COMP551 - Mini Project 3 - MLPs for Image Classification

Group 57: Jean-Pierre Falet, Miguel Ibanez Salinas, Alex Le Blanc
{alex.leblanc,miguel.ibanezsalinas,jean-pierre.falet}@mail.mcgill.ca
April 1, 2021

Abstract

In this project, we investigated the performance of multilayer perceptrons (MLP) on the MNIST dataset: an image classification benchmark dataset. We investigated various hyperparameters and architectural choices, such as the number and width of hidden layers, activation functions, regularization parameters, and dropout. We also investigated data normalization, and dimensionality reduction with principal component analysis. We found that the MNIST classes were easy to separate with minimal overfitting over a wide range of epochs, benefiting little from heavy regularization. Adding more hidden layers and increasing layer width was beneficial, but too many hidden layers impaired performance slightly under some conditions. We achieved our best test accuracy of 98.13% using an MLP with 2 hidden layers and leaky ReLU activation functions in the hidden layers. We compared this performance with a CNN which performed even better, with a test accuracy of 99.11%.

1 Introduction

In this project we investigated the use of MLPs for classification of images of handwritten digits in the MNIST dataset [1]. This task corresponds to the field of optical character recognition and involves a large body of research in machine learning and computer vision, including linear classifiers, non-linear classical machine learning models, neural networks, and convolutional neural networks (CNNs) [2].

The MNIST dataset is very commonly used as a benchmark for image classification algorithms. The current SOTA model is a branching and merging CNN with homogeneous filter capsules, and can achieve a test accuracy of 99.84% on the MNIST data [3]. With a basic 2-hidden-layer MLP, we were able to achieve a test accuracy of 98.13%, which comes quite close (1.71% difference). We found that regularization was only slightly helpful, and the validation errors were very stable over hundreds and even thousands of epochs, suggesting it is difficult to significantly overfit on this dataset. Greater width and depth generally improved accuracy, although there was a ceiling effect for some architectures. By testing additional activation functions, we found leaky ReLU to perform best, with very little difference among the other activation functions. We found normalization of the dataset to be crucial to speed up and stabilize gradient descent, and principal components analysis (PCA) to help improve computational efficiency without sacrificing accuracy. Finally, we compared the performance of our best MLP with a basic implementation of a CNN and found the CNN to outperform the MLP (99.11% test accuracy).

2 Dataset and Model Architecture

2.1 Dataset

The MNIST dataset is a collection of images of handwritten digits. It is comprised of a training set with 60,000 instances and a test set with 10,000 instances. Each instance represents a grayscale image of a digit between 0 and 9, representing ten classes. The images are stored in 28x28 arrays where each element represents the corresponding pixel's 8-bit value (between 0 and 255). The classes are fairly evenly distributed: the least prevalent class (5) represents 9.035% of the dataset whereas the most prevalent class (2) represents 11.27% of the dataset. After vectorization, each input instance has 784 dimensions. We checked for abnormal pixel values and found no issues with the data.

2.2 Model Architecture

A multilayer perceptron is a feedforward artificial neural network, consisting of input, hidden and output layers. In our case, the input layer has 784 units and receives the vectorized array representation of the image; the output is a 10-unit softmax layer which outputs the class probabilities, the maximum of which is the class prediction. Our MLP models are fully connected with no weight sharing. Each neuron has a nonlinear activation function (tanh, sigmoid, ReLU or leaky ReLU). We used backpropagation and mini-batch gradient descent (batch size of 264) for learning optimal weights.

2.3 Early Stopping and Accuracy Smoothing

To reduce training time for each model and prevent overfitting, we implemented an early stopping class. Early stop-

ping monitors the validation accuracy at each training epoch and stops training once there is no further improvements after a set number of epochs. It saves checkpoints to reload the best-performing model weights at the end. The best epoch identified during K-fold cross-validation (CV) is defined as the maximum of the early-stopping epochs across all folds and is used as the epoch duration for training the final model on the entire training dataset. Furthermore, to reduce the effect of fluctuations in the validation accuracy, we smoothed the validation accuracy using an exponential moving average with $\beta = 0.9$. This smoothed accuracy curve provides a more accurate estimate of the model's overall performance and helps in selecting hyperparameters that will perform well once trained on the entire training dataset.

