Scaling Down Deep Learning with MNIST-1D

Sam Greydanus [1 2] **Dmitry Kobak** [3 4]

Abstract

Although deep learning models have taken on commercial and political relevance, key aspects of their training and operation remain poorly understood. This has sparked interest in *science of deep learning* projects, many of which require large 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, procedurally generated, low-memory, and low-compute alternative to classic deep learning benchmarks. Although the dimensionality of MNIST-1D is only 40 and its default training set size only 4000, MNIST-1D can be used to study inductive biases of different deep architectures, find lottery tickets, observe deep double descent, metalearn an activation function, and demonstrate guillotine regularization in self-supervised learning. All these experiments can be conducted on a GPU or often even on a CPU *within minutes*, allowing for fast prototyping, educational use cases, and cutting-edge research on a low budget.

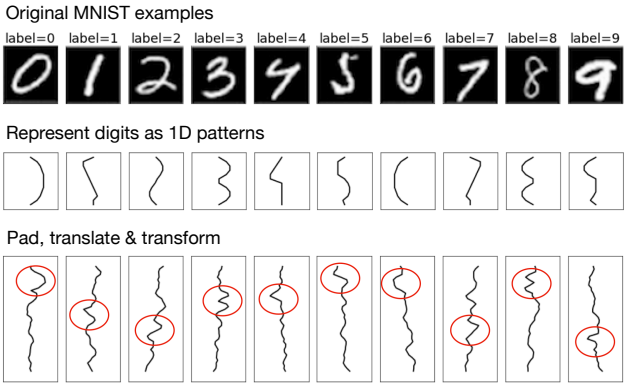

Figure 1: Constructing the MNIST-1D dataset. Unlike MNIST, each sample is a one-dimensional sequence. To generate each sample, we begin with a hand-crafted digit template loosely inspired by MNIST shapes. Then we randomly pad, translate, and add noise to produce 1D sequences with 40 points each. **[CODE]**

1. Introduction

The deep learning analogue of *Drosophila melanogaster* is the MNIST dataset. Drosophila, the fruit fly, has a life cycle that is just a few days long, its nutritional needs are negligible, and it is easier to work with than mammals, especially humans. Like Drosophila, MNIST is easy to use: training a classifier on it takes only a few a minutes whereas training full-size vision and language models can take months of time and millions of dollars (Sharir et al., 2020).

But in spite of their small size, both test systems have had a major impact on their respective fields. A number of seminal discoveries in medicine, including multiple Nobel prizes, have been awarded for work performed with fruit flies. Early work in genetics — work that paved the way for the Human Genome Project, which involved billions of dollars of funding, dozens of institutions, and over a decade of accelerated research (Lander et al., 2001) — was performed on fruit flies and other simple organisms. To this day, experiments on Drosophila are a cornerstone of biomedical research.

Meanwhile, MNIST has served as the initial proving ground for a large number of deep learning innovations including dropout, Adam, convolutional networks, generative adversarial networks, and variational autoencoders (Srivastava et al., 2014; Kingma and Ba, 2014; LeCun et al., 1989; Goodfellow et al., 2014; Kingma and Welling, 2014). Once a proof of concept was established in small-scale experiments, researchers were able to justify the time and resources needed for larger and more impactful applications.

However, despite its historical significance, MNIST has three notable shortcomings. First, it is too simple. Linear classifiers, fully-connected networks, and convolutional models all perform similarly well, obtaining above 90% accuracy (Table 1). This makes it hard to measure the contribution of a CNN's spatial priors or to judge the relative

[1]Oregon State University, USA [2]The ML Collective [3]University of Tübingen, Germany [4]Heidelberg University, Germany. Correspondence to: Sam Greydanus <samgreydanus@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Table 1: Test accuracies (in %) of common classifiers on the MNIST and MNIST-1D datasets. Most classifiers achieve similar test accuracy on MNIST. By contrast, the MNIST-1D dataset is capable of separating different models based on their inductive biases. The drop in CNN and GRU performance when using shuffled features indicates that spatial priors are important on this dataset. All models except logistic regression achieve 100% accuracy on the training set. Standard deviation is computed over three runs. **[CODE]**

Dataset	Logistic regression	MLP	CNN	GRU	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	$\sim 30 \pm 10$

effectiveness of different regularization schemes. Second, it is too large. Each sample in MNIST is a 28×28 image, resulting in 784 input dimensions. Together with its sample size $n = 70\,000$, this requires an unnecessarily large amount of computation to perform a hyperparameter search or debug a metalearning loop. Third, it is hard to hack. MNIST is a fixed dataset and it is difficult to increase the sample size or to change the noise distribution. The ideal toy dataset should be procedurally generated to allow researchers to vary its parameters at will.

In order to address these shortcomings, we propose the MNIST-1D dataset (Figure 1). It is a minimalist, low-memory, and low-compute alternative to MNIST, designed for exploratory deep learning research where rapid prototyping and short latency are a priority. MNIST-1D has 40 dimensions, many fewer than MNIST's 784 or CIFAR's 3,072. The sample size can be arbitrarily large, but the frozen default dataset contains 4000 training and 1000 test samples, many fewer than the 70,000 in MNIST and 60,000 in CIFAR-10/100. Although our dataset is procedurally generated, its samples are intuitive enough for a human expert to match or even outperform a CNN.

