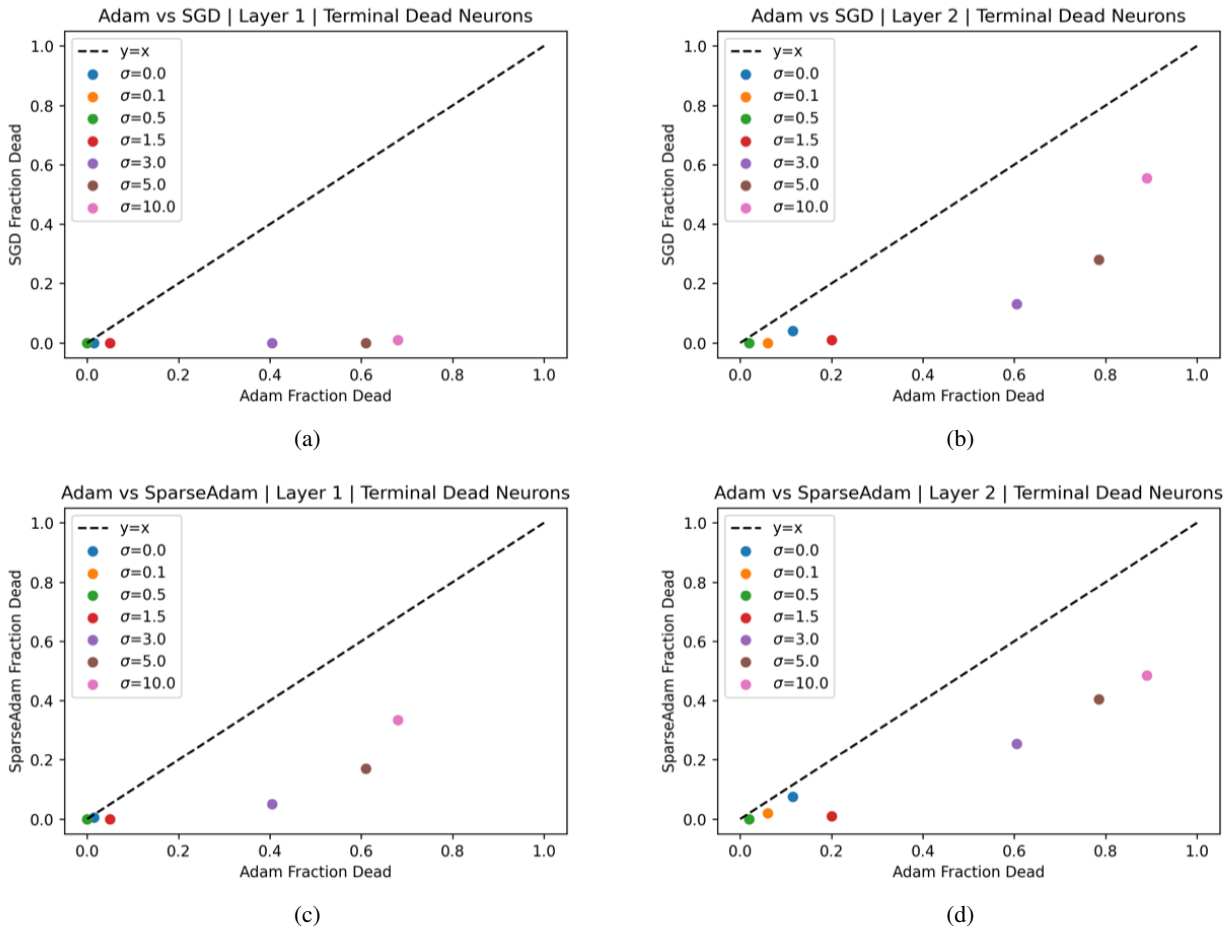
Figure 25: **SGD Sparsity as a function of Batch Size and Learning Rate.** We train 1,000 neurons in a single hidden layer on the reconstruction task for latent CIFAR10 with three noise amounts (one for each plot). We allow the lower learning rates to have more gradient steps where the [learning rate, training steps] pairs are: $[0.001, 800k]$, $[0.01, 600k]$, $[0.1, 400k]$, $[1.0, 200k]$ and we confirm that all runs have converged to the same training loss and there are no dead neurons. To make the sweeps more efficient, we randomly sample 5,000 datapoints out of the 50,000 from the CIFAR10 256 dimensional latent embeddings and use this same subset for all experiments. The only anomaly is in the bottom subfigure ($\sigma = 8$) on the bottom row where batch size is 128 and learning rate is 1.0 and the model is denser than other batch sizes. This anomaly is robust to random seeds and deserves further investigation.

I. Dead Neurons

I.1. Stale Momentum

In order to avoid Stale Momentum where the moving average are lagging and noisy for sparse models, SGD must be used but this removes the very advantages that adaptive optimizers were developed for – namely faster convergence that is more robust to hyperparameter choices. As a result, we introduce SparseAdam which was originally developed to make backpropagation more computationally efficient for highly sparse models in the "Sparse Layers" library (PyTorch, 2022). We implemented SparseAdam to work with dense layers and found that it reduces the number of dead neurons about as effectively as SGD for the three layer MLP (Fig. 26). We believe this is because, while it prevents inactive neurons from being updated by an out of date moving average, it fails to prevent gradient spikes when neurons are re-activated after a long period of quiescence.