3 Results

We evaluated performances with 3-fold CV. We use the average CV accuracy to tune each model's hyperparameters and report the test accuracy for the tuned models. The execution time was long for some experiments, so we implemented a random search function for hyperparameter tuning.

3.1 Experiment 1: Number of hidden layers

In the first experiment, we studied the effect of network depth when the width of each layer is fixed to 128, and the activation function is set to ReLU. We only tuned for learning rate ($\alpha \in \{0.25, 0.1, 0.01\}$). We see from table 1 that test accuracy and average CV accuracy is significantly lower when we do not use hidden layers, as opposed to having a nonzero amount of hidden layers (at least 5.32% difference). Moreover, we note that both the average CV accuracy and test accuracy peak with one hidden layer. It is expected for these accuracies to peak at a certain number of hidden layers and to pleateau or even decrease with increasing depth. Potential explanations include insufficient data for a growing number of model parameters, higher propensity to overfit due to increased model complexity, and more complex loss landscape optimization.

3.2 Experiment 2: Activation function

In the second experiment (table 2), we compared the performance of four different activation functions (leaky ReLU, ReLU, sigmoid and tanh) in a 2-hidden layer MLP with a width of 128. The hyperparameters were batch size (264 and 1280), and learning rate (0.01, 0.1 and 0.25). Best CV accuracy results are very similar between activation functions (from 97.36% to 97.48%). Nevertheless, leaky ReLU's test accuracy (98.13%) is slightly above the others. During CV, the best results were obtained for batch size 264 and the relatively high learning rate of 0.25 but results with batch size 1280 and lower learning rates were comparable (except for sigmoid activation, which went down to 92.31% CV accuracy for the smallest learning rate of 0.01 and batch size of 1280). Interestingly, the test error is slightly lower than the CV error for all activation functions, which could be a result of random dataset shuffling placing easier instances in the test set than the validation set. Also, when comparing training and validation accuracy, we observe that, in most cases, we obtain 100% training accuracy in less than 150 epochs while validation accuracy ranges between 97 and 98% (likely representing over-fitting). Figure 1 provides an example of this behavior for the leaky ReLU activation. Regarding execution time, leaky ReLU was faster to train than other activation functions, presumably because it enabled more gradient flow due to it's non-zero and non-plateauing derivative throughout its range.

3.3 Experiment 3: L2 regularization

We then investigated the effect of L2 regularization when added to a 2-hidden-layer MLP with a width of 128 using ReLU activations. The results of eight different regularization rates λ are shown in Table 3. We fixed the learning rate $\alpha = 0.25$.

We note that accuracy improves as λ increases and peaks at $1e-2$ before decreasing again. This corresponds to the expected overfitting with lower regularization coefficients and underfitting with higher regularization coefficients.

We then performed a random search with 20 hyperparameter combinations sampled for λ and the learning rate α which yielded an improved CV accuracy (97.73%) and test accuracy (98.1%) with $\lambda = 1e-1$ and $\alpha = 1e-1$. This emphasizes the need to tune each hyperparameter simultaneously rather than sequentially given their interdependence.

3.4 Experiment 4: Un-normalized data

To understand the effect of data normalization, we performed a grid search for $\alpha \in \{0.5, 0.25, 1e-1, 1e-2, 1e-3, 1e-4\}$ for a 2-layer MLP with ReLU activations. The resultant best CV accuracy was 93.43% with test accuracy of 95.51% using $\alpha = 1e-3$. We then performed the same random search from experiment 3 with 20 random hyperparameter combinations for both α and λ. The resultant best CV accuracy was 94.52% with a test accuracy of 95.96%. This was achieved with a learning rate $\alpha = 1e-3$ and L2-regularization coefficient $\lambda = 1$. This learning rate is much smaller and the L2-regularization is much larger than those achieving the highest CV accuracy on the normalized data, which suggests that some of the gradients and weight updates are too large using the unnormalized data. In fact, only $\alpha \leq 1e-3$ resulted in acceptable accuracies in the range of 90-95% accuracy—all other learning rates resulted in peak accuracies around 10%, suggesting no learning could occur.

#Hidden Layers	Best Learn. Rate	CV Acc.	Test Acc.
0	0.1	92.03	92.5
1	0.25	97.47	**97.87**
2	0.25	97.42	97.82

Table 1: Experiment 1. Effect of MLP network depth on performance

	Best parameters		Accuracy (%)	
activation	Batch Size	Learn. Rate	CV	test
leaky ReLU	264	0.25	97.36	**98.13**
ReLU	264	0.25	97.41	97.78
sigmoid	264	0.25	97.48	97.94
tanh	264	0.25	97.38	97.86

Table 2: Experiment 2. Effect of activation functions on performance (two hidden layers).