MNIST-1D does a much better job than the original MNIST at differentiating between model architectures: a linear classifier can only achieve 32% accuracy (Table 1), while a CNN reaches 94%. Below we show that MNIST-1D can be used to study phenomena ranging from deep double descent to self-supervised learning. Crucially, the experiments we present in this paper take only a few minutes to run on a single GPU (in some cases just a CPU) whereas they would require multiple GPU hours or even GPU days when using MNIST or CIFAR. This makes MNIST-1D valuable as a playground for quick initial experiments and invaluable for researchers without access to powerful GPUs.

All our experiments are in Jupyter notebooks and are available at `https://github.com/greydanus/mnist1d`, with direct links from figure captions. We provide a `mnist1d` package that can be installed via `pip install mnist1d`.

2. Related work

There are a number of small-scale datasets that are commonly used to investigate *science of deep learning* questions. We have already alluded to MNIST (LeCun et al., 1998), CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). The CIFAR datasets consist of 32×32 colored natural images and have sample sizes similar to MNIST. They are better at discriminating between MLP and CNN architectures and also between different types of CNNs: for example, vanilla CNNs versus ResNets (He et al., 2016). The FashionMNIST dataset (Xiao et al., 2017) has the same size as MNIST and is somewhat more difficult. It aims to rectify some of the most serious problems with MNIST: in particular, that MNIST is too easy and thus all neural network models attain roughly the same test accuracy. None of these datasets is substantially smaller than MNIST and this hampers their use in fast-paced exploratory research or compute-heavy applications such as metalearning.

There are very few datasets smaller than MNIST that are of interest for deep learning research. Toy datasets provided by Scikit-learn (Pedregosa et al., 2011), such as the `two_moons` dataset, can be useful for studying clustering or training very simple classifiers, but are not sufficiently complex for deep learning investigations. Indeed, these datasets are just 2D point clouds, devoid of spatial or temporal correlations between features and lacking manifold structures that a deep nonlinear classifier could use to escape the curse of dimensionality (Bellman and Kalaba, 1959).

To the best of our knowledge, the MNIST-1D dataset is unique in that it is over two orders of magnitude smaller than MNIST but can be used just as effectively — and in a number of important cases, *more effectively* — for studying fundamental deep learning questions. This may be why MNIST-1D was used as a teaching tool in the recent *Understanding Deep Learning* textbook (Prince, 2023)[1].

MNIST-1D bears philosophical similarities to the *Synthetic Petri Dish* by Rawal et al. (2020). The authors make similar

[1]The initial preprint of this manuscript was released on arXiv in November 2020.

Table 2: Default parameters for MNIST-1D generation.

Parameter	Value
Train/test split	4000/1000
Template length	12
Padding points	36–60
Max. translation	48
Gaussian filter width	2
Gaussian noise scale	0.25
White noise scale	0.02
Shear scale	0.75
Final seq. length	40
Random seed	42

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 the amount of noise, translation, and padding. Our dataset is more intuitive to humans: an experienced human can outperform a strong CNN on the MNIST-1D classification task. This is not possible on the Synthetic Petri Dish dataset.

3. The MNIST-1D dataset

Dimensionality. Our first design choice was to use one-dimensional time series instead of two-dimensional grayscale images or three-dimensional tensors corresponding to colored images. Our rationale was that one-dimensional signals require far less computation to train on but can be designed to have many of the same biases, distortions, and distribution shifts that are of interest to researchers studying fundamental deep learning questions.

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 templates consisted of 12 hand-crafted x coordinates. Next we padded the end of each sequence with 36–60 additional zero values, did a random circular shift by up to 48 indices, applied a random scaling, added Gaussian noise, and added a constant linear signal. We used Gaussian smoothing with $\sigma = 2$ to induce spatial correlations. Finally, we downsampled the sequences to 40 data points that play the role of *pixels* in the resulting MNIST-1D (Figure 1). Table 2 gives the values of all the default hyperparameters used in these transformations.

Implementation. Our goal was to make the code as simple, modular, and extensible as possible. The code for generating the dataset occupies two Python files and fits in 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 (Table 2). The following code snippet shows how to install the `mnist1d` package, choose a custom number of samples, and generate a dataset:

