Superposition, Memorization, and Double Descent

AUTHORS

Tom Henighan[*], Shan Carter[*], Tristan Hume[*], Nelson Elhage[*], Robert Lasenby, Stanislav Fort, Nicholas Schiefer, Christopher Olah[‡]

AFFILIATION

Anthropic

* Core Research Contributor; ‡ Correspondence to colah@anthropic.com; Author contributions statement below.

PUBLISHED

January 5, 2023

In a recent paper [1], we found that simple neural networks trained on toy tasks often exhibit a phenomenon called superposition [2, 3, 4], where they represent more features than they have neurons. Our investigation was limited to the infinite-data, underfitting regime. But there's reason to believe that understanding overfitting might be important if we want to succeed at mechanistic interpretability, and that superposition might be a central part of the story.

Why should mechanistic interpretability care about overfitting? Despite overfitting being a central problem in machine learning, we have little mechanistic understanding of what exactly is going on when deep learning models overfit or memorize examples. Additionally, previous work has hinted that there may be an important link between overfitting and learning interpretable features [5].

So understanding overfitting is important, but why should it be relevant to superposition? Consider the case of a language model which verbatim memorizes text. How can it do this? One naive idea is that it might use neurons to create a lookup table mapping sequences to arbitrary continuations. For every sequence of tokens it wishes to memorize, it could dedicate one neuron to detecting that sequence, and then implement arbitrary behavior when it fires. The problem with this approach is that it's extremely inefficient – but it seems like a *perfect* candidate for superposition, since each case is mutually exclusive and can't interfere.

In this note, we offer a very preliminary investigation of training the same toy models in our previous paper on limited datasets. Despite being extremely simple, the toy model turns out to be a surprisingly rich case study for overfitting. In particular, we find the following:

- Overfitting corresponds to storing data points, rather than features, in superposition.
- Depending on dataset size, our models fall into two different regimes: an overfitting regime (characterized by storing data points in superposition), and a generalizing regime (characterized by storing features in superposition).
- We observe double descent [6, 7, 8, 9] as the model transitions between these regimes.

Experiment Setup

We hypothesize that real neural networks perform operations in a sparse, high-dimensional "feature" space, but these features are difficult for us to see directly because they're stored in superposition. Motivated by this, we attempt to simulate this feature space using synthetic input vectors x which are sparse, high-dimensional, and non-negative (similar to our previous paper). Concretely, $x \in \mathbb{R}^n$ is a $n = 10,000$ dimensional vector. We let individual features $x_i = 0$ with probability $S = 0.999$, but otherwise uniformly distributed between $[0, 1]$. However in contrast to our previous work, we then rescale x such that $||x||^2 = 1$, as this will make memorization of training examples easier[1][2]. We also consider training sets of finite size T, whereas our previous work only considered $T = \infty$. We will use $X \in \mathbb{R}^{n \times T}$ to refer to the matrix of training data, where each column X_i is a training data point.

We consider the the "ReLU Output" toy model, defined as

ReLU Output Model

$$h = Wx$$

$$x' = \text{ReLU}(W^T h + b) = \text{ReLU}(W^T W x + b)$$

where W is Xavier-initialized [10]. Models are trained to minimize mean-squared reconstruction error.

$$L = \frac{1}{T}\sum_x \sum_i I_i (x_i - x_i')^2$$

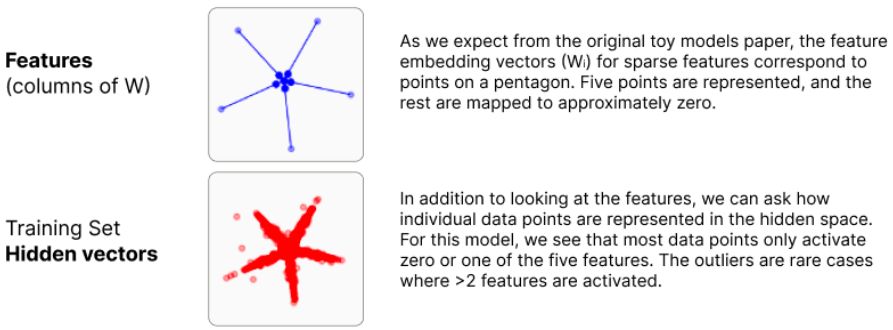
In this work we limit ourselves to uniform importance $I_i = 1 \quad \forall i$.

We use 50,000 full-batch updates, as opposed to mini-batch, using the AdamW [11] optimizer. Our learning rate schedule includes a 2,500 step linear-warmup to 1e-3, followed by a cosine-decay to zero. The number of training updates, the use of full-batch optimization, and the annealed learning rate all seem important for our results. We focus on very low-dimensional hidden spaces which seems to make this a more difficult optimization problem. Additionally, having extremely sparse features makes gradients much noisier. For this reason, it's important to optimize for a large number of steps using full-batches to produce these results. In some experiments, we'll further train multiple models with different random seeds for parameter-initialization and select the one with the lowest training loss in order to avoid local minima. However we qualitatively find these sensitivities are greatly reduced for models with $m > 3$.

Unless otherwise specified, we use a weight decay of $1e-2$. In line with previous work, we find that double descent is strongest with low values of weight decay and vice versa.

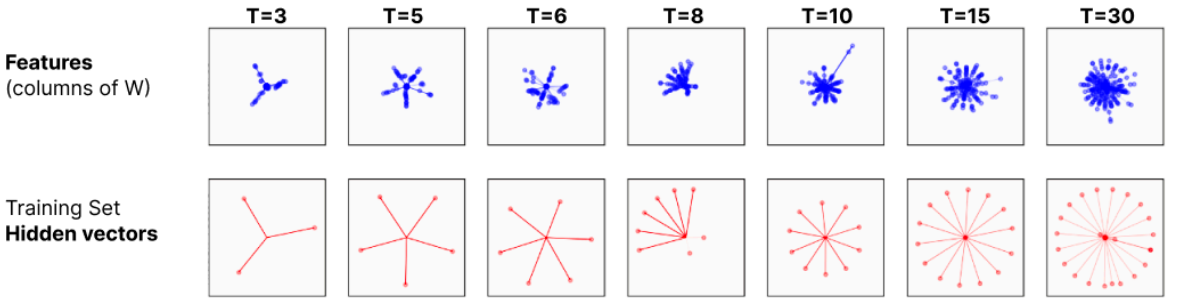
"Datapoint Features" vs "Generalizing Features"

In the "normal superposition" we described in our previous paper, we found that the model embeds more features than it has dimensions, often mapping them to regular polytopes. For example, if the model has a two dimensional hidden space, sparse features will be organized as a pentagon:



But what happens if we train models on finite datasets instead? It turns out that the models we find will often look very messy and confusing if you try to look at them from the perspective of features, but very simple and clean if you look at them from the perspective of *data point activations*.

Let's visualize a few ReLU-output models trained on datasets of different sizes, with many sparse features. We'll focus on models with $m = 2$ hidden dimensions. We pick a two-dimensional hidden space to allow explicit visualization, while the task setup is chosen to produce the most extreme version of the phenomenon we wish to highlight.[3] If we visualize the resulting columns of W, as in the previous paper, everything is "messy", forming complicated scatters of features rather than clean polygons. But if we instead visualize *the training-set hidden-vectors* ($h_i = WX_i$), we see clean structure:



The data points – rather than the features – are being represented as polytopes![4] We might think of these models as operating on a different kind of feature: a "single data point feature" that can be used for memorization, instead of a "generalizing feature". This suggests a naive mechanistic theory of overfitting and memorization: memorization and overfitting occur when models operate on "data point features" instead of "generalizing features". We expect this naive theory to be overly simplistic, but it seems possible that it's gesturing at useful principles!

How Do Models Change with Dataset Size?