λ	CV Accuracy (%)
0	97.30
1e-5	97.31
1e-4	97.32
1e-3	97.32
1e-2	***97.63**
1e-1	97.50
0.5	96.61
1	95.67

Table 3: Experiment 3. Effect of L2 regularization on valid on average CV accuracy. *Best λ determined by the CV procedure.

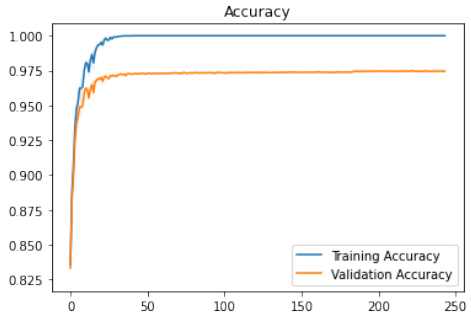

Figure 1: Training vs Validation accuracy by epochs. Experiment 2: leaky ReLU

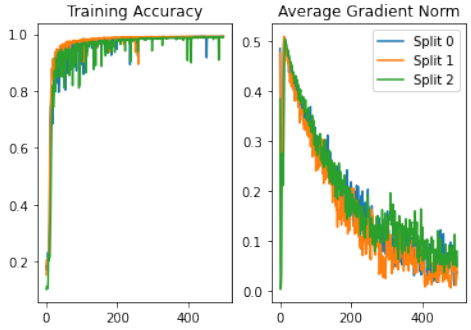

Figure 2: Training accuracy (left) and average gradient norm across all weights for each split (right) over training epochs after gradient clipping.

To prove our suspicion, we performed extra investigations of the average norm of the gradients over epochs for unnormalized images using the best α identified in experiment 1 ($\alpha = 0.25$). We see that at least one of the splits has extremely large gradient norms, beyond $3e15$ (figure shown in Notebook), and while it does begin to reduce after the first epoch, the reduction is insufficient and the training accuracies remain very low. We tried to get around this issue of large gradient variations by gradient clipping. Gradient clipping was implemented by setting a maximum gradient norm, such that gradient vectors g with norms exceeding a set threshold θ are rescaled as follows: $g = \theta \frac{g}{\|g\|}$. Doing so with $\theta = 1.0$ significantly improves the stability of training, as shown in Figure 2. Nonetheless, learning is still slower, with more oscillations in the training accuracy, than what we saw with normalized images, and the peak accuracy after 500 epochs still seems to be increasing and does not seem to be at the optimum (95.76% CV accuracy, and 96.41% testing accuracy).

3.5 Extra Experiments

First, we investigated the prospect of additional hidden layers increasing model capacity and representational power. We explored the performance of 3 and 4-hidden-layer MLPs with 128 units, leaky ReLU activations and $\alpha \in \{0.2, 0.25, 0, 3\}$. In table 4 we observe that the best CV accuracy in both models is only slightly higher than that from the leaky ReLU 2-hidden layer model (97,85% and 97,73% versus 97.36%). Both extra layers only offer 0.07 pp improvement of accuracy in the test set (98.20%) when compared to that from the leaky ReLU 2-hidden layer model. Early stopping proved relevant during training setting as the best accuracy was found only after around 150 epochs.

activation	# Hidden Layers / units	Epochs	Hyper-parameter Learning Rate	Accuracy (%) CV	test
leaky ReLU	128, 128, 128	189	0.3	97.85	98.20
leaky ReLU	128, 128, 128, 128	143	0.2	97.73	98.20

Table 4: Extra Experiment 1. Performance when adding Extra Hidden layers -128 units

Width	CV Accuracy (%)
32	96.10
64	97.09
128	97.47
264	97.63
528	***97.77**

Table 5: Extra experiment 2. Effect of layer width on accuracy. *Best width determined by the CV procedure.

