1	Σ

Assignment Sheet Nr. 5

Group 107

Question 1

DISCLAIMER: Our network changed within the last hours of the challenge. The network described here is therefore slightly changed (see conclusion, network is marked int the code file).

Network Overview

We chose to use a Convolutional Neural Network design after a lot of testing and investigations. For that, we decided to use two convolutional layers: a bigger one for basic shape recognition, and a smaller one for detail recognition. Both were followed by a pooling layer to reduce the overall amount of data without missing out on information.

Those were followed by a two-layer feed forward network to process the reduced data and determine a result. Each of the layers has 132 nodes and uses the ReLU activation function as it is commonly used in combination with convolutional layers.

The final layer also received a dropout of 50%, which disabled half of the nodes to decrease overfitting.
For our optimizer, we tested a bunch of methods, but (for now) stuck with Stochastic Gradient Descent combined with a Cosine Annealing Scheduler with warm restarts which changes the learnrate from 0.05 to 0.0001 using a cosine function depending on the epochs. This results in an 'annealing effect' where strong weights are hardened while weak weights get destroyed. For our weight decay, we chose 0.005 as a start value.

For our loss function, we chose BCELoss which provides a Cross Entropy loss function for binary classification.

For additional parameters, we reduced the batch size to 16, as a smaller batch size is said to be good for the network performance, and a (current) total of 80 epochs for training.

Training

For training, we used a batch size of 16 with the alread given training function and mentioned BCE Loss function, cosine annealing scheduler and SGD optimization.

Using my GTX 1060, training took about 25 seconds per epoch, where each epoch uses a dataset of 3750 images. With a total of 80 epochs, it took around 30 minutes per network.

Training Set

Because training data is the essence of a neural network, and more data can help stabilizing the network, we enlarged our training set to a total of 18072 images.

As generating and labeling training data is mostly really difficult, we reused the already labeled training data by flipping it horizontally, vertically, and rotating it by 180 degrees.

The mirroring was also done for the validation set, which we then also used for training (except the original validation set) to increase variety and preventing overfitting.

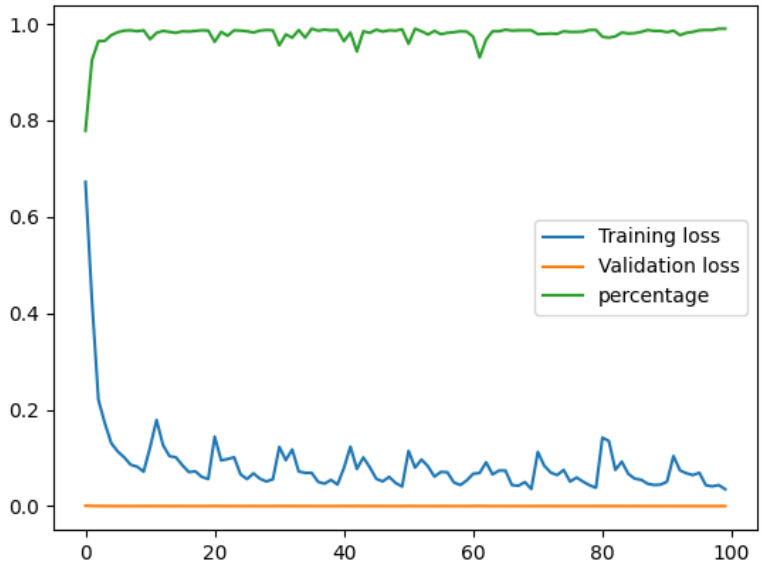

Figure 1: Training history of our network: Underfitting can be observed

0.1 Conclusion:

In retrospective, we found that our model was in fact massively underfitting, and could probably still improve upon the 98.937% correctness reached by increasing our network size.

Our fork can be found at:

`https://github.com/Skyfighter64/SeaDronesSee`

We will publish our code there as soon as the challenge is over.