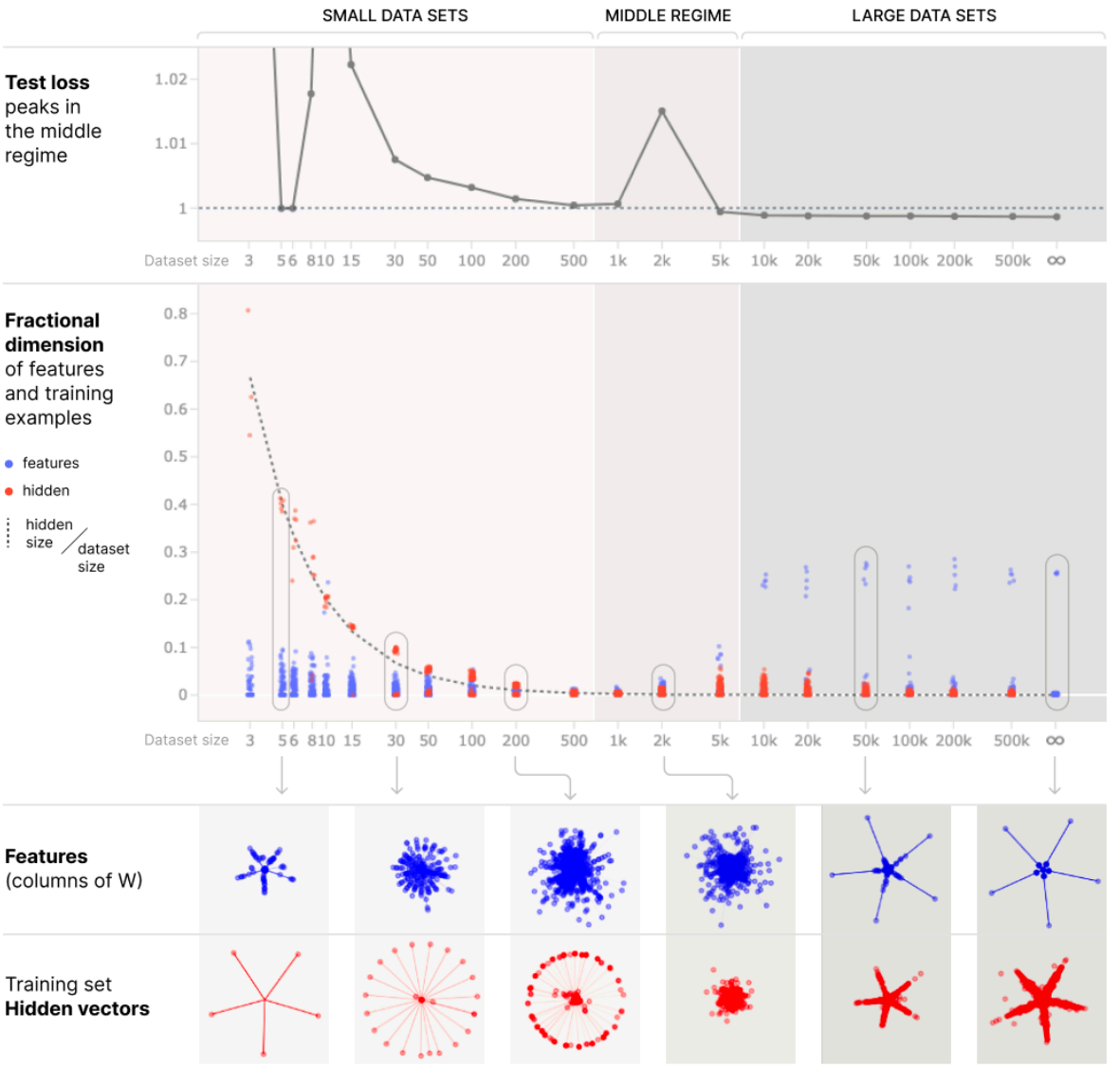
What happens as we make the dataset larger? Clearly our toy models behave very differently in the small data regime where they "use data points as features" and the infinite data regime where they learn the real, generalizing features. What happens in between?

In the original paper, the notion of "feature dimensionality" was helpful for studying how the geometry of features varies as we change models. For this note, we'll extend our notion of feature dimensionality (which we will denote as D_{f_i} for the remainder of this text) to also define the dimensionality of a training example, D_{X_i}, as:

$$D_{X_i} = \frac{||h_i||^2}{\sum_j (\hat{h_i} \cdot h_j)^2}$$

where $h_i = WX_i$ is the hidden-vector associated with a training example X_i and $\hat{h_i}$ is the unit vector in the direction of h_i. If the training examples are embedded in an T-gon in $m = 2$, we should expect each training example to have dimensionality m/T.

We can now visualize how the geometry of features and data points changes as we vary the size of the dataset. In the middle pane below is a scatter plot of both feature and training-example dimensionalities for varying the dataset size (we will discuss the test loss in the top pane in a later section).



In the small data regime on the left, we see that while the feature dimensionalities are small, the training-example dimensionalities follow m/T as expected. In the red vector-plots on the bottom, we see that the models are forming T-gons with the h_i. In fact, the ReLU output model can memorize any number of orthonormal training examples using T-gons if given infinite floating-point precision. This motivated our choice to normalize our training examples, though this constraint can be lifted if the W and W_{up} are untied. See this Colab notebook for more details.

In the large data regime on the right, we see 5 features whose dimensionalities are large, while the rest of the feature and training-example dimensionalities are small. The blue vector plots show that those 5 features are represented in a pentagon, while the rest are largely ignored. We provide some intuition as to why one should expect this ~5 feature solution in this colab. The fractional dimension of the pentagon features is notably less than the expected 2/5. We believe this is due to there being many other features (9,995) whose individually small contributions add-up to a significant fraction of the denominator in D_{f_i}.

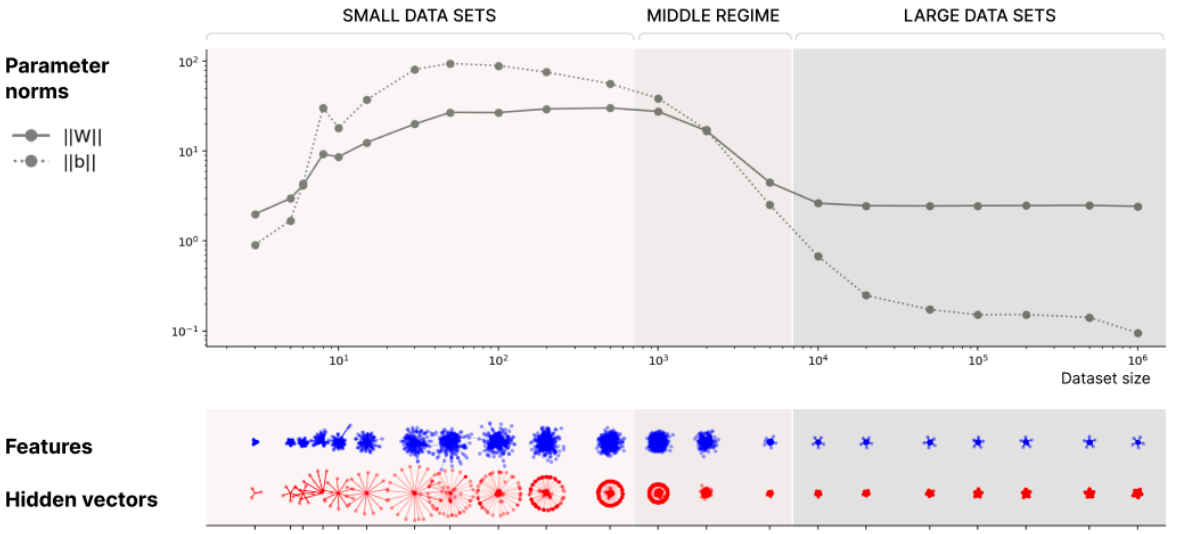
Most data examples have nonzero values for only zero or one of the 5 pentagon features, causing the hidden-vectors to also trace out a pentagon in the bottom-right red subfigure. The outliers represent rare cases with >1 nonzero values.

In between these two extremes, things are messier and harder to interpret.

We did not use a consistent scale for the red and blue vector plots in the previous figure. Using a consistent scale (see below figure) reveals that lengths of both hidden and feature vectors vary widely with dataset size, peaking around $T = 30$ before rapidly declining in the middle regime, and plateauing in the large-data regime. Plotting the Frobenius norm of W and the l2-norm of b reveals the same trend for the model parameters.

A few comments on these trends:

- **Small Data Regime:** One can show that a model memorizing orthonormal training examples via T-gons should have larger W and b for larger T (see this colab for more details). This agrees with the observed behavior for $T < 30$.
- **Large Data Regime**: It seems intuitive that once the model has enough data to generalize (ie the pentagon), adding more data causes relatively minor changes. This matches our observation of a plateau in the large-data regime. In our oversimplified solution for infinite data, we found $||W|| = \sqrt{5} \approx 2.2$, which appears surprisingly close to the true value (though our assumption of $b = 0$ was clearly unrealistic).



Superposition Double Descent

The phenomenon of models behaving very differently in two different regimes, with strange behavior in between, is eerily reminiscent of double descent [6, 7, 8], especially "data double descent" (eg. [9]). One striking phenomenon of data double descent is that test loss gets *worse* before it gets better – in violation of naive intuitions that more data should always reduce overfitting!

