Scaling down Deep Learning

Sam Greydanus [1]

Abstract

Though deep learning models have taken on commercial and political relevance, many aspects of their training and operation remain poorly understood. This has sparked interest in "science of deep learning" projects, many of which are run at scale and require enormous amounts of time, money, and electricity. But how much of this research really needs to occur at scale? In this paper, we introduce MNIST-1D: a minimalist, low-memory, and low-compute alternative to classic deep learning benchmarks[1]. The training examples are 20 times smaller than MNIST examples yet they differentiate more clearly between linear, nonlinear, and convolutional models which attain 32, 68, and 94% accuracy respectively (these models obtain 94, 99+, and 99+% on MNIST). Then we present example use cases which include measuring the spatial inductive biases of lottery tickets, observing deep double descent, and metalearning an activation function.

1. Introduction

By any scientific standard, the Human Genome Project was enormous: it involved billions of dollars of funding, dozens of institutions, and over a decade of accelerated research (Lander et al., 2001). But that was only the tip of the iceberg. Long before the project began, scientists were hard at work assembling the intricate science of human genetics. And most of the time, they were not studying humans. The foundational discoveries in genetics centered on far simpler organisms such as peas, molds, fruit flies, and mice. To this day, biologists use these simpler organisms as genetic "minimal working examples" in order to save time, energy, and money. A well-designed experiment with Drosophilia, such as Feany and Bender (2000), can teach us an astonishing amount about humans.
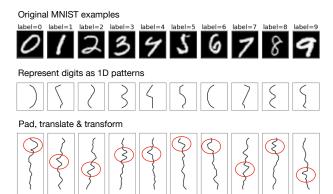
[1] Oregon State University; the ML Collective. Correspondence to: Sam Greydanus <greydanus.17@gmail.com>.

[1] Code at `github.com/greydanus/mnist1d`

Figure 1. Constructing the MNIST-1D dataset. Like MNIST, the classifier's objective is to determine which digit is present in the input. Unlike MNIST, each example is a one-dimensional sequence of points. To generate an example, we begin with a digit template and then randomly pad, translate, and transform it to produce sequences like the ones shown.

The deep learning analogue of Drosophilia is the MNIST dataset. A large number of deep learning innovations including dropout, Adam, convolutional networks, generative adversarial networks, and variational autoencoders began life as MNIST experiments (Srivastava et al., 2014; Kingma and Ba, 2014; LeCun et al., 1989; Goodfellow et al., 2014; Kingma and Welling, 2013). Once these innovations proved themselves on small-scale experiments, scientists found ways to scale them to larger and more impactful applications.

They key advantage of Drosophilia and MNIST is that they dramatically accelerate the iteration cycle of exploratory research. In the case of Drosophilia, the fly's life cycle is just a few days long and its nutritional needs are negligible. This makes it much easier to work with than mammals, especially humans. In the case of MNIST, training a strong classifier takes a few dozen lines of code, less than a minute of walltime, and negligible amounts of electricity. This is a stark contrast to state-of-the-art vision, text, and game-playing models which can take months and hundreds of thousands of dollars of electricity to train (Sharir et al., 2020).

Yet in spite of its historical significance, MNIST has three notable shortcomings. First, it does a poor job of differentiating between linear, nonlinear, and translation-invariant

models. For example, logistic, MLP, and CNN benchmarks obtain 94, 99+, and 99+% accuracy on it. This makes it hard to measure the contribution of a CNN's spatial priors or to judge the relative effectiveness of different regularization schemes. Second, it is somewhat large for a toy dataset. Each input example is a 784-dimensional vector and thus it takes a non-trivial amount of computation to perform hyperparameter searches or debug a metalearning loop. Third, MNIST is hard to hack. The ideal toy dataset should be procedurally generated so that researchers can smoothly vary parameters such as background noise, translation, and resolution.

