U-Net for Segmentation of Neuronal Structures in EM stacks

Anubhav Goel Christina Bartozzi Madhur Sudarshan

{s201850, s205745, s206641}@dtu.dk

Problem

Convolutional Neural Networks have gained immense popularity in recent years in numerous image processing tasks and image segmentation in the medical domain is no exception. However, the general understanding of neural networks shows that training a neural network requires a large annotated dataset.
This requires careful manual annotation of the training data and for large amounts of medical data, this may not always be possible. We need to look at strategies which make the maximum use of the available data. Along with the amount of data, the training of the network is also influenced by the various network parameters like learning rate, use of optimizers, the loss function as well as the depth of the neural network. We study the effect of these factors as well.

Proposal

In this project, we explore the effects of the following factors in a deep learning setup by using the U-Net as our basic architecture

- **Effect of Depth** A neural network with more layers possesses higher power of representation but computation power requirements and the possibility of overfitting increase with depth.
- **Data Augmentation** Limited training data can lead to overfitting, which can be observed on the validation error. Augmentation of data gives the model more opportunity to learn the data features.
- **Effect of Learning Rate** Learning rate is the speed at which the model parameters are learnt. An optimal learning rate must be set in order for the model to be learnt in a reasonable time and avoid the divergence of loss.
- **Use of Optimizers** Different optimizers use different mathematical gradient traversal schemes to reach the loss minima and avoid vanishing gradients. The correct optimizer depends on the type of data and the architecture of the model.

We try to understand how these parameters affect the performance of a neural network and try to select an optimal model for our dataset.

Model Architecture

U-Net was first introduced in 2015 for the purpose of image segmentation in the biomedical domain. The architecture can be seen in two different parts: a contracting and an expansive path. The contracting path can be seen as a typical CNN which consists of convolutional layers which perform convolution with filters which have learnable weights and then pooling layers which discard information to extract the low level information.
Generally, during pooling the size of the image decreases therefore resulting in loss of spatial information. When we output the result of the segmentation, we need the high level spatial information again which results in the need for an expansive path following the contracting path. To obtain the spatial information that we lost during pooling, we make the use of skip connections. The connections take information from earlier layers and pass them on to the layers that do not contain them anymore.
The U-Net is a fully convolutional architecture which has the advantage that the size of the input does not affect the architecture. The number of layers decide the complexity of the model and can be tuned depending on the availability of the data and the computational power.

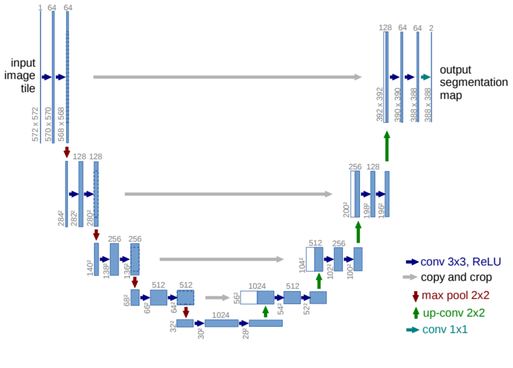
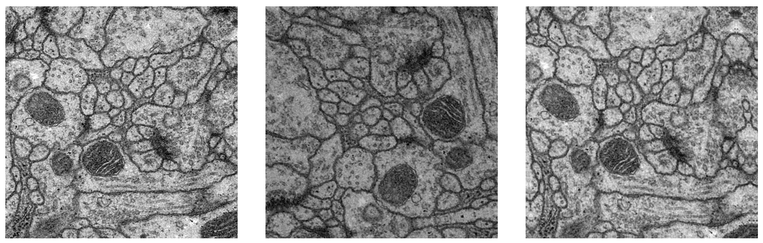


Figure 1:U-Net Architecture

Data Augmentation

We apply affine transformations, changes in brightness range as well random elastic transformations to the data and to the corresponding masks and append these to the data to generate a bigger dataset. This allows the model to learn invariance to transforms of this kind which are quite common in the biomedical imaging domain resembling the natural variation in tissues. The first pair of pictures shows a train sample along with the affine transformation and random brightness scale. The third image shows the random elastic deformed image of the first one.



Effect of Learning Rate

Learning Rate controls the rate of update of model parameters. Setting the right learning rate is an important aspect of neural network training process. When the learning rate is set too low, the model takes considerably more number of epochs to reach the loss minima. A higher learning rate may decrease the learning time but may lead to the divergence of loss. We perform learning rate analysis for the Adam optimizer, the graphs for which are shown below.

