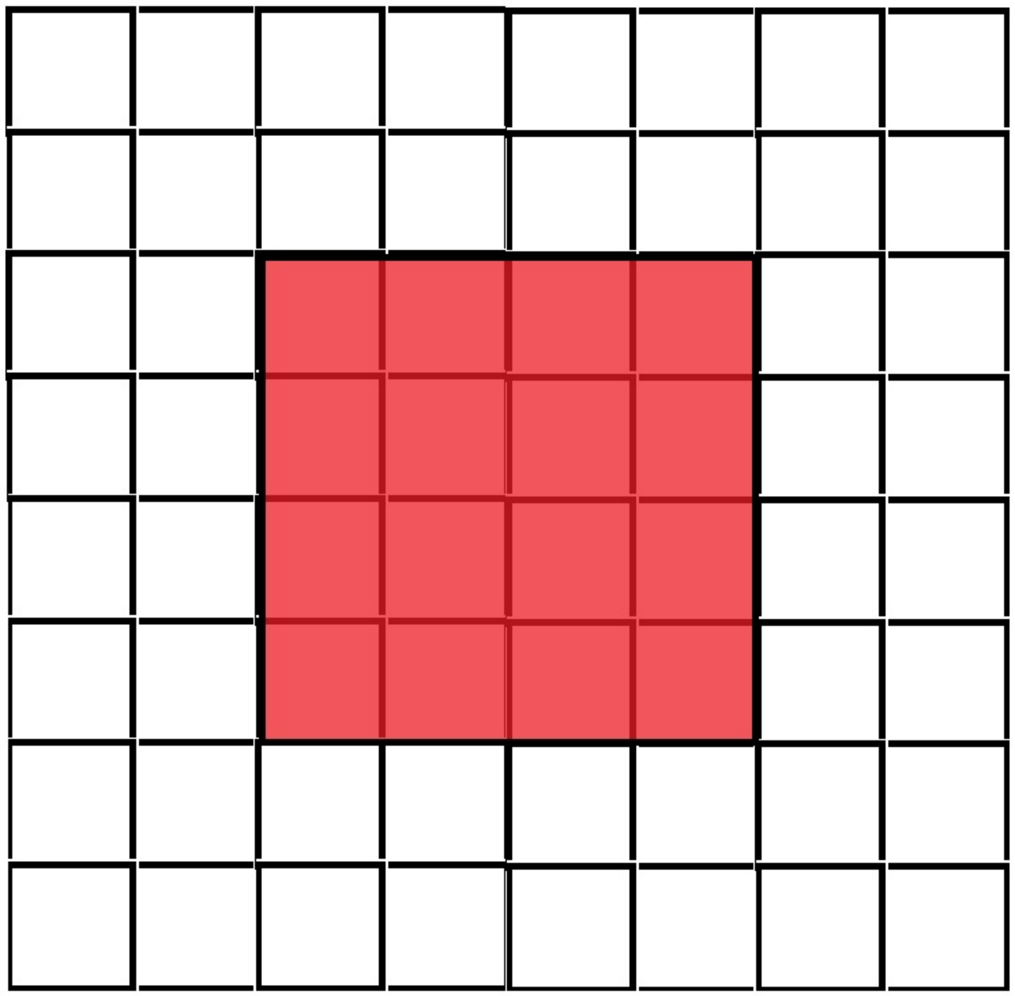
blog.md

A NEURAL NETWORK BASED CORNER DETECTION METHOD AND MOTION DETECTION

By: Noah Jadoenathmisier (4684907 TU-Delft)