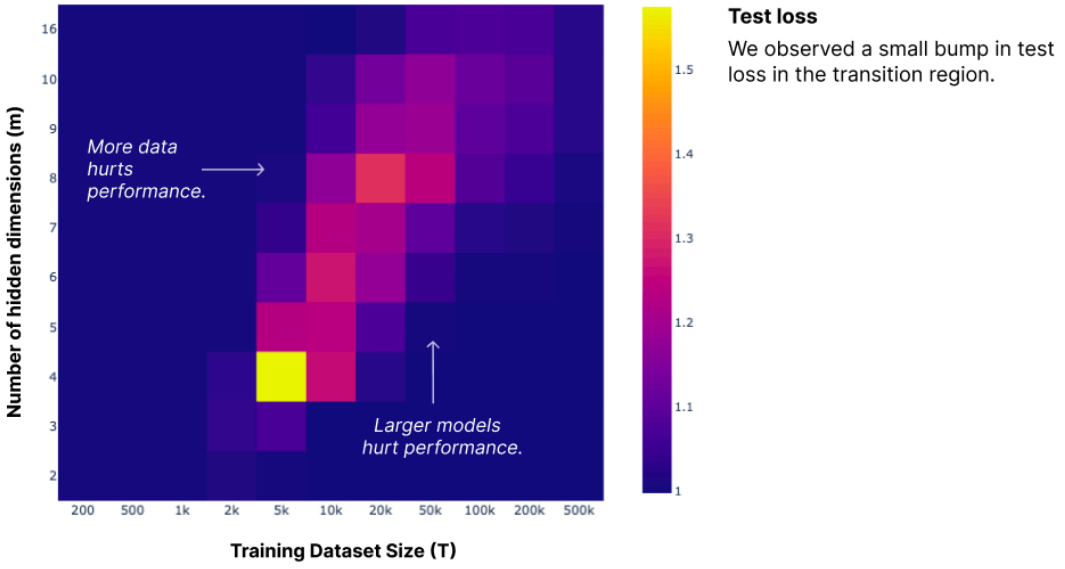
For a given T, the model's solution will depend on the randomly-chosen training set, where some will lend themselves to memorization (e.g. orthogonal training examples) and others to generalization. To ensure our results aren't a fluke, we trained 4 models with different dataset random seeds for each dataset size. We then plotted the average test loss (top pane in our first figure), revealing a clear bump at the transition between the "data points in superposition" regime and the "generalizing features in superposition" regime.

It's interesting to note that we're observing double-descent in the absence of label noise. That is to say: the inputs and targets are *exactly* the same. Here, the "noise" arises from the lossy compression happening in the downprojection. It is impossible to encode 10,000 features into 2 neurons with a linear projection, even in the sparse limit. Thus the reconstruction is necessarily imperfect, giving rise to unavoidable reconstruction error and consequently, double-descent [12].

The Effect of m on Double Descent

At this point, it's natural to wonder whether the double descent might be an artifact of only having $m = 2$ hidden dimensions,or difficulties with optimization. In this section, we'll confirm that this isn't the case. We'll also be able to explore a theme in some of the double descent literature – understanding it not as a one-dimensional phenomenon, but as a multi-dimensional interaction between model size, dataset size, and training [9].

We visualize double descent as a two-dimensional function varying both the number of training examples, T, and the number of hidden dimensions, m. All other hyperparameters are the same as above. We train four models for each (T, m) configuration, averaging the resulting losses. We empirically found optimization to yield much more consistent results for $m > 3$.



There are clearly regions where "double descent" occurs – regions where bigger models or more data hurt performance.

Consistent with prior work on double descent, these results are sensitive to weight decay and the number of training epochs. In the appendix, we show that for $m = 4$ models

- The test-loss bump increases when training for more epochs
- Increasing weight-decay from 1e-2 to 1.0 removes the bump entirely

Conclusion

We find that, in toy models, memorization can be understood as models learning "single data point features" in superposition. These models exhibit double descent as they transition from this strategy of representing data points to representing features.

There is much more to explore. The most obvious question is whether the naive mechanistic theory of overfitting that these results suggest generalizes at all to real models. But there's also a lot to ask in the context of toy models:

- What happens when models learn a mixture of features and single data point features? (This could perhaps be studied using partially repeated data, as in [13].)
- What happens in the "middle regime" as models transition from one strategy to the other and losses spike? What are these models doing mechanistically?
- Is there some notion of "what is a feature" or "how to recognize features" which can encompass generalizing features, as well as single data points?

Comments & Replications

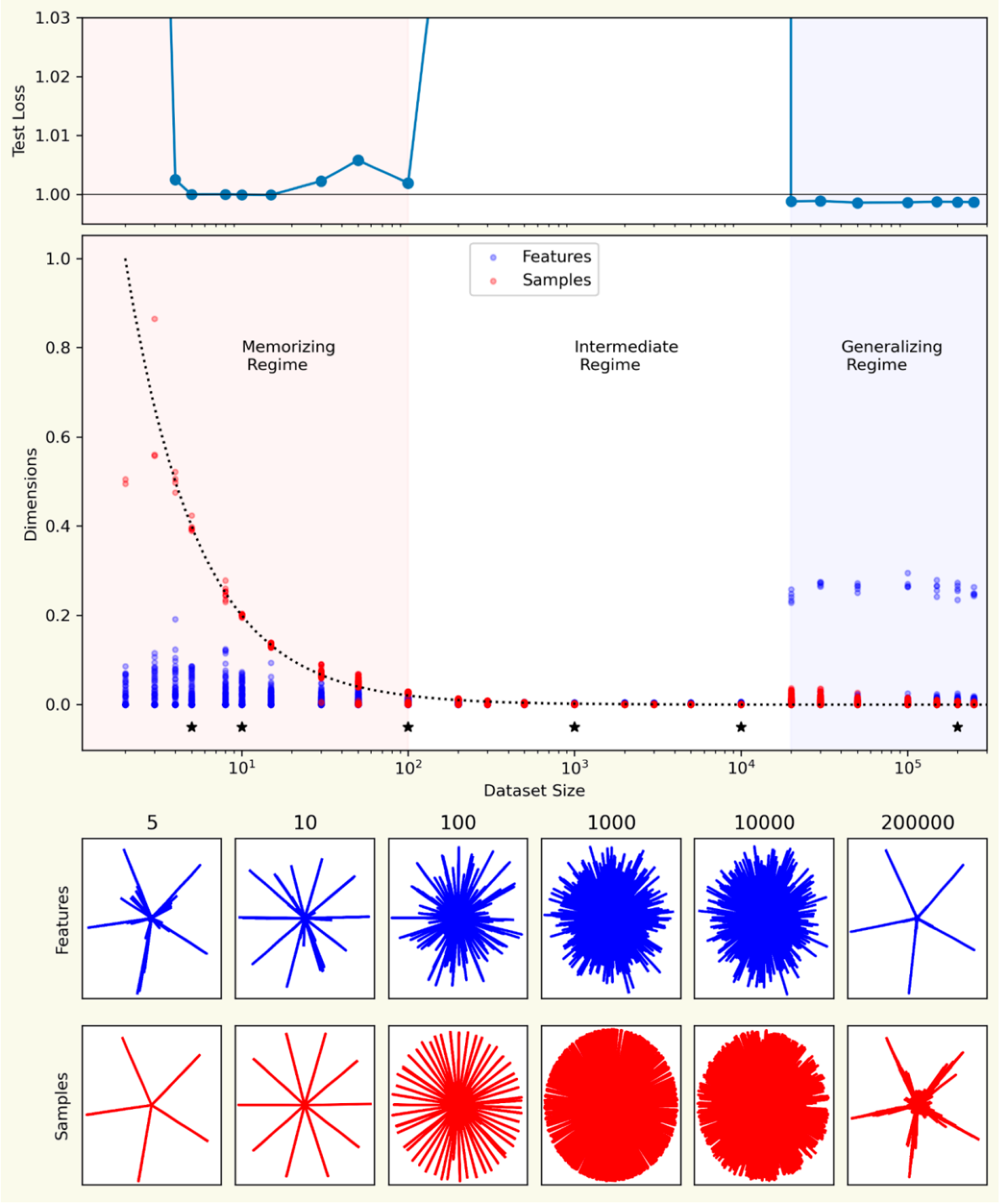
Inspired by the original Circuits Thread and Distill's Discussion Article experiment, the authors invited several external researchers who we had previously discussed our preliminary results with to comment on this work. Their comments are included below.

REPLICATION

Adam Jermyn is an independent researcher focused on AI alignment and interpretability.

After seeing preliminary results, I replicated the results in the section "How Do Models Change with Dataset Size?" for models with hidden dimension $m = 2$. Overall I found good qualitative agreement. There are some quantitative differences between my results and those shown in the paper, but nothing that I expect to affect any of the conclusions.

