Investigating the impact of classification, weight sharing and auxiliary losses in deep learning architectures

Project 1 - EE-559

Jodok Vieli[1], Puck van Gerwen[2], Felix Hoppe[3]

Abstract

Given two images of digits, we investigated different model architectures to predict whether the digit in one image was greater than or equal to the other. Our best model, `Digits`, uses a siamese LeNet-like architecture and auxiliary losses to predict the digit in each image, which are arithmetically compared to predict the target class with 98.29% accuracy. Our `PreTrain` model achieves nearly the same target accuracy (97.45%) by pretraining a network on the arithmetic comparison task before passing it to the same siamese architecture, illustrating the importance of human domain knowledge in building accurate models.

1 Introduction

The aim of this project is to compare digits visible in two 14x14 (grayscale) images. If the digit in the second image is greater than or equal to the digit in the first, a class label of 1 is assigned. Otherwise, a class label of 0 is assigned. An example is illustrated in figure 1. Different model architectures are investigated to predict these class labels, in particular, using weight sharing and auxiliary losses.

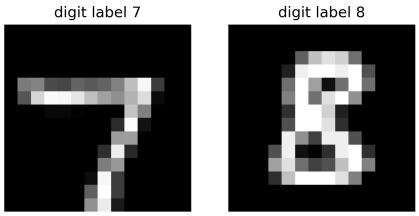

Figure 1: For this pair of images, the class label is 1, as the second digit (8) is greater than or equal to the first (7).

Train and test sets are 1,000 pairs each, where the train set is again split into a train and validation set (80-20 split). In all cases models are trained using the Adam optimizer [1] with a learning rate of 0.001, with minibatches of size 32 and 25 training epochs. All models have around 70,000 parameters and are trained in less than 3 seconds on a single CPU (Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz).

2 Results & Discussion

An overview of the implemented models and their target accuracy is summarised in table 1. The models are listed in the order they were developed.

Model name	Target accuracy (std)
`FCBaseline`	79.71 (1.36)
`FCAux`	86.23 (1.87)
`ConvNet`	86.73 (0.74)
`ConvAux`	88.04 (1.97)
`Digits`	98.26 (0.34)
`ConvTail`	91.02 (0.51)
`ConvSoftMax`	94.64 (3.12)
`ConvArgMax`	97.23 (0.67)
`PreTrain`	97.45 (0.53)
`PreTrainRandom`	97.27 (0.77)

Table 1: Target accuracy and standard deviation (std) of our models, averaged over 10 rounds of random data and weight reinitialization.

We begin with a fully connected network as a baseline (`FCBaseline`), using a sequence of 7 fully connected layers to take the flattened input images and return two output nodes corresponding to class predictions 0 or 1. The loss

[1]SCIPER: 336284
[2]SCIPER: 283530
[3]SCIPER: 276062

and accuracy are computed on the class predictions only. This is illustrated in figure 2a. This baseline model achieved an accuracy of 79.71%. Modifying the loss function to include *auxiliary* loss terms for the digit prediction, the loss has the form $L = w\,L_{\text{digit}} + L_{\text{target}}$, where w is a parameter weighting the auxiliary loss, L_{digit} is the average loss of the individual digit classification, and L_{target} is the target loss. Using this combined loss, the accuracy of the fully connected model `FCAux` increases to 86.23%. The performance of the fully connected models over training epochs is shown in figure 3a.

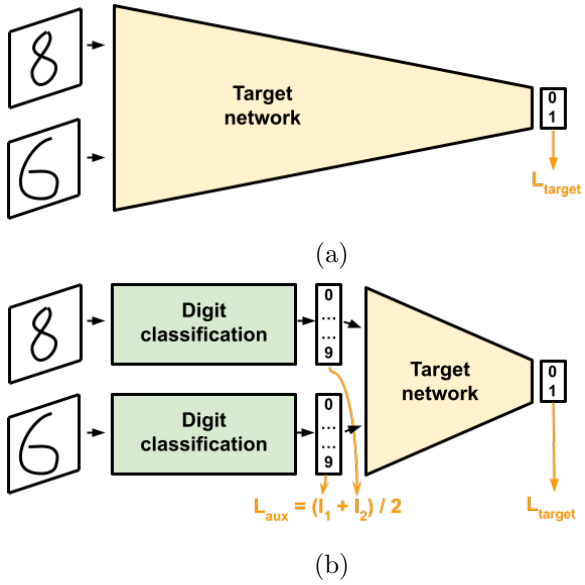

Figure 2: Images of digits are passed into two possible architectures to predict target labels 0/1: (a) a simple network that uses only the target loss to compute L_{target} and does not take advantage of weight sharing and (b) a siamese architecture that passes the images along networks with identical weights to provide digits predictions with losses l_1 and l_2 respectively before passing these predictions into a target network to predict the target labels 0/1. The auxiliary loss L_{aux} is a combination of the individual network losses, and the target loss L_{target} is as before.

We then move to convolutional networks based on the LeNet [2] architecture, which we expect to give better performance as convolutions inherently encode some weight sharing between neighbouring pixels in images. We encode explicit weight sharing by using a siamese network architecture that passes the two digit images along separate identical (*siamese*) networks that predict digits (a 10-dimensional tensor) before concatenating the outputs and passing these through a single fully connected layer to get a class prediction. This is illustrated in figure 2b. The baseline convolutional network, `ConvNet`, starts with a baseline target accuracy of 86.73%, which indeed is higher than the `FCBaseline` model, which illustrates the effectiveness of explicit (siamese) and implicit (convolutional) weight sharing. Adding the auxiliary loss to the target loss again improves the result, leading to an accuracy of 88.04%.