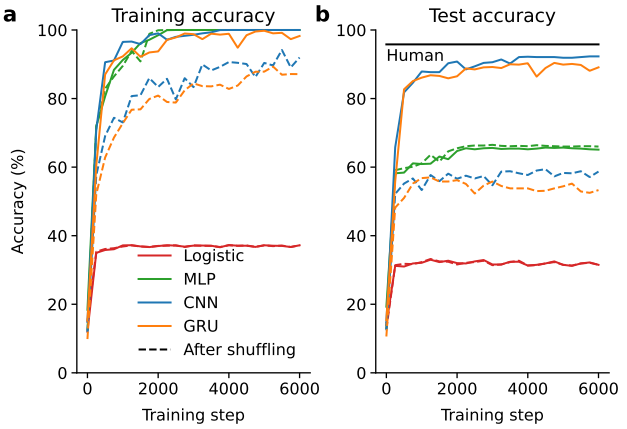

Figure 2: Train and test accuracy of common classification models on MNIST-1D. The logistic regression model fares worse than the MLP. Meanwhile, the MLP fares worse than the CNN and GRU, which use translation invariance and local connectivity to bias optimization towards solutions that generalize well. When local spatial correlations are destroyed by shuffling feature indices (dashed lines), the MLP performs the best. CPU runtime: ∼10 minutes. **[CODE]**

```
# install the package from PyPI
# pip install mnist1d

from mnist1d.data import make_dataset
from mnist1d.data import get_dataset_args

args = get_dataset_args() # default params
args.num_samples = 10_000
data = make_dataset(args)
x, y = data["x"], data["y"]
```

The frozen dataset with $4000 + 1000$ samples can be found on GitHub as `mnist1d_data.pkl`.

4. Classification

We used PyTorch to implement and train simple logistic, MLP (fully-connected), CNN (with 1D convolutions), and GRU (gated recurrent unit) models. We used the Adam optimizer and early stopping for model selection and evaluation. We obtained 32% accuracy with logistic regression, 68% using an MLP, 91% using a GRU, and 94% using a CNN (Table 1). Even though the test performance was markedly different between MLP, GRU, and CNN, all of them easily achieved 100% accuracy on the training set (Figure 2). While for the MLP this is a manifestation of overfitting, for

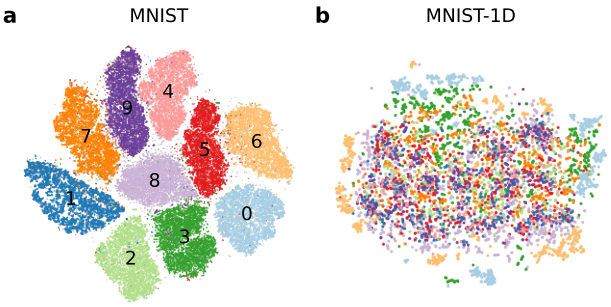

Figure 3: Visualizing the MNIST and MNIST-1D datasets with t-SNE. The well-defined clusters in the MNIST embedding indicate that the classes are separable via a simple kNN classifier in pixel space. The MNIST-1D plot reveals little structure and a lack of clusters, indicating that nearest neighbors in pixel space are not semantically meaningful, as is the case with natural image datasets. **[CODE]**

the CNN this is an example of *benign overfitting*.

For comparison, we also report the accuracy of a human expert (one of the authors) trained on the training set and evaluated (one-shot) on the test set. His accuracy was 96%. The purpose of this comparison was to show that MNIST-1D is a task that is as intuitive for humans as it is for machine learning models with spatial priors. This suggests that the models are not achieving high performance by exploiting some unintuitive statistical artifacts. Rather, they are using the relative position of various features associated with each 1-D digit. Interestingly, the CNN and the human expert had similar per-digit error rates (Figure S1).

Shuffling sanity check. We also trained the same models on a version of the dataset which was permuted along the spatial dimension. This 'shuffled' version measured each of the models' performances in the absence of local spatial structure (Zhang et al., 2017; Li et al., 2018). The test accuracy of CNNs and GRUs decreased by about 35 percentage points after shuffling whereas the MLP and logistic models performed about the same (Table 1, Figure 2). This makes sense, as the former two models have spatial and temporal locality priors whereas the latter two do not.

Dimensionality reduction. We used t-SNE (Van der Maaten and Hinton, 2008) to visualize MNIST and MNIST-1D in two dimensions. We observed ten well-defined clusters in the MNIST dataset, suggesting that the classes are separable with a kNN classifier in pixel space (Figure 3a). In contrast, there were few well-defined clusters in the MNIST-1D visualization, suggesting that the nearest neighbors in pixel space are not semantically meaningful (Figure 3b). This is well known to be the case for natural image datasets such as CIFAR-10/100, and is therefore a *benefit* of MNIST-1D, making it more interesting.

5. Science of deep learning with MNIST-1D

In this section we show how MNIST-1D can be used to explore empirical science of deep learning topics.

5.1. Lottery tickets and spatial inductive biases

It is not unusual for deep learning models to have many times more parameters than necessary to perfectly fit the training set (Prince, 2023). 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 can be trained in isolation to equivalent or even higher accuracies. These *lottery ticket* subnetworks can be found through a simple iterative procedure: train a network, prune the smallest weights, reset the remaining weights to their original values at initialization, and then retrain and repeat the process until the desired sparsity threshold is reached.

Since the original paper was published, many 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. We were able to demonstrate the existence of lottery tickets in a MLP classifier trained on MNIST-1D (Figure 4a–b). Lottery ticket subnetworks that we found performed better than random subnetworks with the same level of sparsity. Remarkably, even at high (>95%) rates of sparsity, the lottery tickets we found performed *better* than the original dense network.