In order to address these shortcomings, we propose the MNIST-1D dataset. It is a minimalist, low-memory, and low-compute alternative to MNIST, designed for exploratory deep learning research where rapid iteration is a priority. Training examples are 20 times smaller but they are still better at measuring the difference between 1) linear and nonlinear classifiers and 2) models with and without spatial inductive biases (eg. translation invariance). The dataset is procedurally generated but still permits analogies to real-world digit classification.
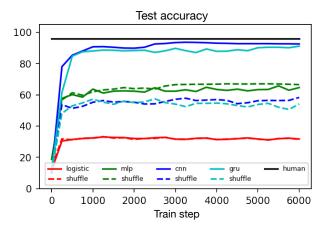

Figure 2. Visualizing the performance of common models on the MNIST-1D dataset. Whereas most ML models perform within a few percent accuracy on MNIST, this dataset separates them cleanly according to their characteristics. Logistic regression models fare worse than MLPs because they cannot use nonlinearities. MLPs, meanwhile, fare worse than CNNs because they cannot use translation invariance and local connectivity to bias optimization towards solutions that generalize well. These results suggest MNIST-1D is a good dataset for studying the inductive biases of ML models.

2. Context

The machine learning community has grown rapidly in recent years. This growth has accelerated the rate of scientific innovation, but it has also produced multiple competing narratives about the field's ultimate direction and objectives. In

this section, we will explore three such narratives in order to place MNIST-1D in its proper context.

Scaling trends. One of the defining features of machine learning in the 2010's was a massive increase in the scale of datasets, models, and compute infrastructure (Amodei and Hernandez, 2018). This scaling pattern allowed neural networks to achieve breakthrough results on a wide range of benchmarks (Krizhevsky et al., 2012; Szegedy et al., 2015; Radford et al., 2019). Yet while this scaling effect has helped neural networks take on commercial and political relevance, opinions differ about how much more intelligence it can generate. One one hand, many researchers and organizations argue that scaling is a crucial path to making neural networks behave more intelligently (Amodei and Hernandez, 2018). On the other hand, there is a healthy but marginal population of researchers who are not primarily motivated by scale. They are united by a common desire to change research methodologies, advocating a shift away from human-engineered datasets and architectures (Clune, 2019), an emphasis on human-like learning patterns (Chollet, 2019), and better integration with traditional symbolic AI approaches (Marcus, 2018).

Once again, the genetics analogy is useful. In genetics, scale has been most effective when small-scale experiments have helped to guide the direction and vision of large-scale experiments. For example, the organizers of the Human Genome Project regularly used yeast and fly genomes to guide analysis of the human genome (Lander et al., 2001). Thus one should be suspicious of research agendas that place disproportionate emphasis on large-scale experiments, since a healthy research ecosystem needs both. The fast, small scale projects permit creativity and deep understanding, whereas the large-scale projects expose fertile new research territory.

Understanding vs. performance. Researchers are also divided over the value of understanding versus performance. Some contend that a high-performing algorithm need not be interpretable so long as it saves lives or produces economic value. Others argue that hard-to-interpret deep learning models should not be deployed in sensitive real-world contexts. Both arguments have merit, but the best path forward seems to be to focus on understanding high-performing algorithms better so that this tradeoff becomes less severe. One way to do this is by identifying things we don't understand about neural networks, reproducing these things on a toy problem like MNIST-1D, and then performing ablation studies to isolate the causal mechanisms.

Ecological impacts. A growing number of researchers and organizations claim that deep learning will have positive environmental applications (Loehle, 1987; Rolnick et al., 2019). This may be true in the long run, but so far artificial intelligence has done little to solve environmental problems. In the meantime, deep learning models are consuming mas-

Table 1. Test accuracies of common classifiers on the MNIST and MNIST-1D datasets. Most classifiers achieve similar test accuracy on MNIST. The MNIST-1D dataset is much smaller than MNIST and does a better job of separating models with different inductive biases. The drop in CNN and GRU performance when using shuffled data indicates that spatial priors are important on this dataset.