The AlexNet (Sec. J.3) and Deep MLP (Sec. J.2) appendices both have futher analysis of the different optimizers effect on dead neurons and sparsity. Meanwhile, the next section I.2 provides further analysis on when and why Adam results in dead neurons.



(a) (b)

(c) (d)

Figure 26: **Adam leads to more dead neurons than Sparse Adam or SGD in the later layers of Deep MLPs.** We compare the final fraction of dead neurons for Adam and SGD (top row) and Adam and Sparse Adam (bottom row) on the full CIFAR10 latent embedding dataset on a classification task. We train each model for 2,000 epochs and anneal all initial noise levels down to zero between epochs 500 and 1,000. We leave out Layer 0 because there are no dead neurons for any of optimizers or noise levels.

I.2. Dead Neurons from Adam

We investigated more deeply when and why Adam produces dead neurons resulting in a few conclusions:

- Adam kills fewer neurons when it has a larger batch size – this is because more neurons are activated by at least one batch element resulting in a non-stale gradient update (Fig. 27).
- Adam kills neurons only after it has minimized its training and validation losses and converged to its sparse coding solution (Fig. 28).
- The total number of dead neurons is a function of dataset complexity. This includes the size of the dataset and the intrinsic dimensionality of the datapoints (Fig. 29).
- Neurons die more as a function of epochs through the whole dataset rather than the number of gradient steps (Fig. 30).

As shown in Fig. 27, there are fewer dead neurons with a larger batch size or smaller learning rate. This is because a large enough batch size results in more neurons being activated at least once and receiving a gradient update that is not stale.

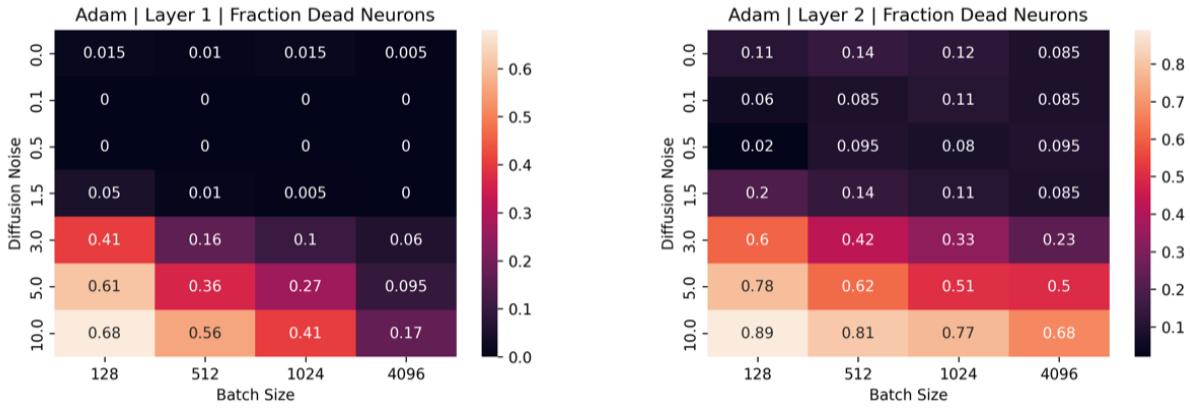
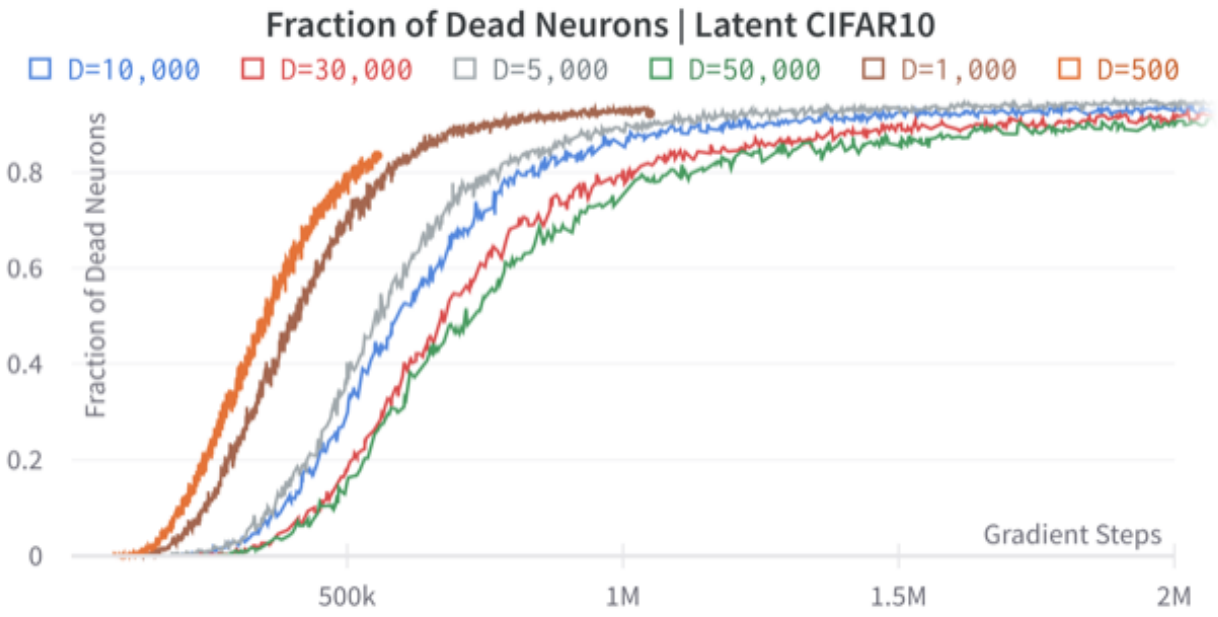


Figure 27: **Larger batches reduce dead neurons with Adam.** Across noise levels, the larger the batch size the fewer the dead neurons. All models achieve the same validation accuracy. The models are sparser in direct proportion to the number of neurons. We don't show the first layer (Layer 0) because it does not have any dead neurons. For these experiments the learning rate is always 0.0001.

