PROJECT 1 – CLASSIFICATION, WEIGHT SHARING, AUXILIARY LOSSES

Oganes Manasian
EPFL, Switzerland

Davit Martirosyan
EPFL, Switzerland

Igor Abramov
EPFL, Switzerland

I. Introduction

In this report we present the different architectures we developed to compare two digits visible in a two-channel image. Section II gives a thorough explanation about all the different models we have come up with. The next section provides the training tips used by us. Afterwards, in Section IV, we tell about our experiments, explaining the motivation behind the creation of each of our models. Finally, in Section V we describe the achieved results.

II. Models

To solve the given binary classification problem we developed several architectures. Let's look at them in turn.

Base model: As a baseline solution we used the well-known LeNet-5 [1] architecture with a few modifications imposed by us. Following modern trends in neural network architectures, we reduced convolution kernel sizes from 5x5 to 3x3 and increased number of filters to 16 and 32 in first and second convolution layers respectively. In addition, to fight against overfitting we added dropout layers [2]. As a result we obtained a model with approximately $45,000$ parameters. Cross-entropy loss was used as loss function for model training.

Siamese models: One special feature that our data possesses is that both of the channels are downsampled MNIST digits [3]. Hence, we can use the same model with the same weights to predict digits from both channels, i.e. we can make use of what is referred as Siamese Network. The siamese neural network that we use is the same as our baseline model except that for the last layer the network outputs a tensor of size 10 instead of 2. After applying the network to each of the channels separately, we propose several methods to combine the two outputs and do $\{0,1\}$ class prediction. One of our approaches concatenates the digit predictions on each channel and gives the resulting tensor as an input to a fully connected layer which then predicts the class label (*Siamese v1*). Another approach does exactly the same except that it concatenates the digit encodings, that is it concatenates the outputs of the first fully connected layer instead of the second one (*Siamese v2*). Besides, we have also tested a network that subtracts the two encodings instead of concatenating them before feeding into the dense layer responsible for prediction (*Siamese v3*). Note that all of these models have the same loss function, namely cross-entropy. It is also worth mentioning that *Siamese v1, v2,* and *v3* have almost the same number of parameters as the *Base model*.

Siamese models with auxiliary loss: In our next approach we went further and added auxiliary losses on digit predictions. This can be seen as extra regularization which forces the network to learn to see digits in each channel of the input, which is logical given the knowledge that the class label is just the comparison of the two digits. Here we do 3 predictions: $\{0,1\}$ class prediction, digit prediction in channel 1 and digit prediction in channel 2. The final loss is the sum of the cross-entropy losses coming from each prediction. It is important to note that such siamese architecture give us the extra benefit of having double the data size for digit prediction since each data sample consists of two digits.

Siamese model for digit prediction: In this model we get rid of the extra layers which use the digit predictions to do $\{0,1\}$ classification. Instead, we do digit prediction for each channel straight away and with a use of a simple arithmetical comparison obtain class labels. For training our model we use as loss function the sum of cross-entropy losses of each digit prediction[1]. We refer to this model as *Siamese v4*.

Enhanced siamese models: Besides the models discussed so far we developed two more architectures which use additional layers upon the shared ones (siamese part). The first of them (*Siamese v5*) applies softmax to digit predictions, concatenates the results and gives it as an input to a fully connected layer. This facilitates the learning process and makes it easier for the network to see digits in the output of siamese part of model and therefore to see connection between class label and predicted digits. Another way to do this is to not use softmax and give the model (*Siamese v6*) freedom to learn it by itself (or learn something even more useful). For that we just concatenated the digit predictions and gave it to a fully connected layer with relu activation. The latter is then fed into the last fully connected layer which does the $\{0,1\}$ classification. As a loss function, for both of the models, we used sum of the cross-entropy losses coming from each prediction, namely prediction of class label and predictions of digits.

Comparison and analysis of all the models with the motivation behind creating them is presented in section IV.

[1]NOTE: This model is only used for analysis purposes

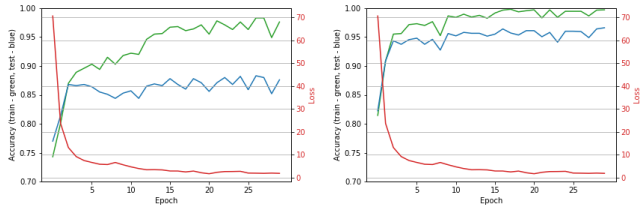
III. Training Tips

Grid Search: A grid search was done to find the best learning rate and regularization parameter pair. Learning rate was chosen from the set $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$, and regularization parameter from $\{0.25, 0.1, 0.05, 0.01\}$,

Early stopping: To prevent overfitting, we used the early stopping technique. In simple terms, we kept the model with the best accuracy on test data during training and returned it as the result of training.

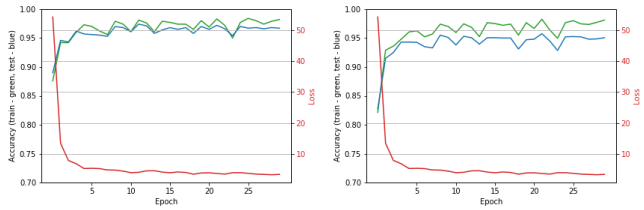
IV. Experiments

Train and test accuracy analysis: In all *Siamese v1, v2,* and *v3* models trained with an auxiliary loss, we found the following behaviour: models overfitted {0,1} class prediction but did quite well (i.e. no overfitting was observed) on digit predictions. The results are provided for *Siamese v2* as an example in Figure 1 (a), (b). Since class prediction is done using digit predictions as an input, it was clear that the cause of overfitting originated from the last layer.