Dataset	Logistic regression	Fully connected model	Convolutional model	GRU model	Human expert
MNIST	94 ± 0.5	> 99	> 99	> 99	> 99
MNIST-1D	32 ± 1	68 ± 2	94 ± 2	91 ± 2	96 ± 1
MNIST-1D (shuffled)	32 ± 1	68 ± 2	56 ± 2	57 ± 2	$\approx 30 \pm 10$

sive amounts of electricity to train and deploy (Strubell et al., 2019). Our hope is that benchmarks like MNIST-1D will encourage researchers to spend more time iterating on small datasets and toy models before scaling, making more efficient use of electricity in the process.

3. Methods

Given modern machine learning's emphasis on scale and performance, we see a hidden demand for alternative projects – projects that prioritize creativity over scale and understanding over performance. We designed MNIST-1D for that sort of research. In particular, we wanted a dataset that was

- Extremely small: smaller than MNIST
- Able to identify models with (spatial) inductive biases
- Easy to hack, extend, or modify
- Analogous to large-scale problems

Dimensionality. Our first choice was to make the data one-dimensional (eg time series) rather than two-dimensional (eg images). Our rationale was that there were already many good image datasets so the value in adding another was small. Meanwhile, one-dimensional signal processing requires less computation but has many of the same scientific properties.

Constructing the dataset. We began with ten one-dimensional template patterns which resemble the digits 0-9 when plotted as in Figure 1. Each of these platonic forms consists of 12 hand-selected x coordinates. Next, we padded each sequence with 36-60 additional points, translated the digit at random, scaled it, added Gaussian noise, and added a constant linear signal analogous to shear in a 2D image. We used a Gaussian filter with $\sigma = 2$ to induce correlations that would be easy to confuse with the templates. Last of all, we downsampled the pattern to 40 data points. Figure 1 shows class-wise examples before and after these transformations.

Implementation. Our goals during implementation were to make the code as simple, modular, and hackable as possible. The code for generating the dataset occupies two Python files and a total of 150 lines. The `get_dataset` method

has a simple API for changing dataset features such as maximum digit translation, correlated noise scale, shear scale, final sequence length, and more. The default train/test split is 4000/1000.

Benchmarking the dataset. We used PyTorch to implement and train simple logistic, MLP, CNN, and GRU baselines. All models used the Adam optimizer and early stopping for model selection. We also trained the same models on a version of the dataset which was permuted along the spatial dimension. We refer to this as the "shuffled" version since it measures each model's performance in the absence of local spatial structure. A number of related works use the same shuffling process to remove spatial priors (Zhang et al., 2016; Li et al., 2018). Table 1 shows that the test accuracy of CNNs and GRUs decreases by about 38% on the shuffled data whereas the MLP and linear models perform about the same. This is a good sanity check since the former two models have spatial and temporal locality priors whereas the latter two do not. As with the dataset itself, the code for implementing and training the benchmark models occupies roughly 150 lines and is clean and modular. You can reproduce the benchmarks in your browser in a few minutes[2].

4. Example use cases

In this section we will explore several examples of how MNIST-1D can be used to study core "science of deep learning" phenomena.

Finding lottery tickets. It is not unusual for deep learning models to have ten or even a hundred times more parameters than necessary. This overparameterization helps training but increases computational overhead. One solution is to progressively prune weights from a model during training so that the final network is just a fraction of its original size. Although this approach works, conventional wisdom holds that sparse networks do not train well from scratch. Recent work by Frankle and Carbin (2019) challenges this conventional wisdom. The authors report finding sparse subnetworks inside of larger networks that train to equivalent or even higher accuracies. These "lottery ticket" subnet-
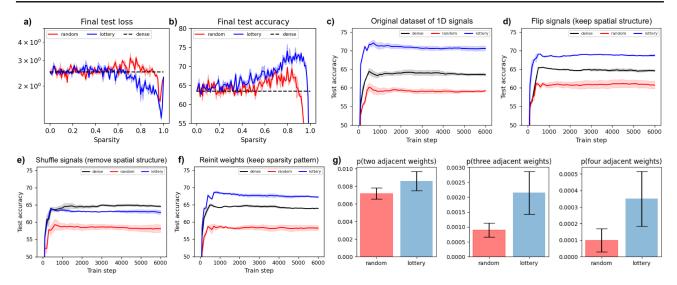
---
[2]`bit.ly/3fghqVu`

