A Comparison of MLP and CNN in Predicting Noisy Handwritten Digits

Dimitrios Megkos
School of Mathematics, Computer Science and Engineering
City, University of London

Abstract: The aim of this study is to create, optimize and compare the performance of Multilayer Perceptron and Convolutional Neural Network models on an image classification problem with added noise. Both models were trained with augmented handwritten digits images, each image having a random rotation between zero and thirty degrees, and random pixel erasing. Each algorithm's best models were tested on a clean test with no noise and a noisy set with added Gaussian Blur. Both algorithms achieved accuracies over 90%. However, the study showed that Convolutional Neural Network performed better on the clean set and was able to recognise more blurry digits than Multilayer Perceptron.

1. Brief description and motivation of the problem

Image classification is one of the most fundamental tasks in computer vision. It has contributed greatly to the advancement of many fields like identifying cancer cells in healthcare, self-driving cars in the automobile industry, biometric authentication in computer science, and more [1].

The purpose of this exercise is to build and optimize two supervised learning neural networks in order to compare their performance on the classification of handwritten digits. The best performing algorithm could be applied in a "real world" scenario, such as recognising and automatically converting handwritten digits that are written using a stylus and an input device or taking a picture of a handwritten phone number with a smartphone, and then using that picture to call the number.

2. Description of the dataset

The Neural Networks were trained and tested on datasets based on the MNIST (Modified National Institute of Standards and Technology) Database. This is a large database containing images of handwritten digits, numbers between zero and nine, that were normalized to fit into a 28x28 pixel box and anti-aliased, which introduced grayscale levels [2]. Each image contains one channel, the grayscale channel, having a distinct number from 0 to 255.

The database contains 60,000 training images and 10,000 testing images. The digits were drawn by employees of the American Census Bureau and by American high school students.

The training set was checked for class imbalance. Figure 1 below shows that all ten classes are fairly balanced.

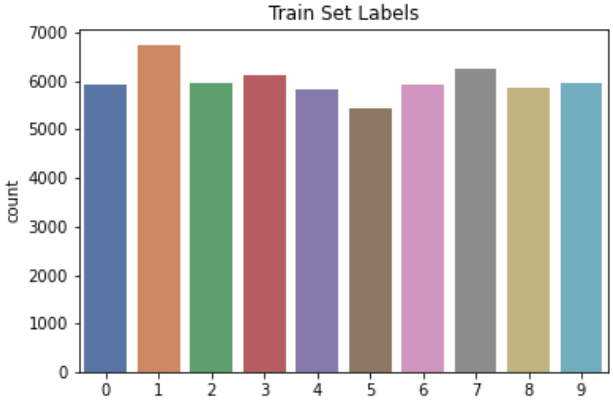

Figure 1: Bar chart showing the class balance

In order to add complexity to the problem and create more robust algorithms, extra noise was applied on the training set, using TorchVision transformations. First, a random rotation between zero and thirty degrees was applied to the digits, inspired by slanted handwriting. Second, random pixel erasing was applied to the images to further push the limits of the neural networks [3]. Figure 2 below shows the results of these augmentations on the training set. All images were normalized.

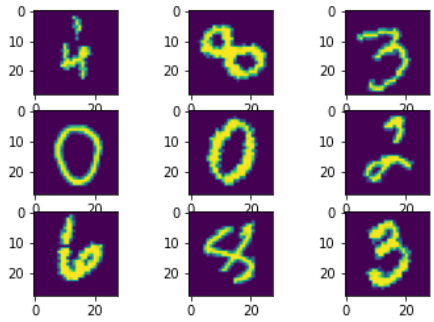

Figure 2: Training images augmented with random rotation and random pixel erasing

For the final testing of the neural network, two versions of the MNIST test set were used, a clean set with zero noise, figure 3, and a noisy set with Gaussian Blur, figure 4 below. All images were normalized.