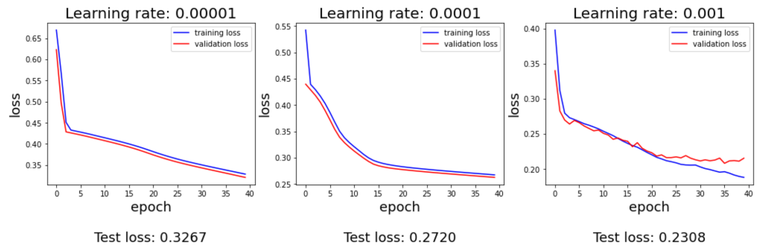


Figure 2:Learning rate effect on train and validation loss

Use of Optimizers

Different optimizers follow different algorithms to achieve the local optima for the loss function that the neural network is backpropagating on. We look at three different popular optimizers and decide by the relative performance which one works the best in the current setup.

- **Stochastic Gradient Descent**: This is the naive approach which can be summarized by the following algorithm

Algorithm 1: Stochastic Gradient Descent

Choose a random set of parameters w and a learning rate η;
while *While loss is less than a certain minimum* **do**
  Randomly shuffle samples in the dataset;
  $\mathrm{w} = \mathrm{w} - \eta \nabla \mathrm{E_i(w)}$;
end

- **Adam**: Adam stands for Adaptive Moment Estimation. Instead of maintaining a single learning rate for all weight updates, a learning rate is maintained for every network parameter which is adaptively changed during the course of training. The estimates of the first and second moments of gradients are used as opposed to the use of only the first gradient of the loss in SGD, which makes Adam less susceptible to the problem of vanishing gradients.
- **Adamax**: Adamax is quite similar to Adam, except the fact that it uses the infinity norm in its calculations as against the second order moments.

The three optimizers were used for U-Net with a learning rate of 0.0001 and the training and validation losses were recorded which have been plotted in the following graphs.

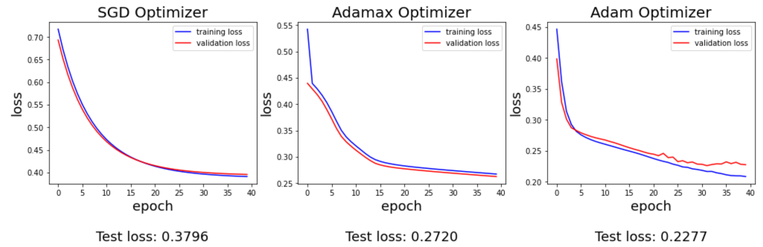


Figure 3:Optimizer effect on training and validation loss

Conclusion

Depth analysis
We started with a U-Net which consisted of 3 contracting layers and 3 expansion layers (upconvolution) along with a bottle neck layer. With this model, the training loss decreased consistently but the validation loss started stagnating so we decided to move from this model to a denser model with 4 contracting and expansion layers. With this model, both the training and validation losses decrease consistently. This can be observed in the left graph which has been shown below.
Data augmentation analysis
Along with this, we also look at the effect of using the augmented dataset against the raw dataset. Even though the raw dataset has lower errors, we can see that the validation loss for the raw dataset starts to increase and diverge which is a clear indication of overfitting. This was expected since overfitting is extremely common in cases where we do not have sufficient data.
Overfit models may perform well on the training data but can lead to poor results on the testing data, therefore we move to the augmented dataset and see that the training and the validation losses keep on decreasing over the epochs, thus having corrected the problem of overfitting.

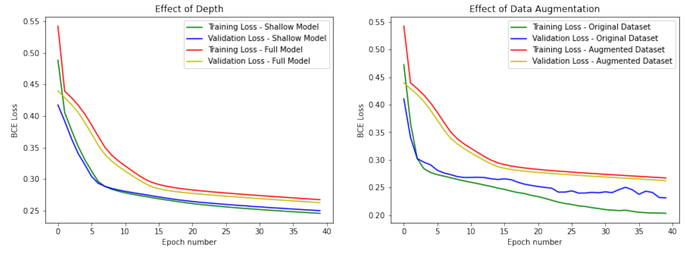


Figure 4:Effect of Depth and Data Augmentation

Learning rate analysis
For the learning rate analysis, we can see that a very low learning rate of 0.00001 does not converge to a low loss even in high number of epochs. If we increase the learning rate as high as 0.001, then the loss starts to diverge which can be seen in third graph where the validation loss starts to shoot up. We see that a learning rate of 0.0001 performs quite well and we set this as the ideal learning rate for our task.
Optimizer analysis
For the optimizer analysis, we see that the SGD optimizer does not reduce the loss even after 40 epochs. However, the Adamax and the Adam optimizer do quite well in reducing the loss. Even though the Adam optimizer has lower loss values towards the end, we see that the validation loss shows signs of divergence, therefore we prefer Adamax optimizer in the model because the losses seem to follow a smooth downward trajectory.

References

- https://github.com/milesial/Pytorch-UNet
- https://github.com/ChristianIngwersen/Advanced-Image-AnalysisF20
- https://www.kaggle.com/bguberfain/elastic-transform-for-data-augmentation
- https://pytorch.org/docs/stable/optim.html
- Model Architecture Image Credits: U-Net: Convolutional Networks for Biomedical Image Segmentation, *Olaf Ronneberger, Philipp Fischer, and Thomas Brox*