The asymptotic performance of lottery tickets with 92% sparsity was around 70% (Figure 4c). When we reversed all the 1D patterns in the dataset, effectively preserving the spatial structure but changing the actual locations of all features (analogous to flipping an image upside down), the original lottery tickets continued to perform at around 70% accuracy (Figure 4d). This suggests that the lottery tickets did not overfit to the original dataset; instead, something about their connectivity and initial weights gave them an inherent advantage over random sparse networks. This reproduces the findings of Morcos et al. (2019), which showed that lottery tickets can transfer between datasets.

Next, we asked whether spatial inductive biases were a factor in the high performance of the lottery tickets we had found. To answer this question, we trained the same tickets on a feature-shuffled version of the MNIST-1D dataset.

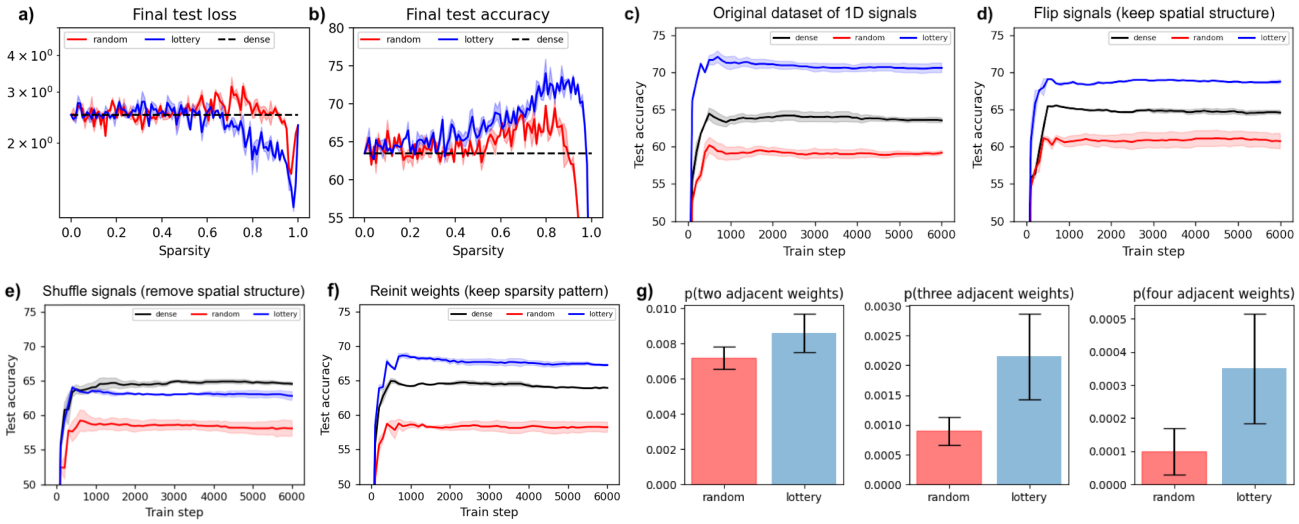

Figure 4: Finding and analyzing lottery tickets. **(a–b)** The test loss and test accuracy of lottery tickets at different levels of sparsity, compared to randomly selected subnetworks and to the original dense network. **(c)** Performance of lottery tickets with 92% sparsity. **(d)** Performance of the same lottery tickets when trained on flipped data. **(e)** Performance of the same lottery tickets when trained on data with shuffled features. **(f)** Performance of the same lottery tickets but with randomly initialized weights, when trained on original data. **(g)** Lottery tickets had more adjacent non-zero weights in the first layer compared to random subnetworks. Runtime: ∼30 minutes. **[CODE]**

In other words, we permuted the feature indices in order to remove any spatial structure from the data. Shuffling greatly reduced the performance of the lottery tickets: they performed appreciably worse — worse, in fact, than the original dense network (Figure 4e). This suggests that part of the lottery tickets' performance can be attributed to a spatial inductive bias in their sparse connectivity structure.

Furthermore, on the original (non-shuffled) MNIST-1D, when we froze the sparsity patterns of lottery tickets but initialized them with different random weights, they still continued to outperform the original dense network (Figure 4f). This suggests that not the weight values but rather the sparsity patterns represent the spatial inductive bias of lottery tickets. We verified this hypothesis by measuring how often non-zero weights in a lottery ticket were adjacent to each other in the first layer of the model. The lottery tickets had more adjacent weights than expected by chance (Figure 4g), implying a bias towards local connectivity. See Figure S2 for a visualization of the actual sparsity patterns of several lottery tickets.

The original lottery ticket paper (Frankle and Carbin, 2019), as well as some of the follow-up studies (Morcos et al., 2019), required a large number of GPUs and multiple days of runtime. By contrast, all the experiments we presented here took around ∼30 minutes to complete on a single GPU.

