Reproducibility Track Proposal

Title: **Harnessing Minimalism: MNIST-1D for Rapid Deep Learning Prototyping.**

Selected Research Paper:

Title: Scaling Down Deep Learning with MNIST-1D

Conference: ICML 2024

Link to Paper: [2011.14439] Scaling Down Deep Learning with MNIST-1D (arxiv.org)

Objective:

Confirming the validity of the reported findings and **improving** and evaluating the performance of different deep learning architectures on the newly introduced MNIST-1D dataset. By assessing how various models perform, we aim to benchmark their efficiency in training compared to traditional datasets like MNIST and CIFAR. The project also seeks to facilitate learning and experimentation by enabling quick prototyping, making it easier for researchers and practitioners to conduct experiments on low-budget hardware. Finally, we aim to contribute to open science by sharing our findings and making the MNIST-1D dataset available for further research and exploration.

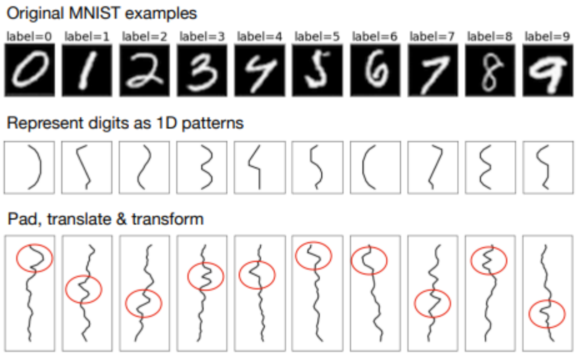


Figure 1: Constructing the MNIST-1D dataset. Unlike MNIST, each sample is a one-dimensional sequence. To generate each sample, we begin with a hand-crafted digit template loosely inspired by MNIST shapes. Then we randomly pad, translate, and add noise to produce 1D sequences with 40 points each. [CODE]

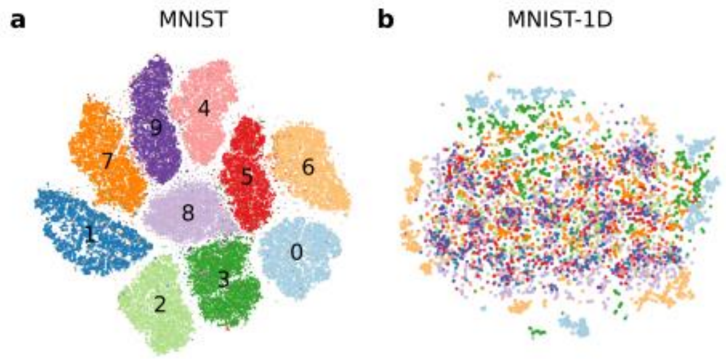
Tradition datasets require high computational resources and large datasets. The substantial time and cost associated with training deep learning models on larger datasets often prevent researchers from conducting timely analyses and prototyping new ideas. But using MNIST1D as mentioned above we can reduce the research cost and can improve efficiency.

Understanding the Paper:

The MNIST-1D dataset was designed using one-dimensional time series to minimize computational demands while still enabling exploration of relevant biases in deep learning.

Research implements and trains various models like logistic regression, MLP, CNN, and GRU using PyTorch, with Adam optimizer and early stopping for evaluation. Results showed that **logistic regression achieved 32% accuracy, MLP 68%, GRU 91%, and CNN 94%, while all**

models reached 100% accuracy on the training set. A human expert achieved 96% accuracy.



Visualizing the MNIST and MNIST-1D datasets with t-SNE. .