The figure below corresponds to the first figure in that section, and shows qualitatively similar features:



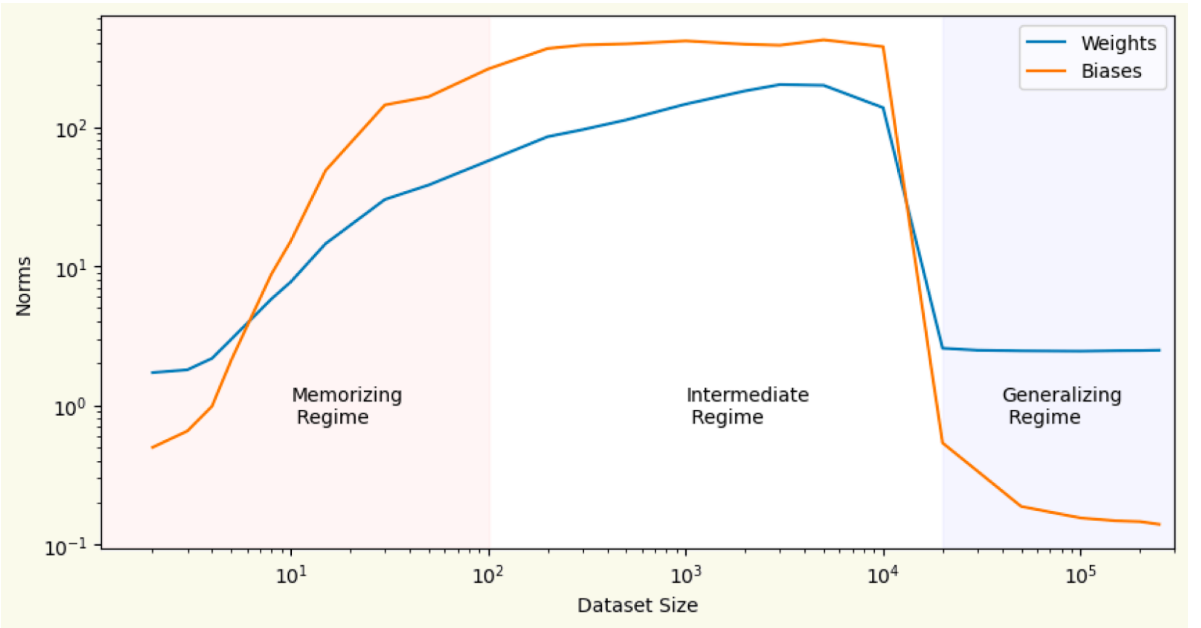
In particular, this replication shows the same division into three regimes, of memorizing samples from small datasets, learning generalizing features from large datasets, and doing something more complicated in between, and the sample and feature embeddings look qualitatively similar between my models and the ones shown in the paper..

There are three differences between this and the corresponding figure in the paper that I can see, and I think they may be related:

1. The onset of the generalizing feature regime is at larger dataset sizes for my models. I see it at T=20,000 and the paper has it starting at T=5,000.
2. The onset of the intermediate regime is at smaller dataset sizes for my models (T ~ 100 versus T ~ 700), as measured either by deviations from uniform sample embeddings or by jumps in the test loss.
3. The test losses in the middle regime are much larger for my models: as high as 5,000 versus ~1.01 in the paper.

I ran my models multiple times and verified that the different instances replicate these differences. I have not been able to pin down where these differences come from, and as far as I can tell I have trained my models precisely as described in the text, though it is certainly possible that I have missed something!

I also reproduced the second figure of the same section:



The general trends are very similar. In particular:

1. The bias norms start out smaller than the weight norms for small dataset sizes, become larger than the weight norms for somewhat larger datasets, and are then smaller in the generalizing regime.
2. Both norms rise with increasing dataset size, and rapidly fall back down once the models learn generalizing features.

There are again differences, though these are quantitative rather than qualitative. In particular, the peak bias norms in my models are roughly 3 times larger than those in the paper, and I see a rise in the weight norms over the range T=100-1000 whereas the figure in the paper shows more of a plateau.

Original Authors' Response: Thanks for replicating this! It's really nice to see that everything qualitatively reproduced. We're uncertain what caused the shift in the dataset size at which the transition occurs. It seems like there must be some hyperparameter difference between our setups, but we're uncertain what it is! However, since we only really care about the existence of the transition, and not exactly where it falls for this toy problem, we're not that concerned about identifying the exact difference.

REPLICATION

Marius Hobbhahn *is a PhD student at the University of Tuebingen.*

I replicated most findings in the "Superposition, Memorization, and Double Descent" paper. I changed the setup by reducing the sparsity and the number of features by 10x respectively. I still find the double descent phenomenon as described in the paper with very similar constellations for features and hidden vectors. I also found double descent in multiple other settings, e.g. with different loss functions or when adding a ReLU activation between the layers. My preliminary takeaway from these findings is that the double descent is a fairly regular phenomenon that we should expect to happen in many settings. (Details can be found in my post More Findings on Memorization and Double Descent.)

EXTENSION: WHAT CONTROLS THE GENERALIZATION SCALE?

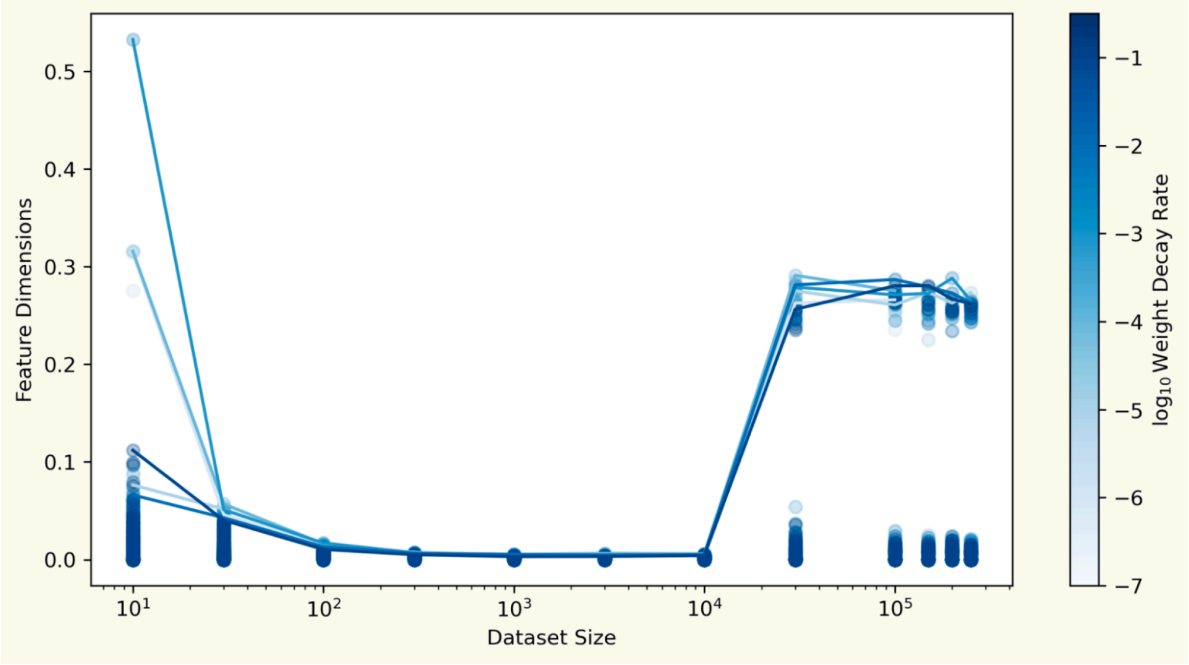
Adam Jermyn is an independent researcher focused on AI alignment and interpretability.

One question I had reading this paper is: what sets the scale at which models learn generalizing features? When I asked this, the authors proposed two potential hypotheses:

1. This is the scale where superposition of data points fails (e.g. due to weight decay constraining the weight norm)
2. This is the scale where features occur multiple times, in different combinations, allowing them to be distinguished.

The first hypothesis predicts that increasing the weight decay rate should decrease the generalizing scale.

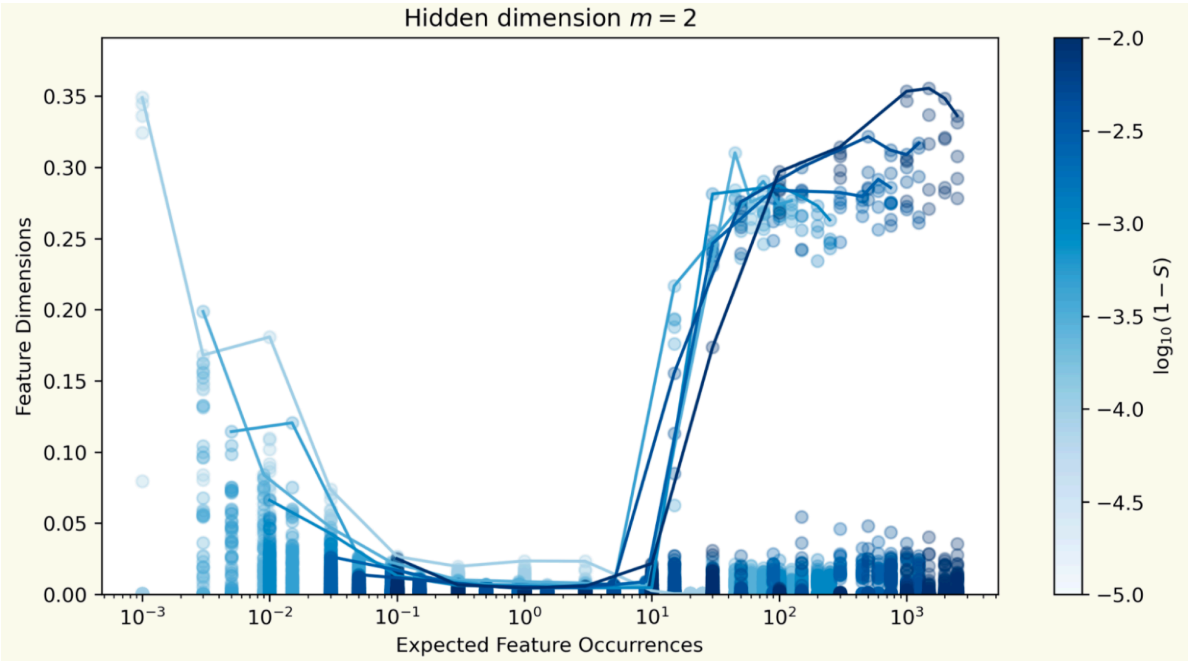
The figure below shows the dimensionalities of features for models trained with different weight decay rates. Lines show the maximum feature dimension and points and lines are colored by the weight decay rate.



