Project 1
Classification, Weight Sharing, Auxiliary Loss

Nicolas Baldwin[1], Ana Lucia Carrizo[2], Christopher Julien Stocker[3]
EE-559 Deep Learning, Swiss Federal Institute of Technology Lausanne

I. Objective

THE objective of *Project 1* is to test different architectures to compare two digits visible in a two-channel image. The model predicts, for each pair, if the first digit is less or equal to the second. The different architectures highlight the utility of weight sharing and auxiliary loss functions on the performance.

II. Project Structure

Data Set: The *MNIST* hand written data set consists of a series of $2 \times 14 \times 14$ tensors, corresponding to pairs of grayscale images 14×14 each.

- **Batch Normalization:** During training, the distribution of each layer's input changes causing the learning rates to slow down. One can reduce this problem by performing normalization for each training mini-batch. This normalization shifts and rescales according to the mean and variance estimated on the batch. This allows for higher learning rates and less sensitivity to data initialization [1].
- **Spatial Dropout:** During the forward pass, channels are removed at random with a probability rate of 50%. This prevents co-adaptation by making the presence of other hidden units unreliable and avoiding over-fitting [2] .
- **Max Pooling:** This pooling method chooses the max values over non-overlapping blocks. This works particularly well with MNIST data since the background is black and we are interested only in the lighter pixels of the image.

III. Different Networks Architecture

Five architectural variations were considered to perform the objective. The first four models have the same type of layers except that they vary whether the model contains or not weight sharing or auxiliary loss. The last model (Model 5) is an attempt to achieve higher accuracy on the data set. It was designed after observing the effects of weight sharing and auxiliary loss on the first four models. Fig. 1 illustrates the different architectures.

1. **No Weight Sharing & No Auxiliary Loss**
   This model uses the architecture of Fig.1. However, there is no weight sharing and no use of auxiliary loss. Therefore digit A and B run in parallel through separate but identical layers. First, they go through the convolutional layers, then through 2 fully connected layers (layers where its written models 1-4 in Fig.1.). Finally, the 2 digits are joined on the final layer (Linear(20 to 2)). This architecture achieved the lowest average test accuracy of 0.7352 and the highest average test loss of 0.5543.

2. **Weight Sharing & No Auxiliary Loss**
   This model uses the architecture of Fig.1. It uses weight sharing but doesn't use an auxiliary loss. Therefore digit A and B run through the same layers. First, they go through the convolutional layers, then through 2 fully connected layers (layers where its written models 1-4 in Fig.1.). Finally, they go through a final layer (Linear(20 to 2)). The *Siamese network*, lead to a significant reduction in the memory and speed up training. Our parameters were reduced by half with no apparent effect on the results. Test accuracy was 0.8004 and test loss of 0.4816.

3. **No Weight Sharing & Auxiliary Loss**
   This model uses the architecture of Fig.1. It uses an auxiliary loss but does not use weight sharing. Therefore digit A and B run in parallel through separate but identical layers. As in Model 1, this model goes through the convolutional layers, and through 2 fully connected layers. However, at this point, the auxiliary loss is computed for both digits comparing the output at Linear(20,10) with its target digit (using cross-entropy loss). Finally, the 2 digits are joined on the final layer (Linear(20 to 2)).Auxiliary loss alone made a significant improvement on accuracy but at the expense of high number of parameters (66'934).This architecture achieved a test accuracy of 0.7766 and a test loss of 0.5120.

4. **Weight Sharing & Auxiliary Loss**
   This model merges the best improvements from the previous models. It uses weight sharing and auxiliary loss. Therefore digit A and B run through the same layers sharing weights as explained before. At the end of the 2nd layer the auxiliary loss is taken for both digits comparing the output at Linear(20,10) with its target digit

[1]Msc. Data Science, Swiss Federal Institute of Technology Lausanne, e-mail: `nicolas.baldwin@epfl.ch`.
[2]Msc. Data Science, Swiss Federal Institute of Technology Lausanne, e-mail: `ana.carrizodelgado@epfl.ch`.
[3]Msc. Robotics, Swiss Federal Institute of Technology Lausanne, e-mail: `christopher.stockersalas@epfl.ch`.

(using cross-entropy loss). Lastly, the 2 digits are joined on the final layer (Linear(20 to 2)). This change significantly improved performance by obtaining a test accuracy of 0.7998 and a test loss of 0.4763.