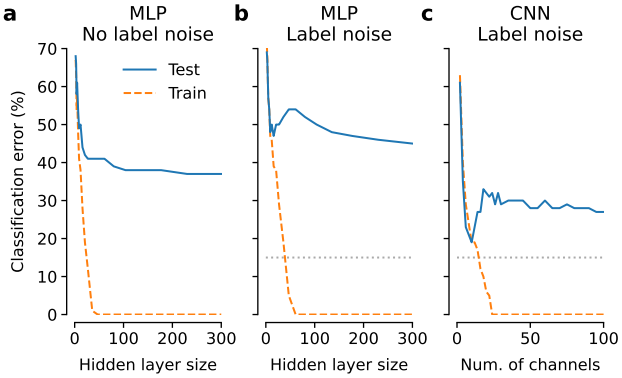

Figure 5: Deep double descent in MNIST-1D classification. Here the test set had 12 000 samples. **(a)** MLP classifier with one hidden layer. **(b)** MLP classifier; 15% label noise. **(c)** CNN classifier with three convolutional layers; 15% label noise. Adapted with permission from Prince (2023, Section 8.4). CPU runtime: ∼60 minutes. **[CODE]**

5.2. Deep double descent

An intriguing property of neural networks is the *double descent* phenomenon. This phrase refers to a training regime where more data, more model parameters, or more gradient descent steps can *reduce* the test accuracy before it increases again (Trunk, 1979; Belkin et al., 2019; Geiger et al., 2019; Nakkiran et al., 2020). This happens around the so-called

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 a single model that can fit the data and this model is very sensitive to label noise and model mis-specification, resulting in overfitting and poor test performance. In contrast, larger models tend to exhibit *benign overfitting* wherein SGD selects a smooth model out of the many possible models fitting the training set (*implicit regularization*).

Despite the above intuition, many aspects of double descent, such as what factors affect its width and location, are not well understood. We argue that MNIST-1D is well suited for exploring these questions. We observed double descent when training a MLP classifier on MNIST-1D, varying the size of the single hidden layer. In the presence of 15% label noise, the test error peaked at the interpolation threshold (training error reaching zero), at around 50 neurons in the hidden layer (Figure 5b). Further increasing the model size led to the test error dropping again. Without label noise, the test error did not peak (Figure 5a). We observed qualitatively similar behavior using the CNN architecture (Figure 5c). The runtime of this experiment was ∼60 minutes on a CPU.

5.3. Gradient-based metalearning

The goal of metalearning is to *learn how to learn*. This can be implemented by having two levels of optimization: a fast inner optimization loop which corresponds to a traditional learning objective and a slow outer loop which updates some meta properties of the learning process. One of the simplest examples of metalearning is gradient-based hyperparameter optimization. This concept was proposed in Bengio (2000) and then scaled to deep learning models by Maclaurin et al. (2015). The basic idea is to implement a fully differentiable training loop and then backpropagate through the entire process in order to optimize hyperparameters such as the learning rate or the weight decay.

Metalearning is a promising line of research but it is very difficult to scale. Metalearning algorithms can consume enormous amounts of time and compute due to their nested optimization, and tend to grow complex because most deep learning frameworks are not well suited for them. This places an especially high incentive on developing and debugging metalearning algorithms on small-scale datasets such as MNIST-1D.

We implemented a metalearning optimization for an MLP classifier on MNIST-1D with an explicitly written inner optimization loop using SGD. The gradient-based hyperparameter optimization converges to the optimal learning rate to be 0.62 regardless of whether the initial learning rate is too high or too low (Figure 6). The whole optimization process took only one minute on a CPU.

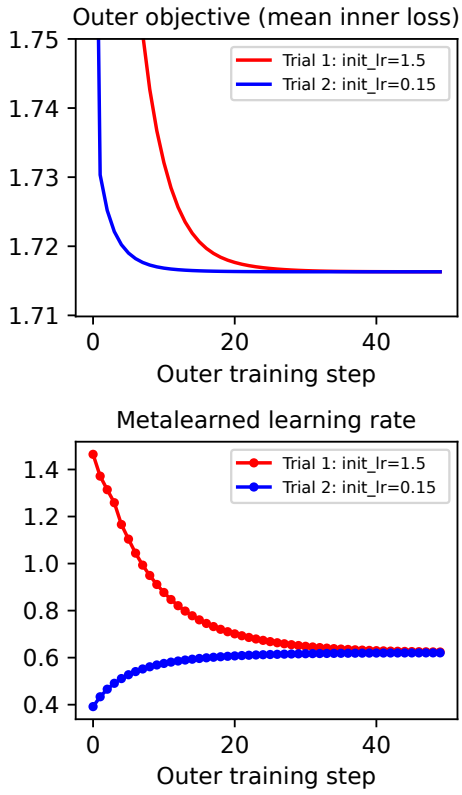

Figure 6: Metalearning the learning rate of SGD optimization of an MLP classifier on MNIST-1D. The outer training converges to the optimal learning rate of 0.62 regardless of whether the initial learning rate is too high or too low. Runtime: ∼1 minute. **[CODE]**

5.4. Metalearning an activation function

The small size of MNIST-1D allows researchers to perform more challenging metalearning optimizations. For example, it permits the metalearning of an activation function — something that to the best of our knowledge has not been studied before. We parameterized our classifier's activation function with a separate neural network (MLP with layer dimensionalities $1 \to 100 \to 100 \to 1$ using `tanh` activations, with outputs added to an ELU function such that it could be trained to produce perturbations to the ELU shape) and then learned its weights using meta-gradients. The learned activation function substantially outperformed common nonlinearities such as ReLU, Elu, and Swish (Figure 7), achieving over 5 percentage points higher test accuracy. The resulting activation function had a non-monotonic shape with two local extrema (Figure 7).

