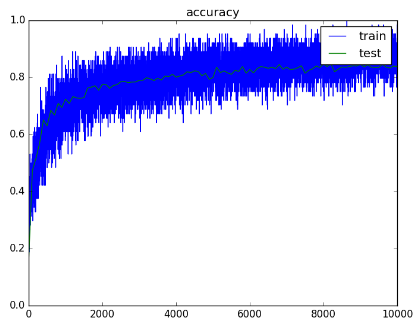
Solution for Project #2
Deep Learning, HHU 2016 / 2017

Thomas Germer **Michael Janschek** **Patrick Brzoska**

1 Code and Architecture

The code is provided as GitHub repository and is available under

`https://github.com/99991/DeepLearningProjects/tree/master/projects/project2`

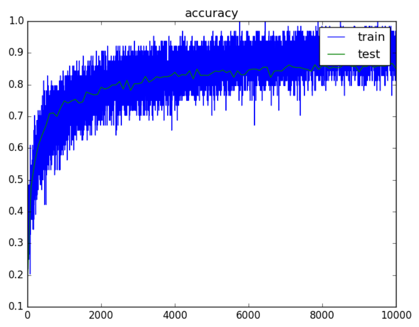


(a) accuracy

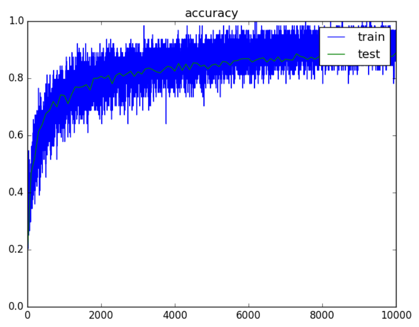
Parameter	Value
batch_size	64
num_batches	10001
kernel_size	3
num_kernels	32
num_hidden	500
depth	3
dropout_keep_probability	0.5

(b) Basic configuration

Figure 1: Performance and parameters for our basic configuration



(a) 64 kernels



(b) 128 kernels

number of batches	number of kernels	Accuracy
30000	32	87%
10000	64	85.1%
10000	128	88.9%
14200	128	91.6%

(c) Increase of iterations and kernels

Figure 2: Accuracy for various parameter configurations

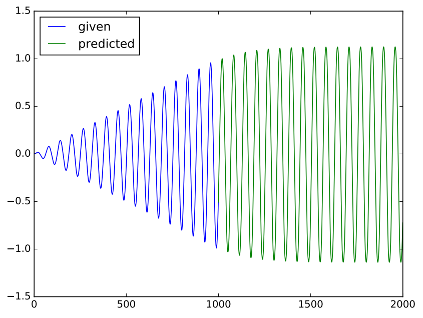
2 Tasks

1. **CNN on CIFAR-10**

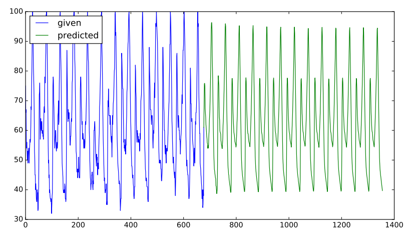
   (a) To accommodate the higher complexity of the CIFAR-10 dataset we chose to train a relatively deep CNN compared to our first project. Also, with the number of batches at 10000 and each batch having 64 samples, the training time was increased.
   The leaky Rectified Linear Unit (ReLU) activation function was used to counter the "dying ReLU" problem. Instead of the function being zero when $x < 0$, it is set to a small negative value of $0.1 * x$.

   (b) Table 1b shows the configuration of our network, while figure 1a shows its achieved accuracy. With these parameters a test accuracy of 82.8% was reached, which seems to be a fairly good number compared to other implementations.

   (c) By increasing the number of batches to 30000 a test accuracy of 87% was achieved. Increasing the kernel size to 64 while retaining a batch size of 10000 was not as successful, but still reached an accuracy of 85.1%. A kernel size of 128 resulted in an even higher test accuracy of 88.9%. The network even reached a test accuracy of 91.6% when increasing the batch size to 14200 with 128 kernels.



