# Classifying and Comparing MNIST Handwritten Digits Using Multi-layer Perceptrons

Andres Ivan Montero Cassab, Ali El Abridi
*School of Computer Science and Communication Systems, EPFL*

***Abstract*— Multi-layer perceptrons networks combined with gradient-based algorithms compose the best techniques to classify complex decisions surfaces and high dimensional patterns such as handwritten characters. This report presents and compares different approaches to classify and compare MNIST handwritten digits using convolutional neural networks, weight sharing, auxiliary loss and different optimization techniques such as dropout, max-pooling, and batch-normalization. These optimizations combined with weight sharing have proven efficient in avoiding over-fitting, and correctly predicting the comparison of MNIST handwritten digits. Our best model has generated excellent results with an accuracy of 98% in the testing set.**

## I. INTRODUCTION

Image comparison and classification is a very challenging task, mainly due to the high dimensionality of the input, and large variability arising from distortion, alignment, or lightening of the images. Current approaches to image classification and comparison make use of powerful and state-of-the-art machine learning techniques including deep learning with multi-layer perceptron and convolutional neural network. Moreover, better and more powerful techniques have been developed to tackle more specific problems as image comparison and prediction of comparable outputs such as Siamese Networks. In this report, we investigate the performance of different models using different network structure such as weight sharing and optimization techniques such as dropout, max-pooling, and batch-normalization.

## II. DATA DESCRIPTION

The objective of this report is to implement a deep learning network such that, given as input a series of two input gray scale pictures of size $14 * 14$ pixels from the MNIST dataset that represent handwritten digits each in the range of [0,9] and in the form of tensors, it predicts whether the first digit is lesser or equal to the second. The data is generated from generate_pair_sets function provided in dlc _practical _prologue module, which generate 1000 sample of training and testing sample each. In our implementation, the training set is divided into 90% training set and 10% validation set to ensure that during the training phase, the models do not over-fit as shown in table I.

TABLE I
TRAINING AND TESTING DATA DESCRIPTION

| Name | Tensor dimension | Type | Content |
|---|---|---|---|
| train_input | 900*2*14*14 | float32 | Images |
| train_target | 900 | int64 | Class to predict {0, 1} |
| train_classes | 900*2 | int64 | Two digit classes{0,..,9} |
| validation_input | 100*2*14*14 | float32 | Images |
| validation_target | 100 | int64 | Class to predict {0, 1} |
| validation_classes | 100*2 | int64 | Two digit classes{0,..,9} |
| test_input | 1000*2*14*14 | float32 | Images |
| test_target | 1000 | int64 | Class to predict {0, 1} |
| test_classes | 1000*2 | int64 | Two digit classes{0,..,9} |

## III. ARCHITECTURES

For the project, five different neural network architectures are being evaluated for the comparison task. All architectures are based on convolutional neural networks trained with the training_set and tested with the validation_set in order to evaluate the different models. For the training phase of all the architectures, we are using the cross entropy as loss function, and Adam as the optimizer provided by the neural network module of Pytorch. The main difference of the architectures are explained in the following sections:

### A. SimpleConvNet Architecture

The SimpleConvNet architecture as shown in figure 1 serves as a base line model for our comparison. It takes as input the two channels input, followed by two convolutional layer with different number of channels, kernel size, and output the prediction 0 or 1 depending on whether the first digit is greater or equal to the second.

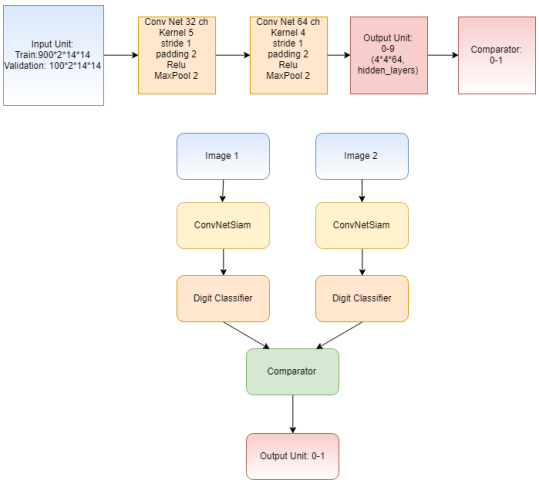


Fig. 1. SimpleConvNet Architecture

### B. ConvNetSiam_noWS_noDr_noBN Architecture

The ConvNetSiam_noWS_noDr_noBN architecture as shown in figure 2 uses a siamese network with no weight