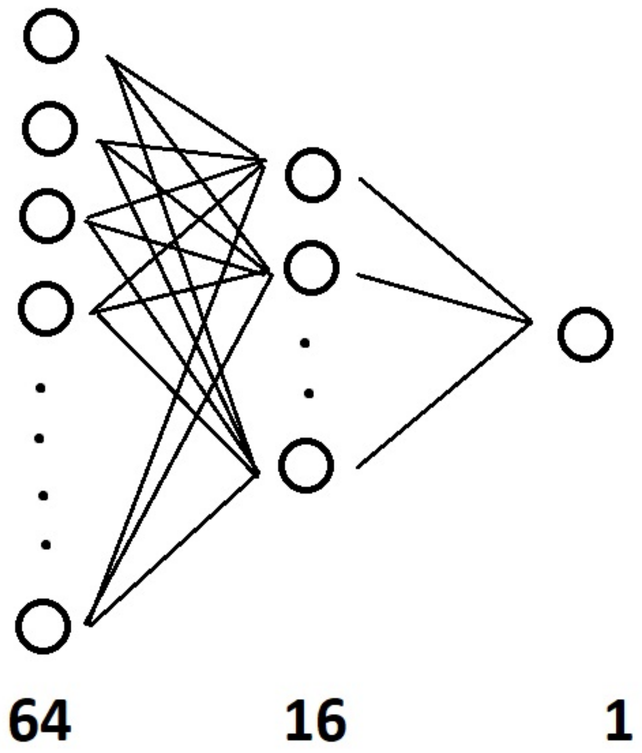
Corner detection is a very useful and common computer vision procedure. In "A Neural Network Based Corner Detection Method" a technique is proposed that uses neural networks to find these corners. In this article, we will reproduce the technique described in this paper and test it on real-world data. Specifically, we will use the corner detection method to do motion detection on a video. For this reproduction Python 3.9 is used with PyTorch.

This corner detection technique works by looking at patches of 8x8 pixels and classifying them as containing a corner in the middle 4x4 pixels or not.



A patch of 8 by 8 pixels, If there is a corner in the red area, the whole patch is classified to be a corner patch

The paper describes a very simple architecture with a single hidden layer of size 16.



In the paper, it is not discussed which activation function they use. In this reproduction, we use the relu activation function for the hidden layer. To train the neural network, they use two sets of 8x8 images. One set for which there is a corner in the center 4x4 pixels, and one for which there is not. The corners are always a multiple of 45 degrees.

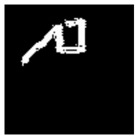
The paper does not give a clear description of how the training data was produced. In this article we will use the following technique:

Generate a set of images and add random lines with corners to those images. While placing the random lines, a list of all the corner locations is made. Now, take random 8x8 patches from those images, and divide them into two groups. One with corners, and one without. Finally, throw away random samples from the largest group, to balance the data.

These images were used to train the neural network. To generate the test data 10.000 images were generated, resulting in about 150.000 8x8 patches. The test set contained about 1500 8x8 patches.

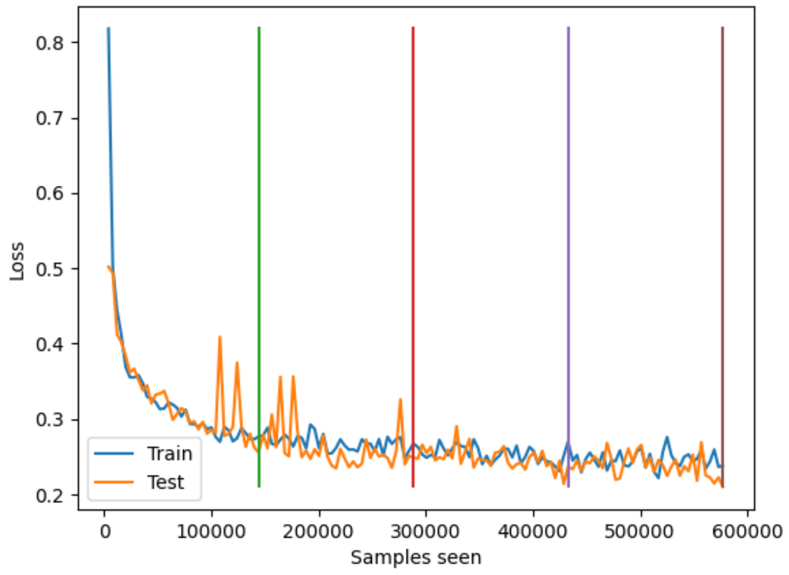
The initial results did not look good at all. Every time the network was done training, it either classified all samples as having a corner or classified all samples as not having a corner. Dropping the learning rate from 0.001 to 0.00005 solved that problem! The accuracy also looked very good, but there was still a little problem. The neural network learned to cheat!