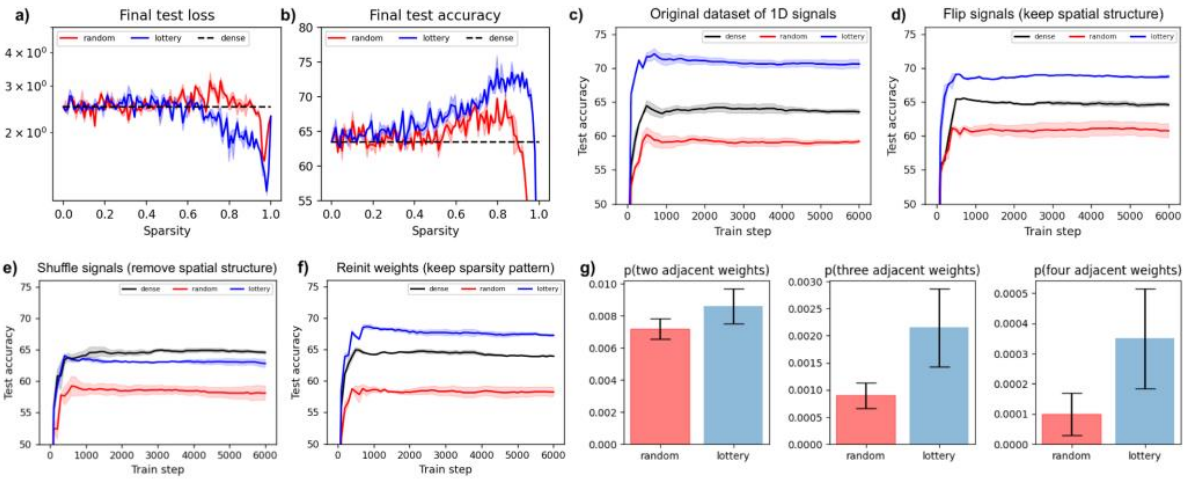
t-SNE analysis shows that MNIST has clear clusters, allowing for easy class separability with a kNN classifier. In contrast, MNIST-1D lacks structure, indicating that nearest neighbours in pixel space are not semantically meaningful. This highlights the unique challenges MNIST-1D presents compared to larger datasets like CIFAR-10/100.

Understanding of lottery tickets and spatial inductive biases:

Lottery ticket pruning is the approach for mitigating computational cost in deep learning models by choosing smaller and efficient subnetworks (lottery tickets) from large networks and trained separately. This paper demonstrates the existence of lottery tickets in an MLP classifier trained on MNIST-1D, revealing that these subnetworks outperform random sparse networks at high sparsity rates. Even at high (>95%) rates of sparsity, the lottery tickets we found performed better than the original dense network.

The performance of lottery tickets dropped and seen when tested on a feature-shuffled version of the dataset, which means that spatial inductive biases within the sparse connectivity structure contribute to their effectiveness. Also if random non zero weights are given and fixing the sparsity pattern resulted in superior performance, emphasizing the importance of connectivity over weight values.

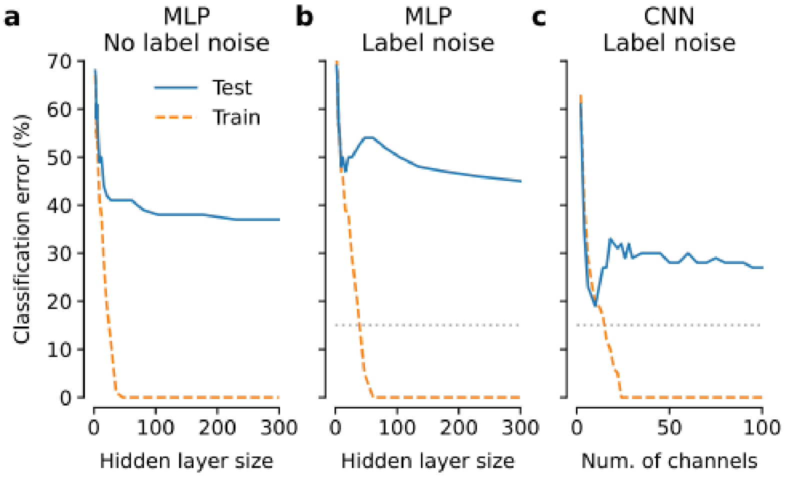
The figure below highlights, Finding and analysing lottery tickets. (a–b) The test loss and test accuracy of lottery tickets at different levels of sparsity, compared to randomly selected subnetworks and to the original dense network. (c) Performance of lottery tickets with 92% sparsity. (d) Performance 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.



Understanding Deep Double Descent:

It describes how test accuracy can initially decrease before increasing as model complexity and data size grows. This occurs at the interpolation threshold, where the model can perfectly fit the training data, making it sensitive to noise and mis-specification, leading to overfitting.

In experiments with MNIST-1D, we found double descent with a multi-layer perceptron (MLP) by varying hidden layer sizes. With 15% label noise, test error peaked around 50 neurons, then decreased as model size increased. Without noise, no peak was observed. Similar results were seen with a CNN, and the experiment took about 60 minutes on a CPU, highlighting MNIST-1D's utility for studying double descent.