sharing, dropout layers, or batch-normalization. This architecture has two parallel two convolutional layers with the same kernel and number of channels, and it uses auxiliary loss to train the model.

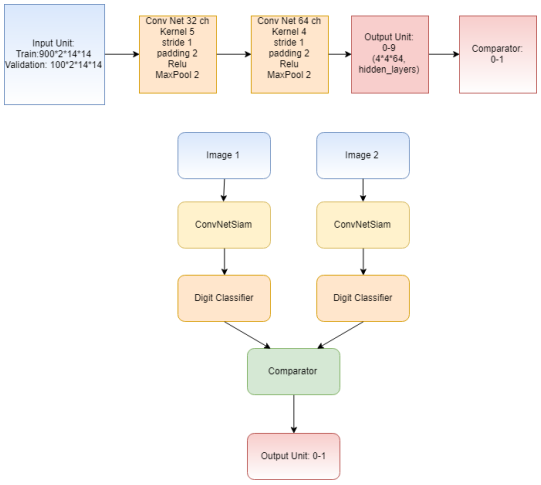


Fig. 2. ConvNetSiam_noWS_noDr_noBN Architecture

### C. *ConvNetSiam_WS_noDr_noBN Architecture*

The ConvNetSiam_WS_noDr_noBN architecture as shown in figure 3 uses a siamese network with weight sharing, but without dropout layers, or batch-normalization. This architecture has two parallel two convolutional layers sharing the same parameters (weights, and bias) with the same kernel and number of channels, and using auxiliary loss to train the model.

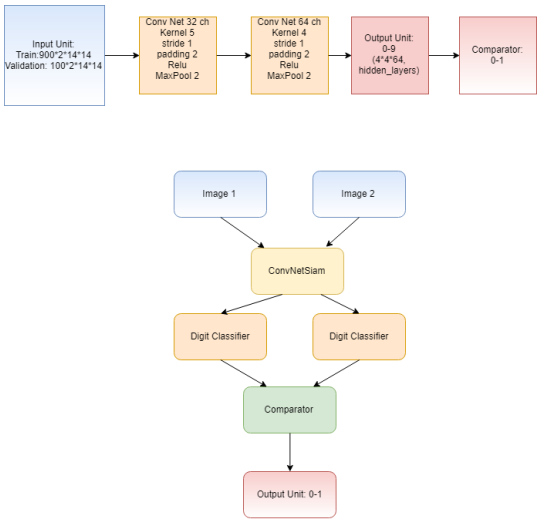


Fig. 3. ConvNetSiam_WS_noDr_noBN Architecture

### D. *ConvNetSiam_WS_Dr_BN Architecture*

The ConvNetSiam_WS_Dr_BN architecture as shown in figure 4 uses a Siamese network with weight sharing, dropout layers, and batch-normalization. This architecture has two parallel two convolutional layers sharing the same parameters (weights, and bias) with the same kernel and number of channels, and using auxiliary loss to train the model.