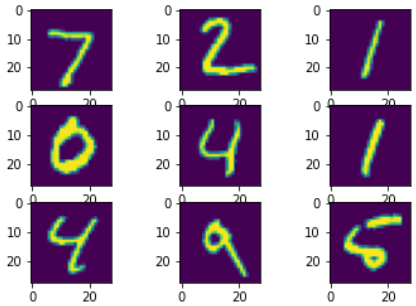

Figure 3: Clean testing images with no augmentations

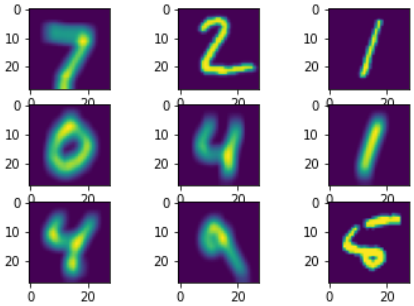

Figure 4: Noisy testing images with Gaussian Blur

3. Brief summary of the two neural network models with their pros and cons

The two models that were used for this classification problem are Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN). Both models can provide a lot of flexibility due to having several hyperparameter optimization options, making both powerful and reliable in a wide range of problems.

3.1. Multilayer Perceptron (MLP)

Multilayer Perceptron is considered the classical type of neural network. It is an extension of the first trainable Artificial Neural Network, the Perceptron, adding extra hidden layers consisting of multiple hidden neurons between the input and the output layers. The input data is received by the input layer, which is then passed through the hidden layers, the computational engine of the network. Finally, the prediction and classification are performed by the output layer. The network is trained using backpropagation, a method which updates the weights of the neuron connections based on the output error or loss. A Multilayer Perceptron with even just one hidden layer can approximate any function that contains a continuous mapping from one finite space to another [9].

Pros

- Easy to implement and train
- Suitable for both classification and regression tasks
- Capable of learning any mapping function, proven to be a universal approximation algorithm
- Can be applied to many types of data (image data, text data, time series data) if data is provided in a tabular format

Cons

- Falls behind in advanced computer vision tasks, because it takes flattened vectors as inputs, disregarding the spatial relationships of images
- Sensitive to feature scaling
- Supervised learning only algorithm

3.2. Convolutional Neural Network

Convolutional Neural Networks were designed to map images to an output variable, and they are the preferred method for image recognition problems. They introduce several convolution and pooling layers that pre-process the images before passing them to traditional feed forward neural networks. A full image is given as an input to the model, the convolution layer applies a filter (kernel) on top of the image computing the dot product. The output is collected in a feature map which is used as an input for the pooling layer. A pooling layer is a technique that is used to compress feature representations and reduce overfitting [8]. The output of the pooling layer is flattened and passed as input vector to a normal feed forward neural network. A Convolutional Neural Network can have multiple convolution and pooling layers and multiple hidden layers in the feed forward network.

Pros

- Suitable for both classification and regression tasks
- Can preserve the spatial relationship of data, making them ideal for image classification problems
- They can automatically learn and generalize features
- Designed to be invariant to object position and distortion in the scene

Cons

- Can take a longer time to train than normal feed forward networks due to the greater number of computations
- Has increased complexity due to the extra convolution and pooling layers
- Supervised learning only algorithm

4. Hypothesis statement

a. Both models will have a high accuracy despite the added noise on the test dataset. However, Convolutional Neural Network will perform better than Multilayer Perceptron, especially on the blurry test set, since the algorithm was designed specifically for image recognition.

b. Convolutional Neural Network will require significantly longer time to train than Multilayer Perceptron, because of the added convolution and pooling layers.

5. Description of choice of training and evaluation methodology

A validation set was created using twenty percent of the transformed images of the training set. The validation set was used after training the model in order to experiment with different model parameters. The test sets were used in the end, as a final test on the optimised model.

Both models were trained using batch learning, since it's a more stable method of learning compared to online learning. Typically, while a model is being trained, it's recommended to pass samples in “minibatches”, reshuffling the data at every epoch to reduce model overfitting. PyTorch's DataLoader is the answer for parallelizing the data loading process with automatic batching [4].

The model selection was determined by two performance metrics, the loss and the accuracy. The model that returned the lowest loss and highest accuracy on the validation set was the one selected as the best model, one for each algorithm.

The hyperparameters of each algorithm’s best models were stored and both models were retrained on the combined training and validation set and tested on the two test sets. The best algorithm was selected by comparing the accuracy and the confusion matrix of both each algorithm’s best model, on both the clean and noisy test sets.