The generalizing scale corresponds to a jump in the dimensionalities. Importantly this scale does not appear to change with the weight decay rate, which is evidence against the first hypothesis.

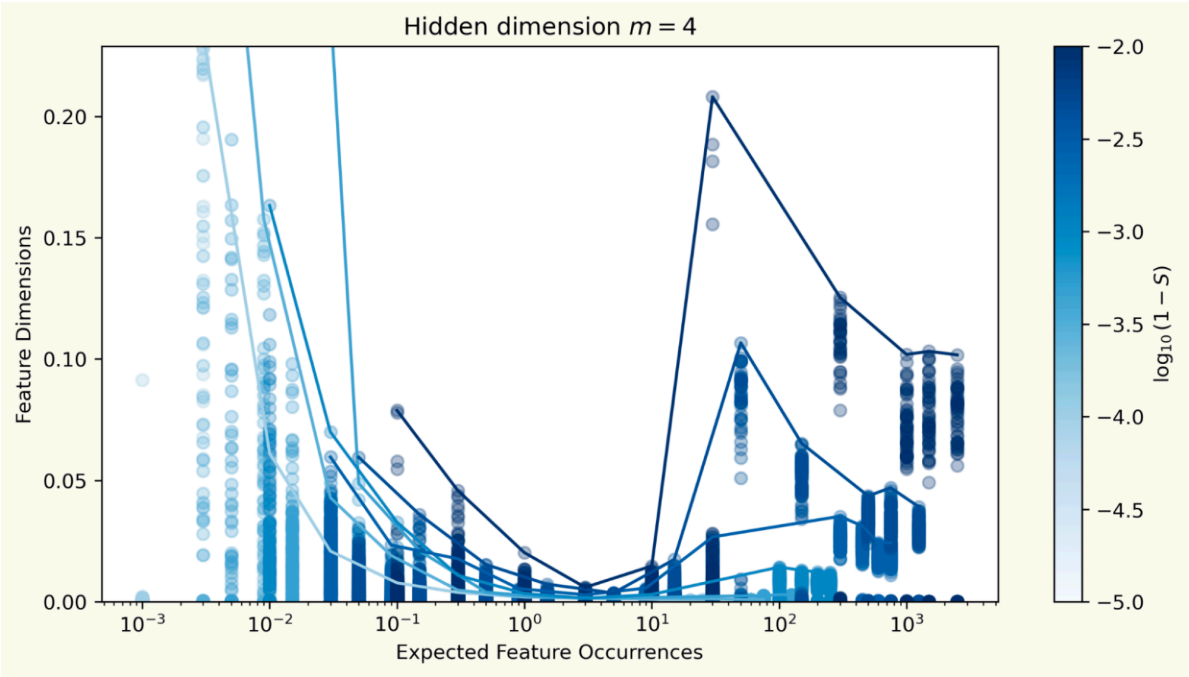
The second hypothesis predicts that the generalizing scale occurs once the dataset is large enough that it contains multiple instances of each feature. That is, it occurs at $T \propto 1/(1-S)$.

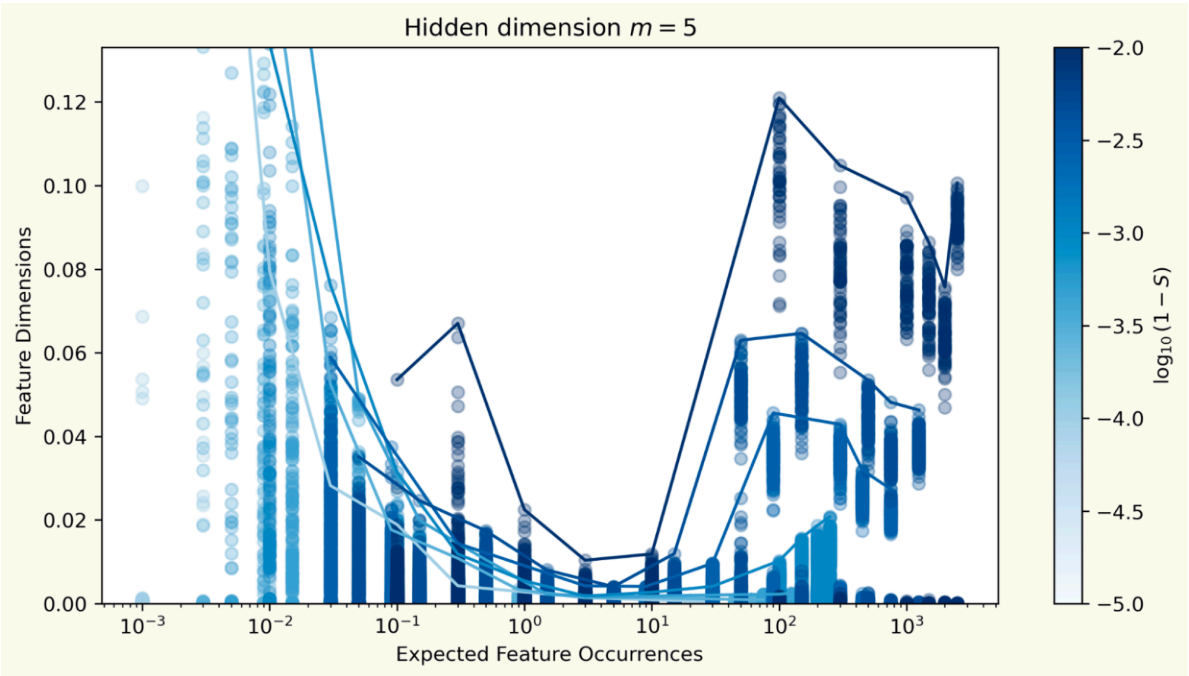
The figure below shows the dimensionalities of features for models trained with different weight decay rates. Lines show the maximum feature dimension and points and lines are colored by the feature frequency $(1-S)$. The horizontal scale now is the expected number of times any given feature appears in the dataset ($T(1-S)$), and a prediction of the second hypothesis is that the generalizing scale should be fixed on this scale.



Indeed that appears to be the case! Models trained with very different sparsities learn generalizing features once datasets are large enough to see each feature roughly 10 times.

While this is suggestive, it is not clear that this is the whole story. For instance, for models with more hidden dimensions the dimensionality curves don't lie as cleanly on top of each other (see below), and there are other trends that are puzzling (e.g. the peak feature dimensions *decrease* as the datasets grow post-generalization), so it seems possible that there is more going on.





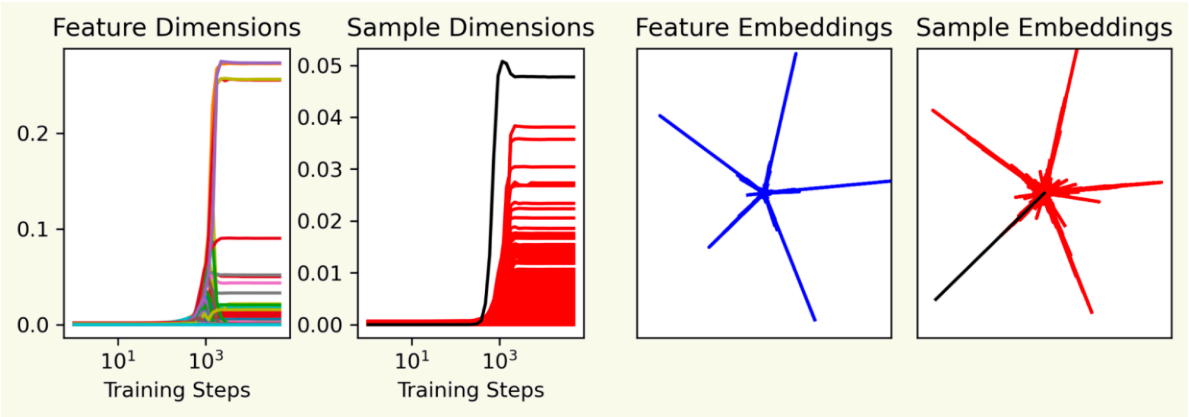
EXTENSION: REPEATED DATA POINTS

Adam Jermyn is an independent researcher focused on AI alignment and interpretability.