Figure 28 shows how Adam starts to kill neurons only after both training loss and the number of dead neurons have converged. Figure 29 shows how dataset size and complexity determines the number of dead neurons Adam eventually kills. These two plots show different datasets (latent CIFAR10 (top) and raw CIFAR10 pixels (bottom)) and we vary the total size of the dataset D where the full dataset size is $D = 50,000$. The two main observations are: (i) the less complex latent CIFAR10 dataset results in many more dead neurons, almost independently of D; (ii) the larger D is, the more slowly neurons are killed off and the lower the total number is.

Finally, neurons die as a function of epochs rather than gradient steps. It is more than just the size of the dataset and noise introduced by the batch size but also the number of passes through the full dataset suggesting a relationship with the novelty of datapoints (Fig. 29). In Fig. 30 we show how the smallest batch size of 2 kills neurons very quickly as a function of epochs but very slowly as a function of gradient steps because it takes much longer to iterate through the dataset of 5,000 latent CIFAR10 images.

Figure 28: **AlexNet Killing Neurons Long After Training Loss and Sparsity Converge.** We use the same settings as Fig. 5 with $\sigma = 3.0$ on the full CIFAR10 latent dataset but 1,000 neurons instead of 10,000 and for a much larger number of epochs. We label the 800th epoch which shows the point where both training loss (top left) and mean number of active neurons (top right) have converged. Meanwhile, the fraction of dead neurons is only just about to go from $\sim$0 to 92% (bottom right). Validation loss looks the same as train loss.



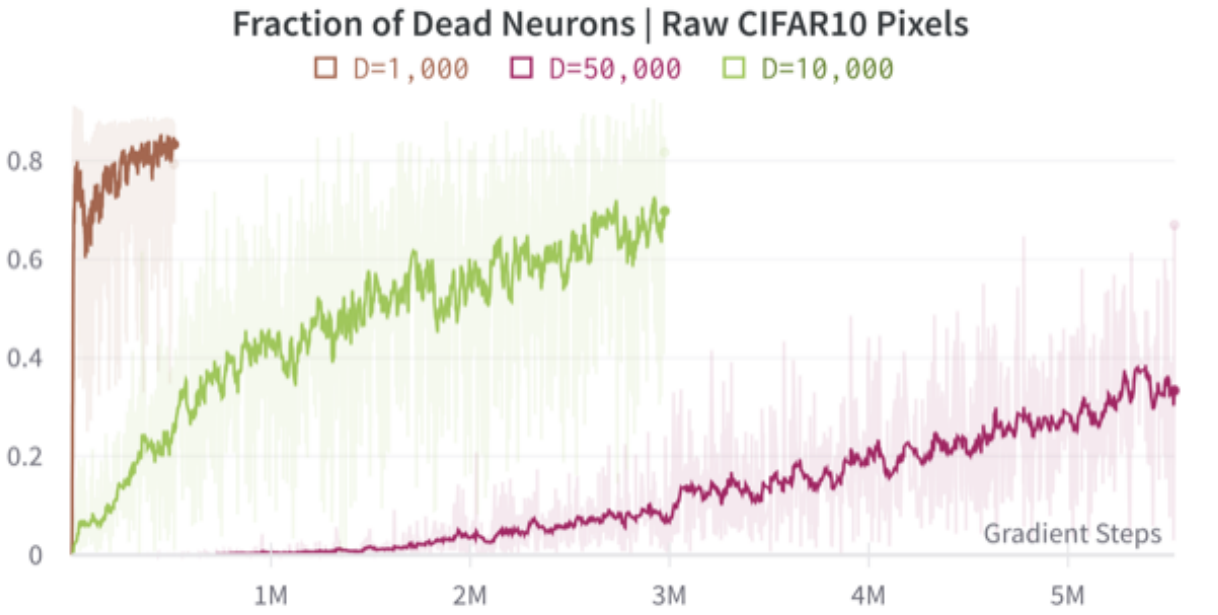


Figure 29: **Adam Dead Neurons as a Function of Dataset Complexity and Size.** We train Adam keeping all parameters including batch size constant and set $\sigma = 3.0$, only varying the size of the dataset denoted by D for latent and raw CIFAR10. The runs with smaller dataset sizes did not complete as many gradient steps due to an issue with model checkpointing but the results are sufficient for a few observations: (i) the 256 dimensional latent CIFAR dataset allows for much higher numbers of dead neurons than the 3072 dimensional raw CIFAR dataset. (ii), the smaller the dataset, the faster dead neurons are created. (iii), the latent CIFAR10 dataset is sufficiently simple that $\sim$92% of neurons eventually die almost independently of dataset size. For clarity, we smooth our lines in the bottom figure.