Figure 3. Finding and analyzing lottery tickets. In **a-b)**, we isolate a minimum viable example of the effect. Recent work by Morcos et al. (2019) shows that lottery tickets can transfer between datasets. We wanted to determine whether spatial inductive biases played a role. So we performed a series of experiments: in **c)** we plot the asymptotic performance of a 92% sparse ticket. In **d)** we reverse all the 1D signals in the dataset, effectively preserving spatial structure but changing the location of individual datapoints. This is analogous to flipping an image upside down. Under this ablation, the lottery ticket continues to win. Next, in **e)** we permute the indices of the 1D signal, effectively removing spatial structure from the dataset. This ablation hurts lottery ticket performance significantly more, suggesting that part of the lottery ticket's performance can be attributed to a spatial inductive bias. Finally, in **f)** we keep the lottery ticket sparsity structure but initialize its weights with a different random seed. Contrary to results reported in (Frankle and Carbin, 2019), we see that our lottery ticket continues to outperform a dense baseline, aligning well with our hypothesis that the sparsity pattern represents a spatial inductive bias. In **g)**, we verify our hypothesis by measuring how often unmasked weights are adjacent to one another in the first layer of our model. The lottery ticket has many more adjacent weights than chance would predict, implying a local connectivity structure which helps gives rise to spatial biases. *Figure 9 in the Appendix visualizes the actual masks, revealing how this local connectivity is manifested.*

works can be found through a simple iterative procedure: train a network, prune the smallest weights, and then rewind the remaining weights to their original initializations and retrain.

Since the original paper was published, a multitude of works have sought to explain this phenomenon and then harness it on larger datasets and models. However, very few works have attempted to isolate a "minimal working example" of this effect so as to investigate it more carefully. Figure 3 shows that the MNIST-1D dataset not only makes this possible, but also enables us to elucidate, via carefully-controlled experiments, some of the reasons for a lottery ticket's success. Unlike many follow-up experiments on the lottery ticket, this one took just two days of researcher time to produce. The curious reader can also reproduce these results in their browser in a few minutes[3].

Observing deep double descent. Another intriguing property of neural networks is the "double descent" phenomenon. This phrase refers to a training regime where more data, model parameters, or gradient steps can actually *reduce* a model's test accuracy (Trunk, 1979; Belkin et al., 2019;

Geiger et al., 2019; Nakkiran et al., 2020). The intuition is that during supervised learning there is an interpolation threshold where the learning procedure, consisting of a model and an optimization algorithm, is just barely able to fit the entire training set. At this threshold there is effectively just one model that can fit the data and this model is very sensitive to label noise and model mis-specification.

Several properties of this effect, such as what factors affect its width and location, are not well understood in the context of deep models. We see the MNIST-1D dataset as a good tool for exploring these properties. In fact, we were able to reproduce the double descent pattern after a few hours of researcher effort. Figure 4 shows our results for a fully-connected network and a convolutional model[4]. We also observed a nuance that we had not seen mentioned in previous works: when using a mean square error loss, the interpolation threshold lies at $n * K$ model parameters where n is the number of training examples and K is the number of model outputs. But when using a negative log likelihood loss, the interpolation threshold lies at n model parameters – it does not depend on the number of outputs.

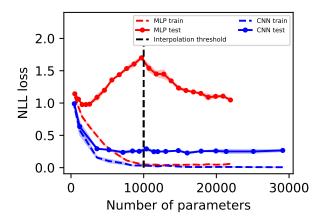[3]bit.ly/3nCEIaL

[4]Run in browser: bit.ly/2UBWWNu

Figure 4. Observing deep double descent. MNIST-1D is a good environment for determining how to locate the interpolation threshold of deep models. This threshold is fairly easy to predict in fully-connected models, less easy to predict for other models like CNNs, RNNs, and Transformers. Here we see that a CNN has a double descent peak at the same interpolation threshold, although the effect is much less pronounced.

This is an interesting empirical observation that may explain some of the advantage in using a log likelihood loss over a MSE loss on this type of task.