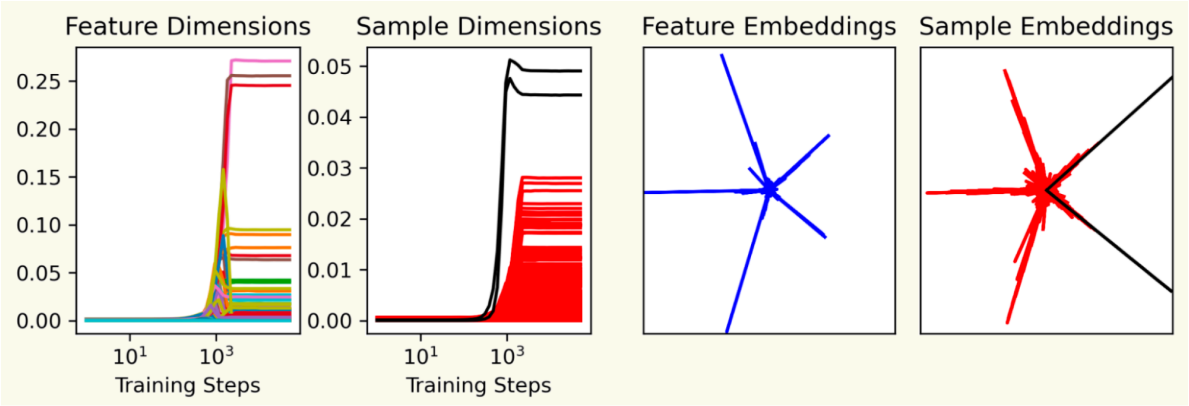
When the authors shared a preliminary draft, they suggested it might be interesting to look at what happens when individual datapoints are repeated in the dataset.

When a datapoint appears a small number of times (2-3) the phenomenology is the same as in this paper, but with more repeats models switch to learning a combination of datapoints and generalizing features.

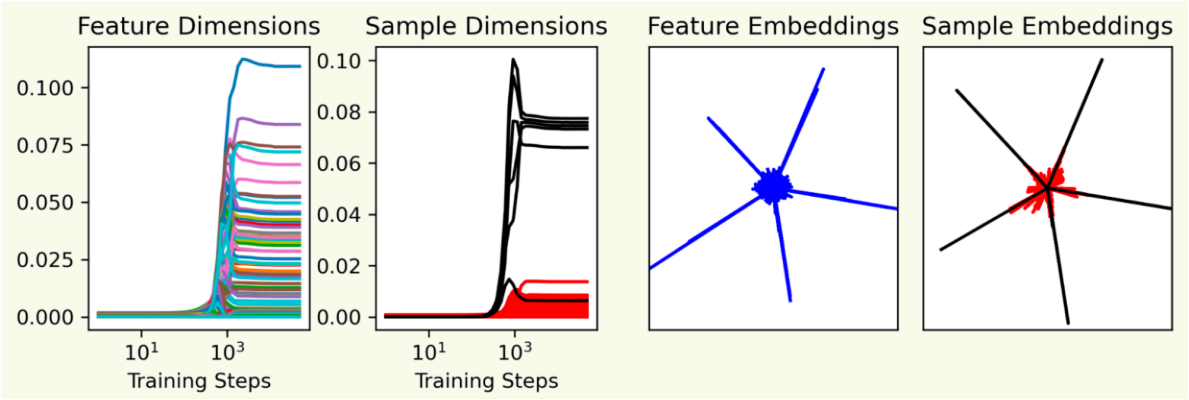
The figure below shows training histories of the feature and sample dimensions (left panels) as well as the final feature and sample embeddings (right panels) for a model with T=30,000 and a single feature (black) appearing 5 times. The repeated feature is embedded alongside four generalizing features and suppresses the fifth, effectively replacing one of the generalizing features that would ordinarily be learned.



When there are multiple repeated datapoints the model preferentially learns these, and each replaces a feature in the embedding space:



When there are more than five repeated datapoints, the model embeds five of them and all five features it would have learned are suppressed:

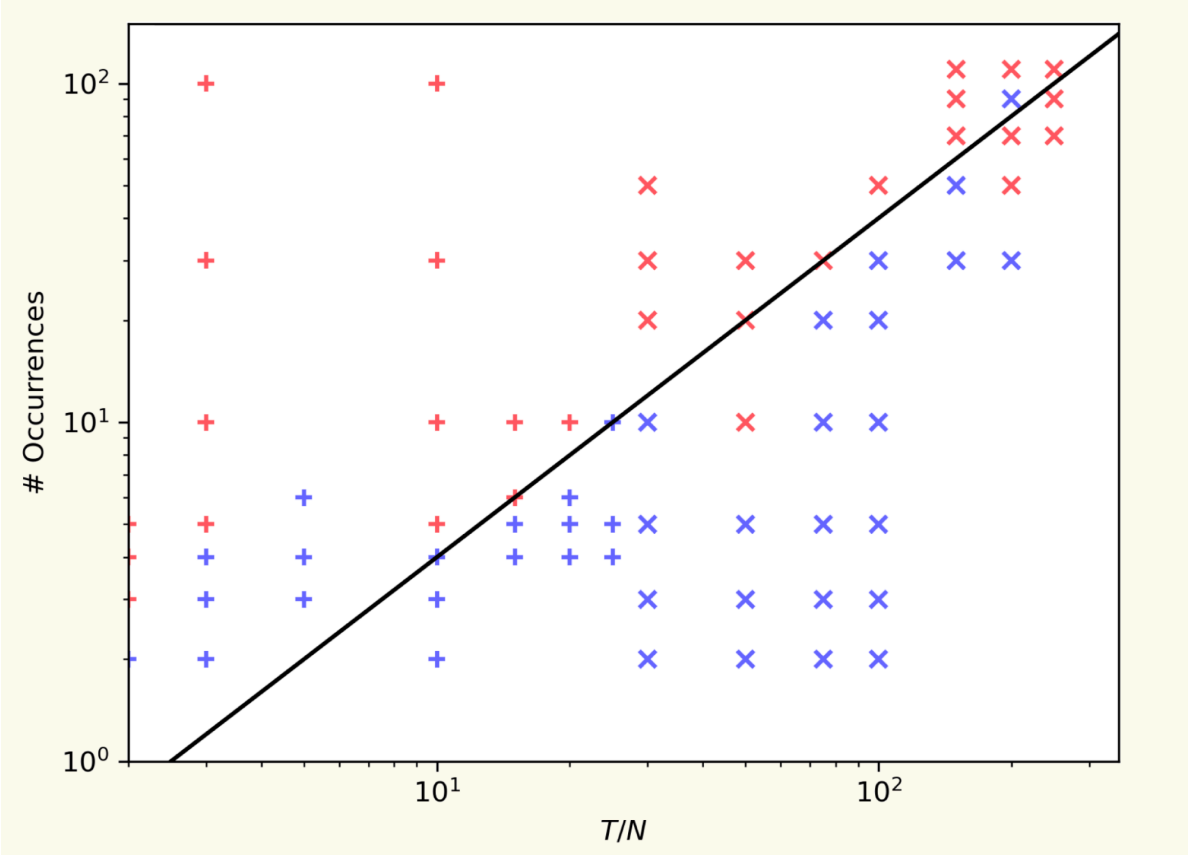


One question that arises here is: how many times does a datapoint need to appear before it is memorized? One way to think about this is in terms of the benefits of memorizing a datapoint versus learning a feature. Very roughly:

1. A given feature appears $T(1-S)$ times in the dataset. If the feature has amplitude A and can be learned perfectly then the loss benefit of learning it is $AT(1-S)$.

2. A given datapoint has $N(1-S)$ features active, each with amplitude of order A. If the datapoint appears R times, the loss benefit of memorizing it perfectly is $RNA(1-S)$.
3. Balancing these loss benefits suggests that datapoints are memorized when $R > T/N$, independent of sparsity. Note that constant factors were dropped in this argument, so there should also be a proportionality constant in this inequality.

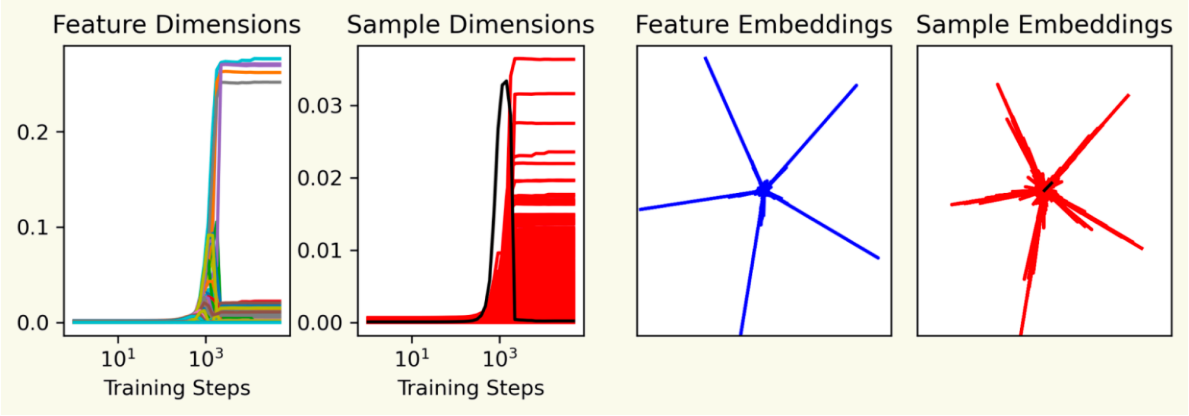
The figure below shows results from models trained with a single repeated feature and hidden dimension $m = 2$. The datasets were large enough for the models to learn generalizing features. The symbols correspond to two different setups: "+" corresponds to $S = 0.999$ and $N = 10,000$ and "x" corresponds to $S = 0.99$ and $N = 1,000$. Symbols are colored by whether the repeated datapoint appeared in the top 0.1% of datapoints by embedding dimension (red for yes, blue for no). This is a rough proxy for whether or not the model memorized the datapoint in question. Finally, the black line shows $R = 0.4T/N$.