There has been work on optimizing activation functions (Clevert et al., 2016; Ramachandran et al., 2018; Vercellino and Wang), but none has used analytical gradients computed

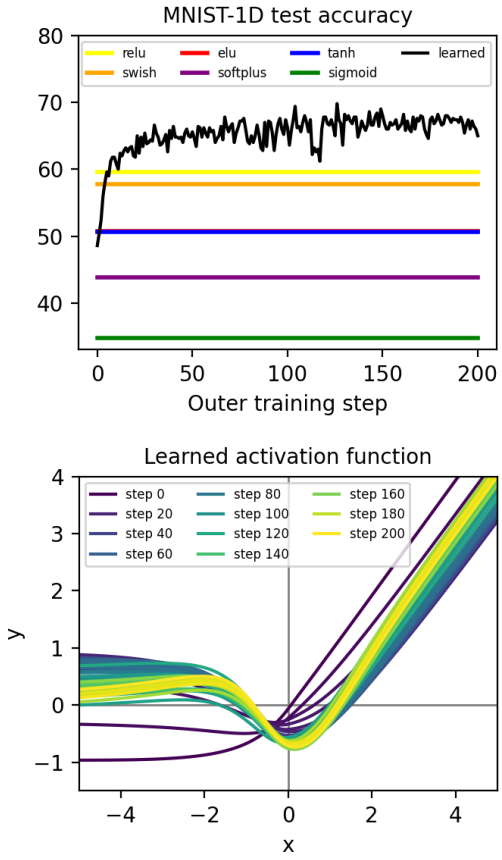

Figure 7: Metalearning an activation function. Starting from an ELU shape, we use gradient-based metalearning to find the optimal activation function for a neural network trained on the MNIST-1D dataset. The activation function itself is parameterized by a second (meta) neural network. Note that the ELU baseline (red) is obscured by the `tanh` baseline (blue) in the figure above. Runtime: ∼1 hour. **[CODE]**

via nested optimization. Moreover, some of these prior experiments (e.g. Ramachandran et al., 2018) used multi-day training runs on large clusters of GPUs and TPUs, whereas our entire training took around 1 hour of CPU runtime.

5.5. Self-supervised learning

As shown in Table 1, logistic classification accuracy for MNIST-1D in pixel space was low (33%). A powerful approach to self-supervised representation learning in computer vision is to rely on data augmentations: each input image is augmented twice, forming 'positive pairs' which the network is trained to map to close locations in its output space while pushing away representations of other input images (Balestriero et al., 2023). In particular, in SimCLR (Chen et al., 2020), each positive pair is repulsed from all other positive pairs in the same mini-batch via the InfoNCE loss function.

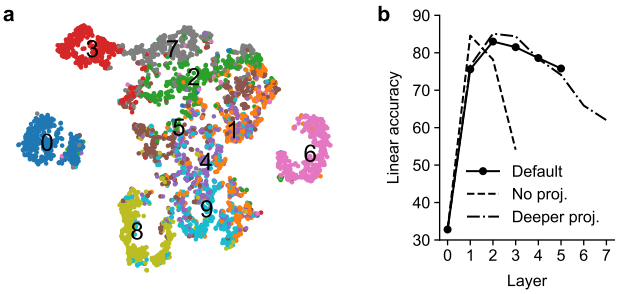

Figure 8: SimCLR-style (Chen et al., 2020) learning on MNIST-1D. **(a)** t-SNE embedding of the output representation after training ($n = 5000$). **(b)** Linear classification accuracy after each layer. Layer 0 stands for the input (pixel space). Accuracy always peaks in the middle (Bordes et al., 2023). CPU runtime: ∼5 minutes. **[CODE]**

We implemented the SimCLR algorithm for MNIST-1D, using a network with three convolutional and two fully-connected layers ('projection head') with output dimensionality 16. Our data augmentations consisted of regressing out the linear slope, circularly shifting by up to 10 pixels, and then reintroducing a random linear slope. We achieved 82% linear classification accuracy before the projection head in ∼1 minute of CPU training (for comparison, training SimCLR on CIFAR-10/100 datasets typically takes ∼10 GPU hours). In the output space, digits 0, 3, 6, and 8 appeared as isolated clusters (Figure 8a). Note that here we used both training and test sets of MNIST-1D for the self-supervised training, and the linear classifier was subsequently trained on the training set and evaluated on the test set.

Empirically, it has been observed that the representation quality (as measured via linear classification accuracy) is higher before the projection head rather than after (Chen et al., 2020); removal of the projection head after training has been dubbed *guillotine regularization* (Bordes et al., 2023) but remains poorly understood. We observed the same effect in our experiment: classification accuracy was the highest after the second layer (Figure 8b). Furthermore, when using a deeper projection head with four layers, or a network without any projection head at all, we achieved similar representation quality, and the accuracy always peaked in the middle (Figure 8b). This suggests that MNIST-1D is sufficiently rich to study cutting-edge open problems in self-supervised learning.