Gradient-based metalearning. The goal of metalearning is to "learn how to learn." A model does this by having two levels of optimization: the first is a fast inner loop which corresponds to a traditional learning objective and the second is a slow outer loop which updates the "meta" properties of the learning process. One of the simplest examples of metalearning is gradient-based hyperparameter optimization. The concept was was proposed by Bengio (2000) and then scaled to deep learning models by Maclaurin et al. (2015). The basic idea is to implement a fully-differentiable neural network training loop and then backpropagate through the entire process in order to optimize hyperparameters like learning rate and weight decay.
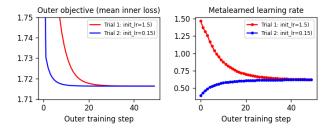

Figure 5. Metalearning a learning rate. Unlike many gradient-based metalearning implementations, ours takes seconds to run and occupies a few dozen lines of code. This allows researchers to iterate on novel ideas before scaling. Best viewed with zoom.

Metalearning is a promising topic but it is very difficult to scale. First of all, metalearning algorithms consume enormous amounts of time and compute. Second of all, implementations tend to grow complex since there are twice as many hyperparameters (one set for each level of optimization) and most deep learning frameworks are not set up well for metalearning. This places an especially high incentive on debugging and iterating metalearning algorithms on small-scale datasets such as MNIST-1D. For example, it took just a few hours to implement and debug the gradient-based hyperparameter optimization shown in Figure 5[5].

Metalearning an activation function. Having implemented a "minimal working example" of gradient-based metalearning, we realized that it permitted a simple and novel extension: metalearning an activation function. With a few more hours of researcher time, we were able to parameterize our classifier's activation function with a second neural network and then learn the weights using meta-gradients. As Figure 6 shows, our learned activation function substantially outperforms baseline nonlinearities such as ReLU, Elu, and Swish.[6] We note that previous works (Clevert et al., 2016; Ramachandran et al., 2018; Vercellino and Wang) have tried to optimize activation functions, but none have done so with analytical gradients computed via bilevel optimization.
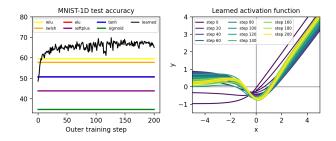

Figure 6. Metalearning an activation function. Starting from an ELU shape, we use gradient-based metalearning to find the optimal activation function for a neural network trained on the MNIST-1D dataset. The activation function itself is parameterized by a second (meta) neural network. Note that the ELU baseline (red) is obscured by the `tanh` baseline (blue) in the figure above.

We transferred this activation function to convolutional models trained on MNIST and CIFAR10 images and found that it achieves middle-of-the-pack performance. It is especially good at producing low training loss early in optimization, which is the objective that it was trained on in MNIST-1D. When we rank nonlinearities by final test loss, though, it achieves middle-of-the-pack performance. We suspect that running the same metalearning algorithm on larger models and datasets would further refine our activation function, allowing it to at least match the best hand-designed activation function. We leave line of inquiry to future work.

---

[5]Run in browser: `bit.ly/38OSyTu`
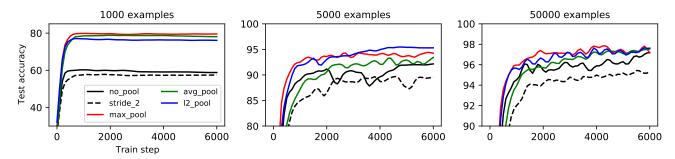[6]Run in browser: `bit.ly/38V4GlQ`

Figure 7. Benchmarking common pooling methods. We observe that pooling helped performance in low-data regimes and hindered it in high-data regimes. While we do not entirely understand this effect, we hypothesize that pooling is a mediocre architectural prior that is better than nothing in low-data regimes but becomes overly restrictive in high-data regimes.

Measuring the spatial priors of deep networks. A large part of deep learning's success is rooted in "deep priors" which include hard-coded translation invariances (e.g., convolutional filters), clever architectural choices (e.g., self-attention layers), and well-conditioned optimization landscapes (e.g., batch normalization). Principle among these priors is the translation invariance of convolution. A primary motivation for this dataset was to construct a toy problem that could effectively quantify a model's spatial priors. Figure 2 illustrates that this is indeed possible with MNIST-1D. One could imagine that other models with more moderate spatial priors would sit somewhere along the continuum between the MLP and CNN benchmarks.