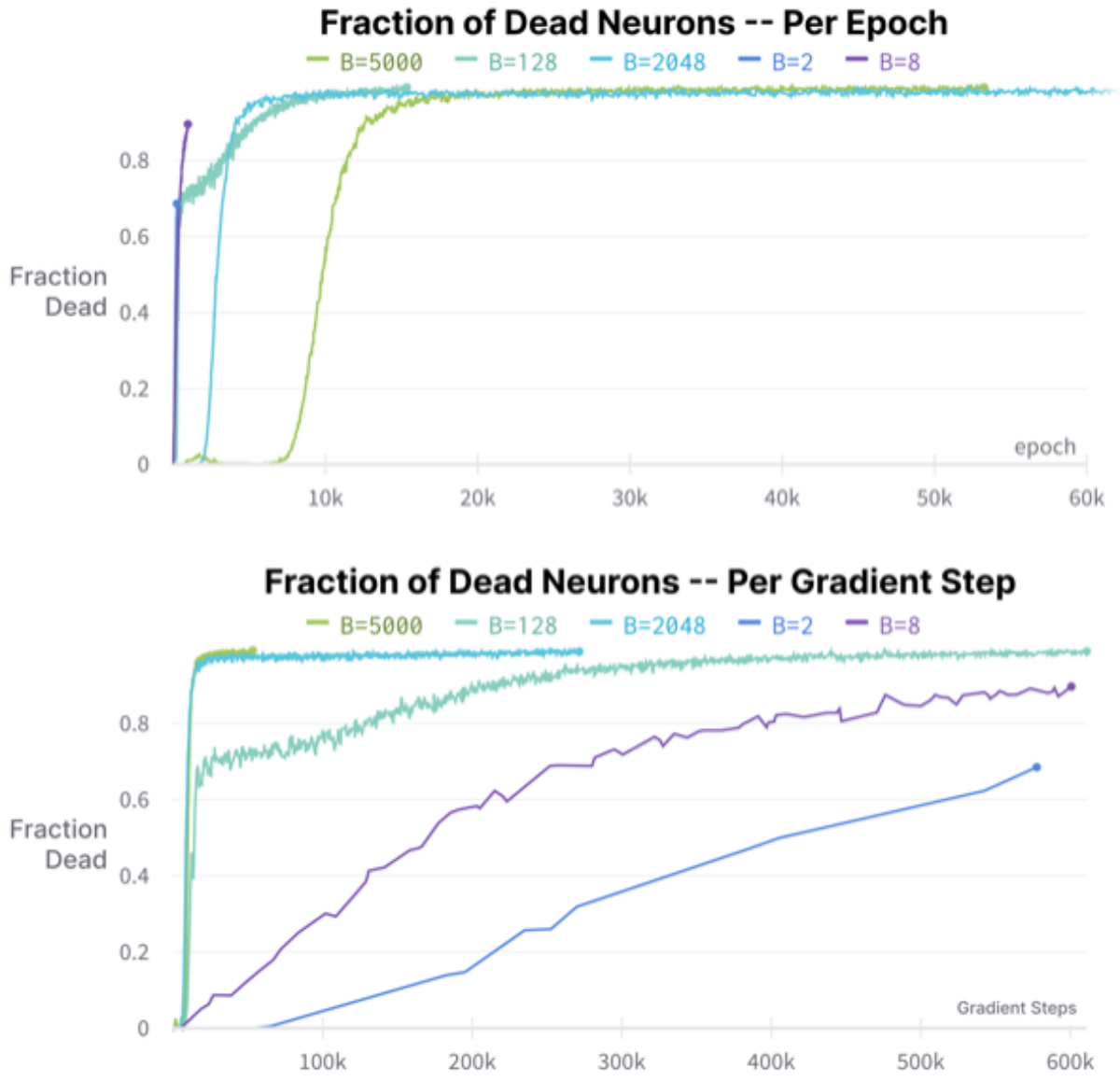


Figure 30: **Neuron dead as a function of epochs vs gradient steps.** Using the random 5,000 latent CIFAR10 images (keeping the subset constant) and $\sigma = 3.0$ with Adam we vary the batch size from full batch $B = 5,000$ (green) to $B = 2$ (dark blue). The top row shows that the smallest batch sizes kill neurons fastest as a function of epochs. The bottom row shows the opposite where the smallest batch sizes die slowest as a function of gradient steps. These results suggest that both larger batch size and more data diversity are important for slowing neuron death.

I.3. Dead Neurons from Noise Annealing

Here we show the number of dead neurons that are present because of noise annealing for reconstruction and classification (Fig. 31). This source of dead neurons is separate from using the Adam optimizer (App. I.2).