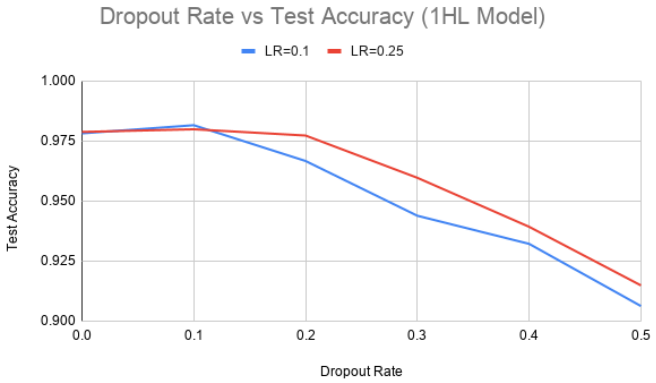

Figure 3: Test Accuracy vs Dropout Rate (ReLU 1HL MLP).

Second, we investigated whether the width of hidden layers had any impact on accuracy, by using a 1HL MLP with ReLU activation, and a learning rate of 0.25. Results shown in table 5 suggest that increased width does improve CV accuracy, however the gains diminish beyond 128.

Third, we investigated feature selection. Because the 784 input feature space translates into relatively long training time, we wanted to investigate whether limiting the input to the 100 most important components identified using principal component analysis (PCA) could preserve prediction accuracy while reducing training time. Table 6 summarizes the results for the 2-hidden 128-unit MLP with leaky ReLU and ReLU activation functions for various learning rates $\alpha \in \{0.15, 0.2, 0.25, 0.3, 0.35, 0.5\}$ and batch sizes (64 and 128). Results point to comparable results as found in experiment 1 and 2, suggesting PCA can be used to reduce MLP training time without having a noticeable accuracy cost.

Fourth, we experimented with a different regularization technique by implementing dropout. Weights were scaled during the training phase to compensate for dropout. In figure 3, we see the effect of dropout rate on the test accuracy of two ReLU 1 hidden-layer types of models with the two highest-performing learning rates: 0.1 and 0.25. We observe that for both types of models, lower dropout rates are preferred. Peak test accuracy values are observed with a dropout rate of 0.1, though a dropout rate of 0 produces very similar test accuracies (0.225% avg. difference). These results are consistent with the results of other regularization experiments, suggesting that the MNIST data is quite sensitive to regularization in MLPs, and that minimal over-fitting is occurring during training.

Finally, we investigated in more detail the predictions of our model. To do so, we plotted a confusion matrix (shown in the Notebook). The most common incorrect predictions are 9s predicted as 4s, followed closely by 5s predicted as 3s. Randomly sampling 5 images from the incorrect predictions reveals that the incorrect cases are often angled, rotated, or squished (shown in the Notebook). We implemented a CNN to see if it could help generalize better to these cases. Our CNN has a basic architecture consisting of two successive modules each consisting of a 32-channel 3x3 convolution with padding 1 and stride 1, a batch normalization layer, a leaky ReLU activation, and a 2x2 max-pooling layer. These two modules are followed by one fully connected layer with leaky ReLU activation and one fully connected layer with a softmax activation. This head is very similar to the one-hidden layer MLP tested previously, so it allows us to compare its performance when convolutional layers are added upstream. With no dropout, learning rate of 0.1, and 2-fold CV, we obtained a CV accuracy of 98.84% and test accuracy of 99.11%.

4 Discussion and Conclusion

In this project we trained MLPs, highly expressive non-linear classifiers, for image classification in the MNIST dataset. Given enough data, MLPs can fit extremely complex distributions, which is reflected in our excellent test accuracies, ranging from 97% to 98%. This could be achieved with relatively high learning rates (>0.2). This suggests the classes are easily separable, which is supported by the fact that even linear classifiers (such as our 0-hidden layer MLP) can achieve >90% accuracy on this dataset [4]. Another reason for the success of MLPs is likely that the MNIST digits are all preprocessed in the same way and digits are

			Hyper-parameters		Accuracy (%)	
activation	# Hidden L. / units	Princ. components	Batch size	Learn. Rate	CV	test
leaky ReLU	128, 128	100	64	0.2	97.81	98.28
ReLU	128, 128	100	64	0.5	97.80	98.36

Table 6: Extra Experiment 3. Performance with 100 Principal components at the data

centered. Therefore, while MLPs are not translation equivariant, they can perform very well on this dataset. CNNs, on the other hand, can achieve near perfect classification (>99.69%) [5]. CNNs have the advantage of translation equivariance, which helps reduce sensitivity to slight translations of parts of the digits, and their weight-sharing can be computationally favorable.