Benchmarking pooling methods. Our final case study begins with a specific question: *What is the relationship between pooling and sample efficiency?* We had not seen evidence that pooling makes models more or less sample efficient, but this seemed an important relationship to understand. With this in mind, we trained models with different pooling methods and training set sizes and found that, while pooling tended to be effective in low-data regimes, it did not make much of a difference in high-data regimes (see Figure 7). We do not fully understand this effect, but hypothesize that pooling is a mediocre architectural prior which is better than nothing in low-data regimes and then ends up restricting model expression in high-data regimes. By the same token, max-pooling may also be a good architectural prior in the low-data regime, but start to delete information – and thus perform worse compared to L2 pooling – in the high-data regime. As with the other examples, you can reproduce these results in your browser in a few minutes[7].

5. When to scale

We should emphasize that this paper is not an argument against large-scale machine learning research. That sort of research has proven its worth time and again and has

come to represent one of the most exciting aspects of the ML research ecosystem. Rather, we are arguing *in favor* of small-scale machine learning research. Neural networks do not have problems with scaling or performance – but they do have problems with interpretability, reproducibility, and iteration speed. We see carefully-controlled, small-scale experiments as a great way to address these problems.

In fact, small-scale research is complimentary to large-scale research. As in biology, where fruit fly genetics helped guide the Human Genome Project, we believe that small-scale research should always have an eye on how to successfully scale. For example, several of the findings reported in this paper are at the point where they should be investigated at scale. We would like to show that large scale lottery tickets also learn spatial inductive biases, and show evidence that they develop local connectivity. We would also like to try metalearning an activation function on a larger model in the hopes of finding an activation that will outperform ReLU and Swish in generality.

6. Related work

The core inspiration for this work stems from an admiration for all that the MNIST dataset (LeCun et al., 1998) has done for deep learning. While it has some notable flaws – some of which we have addressed – it also has underappreciated strengths: it is simple, intuitive, and provides the perfect sandbox for exploring creative new ideas.

Our work also bears philosophical similarities to the *Synthetic Petri Dish* by Rawal et al. (2020). It was published concurrently to this work and the authors make similar references to biology in order to motivate the use of small synthetic datasets for exploratory research. Their work differs from ours in that they use metalearning to obtain their datasets whereas we construct ours by hand. In doing so, we are able to control various causal factors, such as amount of noise, translation, and padding independently. Also, our dataset is more intuitive to humans: a human can outper-

---

[7]`bit.ly/3lGmTqY`

form a strong CNN on the MNIST-1D task. These traits make MNIST-1D a better dataset for investigating "science of deep learning" questions on a small scale. The Synthetic Petri Dish, meanwhile, is not designed to answer the same questions; its objective is specifically to accelerate neural architecture search.

There are a number of other small-scale datasets that are commonly used to investigate "science of deep learning" questions. The CIFAR-10 dataset by Krizhevsky et al. (2009) is larger than the MNIST dataset in that individual examples are four times larger, but the number of images is the same. It generally does a better job of discriminating between MLP and CNN architectures, and between various CNN architectures such as vanilla CNNs versus ResNets (He et al., 2015). The FashionMNIST dataset by Xiao et al. (2017) is the same size as MNIST but somewhat more difficult; it aims to rectify some of the most serious problems with MNIST, in particular, that it is too easy and does not discriminate properly between different machine learning models.

Scikit-learn by Pedregosa et al. (2011) provides dozens of toy datasets – some synthetic and others real – that are meant for evaluating machine learning models on a small scale. These datasets are appropriate for some "science of machine learning" questions but not for others. For example, the `two_moons` dataset is useful for exploring the role of nonlinearity in classification. However, none of these synthetic tasks is good for exploring the role of spatial inductive biases in deep learning architectures. Making real world analogies to, say, digit classification, is not possible with these datasets, and one can often do very well on them using simple linear or kernel-based methods.