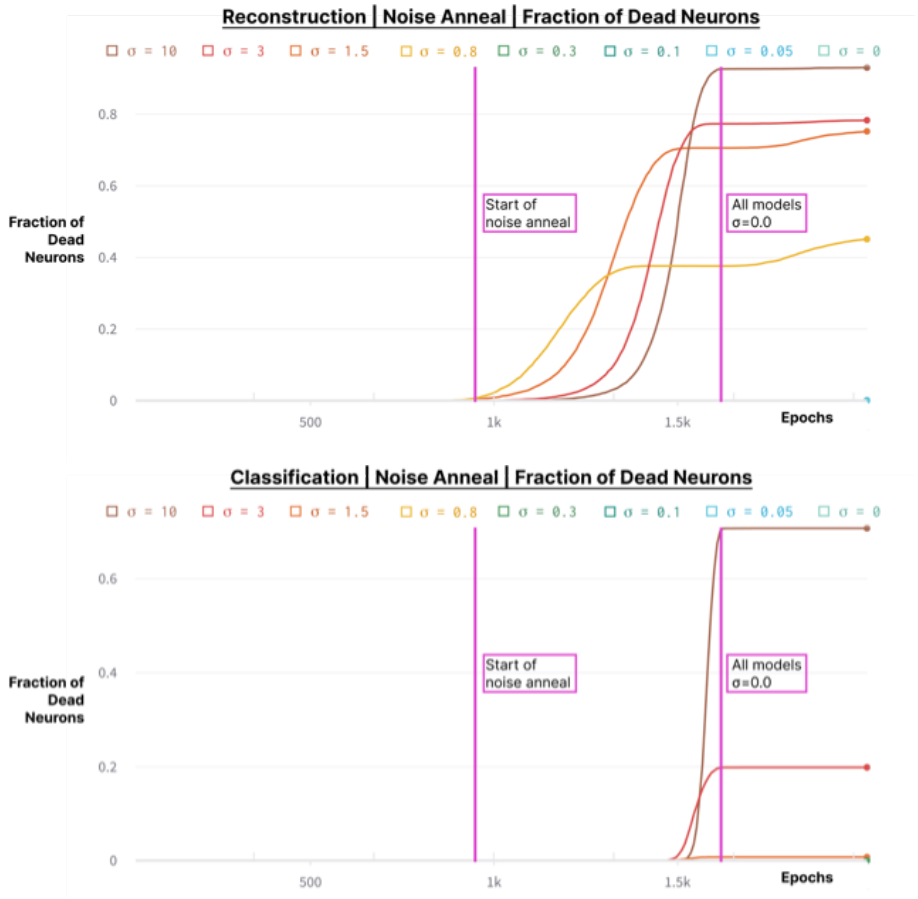


Figure 31: **Dead Neurons during the reconstruction (top) and classification (bottom) tasks.** Reconstruction (top) during noise annealing, the higher noise levels $\sigma \geq 0.8$ see a number of dead neurons. However, this does not full explain their sparsity levels and on this trivial task it does not hurt their reconstruction performance. Classification (bottom) here there are fewer dead neurons, they only appear for $\sigma \geq 3.0$ and they only appear close to when the noise is entirely turned off. Again these results do not fully explain the sparsity amounts found.

J. Deeper Models

J.1. Transformer

Here we provide the training and validation performance for the Transformer as well as sparsity values. This is after 1.7 million training steps when the models look to have converged. Table 1 shows all of the data and Fig. 32 provides a visual depiction of how sparsity levels change for each layer as a function of noise. Only the largest noise level $\sigma = 8$ leads to a slight reduction in model performance.

Noise σ	Layer 1	Layer 2	Layer 3	Train Loss	Val. Loss	Val. Acc
0.0	0.171	0.184	0.296	2.790	3.048	0.437
0.05	0.173	0.178	0.291	2.782	3.038	0.437
0.1	0.164	0.165	0.287	2.793	3.029	0.438
0.3	0.172	0.117	0.252	2.769	3.025	0.439
0.8	0.155	0.056	0.186	2.759	3.002	0.442
1.5	0.133	0.046	0.148	2.778	3.009	0.441
3.0	0.058	0.022	0.137	2.830	3.011	0.440
8.0	0.013	0.001	0.055	3.073	3.134	0.420

Table 1: **Transformer Sparsity and Performance Levels** - We show the mean number of active neurons for each batch across all three layers in addition to the noise levels and training loss obtained at the end of 1.7 million training steps on the WikiText-103 dataset. After 200k training steps the noise levels are annealed down to zero. Only the highest noise level $\sigma = 8$ has a noticeable hit to performance while $\sigma = 3$ has a minor hit to performance while being significantly sparser.

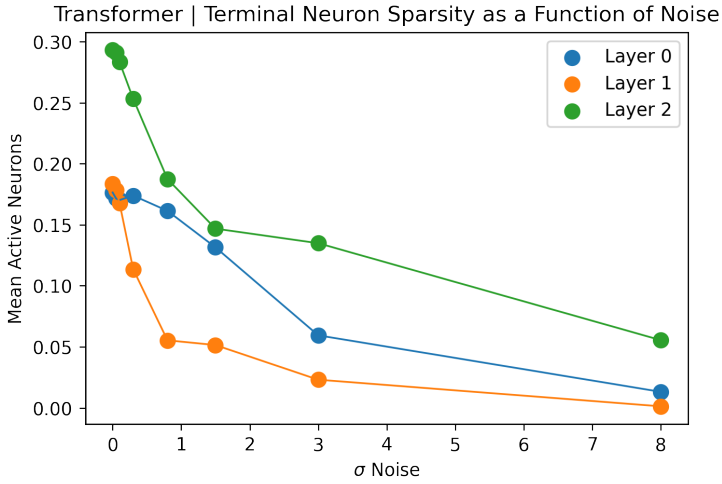


Figure 32: **Transformer Layer Sparsity.** Only the most sparse model $\sigma = 8.0$ has a real hit to train and validation loss.