(a) Sine wave



(b) google trends for term "Urlaub"

Figure 3: The prediction fit

(d) In Zagoruyko and Komodakis [2016] average pooling has yielded better results, but max pooling has worked better within our network. Also, using global average pooling as a regularizer as in Lin et al. [2013] has not had a positiv effect on the test accuracy.

(e) Using random cropping as a way to increase the data size has lead to minor improvements. Random brightness and saturation has not yielded any success.

(f) Adding dropout has not resulted in any success. With the configuration shown in 1b adding dropout resulted in a test accuracy of 74.8%.

(g) Please see figure 4.

2. **Recurrent Neural Network**

   (a) Our architecture consists of a linear layer to scale the input, followed by a lstm layer, followed by three linear layers. The number of stacked LSTMs is chosen such that it covers three consecutive sine waves to capture both the shape of the sine wave as well as the increasing amplitude. More sine waves could be covered at the cost of longer training time.

   (b) A Long short-term memory (LSTM) cell utilizes a hidden state in order to calculate a prediction with each time step using information from the previous value of a given sequence.

   (c) Training went well.

   (d) The figure 3a shows the prediction fit. The network predicts the sine curve quiet smoothly, but sadly does not increase the amplitude in an expected manner. To form a test dataset, the last 100 samples were removed. The mean squared error is 0.000339 and the maximum absolute difference is 0.038844 between the predicted values and the test dataset.

   (e) linear -> lstm -> linear -> linear -> linear -> l2 loss

   (f) The figure 3b shows the number of search queries for the term "Urlaub" starting in the year 2004 until now (blue) as stated by google trends, and our networks prediction (green). The trend shows strong seasonal patterns wich are predicted fairly well.

References

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. 12 2013. URL `https://arxiv.org/abs/1312.4400`.

Sergey Zagoruyko and Nikos Komodakis. 05 2016. URL `https://arxiv.org/abs/1605.07146`.

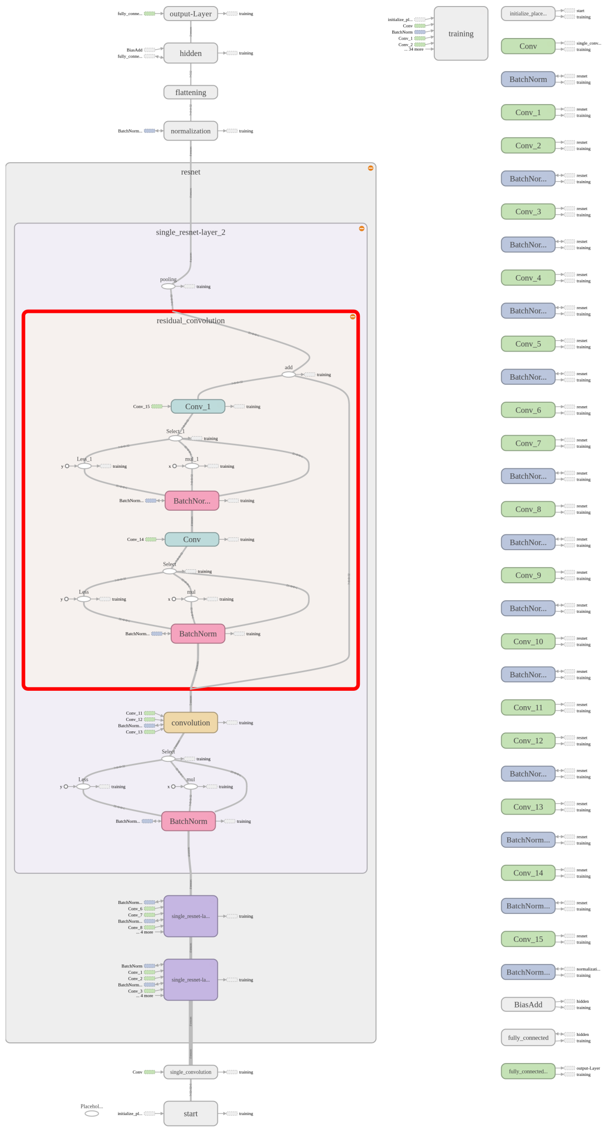


Figure 4: Tensorboard