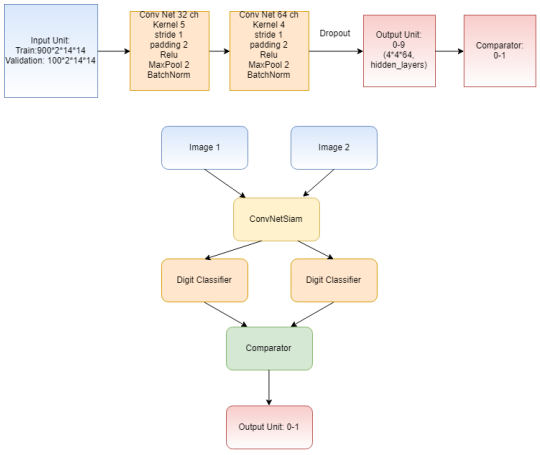


Fig. 4. ConvNetSiam_WS_Dr_BN Architecture

### E. *Advanced Convolutional Network Architecture*

The AdvancedConvNet architecture follows the description in the figure 5. The idea of this architecture is to first input the two images, classify the labels and then do the comparison between the two numbers.This is the reason of why the input is a tensor of $1800*1*14*14$ as the training_set and a tensor of $200 * 1 * 14 * 14$ as validation_set. For this purpose the structure consists of two convolutional layers, with batch-normalization and dropout on the last layer.

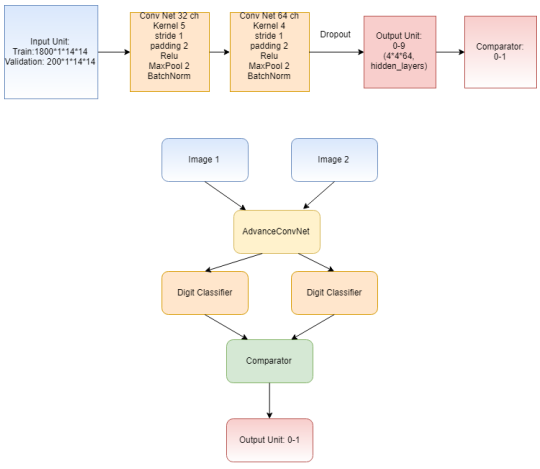


Fig. 5. Fifth Architecture

## IV. Model Training and Discussion

The training of the models resulted in the accuracy described in table II.

TABLE II

ACCURACY OF THE TRAINING, VALIDATION, AND TESTING SET

| **Architecture** | **Training%** | **Validation%** | **Testing%** |
|---|---|---|---|
| SimpleConvNet Architecture | 84.7 ±2.4% | 80.4 ±2.1% | 80.5 ±1.7% |
| ConvNetSiam_noWS_noDr_noBN | 90.9 ±1.3% | 86.3 ±1.4% | 87.4 ±1.2% |
| ConvNetSiam_WS_noDr_noBN | 97.6 ±0.9% | 96.2 ±1.3% | 92.8 ±1.4% |
| ConvNetSiam_WS_Dr_BN | 94.3 ±1.1% | 92.1 ±1.4% | 94.6 ±1.3% |
| Advanced_Convolutional_Network | 98.4 ±1.2% | 97.2 ±1.7% | 96.9 ±1.9% |

As we can observe from table II, the best model to compare MNIST handwritten digits is the *Advanced_Convolutional_Network* with an accuracy of 96.9% on the testing set. Even though the Siamese network is better in comparing the digits, The *Advanced_Convolutional_Network* takes full advantage of the combined data set to improve its training along side with the use of dropout and batch-normalization to avoid overfitting, and finally use these results to compare the digits and output the results.

## V. CONCLUSIONS

The five different architectures developed: SimpleConvNet, ConvNetSiam_noWS_noDr_noBN, ConvNetSiam_WS_noDr_noBN, ConvNetSiam_WS_Dr_BN and AdvancedConvNet show the differences and advantages of using weight sharing, auxiliary loss, dropout and batch normalization. For example, the accuracy difference between SimpleConvNet and ConvNetSiam_WS_Dr_BN is of 14% which is mainly due to the use of methods such as dropout, batch normalization and weight sharing. On the other hand, SimpleConvNet focuses on directly predicting the comparison 0-1 between the digits (80% accuracy), while first predicting the class of the digits, and then using it to compare the digits shows a much better approach, therefore increasing the performance as in AdvancedConvNet architecture (98% accuracy), and thus, outperforming the Siamese network structure .