The data are somewhat noisy, which could in part reflect the difficulty optimizing in $m = 2$, but in general:

1. At low T/N models require more than T/N repeats to memorize repeated datapoints.
2. At larger T/N the division between memorizing and not memorizing repeated datapoints coincides with the black line, corresponding to $R \propto T/N$ and consistent with the rough analytic argument above.

As one final observation, in models on the edge of memorizing datapoints we see a number of "near misses", where the model memorizes a datapoint and then "decides" against it!



Interestingly, this phenomenon is mirrored by a phenomenon in models with no repeated datapoints in the intermediate dataset regime, where some models briefly learn generalizing features and then forget them by the end of training. This is shown in the movie below for a model trained with T=10,000:

1.00

0:00 / 0:09

DATA DIMENSIONALITY OF MNIST

Chris Olah *is one of the authors of the original paper. This comment describes a small extension we may or may not expand on.*

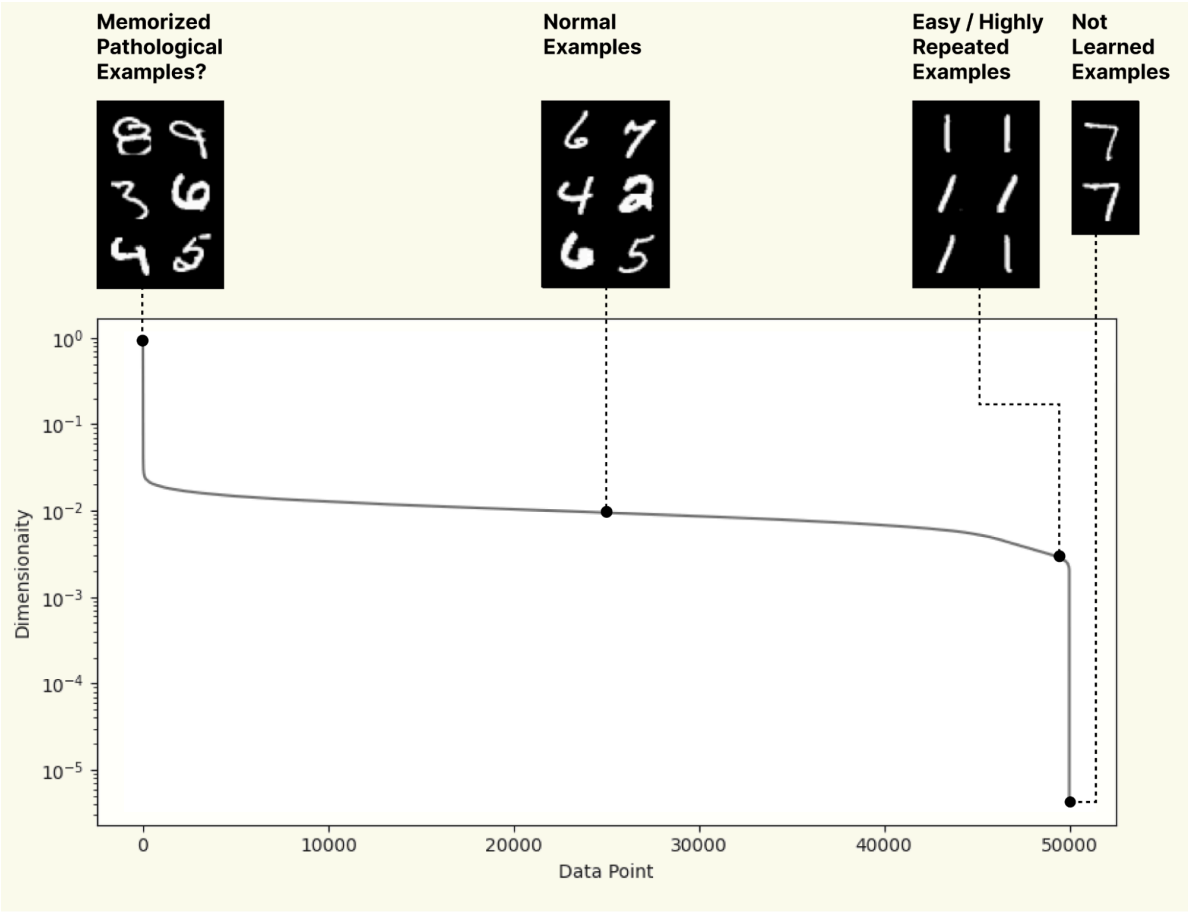
A natural question based on these results is whether we can detect overfit, memorized examples in real neural networks with data dimensionality. As a preliminary investigation, let's look at a model which is only slightly less of a toy – a one hidden layer MNIST model with 512 hidden units. Ordinarily, it's difficult to study superposition in real models because we don't know what the features are. But studying data points in superposition is an exception to this, since we do know what the data points are!

In practice, we expect that features in real models are not orthogonal and that even when a dataset example is memorized, it likely activates some "generalizing" features. To account for this, we'll slightly change our notion of data dimensionality to *maximal data dimensionality*:

$$D_i^* = \sup_v \frac{(v \cdot h_i)^2}{\sum_j (v \cdot h_j)^2}$$

The intuition is that if an example activates multiple features, the supremum can pick the one with the highest data dimensionality. (It turns out that this is closely related to the log-likelihood of data points if you fit a Gaussian to the network activations.)

Below, we plot the data dimensionality of all training examples. We see that most examples have roughly the same dimensionality, but there are a few outliers in both tails. Strikingly, the examples with unusually high data dimensionality – almost 100x higher than typical examples! – tend to be weird outliers. While far from conclusive, it's tempting to believe these examples are "memorized" examples which the model has "special cased".



In addition to detecting overfitting, one might also see this as an example of mechanistic anomaly detection – *detecting that a model is making decisions for a different reason than it normally does*. Of course, we don't mean to suggest that all cases of a model "triggering a special case" can be so easily detected. If anything, it may hint that mechanistic anomaly detection will be even harder than one might think, since it could be hidden by superposition.

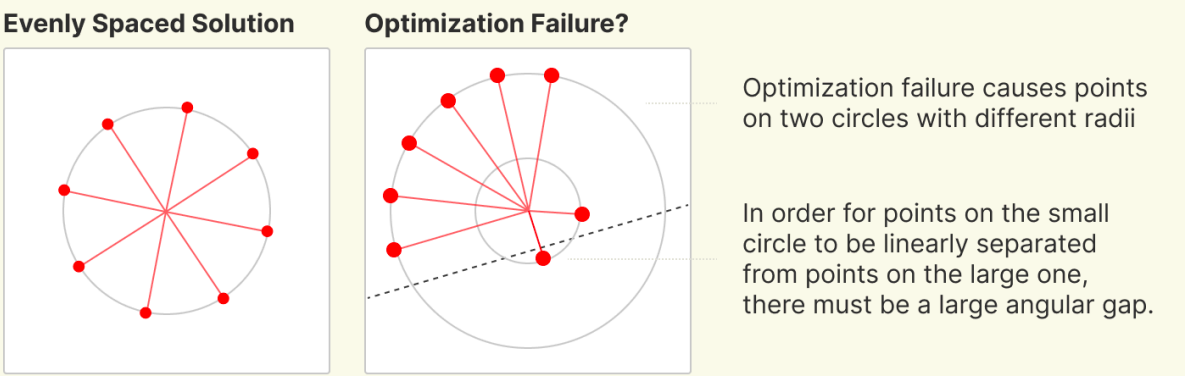
Marius Hobbhahn *is a PhD student at the University of Tuebingen.*

I can replicate these results on maximal data dimensionality up to small deviations. I tested the properties of $D*$ in many additional settings, e.g. at different points in training, dataset sizes and for different labels. My main takeaways are a) that $D*$ can be tricky to get right in practice but still looks interpretable even if it hasn't converged yet, b) $D*$ captures many different properties that we would expect neural networks to have. For example, when applying $D*$ to the double descent setting, we can see the transition from memorization to generalization from the metric and c) it opened a couple of new interesting research questions that might tell us more about how NNs learn during training which would make for interesting follow-up research. (Details can be found in my post More Findings on Maximal Data Dimension.)

OPTIMIZATION FAILURES IN 2D

Chris Olah *and* ***Tom Henighan*** *are authors of the original paper.*

When training $m = 2$ models in the low-data memorization regime, we occasionally observe cases where the points don't organize into a circle. We believe these models are organizing points on two circles of different radii, and fail to merge them. In order to ensure points on the smaller circle can be linearly selected, these solutions require a large angular gap at the interface between points on different circles.