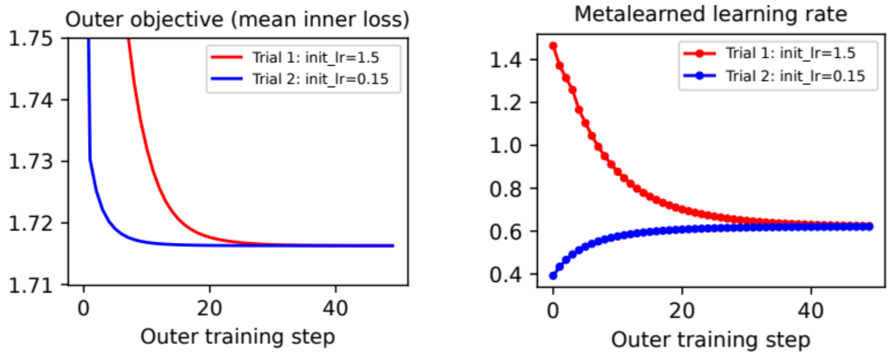


The figure shows: Deep double descent in MNIST-1D classification. Here the test set had 12 000 samples. (a) MLP classifier with one hidden layer. (b) MLP classifier; 15% label noise. (c) CNN classifier with three convolutional layers; 15% label noise. CPU runtime: ~60 minutes.

Understanding Gradient-based metalearning:

Gradient-based metalearning aims to improve the learning process by two-level optimization structure: a fast inner loop for traditional learning objectives and a slower outer loop for updating meta properties.

But metalearning can be expansive as it requires small scale datasets like MNIST-1D for better development and debugging. Here, metalearning is implemented for optimization of MLP classifier on MNIST-1D, using inner SGD optimization loop.

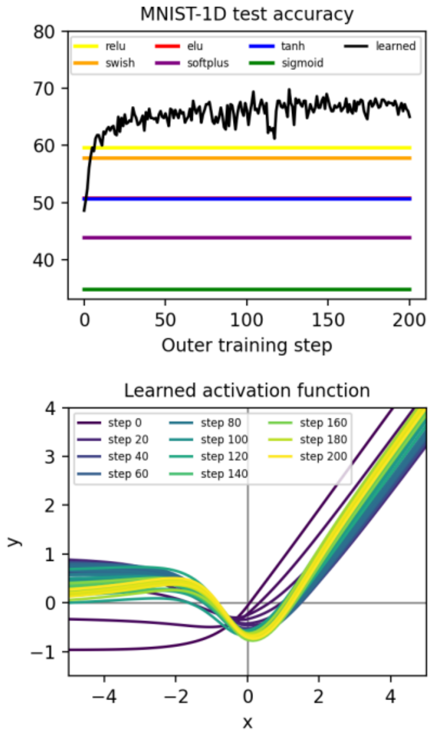


Metalearning the learning rate of SGD optimization of an MLP classifier on MNIST-1D. The outer training converges to the optimal learning rate of 0.62 regardless of whether the initial learning rate is too high or too low. Runtime: ~1 minute.

Understanding Metalearning an activation function:

The smaller size of MNIST-1D enable advance metalearning tasks, including optimization of **activation function,** parameterized the activation function of an MLP classifier using a separate neural network (MLP with dimensions 1 -> 100 -> 100 -> 1 and tanh activations) to create perturbations to an ELU function, optimizing its weights through meta-gradients.

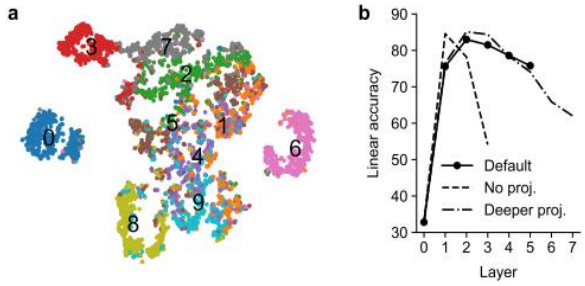
The new activation function created through metalearning worked better than common ones like ReLU, ELU, and Swish, improving test accuracy by over 5 percentage points. It had a non-monotonic shape with two peaks. While other studies have looked at optimizing activation functions, they didn't use analytical gradients from nested optimization, and many required a lot of time and powerful GPUs or TPUs. In comparison, training this activation function took only about one hour on a regular CPU.



The figure shows: 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.

Self-supervised learning:

To improve representation leaning, using SimCLR algorithm with data augmentation, it has achieved 82% linear classification accuracy, costing 1 minute of CPU. The network, with three convolutional and two fully connected layers, used augmentations like circular shifts and slope adjustments. It was found that representation quality was highest before the projection head, consistent with the guillotine regularization effect. Varying the projection head architecture showed similar results, with accuracy peaking after the second layer. This highlights MNIST-1D's potential for studying self-supervised learning.