5.6. Benchmarking pooling methods

In our final case study we asked: *What is the relationship between pooling and sample efficiency?* Here we define pooling as any operation that combines activations from two or more neurons into a single feature. We are not aware of any prior literature on whether pooling makes models more

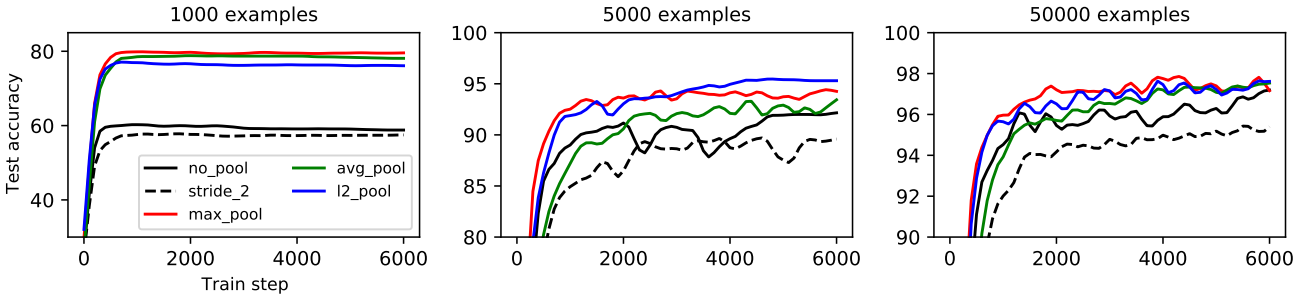

Figure 9: Benchmarking common pooling methods. Pooling was helpful in low-data regimes but hindered performance in high-data regimes. Runtime: ∼5 minutes. **[CODE]**

or less sample efficient.

With this in mind, we trained CNN models for MNIST-1D classification with different pooling methods and training set sizes. Note that here we make use of the procedural generation of MNIST-1D that allows one to generate additional samples at will. We found that, while pooling (but not striding!) was very effective in low-data regimes, it did not make much of a difference when more training data was available (Figure 9). We hypothesize that pooling is a poor-man architectural prior which is better than nothing with insufficient data but restricts model expression otherwise.

6. Discussion

When to scale. This paper is not an argument against large-scale machine learning research. That research has proven its worth and has come to represent one of the most exciting aspects of the ML research ecosystem. Rather, we wish to *promote* small-scale machine learning research. Neural networks do not have problems with scaling or performance — but they do have problems with interpretability, reproducibility, and training 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 could be investigated at scale. It would be interesting to show that large-scale lottery tickets also learn spatial inductive biases and feature local connectivity. It would also be interesting to try metalearning an activation function on a larger model in order to find an activation that can outperform ReLU and Swish in practical deep learning systems.

Understanding vs. performance. There has been some debate over the relative value of understanding neural nets versus increasing their performance. Some researchers contend that a high-performing algorithm need not be interpretable as 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 until we understand them better. Both arguments have merit. However, we believe that the process of identifying things we do not understand about large-scale neural networks, reproducing them in toy settings like MNIST-1D, and then performing careful ablation studies to isolate their causal mechanisms is likely to improve both performance and interpretability in the long run.

Reducing environmental impact. There is hope 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. Deep learning models do, however, require massive amounts of electricity to train and deploy (Strubell et al., 2019). Running experiments on smaller datasets — and waiting to scale until one has a solid grasp of the phenomena involved — is a good way to reduce the electricity costs and environmental impact of this research.

The scaling down manifesto. We would like to provocatively suggest 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 down in a way that preserves the nuances of their behaviors will allow researchers to iterate more quickly on fundamental and creative ideas. This fast iteration cycle is the best way to obtain insights on how to incorporate progressively more complex inductive biases into our models. We can then transfer these inductive biases across scales in order to dramatically improve the sample efficiency and generalization of large models. The MNIST-1D dataset is a first step in that direction.

Acknowledgements

We would like to thank Luke Metz for the interesting conversations, Tony Zador for the encouragement to release this dataset, Simon Prince for improving our initial double descent experiments and for the permission to adapt his code for Figure 5, and Peter Steinbach for his help with preparing the Python package.

This work was partially funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) via Germany's Excellence Strategy (Excellence cluster 2064 "Machine Learning — New Perspectives for Science", EXC 390727645; Excellence cluster 2181 "STRUCTURES", EXC 390900948), the German Ministry of Science and Education (BMBF) via the Tübingen AI Center (01IS18039A), and the Gemeinnützige Hertie-Stiftung.

Impact Statement

The goal of our paper is to advance the field of machine learning. We do not see any potential societal consequences of our work that need to be highlighted in this section.

References

R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

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.

R. Bellman and R. Kalaba. On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

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

F. Bordes, R. Balestriero, Q. Garrido, A. Bardes, and P. Vincent. Guillotine regularization: Why removing layers is needed to improve generalization in self-supervised learning. *Transactions on Machine Learning Research*, 2023.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pages 1597–1607. PMLR, 2020.

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. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

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. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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. *International Conference on Learning Representations (ICLR)*, 2014.

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

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.

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.

S. J. Prince. *Understanding Deep Learning*. MIT Press, 2023. URL http://udlbook.com.

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.

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

L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

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. *International Conference for Learning Representations (ICLR)*, 2017.