5. **Max allowed parameters**
   This model was our last attempt to improve the accuracy by taking advantage of the large parameter space we availability we obtained by using weight sharing. This model uses the same convolutional layers but passes through 3 fully connected layers as highlighted in Fig.1. With more layers available the model can better approximate the ideal function as stated by the *Universal Approximation Theorem* [3]. Finally, the 2 digits are joined on the final layer (Linear(20 to 2)). These improved our performance giving us a test accuracy of 0.8429 and the lowest test loss of 0.3781.

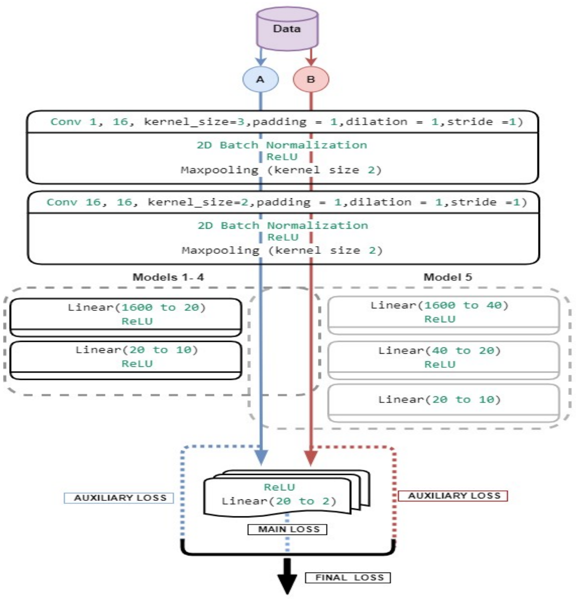


Fig. 1: Structure of models

The five models are compared under the same initial conditions. All four models are tested and trained with 1000 pairs of images and all hyper parameters (learning rates, batch size, momentum, epochs) were hyper tuned through cross-validation.

- **Parameters**: All the models respect the 70,000 parameters limit (see Table I).
- **Epoch**: All models run on 25 epochs
- **Mini-batches**: Varies on model (see Table I)
- **Optimizer**: All models use SGD

IV. Results

1. **Weight Sharing:** As we can see in the plots of Fig.2 - Fig.5, when using weight sharing, even though there are half of parameters ,the accuracy increases and the loss decreases. When thinking of the application of our model, this makes sense since we compare 2 digits that share a symmetrical model. There is no reason that the digits should go through different weights. In fact our model shouldn't be identifying one digit in a different way than the other.

2. **Auxiliary Loss:** As we can see in the plots of Fig.2 - Fig.5, when adding an auxiliary loss, the accuracy increases and the loss decreases. In fact, the auxiliary loss encourages our model to recognize the digits that are fed in and therefore helps it extract the relevant features to accomplish this task. Additionally, auxiliary loss helps update the parameters that are deeper in our model as the total loss is higher and the back propagation for the auxiliary losses does not start from the output of our model.

V. Results

The results of all five models are compared in the Table I. Fig.4 and Fig.5 plot the different accuracy's of the models during testing and training. Conversely Fig.2 and Fig.3 plot the losses of the models during testing and training.

These results were key in building the best performing model. Each new model built on the improvements from the previous. Model 4 (Weight Sharing and Auxiliary Loss) performed the best in accuracy among the first four models, despite having half the parameters of Model 1 and Model 3 (35'0000 parameters).

The relatively small number of parameters of the latter led us to build that a new model (Model 5) with an additional fully connected hidden layer. This model could leverage the extra parameters and reach a higher accuracy. Model 5 doubles the number of parameters (66334) whilst still using weight sharing and auxiliary loss. This model brought an improvement on the error rate by +3.83 and reduced the test loss by −9.45. Model 5 shows us that by adding parameters to our model and using weight sharing and auxiliary losses we can significantly increase the accuracy. Adding parameters must be a take careful consideration as they increase computational power and is prone to overfitting. We did not see that to be the case in our model as the accuracy difference in the test and training sets were not significant.

