EE-559 Deep Learning Project 1 Report

Devrim Celik, Xavier Oliva i Jürgens, Nina Mainusch
M.Sc. Data Science, EPFL, Switzerland

***Abstract*—In this work, we propose and test different network architectures with the task to compare two digits in a two-channel image from the `MNIST` handwritten digits dataset. We demonstrate and explain the superiority of convolutional neural networks over normal multilayer perceptrons for this task and show the impact of an auxiliary loss and weight sharing by implementing networks with a siamese structure. We encounter that these siamese networks, being the architecture best adapted to the learning task, perform best. Using auxiliary losses, the highest test performance of 97.8% is achieved by the convolutional siamese residual network (ConvSiameseResNet). Without taking into account any auxiliary loss, again the convolutional siamese network (ConvSiameseNetTargets) performed best with a test performance of 82.3%.**

I. Introduction

For the first project, the task is to implement different deep networks that predict whether the first digit is less or equal to the second for pairs of gray-scale digit image data. The aim of the project is to assess the quality of different architectures and to investigate how weight sharing, auxiliary losses or other tools can improve the performance of the deep neural network. For educational purposes, we will only rely on the `PyTorch` framework.
Given that we are dealing with image data, a natural choice of network is a convolutional one. Networks with convolutional layers reach significantly better training and test performances than those without, and converge faster due to their explicit inductive bias. This explicit inductive bias is their independence to translation and the underlying assumption that local feature detectors are useful. We note that a convolutional layer is a special case of a dense layer where many neurons have the same weights and where many weights are zero, except in a small neighbourhood.
The nature of the task furthermore ensues the usefulness of a siamese network architecture, since we have to process two images from the same source. A siamese network can process the two input images independently with the same network layers, thus doubling the amount of training data and implicitly employing weight sharing. An auxiliary loss can be used to first train the network for classifying the images directly and then the general loss is calculated for the actual task of digit comparison.
Regarding the structure of our project, we first set up and test a normal multilayer perceptron, but given the presumed superiority of convolutional neural networks we then focus on them and develop several variants. In the first part, we describe all our chosen models and their specifications. In the second part, we evaluate their performance. In the last part, we summarize and discuss our results.

A. Data

The inputs for the network are series of $2 \times 14 \times 14$ tensors, corresponding to pairs of 14×14 gray-scale images from the `MNIST` database. The training and test set contain 1,000 pairs each. Given the nature of the task, the targets that we predict are binary, $T \in \{0, 1\}$. Additionally, we know the classes of the two digits, $C \in \{0, ..., 9\}$, of which we will take advantage when constructing our auxiliary loss models. Both data and weight initialization are randomized.

B. Weight sharing & Auxiliary loss

Knowing the architecture of an artificial neural network, we can see that parameter sharing reduces the training time due to the reduction of the number of weight updates that have to take place during back-propagation. Since it makes our model less flexible, it can also be seen as a type of regularization method that mitigates overfitting. As stated above, weight sharing can be implemented in several ways:

- a **convolutional layer** is a special case of a dense layer where many neurons have the same weight
- a network using **dropout** can be interpreted as an ensemble of 2^N models with heavy weight sharing [1]
- siamese networks employ weight sharing, since the two different inputs are processed by the same network

To test the influence of weight sharing, we will build convolutional and siamese networks, and use dropout. Selected siamese networks will simultaneously be implementing an auxiliary loss, since in addition to the final target loss that compares the digits, they use the class labels C for an auxiliary loss to first train the network on the digit itself.

II. Models & Methods

A. Methods

Our loss of choice is the `CrossEntropyLoss` and the optimizer we use is `ADAM`, with the default learning rate of 0.001. For all networks, the `SELU` activation function is employed. The data is split into 1,000 training and 1,000 test data points. In order to estimate the standard deviation of our test error, we set up 8-fold cross-validation for the training data, dividing it thereby into training and validation data. We calculate a validation accuracy for every fold, and based on these 8 validation accuracies, we estimate the test accuracy mean and its standard deviation. We use early stopping to avoid over-fitting. We set `patience = 10`, which means that training is interrupted if the current validation loss is not lower than the lowest one for 5 consecutive epochs. At the very end, we train our models based on the best validation accuracy and calculate a final test accuracy.

B. Models

To get a first impression of our models, we list all of them and their corresponding number of parameters in Table I.