A. Supplementary Figures

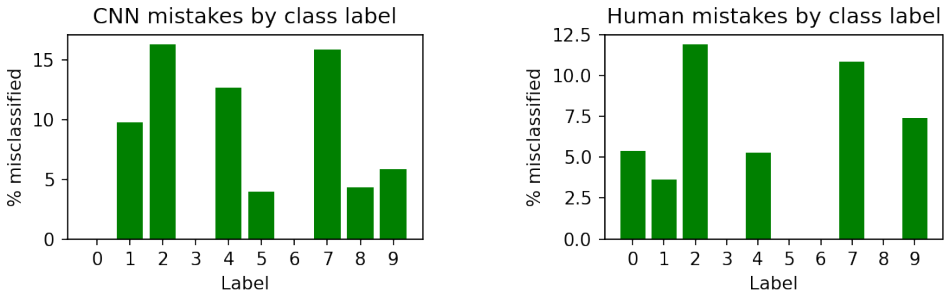

Figure S1: Classwise errors of a CNN and a human subject on the test split of the MNIST-1D dataset. As described in the main text, we estimated human performance on MNIST-1D by training one of the authors to perform classification and then evaluating his accuracy on 500 test images. His accuracy was 96%. The CNN's accuracy was 94%. Both humans and the CNN struggled primarily with classifying 2's and 7's, and to a lesser degree 4's. The human subject 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. It is interesting that a human could outperform a CNN on this task. Part of the reason may be that the CNN was only given 4000 training examples — with more examples it could possibly 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. It is worth noting that modern CNNs can outperform human subjects on most large-scale image classification tasks like ImageNet. But in our tiny benchmark a human was still competitive.

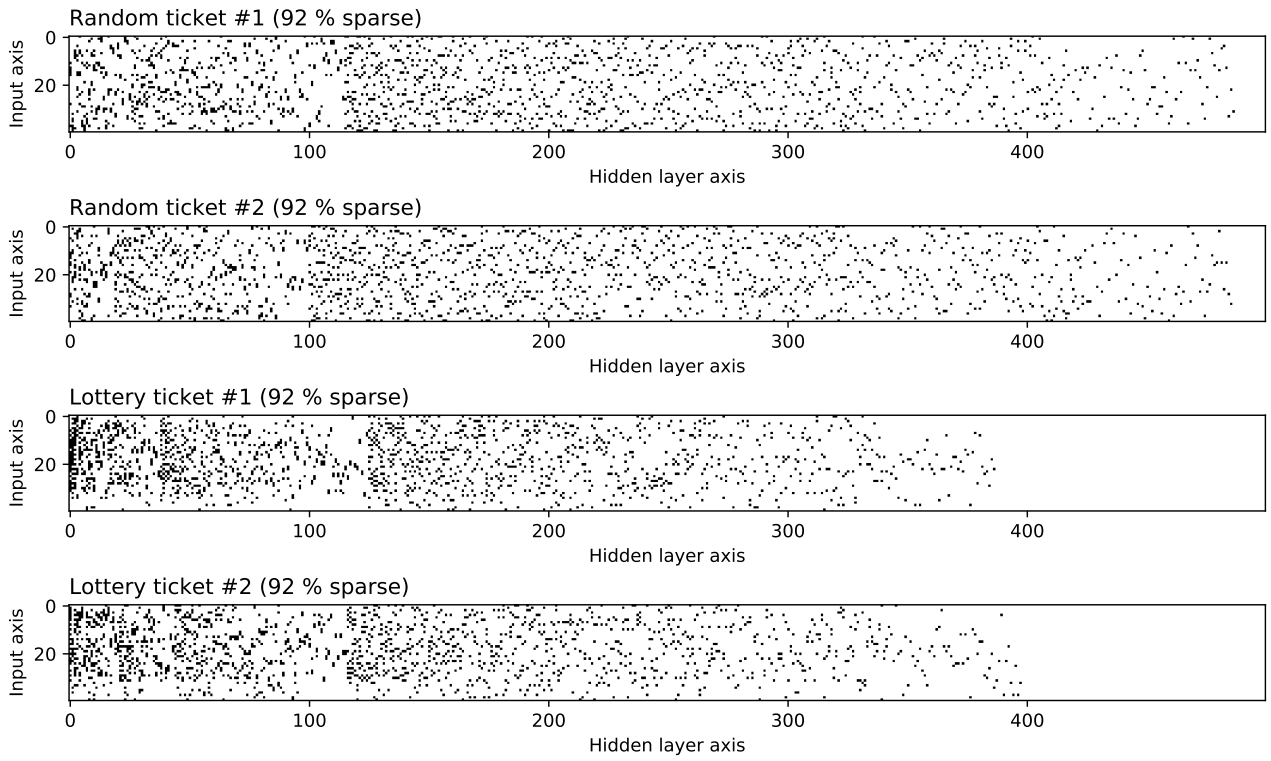

Figure S2: First layer weight masks of random tickets and lottery tickets. We sorted the masks along their hidden layer axes, according to the number of adjacent unmasked parameters. This helps to reveal a bias towards local connectivity in the lottery ticket masks. Notice how there are many more vertically-adjacent 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.