We noted excellent concordance between the CV accuracies and the test accuracies in all our experiments. Our use of three-fold cross validation, early stopping, and a moving average applied on the validation accuracy ensured that our hyperparameter search produced reliable and generalizable results. On occasion, we noted the test accuracies to be slightly higher than the CV accuracy, which we hypothesize is due to random shuffling of instances, where by chance some easier instances are placed in the test set. Neural networks can be difficult to configure and hyperparameter tuning was a critical part of our approach. Our k-fold cross validation training was time intensive, and the hyperparameter space can easily increase in size. Our learnt lessons include pre-selecting a restricted range of hyperparameters to identify a general preference for certain values, and performing a second, deeper hyperparameter search on a focused sub-range of values.

As per experiment 1 and extra experiment 1, we saw an improvement in accuracy by adding at least one hidden layer. Adding more than one layer, however, produced only slight gains in performance and only under certain conditions (e.g. using leaky ReLU). As we saw in extra experiment 2, increasing width similarly resulted in increasing accuracy.

All activation functions performed similarly during training in experiment 2. Nevertheless, leaky ReLU (and ReLU) achieved faster convergence, presumably because it enables more gradient flow because its derivative is non-zero and non-plateauing throughout the possible range of values.

In experiment 3 and in the extra dropout experiment, we saw slight improvements with low levels of regularization. Higher regularization resulted in underfitting. This suggests this task is easily separable and not conducive to overfitting.

In experiment 4, we observed significant increasse in training time, and lower accuracies accross most hyperparameters using unnormalized images. We found large gradient norm variations when using unnormalized images which was partially fold-dependent during cross validation training. We hypothesize this could be due to the different instances that each fold begins training on, whereby some folds may start with larger errors which, when back-propagated, can lead to large gradients if some of the input features have large unnormalized values. This could lead to overshooting in the weight space if these gradient updates are not in the direction of the optimum, leading to unstable learning. Another factor which could slow learning is the fact that unnormalized pixels all have positive values with a mean far from 0, which produces significant zigzagging in the weight space due to weight updates for each neuron being constrained to a single direction [6].

Our final extra experiment identified difficult digits for the MLP to predict, which included angled, rotated, or squished digits. We observed improvement in test accuracy by using our own CNN implementation, which points to a potential advantage of translation equivariance. While CNNs are not equivariant to scale and rotation, they may be better able to identify angled or slightly rotated digits as part of the digit can be said to be translated with respect to the other part. Max pooling layers likely also help reduce the sensitivity to slight local variations.

Future directions include further studies on early stopping. As we have noticed that early stopping leads to stopping training at a variety of times during cross-validation depending on the particular fold, it is not clear how to then select a best epoch from this procedure. In this paper we opted to take the maximum early-stopping epoch across all k-folds as the best number of epochs to then train our final model with, but other options would include taking the mean early-stopping epoch, or taking each k-fold's early-stopped model as a member of a k-sized ensemble which can then be directly used to predict on the unseen test set. Also, given that the MLP had the most difficulty predicting rotated or squished digits, data augmentation techniques such as image rotation could be tried. We would also like to experiment with different weight initialization strategies (He, Xavier, etc.) to see how these effect training.

5 Statement of Contributions

All authors contributed to writing this report. Falet: early stopping, random search, CV, calibration, CNN. Ibanez Salinas: MLP, gradient descent, PCA. Le Blanc: dataloading, dropout, weight initialization, activations.

References

[1] LeCun et al. “The MNIST DatasetOf Handwritten Digits (Images)”. In: (1999). URL: http://yann.lecun.com/exdb/mnist/.

[2] Alejandro Baldominos, Yago Saez, and Pedro Isasi. “A Survey of Handwritten Character Recognition with MNIST and EMNIST”. In: *Applied Sciences* 9.15 (2019). ISSN: 2076-3417. DOI: 10.3390/app9153169. URL: https://www.mdpi.com/2076-3417/9/15/3169.

[3] Adam Byerly, Tatiana Kalganova, and Ian Dear. *A Branching and Merging Convolutional Network with Homogeneous Filter Capsules*. 2021. arXiv: 2001.09136 [cs.CV].

[4] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[5] wikipedia. “MNIST database”. In: (2021). URL: https://en.wikipedia.org/wiki/MNIST_database.

[6] Yann A. LeCun et al. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.