6. Choice of parameters and experimental results

Since this is a multiclass classification problem and the output layer has multiple neurons, one for each class, Cross Entropy was used as a loss function for both algorithms, and a SoftMax activation function was used to output the probability of the network predicting each of the class values. The class with the highest probability was selected in the end as the predicted class.

The activation function that was chosen for both algorithms is the Rectified linear Activation Function (ReLU). ReLU will output the input directly if it is positive, otherwise, it will output zero. It is a nonlinear function that looks and acts like a linear function and is the default activation function for most types of neural networks. The weights of the connections of both algorithms were initialized using He initialization [5] and Glorot initialization [6]. The batch size of all data loaders was set to sixty-four, the number of iterations was set to five thousand, and the number of epochs was calculated using the batch size, the size of the training set and the number of iterations.

Regarding Multilayer Perceptron's number of hidden layers, since the given classification problem is an image recognition problem, it was decided from the start that the network will have two hidden layers. A Multilayer Perceptron with two hidden layers can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy [9]. Similarly, two hidden layers were also used for Convolutional Neural Network's fully connected network. Regarding the number of convolution and pooling layers, because the images have added noise, two convolution and two pooling layers were used to better learn and generalize features.

6.1. Experimental results

Two different optimizers were tested for MLP: Adam with 0.001 learning rate and Stochastic Gradient Descent with 0.01 learning rate and 0.9 momentum. Three values were used for MLP's number of neurons on the hidden layers: 100, 150, and 200. For CNN, only SGD was tested, with 150 and 200 hidden neurons and two different values of number of filters for the Convolution layer: 32 and 64.

6.1.1 Multilayer Perceptron

Using Stochastic Gradient Descent with 0.01 learning rate and 200 neurons on the hidden layers returned the best results, with an accuracy of 94%. The results of the experiments are shown on figure 5 and figure 6 below.

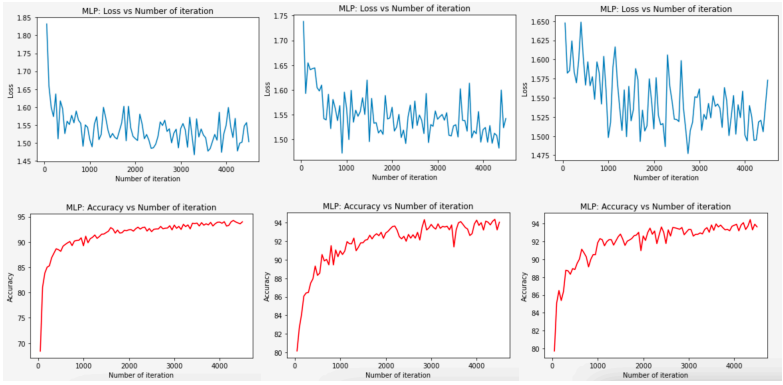

Figure 5: Adam with 100, 150 and 200 neurons

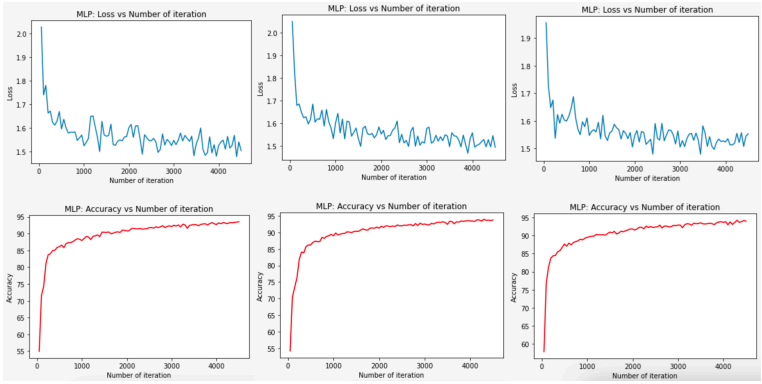

Figure 6: Stochastic Gradient Descent with 100, 150 and 200 neurons

6.1.2 Convolutional Neural Network

Using 64 filters on the convolution layer with 150 neurons on the hidden layer returned the best results, with an accuracy of 96.5%. The results of the experiments are shown on figure 7 below.