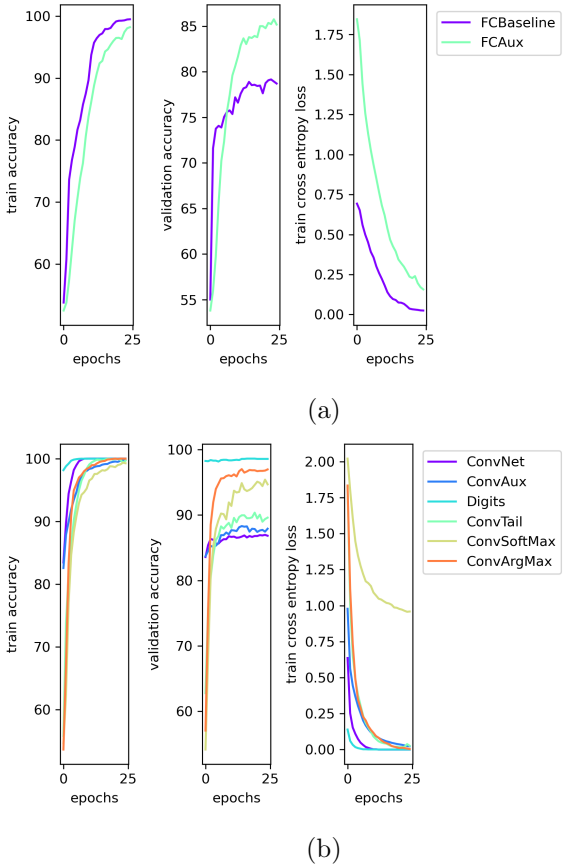

Figure 3: Performance of (a) fully connected networks and (b) convolutional networks over training epochs, averaged over 10 re-initialized models. Using an auxiliary loss function (`FCAux` and `ConvAux`) improves model performance over the baseline models (`FCBaseline` and `ConvNet`). For the convolutional models, using an arithmetic operation to compare digits (`Digits`) yields the best performance as well as converging extremely rapidly compared to other models. Using a larger fully connected network `TailNet` instead of a single fully connected layer in models `ConvTail`, `ConvSoftMax` and `ConvArgMax` improves predictive accuracy.

To get a benchmark for the accuracy we should be aiming for, we trained a network (`Digits`) that predicts digits along siamese networks as before, and directly compares the predicted digits using an arithmetic operation rather than using a target network. This model got a prediction accuracy of 98.26%. We then tried to design the final layers of the model to mimic this arithmetic operation by first designing a fully connected network `TailLinear` with 3 layers that achieved 100% prediction accuracy on the arithmetic task within 5 training epochs. Using `TailLinear` as the target network, we got a target accuracy of 91.02%. We improved this accuracy to 94.64% and 97.23% respectively by adding softmax or argmax operations on the digit predictions before passing them to the target network. Both of these operations likely simplify the classification task: The softmax normalizes the output of the digit classification model which enables the target model to interpret the input as probabilities. The argmax on the other hand returns a single digit prediction, which reduces the dimension of the input space.

Algorithm 1 PreTrain algorithm

$tail \leftarrow TailLinear()$
$body \leftarrow LeNet()$ ▷ initialize models
$e = 0$
while $e < 5$ **do** ▷ iterate over epochs
    $e = e + 1$
    $tail = tail.train()$ ▷ training step
end while
$tail.requires\_grad = False$ ▷ fix parameters
$model = Siamese(tail, body)$ ▷ combine models
$e = 0$
while $e < 20$ **do** ▷ iterate over epochs
    $e = e + 1$
    $model = model.train()$ ▷ training step
end while
return model ▷ return trained model

We finally pushed the accuracy up to 97.45% by pretraining `TailLinear` on the arithmetic task and `LeNet` with an argmax layer on the digit prediction task before putting them together in a siamese model called `PreTrain`, illustrated in algorithm 1. We also tried randomising the loss function such that it randomly chose between using the target loss and digit loss rather than using the auxiliary loss (`PreTrainRandom`), which did not improve the target accuracy.

3 Conclusion

The aim of this project was to develop a machine learning model to predict whether a digit in one image is less than or equal to a digit in another image. We observed a systematic improvement in target accuracy on including an auxiliary loss function in a fully connected architecture, to including auxiliary loss and weight sharing in a siamese LeNet-like architecture. The `Digits` model, which uses such an architecture to predict the digit in each image, then uses an arithmetic comparison to determine whether one digit is less than or equal to the second, achieved the highest target accuracy at 98.29% ($\pm$ 0.55%). Our `PreTrain` model, which separately pretrains a LeNet-like model to classify digits and a fully connected model to perform the arithmetic comparison before combining them in a siamese architecture, achieves a target accuracy of 97.45% ($\pm$ 0.53%), which is within the standard deviation of the `Digits` model.

This project demonstrates the importance of human domain knowledge in Deep Learning. Forcing the network to have a certain structure (eg. between the classification and target network) improved the performance in all experiments. This take-away can be very valuable in future projects.

References

[1] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[2] 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.