TABLE I
NUMBER OF MODEL PARAMETERS

MLP	ConvResNet	ConvSiameseNetClasses
27,490	41,477	30,006
ConvNet	ConvSiameseResNet	ConvSiameseNetTargets
57,854	41,590	30,048
MNISTResNet	MNISTResSiameseNet	
11,174,402	11,174,402	

MLP The first model we built was a standard multilayer perceptron (MLP) with three linear layers, which is supposed to act as a baseline. This network makes no use of the class labels C, but tries to learn the task directly and only outputs the target T. It is a well known fact that the distribution of each layer's input changes during training. Generally this slows down training and requires better fine-tuning of the hyperparameters. Batch normalization is a method to stabilize and speed up training, by re-centering and re-scaling the layer's inputs [2]. This MLP applies batch normalization after the first and second layer.

ConvNet The next model is a standard convolutional neural network with 3 convolutional and 2 fully connected linear layers. For the first version, we used max-pooling (with a kernel size and stride of 2) and dropout. The advantage of dropout is, that a network using it can be interpreted as an ensemble of 2^N models with heavy weight sharing[1]. Building up on this architecture to get to our final basic ConvNet model and knowing that activations benefit the most from it, we added batch normalization after the first and second non-linear activation, which also made dropout redundant.

ConvResNet We know that deeper neural networks are more difficult to train, due to the problem of vanishing gradients. To mitigate this, He et al. [3] proposed to use residual networks. Residual networks use skip connections between building blocks, where these these blocks consist of a convolutional layer (say this is layer $n + 1$) followed by a `SELU` activation and batch normalization and another convolutional layer (say this is layer $n + 2$). Then layer n gets added to layer $n + 2$, giving the network the chance to skip the two layers in between which makes it learn the residual, i.e. the difference between the value before the block and the value after it. After the residual blocks, we apply average-pooling with a kernel size of 8. Residual networks are known to be stable, possibly because they implement ensemble learning with their skip connections [4].

ConvSiameseNetClasses We designed a siamese network [5] with 2 convolutional and 2 fully connected linear layers, using max-pooling and batch normalisation. It is called siamese, because it splits the input into its two components, image 1 and image 2, and processes them both separately in the same network. This time we do use the class labels C and an auxiliary loss to train the network on them and then calculate the actual loss and target for the digit comparison.

ConvSiameseNetTargets Similarly to the ConvSiameseNetClasses network, we designed a siamese network that does not take the class labels into account, but only learns on the targets $\{0, 1\}$, again with 2 convolutional and 2 fully connected linear layers, using max-pooling and batch normalisation. The inputs are processed separately, but in the end we stack them back on top of each other to get the target prediction.

ConvSiameseResNet The ConvSiameseResNet is similar to the ConvSiameseNetClasses network, but as all our residual networks it has additionally 10 residual blocks.

MNISTResNet Using deeper architectures has been a key in improving performance in tasks such as image classification[6]. Furthermore the concept of transfer learning (i.e. the reuse of a pre-trained model on a new problem) encouraged us to approach the problem by using the pre-trained deep model `ResNet18`[3], which is an 18-layer residual network, where we adjusted the input and output layers. In order to stabilize learning, this pre-trained network employs batch normalization, where the resulting gradient explosion is buffered by the skip connections of the residual network structure.

MNISTResSiameseNet Lastly, we implemented the pre-trained `ResNet18` as a siamese network, where we take the class labels with an auxiliary loss into account

III. PERFORMANCE EVALUATION

In Fig. 1, we can see that our models do learn over time, where all of them achieve a training accuracy of above 90%. The two models using the auxiliary loss for the class labels get test accuracies of up to 95%, where the two models that only use the target information display a test accuracy of up to 80%.

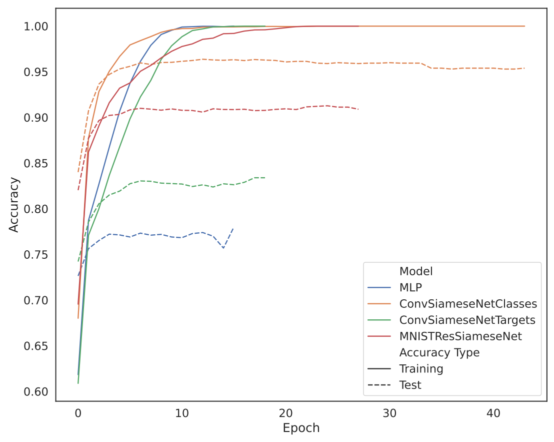

Fig. 1. Training and test accuracies of selected models using early stopping.