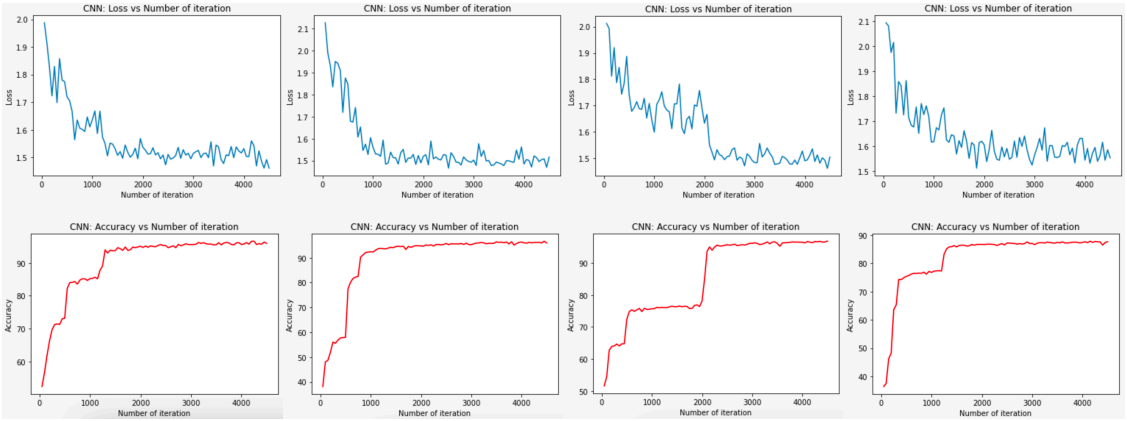

Figure 7: 32 filters with 150 and 200 neurons, 64 filters with 150 and 200 neurons

7. Analysis and critical evaluation of results

Both algorithms' best trained models performed great on the validation set, with Multilayer Perceptron having an accuracy of 94% and Convolutional Neural Network having an accuracy of 96.5%. Although the difference in performance is small, CNN does a better job in recognising rotated numbers, numbers with random pixel erasing or a combination of both.

Both models were retrained with the best hyperparameter values using the combined sets of training and validation in order to increase the amount of training images and were tested on the clean and blurry test sets. Figure 8 below shows that both algorithms had almost no trouble recognising clean images, which had no added rotation, pixel erasing or Gaussian Blur, with MLP having an accuracy of 94% and CNN having an accuracy of 97%.

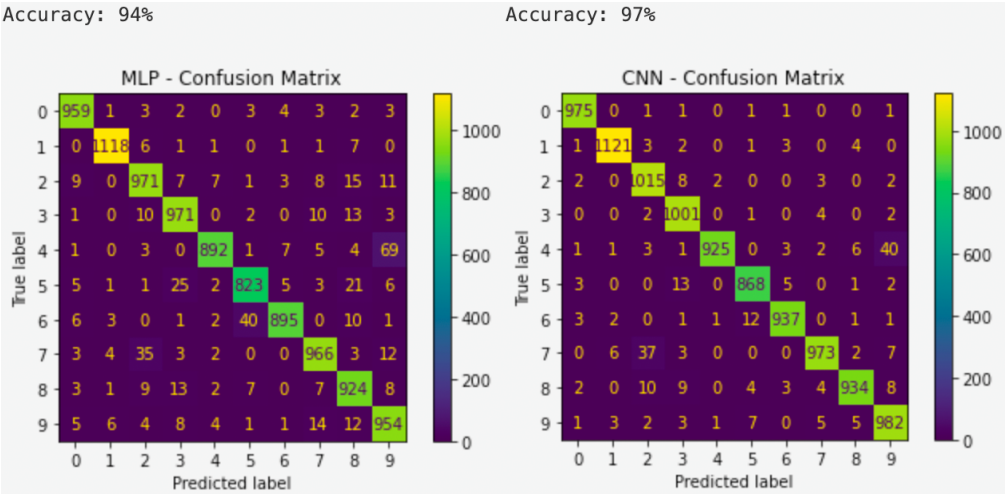

Figure 8: MLP and CNN accuracy and confusion matrix on the clean set

CNN did a better job in recognising all numbers, with number one being the easiest number to recognise for both algorithms. Numbers four and seven were the ones that got the most misclassifications as both models confused them with numbers nine and two respectively, since they look somewhat similar, with MLP also confusing number six with number five.