7. Discussion

There is a counterintuitive possibility that in order to explore the limits of how large we can scale neural networks, we may need to explore the limits of how small we can scale them first. Scaling models and datasets downward in a way that preserves the nuances of their behaviors at scale will allow researchers to iterate quickly on fundamental and creative ideas. This fast iteration cycle is the best way of obtaining insights about how to incorporate progressively more complex inductive biases into our models. We can then transfer these inductive biases across spatial scales in order to dramatically improve the sample efficiency and generalization properties of large-scale models. We see the humble MNIST-1D dataset as a first step in that direction.

Acknowledgements

References

D. Amodei and D. Hernandez. Ai and compute, May 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1903070116. URL https://www.pnas.org/content/116/32/15849.

Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

F. Chollet. The measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations*, 2016.

J. Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019.

M. B. Feany and W. W. Bender. A drosophila model of parkinson's disease. *Nature*, 404(6776):394–398, 2000.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, volume abs/1803.03635, 2019. URL http://arxiv.org/abs/1803.03635.

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

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

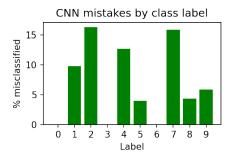A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, et al. Initial sequencing and analysis of the human genome. *Nature*, 2001.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C. Li, H. Farkhoor, R. Liu, and J. Yosinski. Measuring the Intrinsic Dimension of Objective Landscapes. In *International Conference on Learning Representations*, Apr. 2018.

C. Loehle. Applying artificial intelligence techniques to ecological modeling. *Ecological modelling*, 38(3-4):191–212, 1987.

D. Maclaurin, D. Duvenaud, and R. Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

G. Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

A. Morcos, H. Yu, M. Paganini, and Y. Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems*, pages 4933–4943, 2019.

P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *International Conference on Learning Representations*, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *International Conference on Learning Representations (workshop tract)*, 2018.

A. Rawal, J. Lehman, F. P. Such, J. Clune, and K. O. Stanley. Synthetic petri dish: A novel surrogate model for rapid architecture search. *arXiv preprint arXiv:2005.13092*, 2020.

D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al. Tackling climate change with machine learning. *Neural Information Processing Systems Workshop on Climate Change AI*, 2019.

O. Sharir, B. Peleg, and Y. Shoham. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. *57th Annual Meeting of the Association for Computational Linguistics (ACL*, 2019.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *Computer Vision and Pattern Recognition*, Sept. 2015.

G. V. Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on pattern analysis and machine intelligence*, (3):306–307, 1979.

C. J. Vercellino and W. Y. Wang. Hyperactivations for activation function exploration.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.

A. Supplementary figures

More on the human vs. CNN benchmarks. An experienced human can classify MNIST-1D examples at almost 96% accuracy. The CNN can do so at 94% accuracy. Both the human and the CNN struggle primarily with classifying 2's and 7's, and to a lesser degree 4's (see Figure 8). The human had a harder time classifying 9's whereas the CNN had a harder time classifying 1's. Both had zero errors classifying 3's and 6's.

Classification errors were fairly evenly balanced across classes, which is a good sign. If only one or two classes were responsible for most of the mistakes, that would have
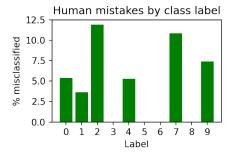
Figure 8. Classwise errors on the MNIST-1D dataset. Some classes contribute more than others, but most represent an appreciable fraction. This is good, because if only one or two classes were responsible for most of the mistakes, that would indicate that those classes are too difficult compared to the others. It's also interesting to note that humans and CNNs struggle with the same classes, such as 7's and 2's.

been a sign that those classes were too difficult compared to the others.

It's interesting that a human can outperform a CNN on this simple task. Part of the issue is that the CNN is only given 4000 training examples – with more examples it can match and eventually exceed the human baseline. Even though the data is low-dimensional, the classification objective is quite difficult and spatial/relational priors matter a lot. It may be that the architecture of the CNN prevents it from learning all of the tricks that humans are capable of using (eg, using relational reasoning about two signals to determine how they work together to form the digit signal).