A. Model comparison

As stated above, we implemented 8-fold cross validation to get an estimate of the test accuracies of each model. In Fig.

2, we summarize our results for each model. The ConvSiameseNetClasses performs best with approx. 95% validation accuracy and only a small standard deviation around its mean, followed by the other two networks using an auxiliary loss with approx. 90% to 94% accuracy. The other networks that do not use the class information show a validation accuracy of about 75 to 80%, with the ConvResNet being the worst network.

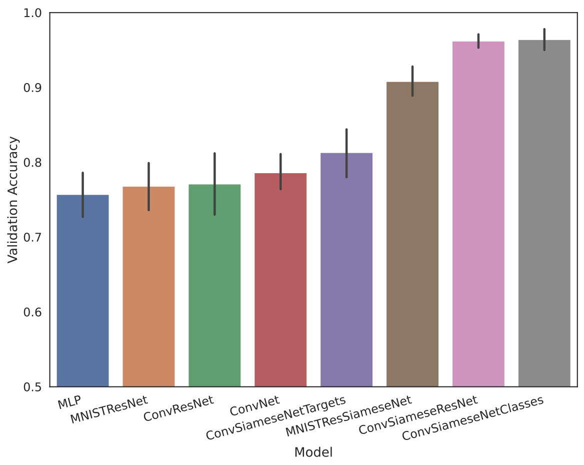

Fig. 2. Mean validation accuracies of our models, where the error bars display the standard deviations calculated with the 8 estimates from our 8-fold cross validation.

Finally we tested all our models on the separate test dataset. The resulting test accuracies are summarized in Table II. We can see that the models implementing the class information yield better test accuracies, where the best test accuracy of 97.8% is achieved by the ConvSiameseResNet network. For the models only relying on the target values, the best model is the ConvSiameseNetTargets network with a test accuracy of 82.3%.

TABLE II
TEST ACCURACIES OF OUR MODELS

MLP 77.4%	ConvResNet 77.0%	ConvSiameseNetClasses 96.2%
ConvNet 78.3%	ConvSiameseResNet **97.8%**	ConvSiameseNetTargets **82.3%**
MNISTResNet 73.6%	MNISTResSiameseNet 90.4%	

IV. DISCUSSION

A. Weight sharing & Auxiliary loss

Inspecting our model results, we can state that weight sharing greatly improves the test performance of our models. All the convolutional networks and especially the Siamese networks, implementing weight sharing inherently in their architecture, perform better or equally good as the MLP. Dropout on the other hand seemed to have less effect on the performance than batch normalization. Regarding the auxiliary loss we can generally state that without this additional loss, a model was not able to exceed a test accuracy of 90%. Only the siamese networks that do use the additional information of the digit's class label show a test performance of more than 90%. The digit recognition task is much more easy for a model to learn than the comparison of two digits, since the former task is a requirement for the second one. It is interesting to note that the best model using only target values reuses the siamese structure.
Our two pre-trained `ResNet18` models have the advantage of being easy to train and resource efficient. In Fig. 1 we can see that because the MNISTResSiameseNet is pre-trained, it quickly gets to a high training accuracy with a decent test performance of about 90%.

B. Further models

In addition to the eight models that are presented above, we created a convolutional network with transpose layers (ConvTransposeNet). Contrary to the standard convolution we have encountered so far, which applies a linear filter at every location of a tensor, the transposed convolution computes at every location a linear combination of kernels thereby increasing the signal size, which is usually applied in a generative context. Surprisingly, the ConvTransposeNet performed well in training (90% validation accuracy with $\sigma \sim 3\%$), but poorly when tested (79.9% test accuracy), which is why we did not list it alongside the other models.

V. SUMMARY

In this work we examined the effect different deep neural network architectures have on the task of classifying and comparing digit image data from the `MNIST` dataset. We set up different architectures, including a standard MLP, convolutional, siamese and residual networks and added different refinements, i.e. dropout, batch normalization and pooling to them. The best model measured by its test accuracy of 97.8% was the ConvSiameseResNet network. Generally, the siamese networks turned out to have the most suitable architecture for the learning task, since they treat the images equally and thereby double the amount of available training data. Without using the additional classes labels, our best model was the ConvSiameseNetTargets network with a test accuracy of 82.3%.

REFERENCES

[1] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1319–1327.
[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[4] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *arXiv preprint arXiv:1605.06431*, 2016.
[5] D. Chicco, "Siamese neural networks: An overview," *Artificial Neural Networks*, pp. 73–94, 2021.
[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.