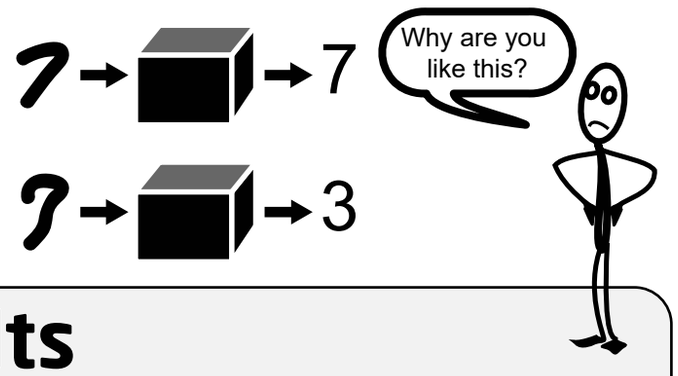
A look into robustness

By Mikkel Odgaard, s174371

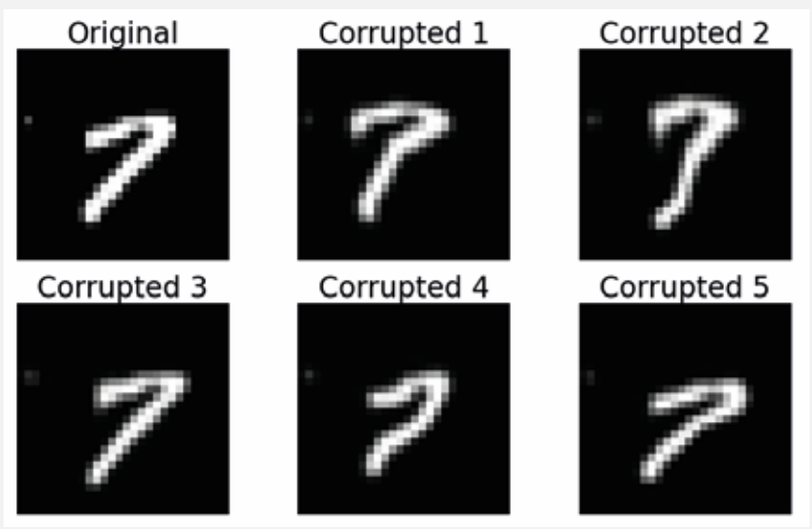


Introduction

The accuracy of neural networks keep going up and they accomplish more and more impressive tasks. Purely looking at accuracy might be fine for lab purposes, but in the real world, robustness starts to matter. Real life data is not as polished and may change over time, which requires neural networks to be adaptable and not susceptible for minor changes. Networks with low robustness are often exposed in real life scenarios and meet a lot of criticism in regards to fairness and bias.