J.2. Deep MLP

Here we provide the validation accuracies for the Deep MLPs in Table 2 and use Fig. 33 to show how sparsity levels and dead neurons change for each layer as a function of noise, comparing the different optimizers.

Optimizer	Noise	Train Acc	Val. Acc
Adam	0.0	1.0	0.915
SGD	0.0	1.0	0.920
SparseAdam	0.0	1.0	0.915
Adam	0.1	1.0	0.925
SGD	0.1	1.0	0.923
SparseAdam	0.1	1.0	0.924
Adam	0.5	1.0	0.926
SGD	0.5	1.0	0.924
SparseAdam	0.5	1.0	0.927
Adam	1.5	1.0	0.924
SGD	1.5	1.0	0.928
SparseAdam	1.5	1.0	0.923
Adam	3.0	1.0	0.917
SGD	3.0	1.0	0.923
SparseAdam	3.0	1.0	0.921
Adam	5.0	1.0	0.916
SGD	5.0	1.0	0.919
SparseAdam	5.0	1.0	0.917
Adam	10.0	1.0	0.915
SGD	10.0	1.0	0.923
SparseAdam	10.0	1.0	0.915

Table 2: **Deep MLP Validation Accuracy is better for small amounts of noisy pretraining but independent of dead neurons** - We show the final validation accuracies for the Deep MLP models that have been noise annealed for the CIFAR10 latent embeddings classification task. Models with small amounts of noise do slightly better than the baseline without noise. There is no significant difference in performance between noise levels or optimizers, even though they kill off different numbers of neurons (Fig. 33).

The SGD optimized model only has dead neurons in its final layer and is almost as sparse as Adam which has the most dead neurons. This is accomplished by the SGD model using negative bias terms in its first layer and then negative weights in its later layers to ensure fewer neurons are activated. This is in contrast to Adam and SparseAdam which only create sparsity through dead neurons. It is worth pointing out that SparseAdam does use negative bias terms to create sparsity and never has dead neurons before noise is annealed, afterwards the neurons die and become the source of sparsity while the bias terms become positive.