Table I: Performance of Models

	M1	**M2**	**M3**	**M4**	**M5**
Nb of Parameters	66934	33504	66934	33504	66344
Avg. Train Accuracy	76.43%	81.50%	79.32%	81.54%	86.06%
Std. Train Accuracy	0.000259	0.000534	0.000677	0.000484	0.000314
Avg. Test Accuracy	0.7351	0.8004	0.776600	0.799800	0.8429
Std. Test Accuracy	0.000635	0.000615	0.000335	0.000271	0.000388
Avg. Train Loss	0.558323	0.469556	0.501004	0.464856	0.353225
Std. Train Loss	0.000582	0.001317	0.000721	0.001069	0.000854
Avg. Test Loss	0.57414	0.481687	0.512044	0.476372	0.378196
Std. Test Loss	0.000302	0.000735	0.000443	0.000521	0.000745

Table II: Optimized Parameters

	M1	M2	M3	M4	M5
Learning Rate	0.1	0.1	0.01	0.01	0.01
Batch Size	50	50	50	25	25
Epochs	25	25	25	25	25
Momentum	0.2	0.0	0.8	0.7	0.5

Fig. 2: Train Losses Plot

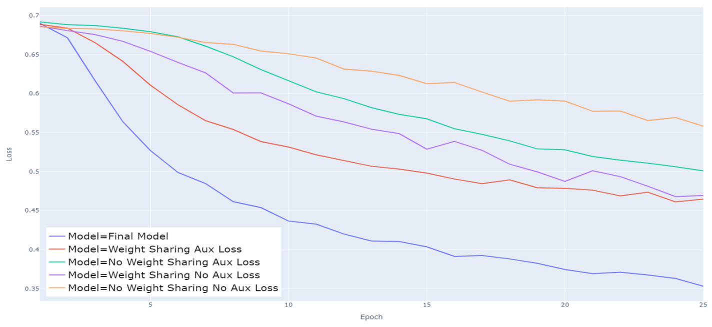


Fig. 3: Test Losses Plot

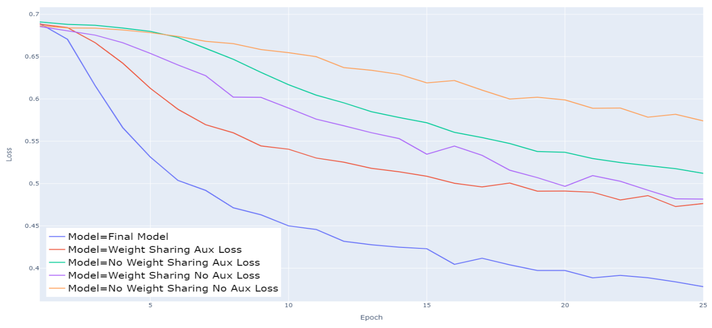


Fig. 4: Train Accuracy Plot

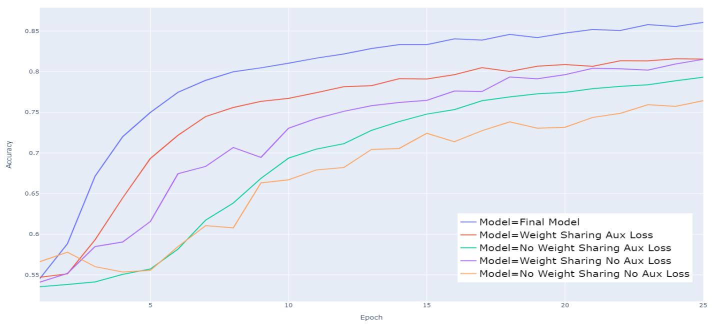
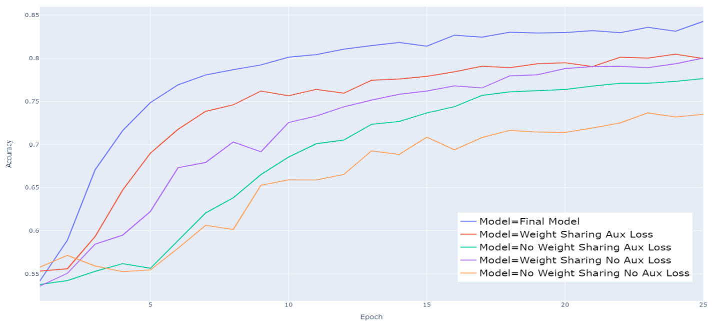


Fig. 5: Test Accuracy Plot



References

[1] S. Ioe and C. Szegedy., *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, In International Conference on Machine Learning (ICML), 2015.

[2] A. Krizhevsky I. Sutskever N. Srivastava, G. Hinton and R. Salakhutdinov., *Dropout: A simple way to prevent neural networks from overfitting*, Journal of Machine Learning Research (JMLR), 2014.

[3] A. Bordes X. Glorot and Y. Bengio., *Deep sparse rectifier neural networks*, In International Conference on Artificial Intelligence and Statistics (AISTATS), 2011.