It's worth noting that CNNs outperform human experts on most large-scale image classification tasks like ImageNet and CIFAR-100. But here is a tiny benchmark where humans are still competitive – this is a nice quality, as it suggests that the key to performing well on the dataset does not rest on shortcut learning such as memorizing specific numbers or patterns to machine precision. A high-performing ML model, we can hope, would have to solve the problem using strategies that would be intuitive to a human.

Further analysis of lottery tickets. In addition to the analysis of lottery tickets shown in Figure 3, we include some

further visualizations in this appendix. In particular, Figure 9 shows the actual masks of the first layer weights of random and lottery ticket masks. This qualitative comparison helps highlight the spatial inductive bias of the lottery ticket we obtained.

Dimensionality reduction. We used tSNE to reduce the dimensionality of MNIST and MNIST-1D so as to plot the datasets in two dimensions. We show the results in Figure 10 with each example colored according to its class label. We observe ten well-defined clusters in the MNIST dataset, suggesting that most examples in most classes are linearly separable from one another. By contrast, we observe few well-defined clusters in the MNIST-1D dataset, suggesting that a classifier must learn a nonlinear representation of the data in order to separate the classes properly. Thanks to Dmitry Kobak for helping with this visualization.

B. Hyperparameters

Table 2. Default hyperparameters of the MNIST-1D dataset.

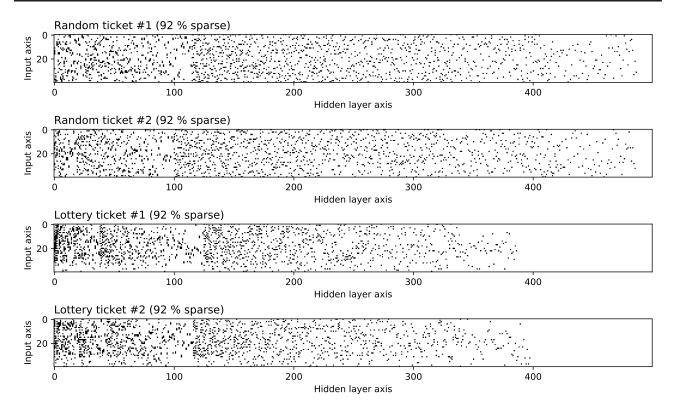| HYPERPARAMETER	VALUE
TRAIN/TEST SPLIT	4K/1K
TEMPLATE LENGTH	12
PADDING POINTS	36-60
MAX TRANSLATION	48
CORRELATED NOISE SCALE	0.25
IID NOISE SCALE	2×10^{-2}
SHEAR SCALE	0.75
SHUFFLE SEQUENCE	FALSE
FINAL SEQ. LENGTH	40
SEED	42

Figure 9. Visualizing first layer weight masks of random tickets and lottery tickets. For interpretabilty, we have sorted the mask along the hidden layer axis according to the number of adjacent unmasked parameters. This helps reveal a bias towards local connectivity in the lottery ticket masks. Notice how there are many more vertically-adjacent unmasked parameters in the lottery ticket masks. These vertically-adjacent parameters correspond to local connectivity along the input dimension, which in turn biases the sparse model towards data with spatial structure.
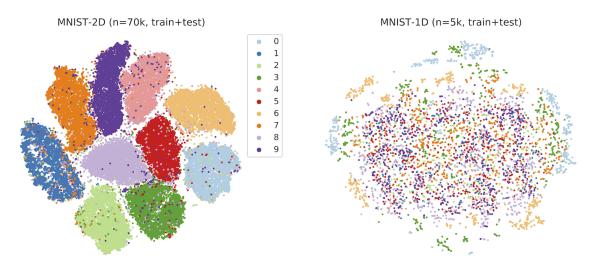

Figure 10. Visualizing the MNIST and MNIST-1D datasets with tSNE. The well-defined clusters in the MNIST plot indicate that the majority of the examples are linearly separable according to class. The MNIST-1D plot, meanwhile, reveals a lack of well-defined clusters which suggests that nonlinear features are much more important for successful classification.