Performance on the blurry test was a bit worse than on the clean set, as expected, but still impressively high. Figure 9 below shows that both algorithms achieved an accuracy of 90%. Comparing the confusion matrices shows that each algorithm had different weaknesses. CNN was better in recognising numbers 0, 1, 2, 3, 5, 6 and 7, while MLP was better in recognising numbers 4, 8 and 9.

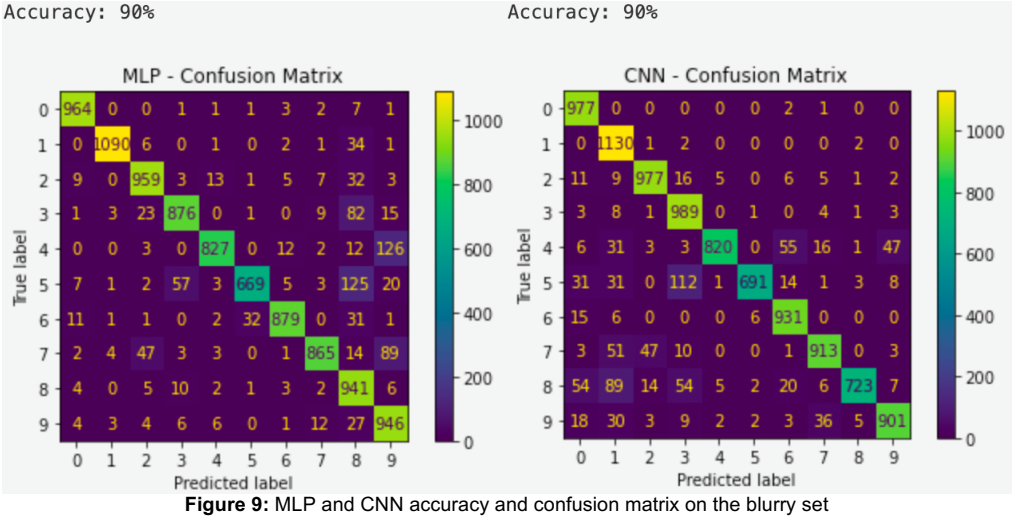

Figure 9: MLP and CNN accuracy and confusion matrix on the blurry set

These results make sense as even the human eye has trouble identifying blurry images without the use of external help (e.g., glasses for myopia). Nevertheless, the performance is impressive, as shown in figure 10 below.

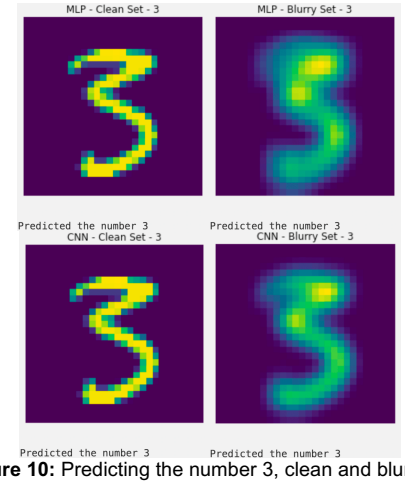

Figure 10: Predicting the number 3, clean and blurry set

8. Conclusions, lessons learned, and future work

8.1. Conclusions

Both hypotheses came true since CNN performed better than MLP overall, but also took five times longer to train. Despite the longer training time, CNN is better when it comes to image recognition with added noise, which explains why it is the preferred algorithm for image recognition tasks.