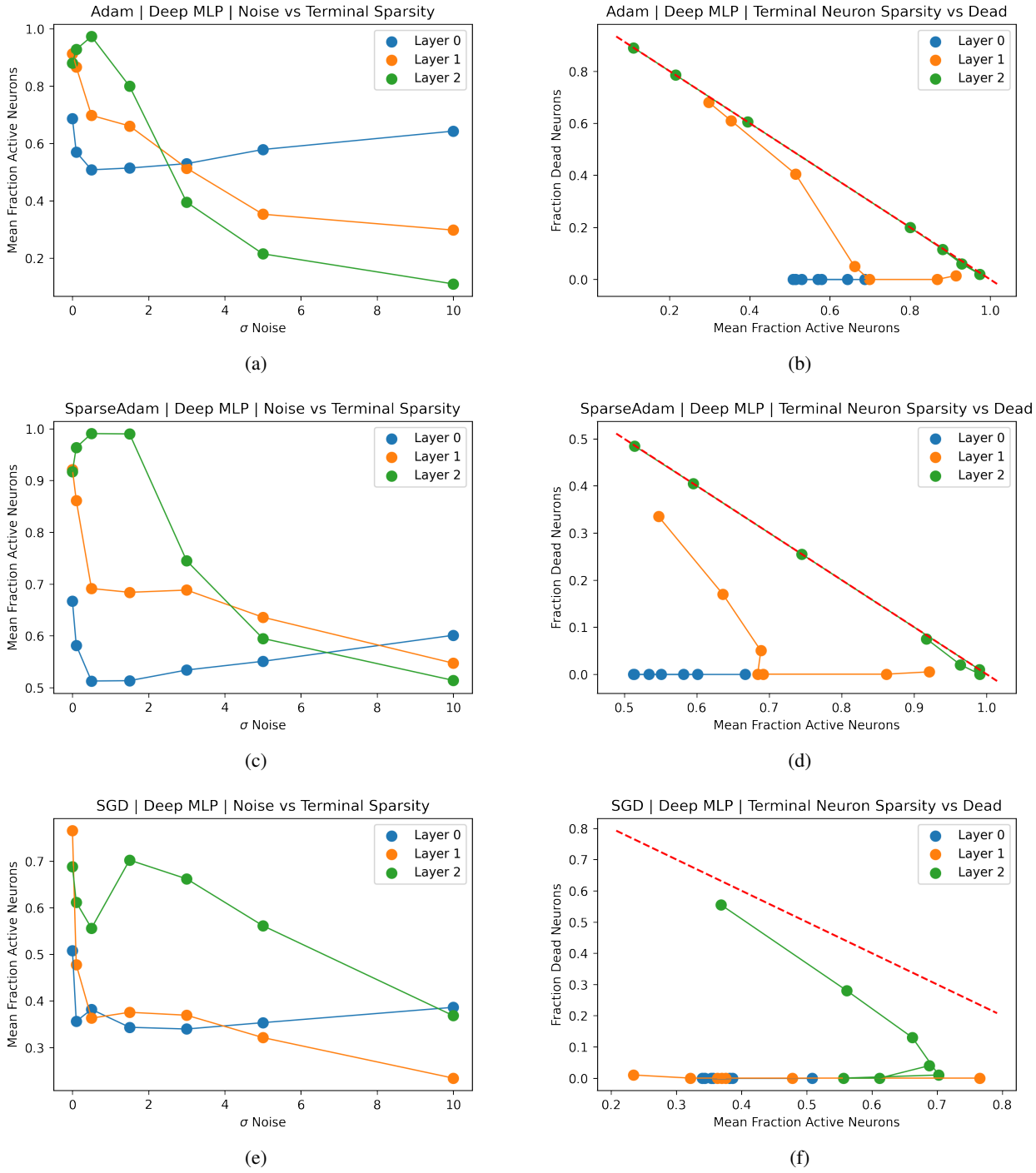


Figure 33: **Deep MLP Layer Sparsity and Dead Neurons across Optimizers.** Adam (top row) produces the most sparse models but this is almost entirely a consequence of dead neurons. Meanwhile, SGD (bottom row) is the only model to avoid dead neurons in its second layer (orange line) and implement a sparse coding solution analogous to the results from our single layer MLP. Note the different ranges for the y-axes.

J.3. AlexNet

Here we provide the validation accuracies for AlexNet in Table 3 and use Fig. 35 to show how sparsity levels change for each layer as a function of noise. Interestingly, even for the smallest noise ($\sigma = 0.5$) the noisy pretraining significantly hurts validation accuracy but not training accuracy (the model always overfits in every case to get 100% train accuracy). The noise was found to produce dead neurons like in the DeepMLP setting. Believing these dead neurons may explain the decrease in validation accuracy, we trained the network using SGD and SparseAdam. While this solved the dead neuron problem (see Fig. 34), Table 3 shows no increase in validation accuracy. Testing AlexNet with Average instead of Max Pooling also did not alleviate the noise performance gap.

Optimizer	Noise	Train Acc	Val. Acc
Adam	0.0	1.0	0.815
SGD	0.0	1.0	0.710
SparseAdam	0.0	1.0	0.820
Adam	0.5	1.0	0.614
SGD	0.5	1.0	0.575
SparseAdam	0.5	1.0	0.618
Adam	1.5	1.0	0.540
SGD	1.5	1.0	0.559
SparseAdam	1.5	1.0	0.551
Adam	3.0	1.0	0.597
SGD	3.0	1.0	0.568
SparseAdam	3.0	1.0	0.602
Adam	5.0	1.0	0.613
SGD	5.0	1.0	0.593
SparseAdam	5.0	1.0	0.618
Adam	10.0	1.0	0.648
SGD	10.0	1.0	0.625
SparseAdam	10.0	1.0	0.647

Table 3: **AlexNet Validation Accuracy Damage From Any Noisy Pretraining** - We show the final train and validation accuracies for the AlexNet models across optimizers. The SGD and Sparse Adam reductions in dead neurons do not translate into validation accuracy improvements.

In this light, it is interesting that the Transformer is immune to dead neurons and that noise does not affect final validation accuracy. This is particularly surprising given that noise is injected at every Transformer MLP layer instead of just at the first layer. We hypothesize this may be due to the Transformer's residual connections and Layer Normalization that occurs right before the MLP weights to rescale the activations (Ba et al., 2016).

We also experimented with injecting noise after the five convolutional layers, just before the MLP layers. However, this also harmed validation accuracy (albeit to a lesser extent). The noisy pretraining results in validation accuracies from 76% (for $\sigma = 0.5$) down to 70% (for $\sigma = 10$) compared to the baseline 82%. Like with injecting noise at the deeper layers of the MLP, doing so caused the convolutional layers to become more active than they would otherwise be and this may have contributed to the reduced performance.

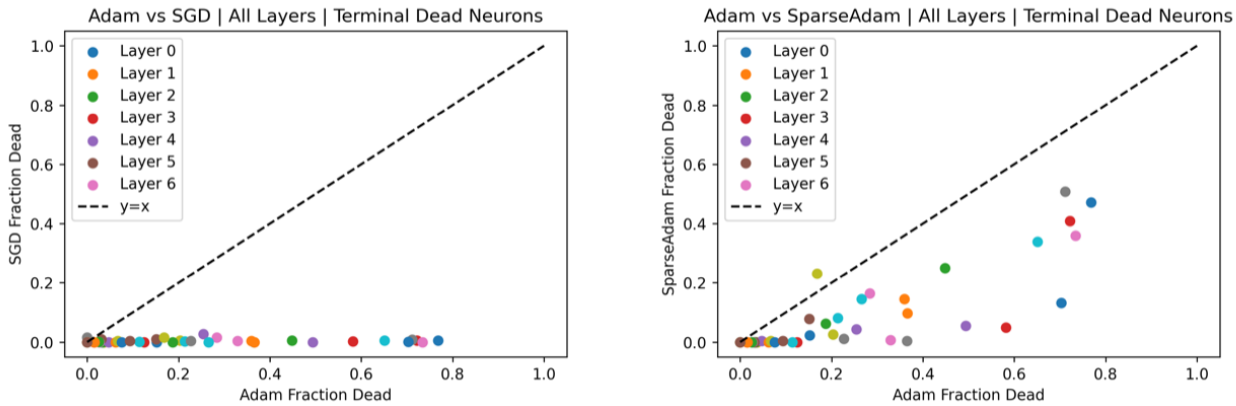


Figure 34: **AlexNet Dead Neurons By Optimizer.** SGD (left plot) results in no dead neurons while SparseAdam (right) creates fewer dead neurons in all but one case (all points aside from one are below the dotted line $y = x$). Dots correspond to the layer and we plot all noise levels on the same plot.