(a) Class prediction accuracy of *Siamese v2*

(b) Digit prediction accuracy of *Siamese v2*



(c) Class prediction accuracy of *Siamese v4*

(d) Digit prediction accuracy of *Siamese v4*

Figure 1: Train and test accuracy curves of models *Siamese v2* and *Siamese v4* with auxiliary loss

To prove our conjecture, we constructed the *Siamese v4* where we removed those layers and put simple arithmetical comparison to get class label from digit predictions. As we can see from Figure 1 (c), (d) *Siamese v4* does not have the problem of overfitting in {0,1} classification.

Performance analysis of Siamese v1, v2 and v3: During our experiments, we observed that *Siamese v4* greatly outperforms *Siamese v1, v2,* and *v3* in terms of accuracy, although those models differ only in the last layer - a layer which aims to predict class labels from digit predictions. To investigate this phenomenon, we trained *Siamese v1* with an auxiliary loss and took its last layer. Afterwards, we constructed an input for this layer - a tensor of dimension $(1, 20)$ whose first 10 values encode the first digit and the next 10 values encode the second digit. The input was constructed in the following way: for both parts of the tensor, we took an index $\in [0, 9]$ which would represent our digit and put a relatively large value at that index compared to the values at other indices, thus giving a signal about the encoded digit. In this way, we encoded the output of siamese part of the network. In total we built 100 such input tensors - a single example for each possible combination. Since for each tensor we knew which digits it represented, we could evaluate network predictions.

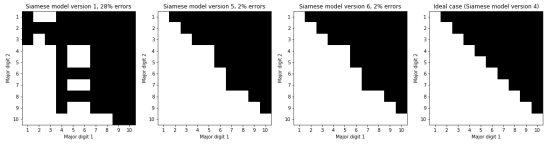


Figure 2: Investigation of the last layer performance

Let's now take a look at Figure 2, where each matrix cell represents one of our 100 input tensors. White cells correspond to a model outputting class 1, meaning that the model thinks that the first digit is less than or equal to the second digit. Black cells correspond to class 0. Ideally, for each model, the matrix plot should be full black (class 0) above the main diagonal and full white (class 1) otherwise. Looking at the leftmost matrix which is the result for *Siamese v1*, we can see that it is far from being perfect.

To improve the results we tried to help the model detect the digits by applying softmax activation and so we came up with *Siamese v5*. Besides, we made another version (*Siamese v6*), in which we added one extra fully connected layer compared to *Siamese v1* helping the model learn to do something equivalent to softmax on its own. The results for *Siamese v5* and *v6* are presented in Figure 2. As we can see, our changes led to almost ideal results.

V. Results

Table I summarizes the results of different model architectures on the test set. As we can see, *Base* model is the worst in terms of test accuracy[2]. In addition, as one can observe, we have a great improvement with the addition of auxiliary loss - accuracy increased from 86.4% to 95.5%.

Table I: Mean test accuracies of the *Base* model, the best *Siamese* model with and without auxiliary loss

Model	Mean Accuracy (%)
Base model	82.4
Siamese model	86.4
Siamese model with auxiliary loss	95.5

[2]Please note that all performance results are estimated through 20 rounds

In Table II, we provide the test accuracy scores for each of our siamese models trained with an auxiliary loss. Models are sorted in descending order in terms of accuracy.

Table II: Mean test accuracies of different versions of Siamese network with auxiliary loss

Siamese model	**Model description**	**Mean accuracy**
v4	Predicts class simply by comparing digit predictions (no use of any trainable layers)	97.8
v5	Predicts class from digit predictions followed by softmax layer	95.5
v6	Predicts class from digit predictions using 2 fully connected layers	93.7
v1	Predicts class from digit predictions using 1 fully connected layer	89.3
v3	Predicts class from subtracted encodings of digits using 1 fully connected layer	88.6
v2	Predicts class from concatenated encodings of digits using 1 fully connected layer	87.7

From the last two tables shown above we can derive the following observations:

- Weight sharing improves model accuracy
- Adding auxiliary losses increases the accuracy even further
- Our best model, namely *Siamese v5*, achieved accuracy close to 98%
- The more we remove knowledge based operations on the data (e.g. comparison to predict class from digit predictions, applying softmax to digit predictions) the worse accuracy we get

Figure 3 illustrates the mean and std of test accuracy as well as loss for our best model - *Siemese v5* trained with an auxiliary loss.

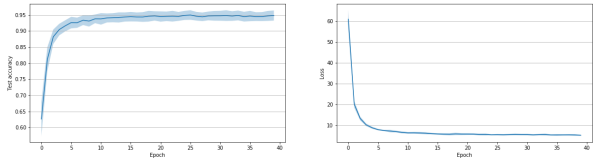


Figure 3: Mean and standard deviation of test accuracy and loss for the *Siamese v5* model with auxiliary loss

In addition, in Table III, we provide the standard deviation of accuracy values (also mean accuracy scores once again) for two of our best models.

Table III: Performance estimates of siamese models with auxiliary losses of version 5 and 6

Siamese model	**Mean accuracy (%)**	**Standard deviation**
v5	95.5	0.011
v6	93.7	0.008

REFERENCES

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[3] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/