SimCLR-style 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. CPU runtime: ~5 minutes

Benchmark pooling methods:

Pooling was highly effective in low-data scenarios but made little difference with larger datasets. Pooling serves as a simple architectural prior, beneficial when data is limited but restrictive as data availability increases.

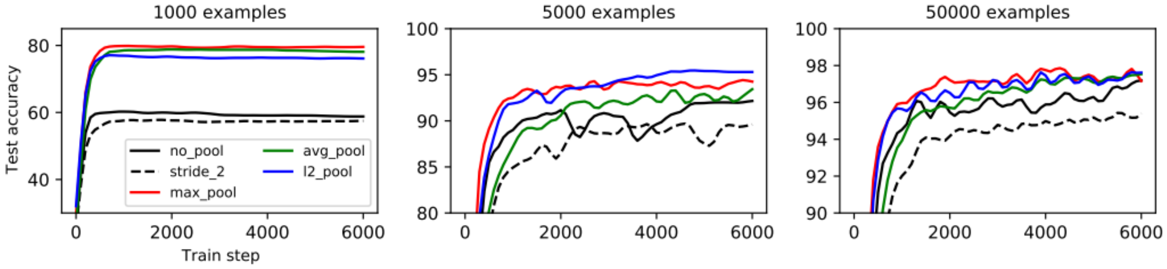


Figure shows: Benchmarking common pooling methods. Pooling was helpful in low-data regimes but hindered performance in high-data regimes. Runtime: ~5 minutes.

Challenges and resources Availability:

The code and the data are provided by the author in the GitHub repo along with the book reference. The dependencies are not stated in repo but can be understood from the paper.

According to my understanding of paper, It may be difficult to:

1. Handle sparse network like optimizing and pruning lottery tickets while desiring high performance
2. In double descent phenomena where model performance may degrade significantly before increasing
3. Gradient based metalearning involves computationally heavy algorithm and optimization.

Ablation Study:

1. Varying the pruning rate to access how different levels of sparsity effect the performance of lottery tickets.
2. How fixing different sparsity patterns effect but randomizing weights to know the effect on structure and performance?
3. How shuffled and non-shuffled dataset effect lottery tickets?
4. Varying the size of hidden layers or filters to explore the effect on double descent phenomena.
5. Effect of different levels of noise on the double descent curve.
6. Disturbing the use of meta-learning to optimize learning rate.
7. Evaluate the impact of learning custom activation functions on test accuracy as compared of using standard ones like ReLU or ELU.
8. Varying the number of project head to explore how representation quality gets affected before and after projection head.
9. We may also apply different pooling methods and striding to evaluate sample efficiency and model performance.
10. We may ablate the use of the InfoNCE loss function and compare it with alternative losses to measure its impact on representation quality.

Methodology:

First, I will try to follow the codes given in the repository, exploring the science the deep learning with MNIST-1D. And considering the deviation in accuracy or performance occurs in my implementation and how author has implemented it.

I will be using google colab and also as the project proceed will try to use my own's laptop gpu (NVIDIA GETFORCE RTX 3050), but I don't think it will perform better and this gpu is for graphical processing more, whereas T4 is better for machine leaning and AI workloads.

Everything depends on **time**, will doing ablation study as much as possible, comparing the performances with changes I do during ablation study.

Expected timeline (Considering the ETE prep and academics):

1. 24-Oct-24 to 1-Nov-24: Understanding codes and setting up.
2. 2-Nov-24 to 5-Nov-24: Running the models on the dataset and assessing accuracy and performance.
3. 28-Nov-24 to 8-Jan-24: Carrying out ablation study, considering the points under the section of ablation study.

4. 9-Jan-24 to 13-Jan-24: Detailing things done in project, comparing the performances in ablation study with the implemented earlier.

Expected Outcome:

1. Better performance with MNIST-1D and reduced complexity as if done with already available dataset.
2. Reduce the time for training and testing and power consumption and work load on machine.
3. While doing ablation, I may find better methods like for pooling and striding the can effect the complexity or performance.
4. Increase in the performance with brute forcing sparsity, just know to know at what level is better.
5. Smaller models may overfit at the interpolation threshold.
6. Meta-learned parameters could lead to better convergence and better accuracy.
7. Meta-learned activation function might outperform traditional ones
8. Impact of label noise on double descent and adjusting threshold.