Author Contributions

Experiments - The experiments in this paper were conducted by Tom Henighan, with help from Tristan Hume and Nelson Elhage. This was based on a prediction by Chris Olah that this behavior could be observed in roughly this experimental setup. Actually eliciting this behavior required careful tuning of hyperparameters, which was done by Tom Henighan. Robert Lasenby contributed significantly to our theoretical understanding of why features organize into pentagons. Nicholas Schiefer helped clarify our interpretation of the experimental results. Stanislav Fort did an independent reproduction of some of the experiments in this paper.

Diagrams - Diagrams were made by Tom Henighan and Shan Carter, with some help from Chris Olah.

Writing - This paper was drafted by Tom Henighan, with significant editing from Chris Olah and Shan Carter. Other authors also contributed to editing.

Acknowledgements

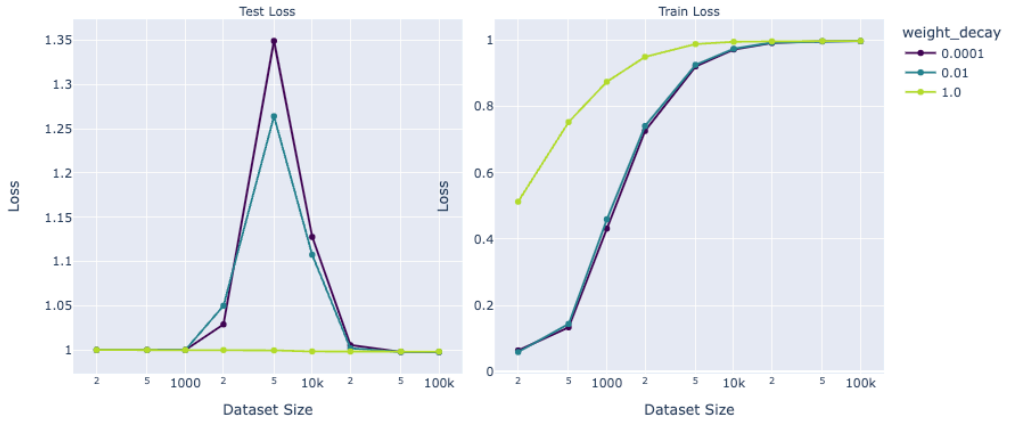
We're grateful to Sheer El Showk, James Sully, Adam Jermyn, Trenton Bricken, Neel Nanda, Eric Winsor, Ilya Sutskever, Gabriel Goh, William Saunders, Martin Wattenberg, Jacob Hilton, Sam Marks, Eric Neyman, Drake Thomas for useful comments and suggestions on early drafts.

We're also grateful to all our colleagues for creating a supportive environment for us to do our work: Jared Kaplan, Dario Amodei, Daniela Amodei, Jack Clark, Tom Brown, Sam McCandlish, Ben Mann, Nick Joseph, Catherine Olsson, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Timothy Telleen-Lawton, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Nova DasSarma, Jia Yuan Loke, Jackson Kernion, Tom Conerly, Scott Johnston, Jamie Kerr, Sheer El Showk, Shauna Kravec, Stanislav Fort, Rebecca Raible, Saurav Kadavath, Rune Kvist, Eli Tran-Johnson, Rob Gilson, Guro Khundadze, Ethan Perez, Sam Bowman, Sam Ringer, Jeeyoon Hyun, Michael Sellitto, Jared Mueller, Joshua Landau, Cameron McKinnon, Sandipan Kundu, Carol Chen, Roger Grosse, Robin Larson, Noemí Mercado, Anna Goldie, Azalia Mirhoseini, Jennifer Zhou, Erick Galankin, Dustin Li, James Landis, Neerav Kingsland, Tamera Lanham, Miranda Zhang, Bryan Seethor, Landon Goldberg, Brian Israel, Newton Cheng, Mike Lambert, Oliver Rausch, Matt Bell, Hongbin Chen, Kamile Lukosiute, Martin Lucas, Ivan Vendrov, Karina Nguyen, Peter Lofgren, Orowa Sikder, Logan Graham, Thomas Liao, and Da Yan.

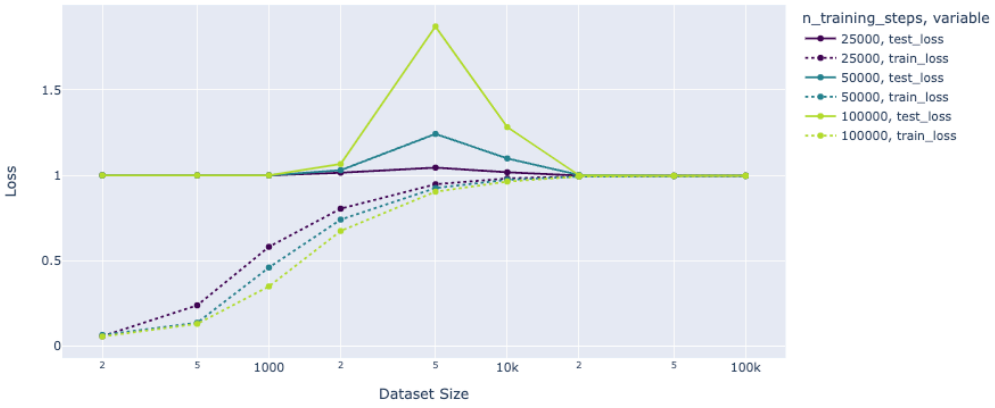
Effect of Weight Decay and Epochs on Double Descent

Here we use $m = 4$ models because they are much less prone to optimization failures than $m = 2$, giving us more confidence in the results.

Keeping other hyperparameters the same as the text, we find that decreasing weight decay increases the test loss bump, whereas increasing it all the way up to 1.0 removes it entirely. This is consistent with prior double-descent work.



Here we stick to a weight-decay of 1e-2, but vary the number of training epochs. We still use a linear warmup for the first 5% of training, followed by a cosine decay for the learning rate. Again we find results in-line with prior double-descent work: more epochs leads to a larger bump in test loss.



Footnotes

1. One could instead untie model weights W and $W_{up} \neq W^T$ to achieve this, though the resulting h_i will generally have different lengths, and so not quite form perfect T-gons. Untied weights like this are likely a more faithful representation of most real neural networks. [↩]

2. in the limit of very large n, one should expect training examples x_i to have equal norms thanks to the central limit theorem. [↩]

3. We find that models most clearly "treat individual data points as features" when the data points are orthogonal and of similar magnitudes (see this Colab notebook for more details). Highly sparse features increase the probability of orthogonal data points. Having lots of features avoids the trivial case where data points are just a single feature activating. Ideally, we'd have preferred to just use enough features that the central limit theorem causes data points to have similar norms, but achieving this with high sparsity would require annoyingly large vectors. [↩]

4. Intuitively this makes sense: if $T \ll n$, it must be easier to represent the T training examples, rather than the n features, in the hidden m-dimensional space. [↩]

References

1. Toy Models of Superposition
   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., 2022. Transformer Circuits Thread.

2. Linear algebraic structure of word senses, with applications to polysemy
   Arora, S., Li, Y., Liang, Y., Ma, T. and Risteski, A., 2018. Transactions of the Association for Computational Linguistics, Vol 6, pp. 483--495. MIT Press.

3. Decoding The Thought Vector [link]
   Goh, G., 2016.

4. Zoom In: An Introduction to Circuits
   Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M. and Carter, S., 2020. Distill. DOI: 10.23915/distill.00024.001

5. Understanding RL Vision
   Hilton, J., Cammarata, N., Carter, S., Goh, G. and Olah, C., 2020. Distill. DOI: 10.23915/distill.00029

6. Reconciling modern machine-learning practice and the classical bias--variance trade-off
   Belkin, M., Hsu, D., Ma, S. and Mandal, S., 2019. Proceedings of the National Academy of Sciences, Vol 116(32), pp. 15849--15854. National Acad Sciences.

7. High-dimensional dynamics of generalization error in neural networks
   Advani, M.S., Saxe, A.M. and Sompolinsky, H., 2020. Neural Networks, Vol 132, pp. 428--446. Elsevier.

8. Jamming transition as a paradigm to understand the loss landscape of deep neural networks
   Geiger, M., Spigler, S., d'Ascoli, S., Sagun, L., Baity-Jesi, M., Biroli, G. and Wyart, M., 2019. Physical Review E, Vol 100(1), pp. 012115. APS.

9. Deep double descent: Where bigger models and more data hurt
   Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B. and Sutskever, I., 2021. Journal of Statistical Mechanics: Theory and Experiment, Vol 2021(12), pp. 124003. IOP Publishing.

10. Understanding the difficulty of training deep feedforward neural networks
Glorot, X. and Bengio, Y., 2010. Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249--256.

11. Decoupled weight decay regularization
Loshchilov, I. and Hutter, F., 2017. arXiv preprint arXiv:1711.05101.

12. A Solvable Model of Neural Scaling Laws
Maloney, A., Roberts, D.A. and Sully, J., 2022. arXiv preprint arXiv:2210.16859.

13. Scaling Laws and Interpretability of Learning from Repeated Data
Hernandez, D., Brown, T., Conerly, T., DasSarma, N., Drain, D., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Henighan, T., Hume, T. and others,, 2022. arXiv preprint arXiv:2205.10487.