To understand what was happening we should take a look at an example. We will show a randomly generated line with corners, and next to it an image of where the neural network thinks the corners are.

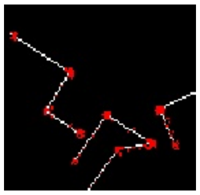


As you can see, the neural network thinks that all parts of the line are corners! Looking at the training samples, this makes a lot of sense. Nearly all the images without corners are exactly the same. Just completely black 8x8 patches. In order to solve this problem the number of empty 8x8 patches in the training data is limited. After running lots of experiments it turned out that having about one empty patch per 15 non-empty patches worked well.

The learning curve looks very smooth, and after 4 epochs the curve flattens out. Every vertical line in the learning curve plot represents the end of an epoch.



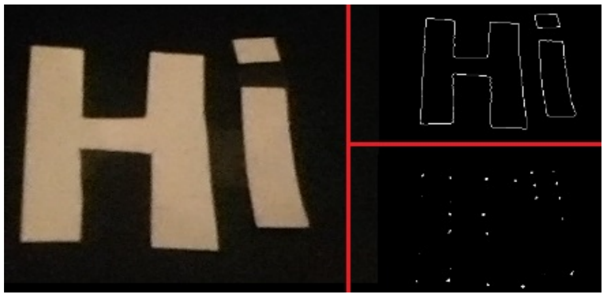
With these changes the network performs very well, getting an accuracy of 93% on the test set. Manually looking at the output of the network also shows that it actually learned to detect corners now. In the following image all locations where the network thinks there is a corner are made red.



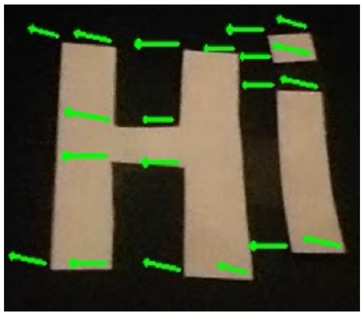
All the data that has been used for training and testing in the original paper, as well as in this reproduction, up to this point, is synthetically generated. We thought it would be a good way to validate how well it works in real-world scenarios. In the original paper, motion detection is mentioned as a use for corner detection. We created a motion detection algorithm with this deep learning corner detection technique at the core to test the real-world performance of the algorithm.

First, every frame of the video is processed with an edge detection filter. Then the neural network is used to find all the locations that contain a corner. Because the neural network outputs for every pixel if it contains a corner or not, we need a method to convert this 2d array to a list of corner locations. We first blur this 2d array and apply a threshold function. Then on every pixel in this array that is true, a flood-fill is used to find all its neighbours that make up the same corner. The average location of all those neighbours is taken and added to the list of corners.

In the following image, a random frame of a video is shown. Next to it, on the top right, there is the same frame with the edge detection applied to it. On the bottom right, the locations of the detected corners are shown. On this specific frame, the network did a perfect job on the 'H', but it missed the bottom left corner of the 'i'.



Then for every two consecutive frames, each corner from the first frame is matched to the closest corner in the second frame. When the distance is within 15 pixels, an arrow is drawn in the direction of the movement, with a length equal to 5 times the moved distance. For a single frame this looks like this:



All the arrows point to the left. This is correct! In this video, the text was indeed moving to the left (actually the camera was moving to the right). This shows that this deep learning corner detection method can be used on real-world problems. Not every frame is as perfect as this one, but on average the motion detection works very well.

The complete video can be seen here (https://youtu.be/3Wegz9sT6Ig). It is slowed down from 60fps to 10 fps to make it easier to see what happens.

Conclusion

Just like the original paper, we implemented a corner detection method using deep learning. The original paper got an accuracy of 97.55% while we got 93%. This difference in performance can be explained by differences in the difficulty of the test set. Our implementation also works well on real-world images. The implementation that we made is however not very optimized for motion detection, because it took about 5 hours to analyse a 4-second video. So, while analysing video with this method, make sure to have plenty of tea (or other beverage of your choosing) to ease the wait :)