8.2. Lessons Learned

- How to apply augmentations on images and how added noise affects the classification
- How MLP and CNN work and how different parameter options affect performance and training time

8.3. Future work

- Apply more image augmentations to further push the limits of algorithms
- Implement a grid search function for better hyperparameter tuning
- Compare CNN with other neural networks, for example Capsule Networks

9. References

[1] Image Classification Explained: An Introduction, [link]

[2] The MNIST Database of handwritten digits, [link]

[3] Illustration of transforms, [link]

[4] Datasets & DataLoaders, [link]

[5] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026-1034, doi: 10.1109/ICCV.2015.123.

[6] Glorot, Xavier and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks.” AISTATS (2010).

[7] Crash Course On Multi-Layer Perceptron Neural Networks, [link]

[8] Crash Course in Convolutional Neural Networks for Machine Learning, [link]

[9] The Number of Hidden Layers, [link]

[10] A Gentle Introduction to the Rectified Linear Unit (ReLU), [link]

[11] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

10. Appendix

10.1. Glossary

Abbreviation/Term	Definition
MNIST (Modified National Institute of Standards and Technology)	Database containing images of handwritten digits, numbers between zero and nine
TorchVision	Python library for importing and transforming images
Training set	Part of the original data that is used for training the neural network
Validation set	Part of the original data that is used for testing different parameters
Clean testing set	Part of the original data that is used for the final testing of the neural network with no added noise
Blurry testing set	Part of the original data that is used for the final testing of the neural network with added Gaussian Blur as noise
Hyperparameter	Parameter that controls the learning process of the neural network
NN/Neural Network	A machine learning algorithm based on the human mind
MLP (Multilayer Perceptron)	A Neural Network with input, (multiple) hidden and output layers
CNN (Convolutional Neural Network)	A Neural Network that uses convolution, pooling and fully connected layers
Tensor	A multi-dimensional matrix containing elements of a single data type
Confusion matrix	A table that summarizes how successful a classification model's predictions were
Gaussian Blur	An image augmentation transformation, adding blur
Adam	An optimizer used by the neural network
SGD (Stochastic Gradient Descent)	An optimizer used by the neural network
Accuracy	Performance metric for the neural network
ReLU (Rectified linear Unit)	Activation function applied on each layer
SoftMax	Activation function applied on the output layer

10.2. Implementation details

Cross-validation and batch training

It is a common practice to use cross-validation when training machine learning and neural network models, especially when the training sample is small or there is class imbalance. However, because MNIST training set contains 60.000 images of evenly distributed classes, it was decided that cross-validation was not required. The data was loaded in PyTorch Dataloaders and batch learning was used instead, with a batch size of sixty-four, a number commonly used on MNIST classification tasks.

Image augmentations

Initially, both models were trained and tested on the clean versions of the MNIST dataset. The classification task proved to be very easy, as both algorithms scored very high accuracy, leaving little to no room for comparisons.

The answer to that problem came in the form of image augmentations. Two image augmentations were applied on the training set and two versions of the test set were created, a clean one and a noisy one. The first augmentation that was applied to each image, was a random rotation between zero and thirty degrees. The inspiration for this augmentation was slanted handwriting. Extra thought was given to this augmentation, as it needed to be something that made sense and could be encountered in a real-world scenario. The second augmentation that was applied was random pixel erasing and was chosen for the purpose of pushing the models' limits. A different kind of noise was chosen for the noisy set, as it had to be a noise that the models weren't trained to recognise. Gaussian Blur was applied to the digits, drawing inspiration from human Short-sightedness or myopia.

Custom functions

Two extra python files were created, one for each algorithm, each containing the model class and custom train and test functions. There are three reasons behind this implementation. First, it was done in order to keep the notebook as clean as possible, having to call only the functions to create, train, validate and test the models. Second, it made it easier to debug the code, since any problem could be traced back to the specific file and the specific function that was broken. Third, because these functions were custom made, they were made to accept hyperparameters as inputs flags. This made the hyperparameter experimentation process much easier, since the values that were chosen to be tested were being changed in their own cell and passed on to the models automatically, rather than going to each model's code block and hardcoding the values directly.

Confusion matrix

The best performing algorithm on this classification task was decided using confusion matrix as a performance metric. The reason behind this choice was that confusion matrix provided an easy way of comparing each algorithm's performance on each digit and discover which algorithm confused which digit the most and with which.