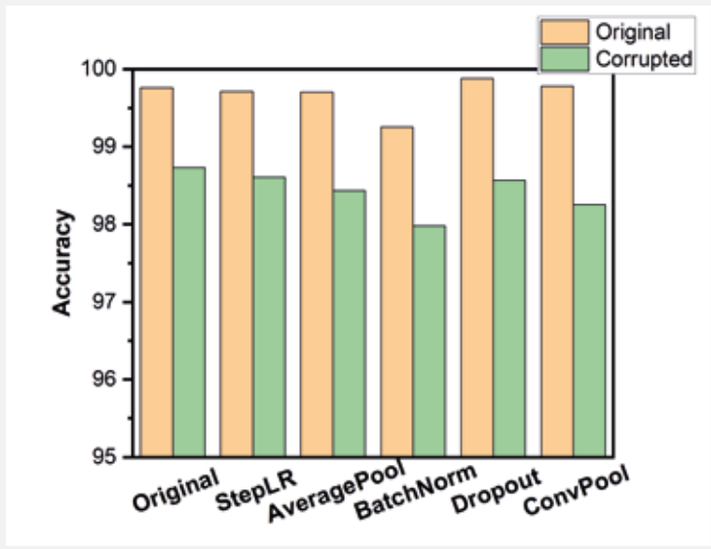
Methodology

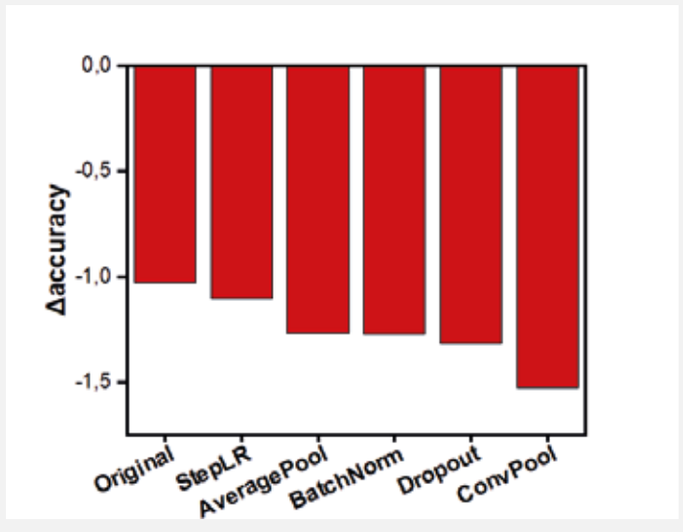
The CNNs are tested on two different datasets, the original MNIST and AlignMNIST, a dataset with deformations created from the original data. The deformed dataset is denoted 'corrupted'. The complete network (see architecture) is the baseline accuracy. Five iterations of this network is then created by removing/substituting a key feature in the network, e.g. removing dropout. These iterations are then tested and compared to the original dataset, giving us (Δaccuracy). Δaccuracy will serve a simple metric to measure robustness, however, it can also indicate overfitting. This is why it is also important to look at the accuracy on the initial dataset and compare the results, such that overfitting does not invalidate the findings.



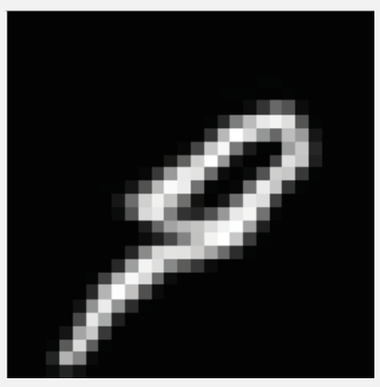
Results

The difference in Δ accuracy isn't as much as initially expected, however, it still tells an interesting story. Firstly, we see that a network without dropout results in overfitting, as it's accuracy is marginally higher for the original dataset, but lower for the corrupted. However, we see that a network, e.g. without batch normalization, suffer a larger drop in accuracy, when tested on 'corrupted' datasets.





Original: 9
Avg: 4
BatchNorm: 8
StepLR: 9
ConvPool: 7
Dropout: 0



It is therefore seen that some of these features directly influences robustness and allows the networks to better generalize it's learning and thus makes the networks adaptable and less susceptible for change.

This is also evident in the bottom left picture, where only the original network and the StepLR network managed to correctly predict the number. We see that when dropping key features such as dropout or batch normalization, that the networks only take some features into account (Predicting 0 due to the upper part of a 9) or incorrectly assuming that the presence of a 9 would indicate an 8, even though the last line is missing.

Conclusion

While the differences where small, the results suggest that features such as batch normalization play a key role, not only in accuracy, but also in the robustness of the network. Although a simply experiment, and an even simpler definition of robustness, this project highlights the importance of robustness in modern networks. While it is hard to decipher the black box that is neural networks, if we can put more trust into it's predictions, by knowing it is fair and robust, neural networks can be applied more critical fields. My hope for the future is that, when evaluating a neural network, robustness is an important metric that stands side by side with accuracy, such that fairness can be achieved. Robustness doesn't promise fairness, but I believe it's an important first step in the right direction.

Architecture

Inspired by LeNet-5 design,
with modern improvements

Two stacked 3x3 filters
Max pooling 5x5 (stride 2)
ReLU activation
Batch normalization
Dropout

Adam optimization
Decaying LR