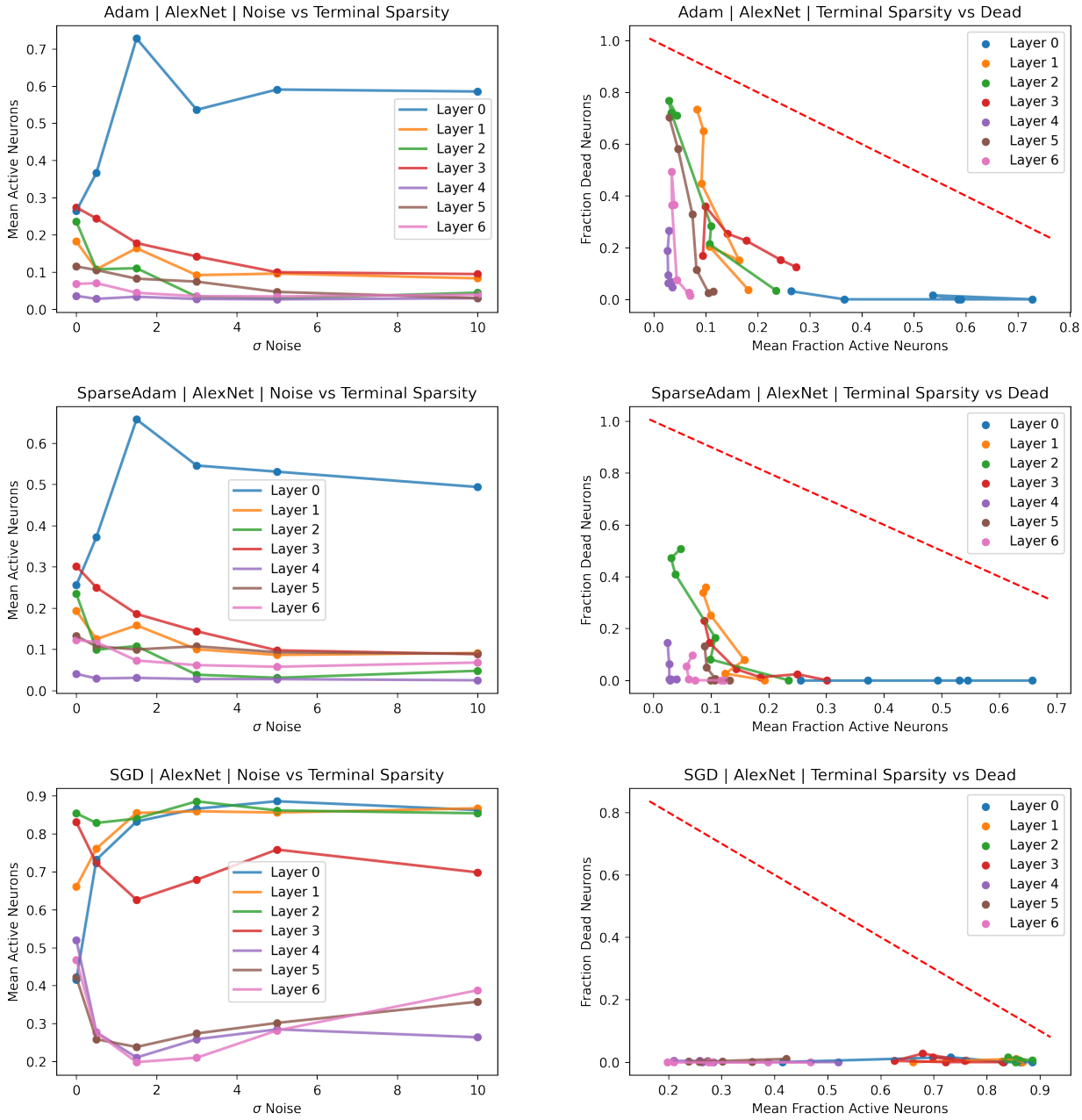


Figure 35: **AlexNet Sparsity and Dead Neurons By Layer.** We see the same effect upon noise annealing here as with the DeepMLP where the first layer starts sparse but then becomes significantly more dense than baseline while the later layers all remain more sparse. Note the different y-axes scales for each plot.

K. Why Transformer Attention, Hopfield Nets, and Sparse Distributed Memory show the inverse relationship between sparsity and noise

For Transformer Attention, the difference is that it has direct access to every input within its receptive field but no further. This provides for high accuracy when dealing with these inputs (keys) but an inability to store long term memories. As a result, in the low noise regime, it is optimal for Attention to implement a form of nearest neighbour lookup, activating few keys. When there is more noise, especially noise that moves the query so far from its target key that it is no longer the nearest, it is optimal to activate more keys and average over them, losing accuracy but giving a solution in the correct neighbourhood.

By contrast, SDM can store long term memories. However, it has the same negative correlation between noise and sparsity as Attention because it attempts to store full patterns with maximum fidelity within each neuron instead of distributing the features of each pattern across neurons (Kanerva, 1988). This results in SDM attempting to also perform a nearest neighbour lookup, activating only as many neurons around the noisy query as is necessary to activate those storing the target pattern.

Our sparse coding networks in this work take the opposite approach where they learns subcomponents that generalize across images instead of all the features of specific images. As a result, they always want to use as many neurons as possible, but must reduce the number used in the presence of noise to avoid interference.