Hyperparameter: Learning Rate for Optimizers

Motivation

When we train our model on datasets, we need the error to converge to a minimum for efficient analysis. Choosing a good optimizer out of the plethora of options is one of the most important decisions that will after this process. A good optimizer will make sure that we get the required accuracy after few epochs of training. The best optimizer for training varies from problem to problem and dataset to dataset. Another important aspect of faster convergence is the learning rate of the optimizer. Since, an optimizer can proceed with many different learning rates, it is important to identify a learning rate that works best for the problem we have. Too high a learning rate might lead to lack of convergence and too low a learning rate will slow down the training process. So, it is essential to test different learning rates to find the best one which converges faster.

It has been observed that having a high learning rate initially is better for faster searching of the loss landscape. Subsequently, it has also been observed that reducing the learning rate as the number of epochs increases helps the optimizer to not make sudden jumps out of a minimum it is approaching, thus helping it to reach a minimum. So, we use different learning rate schedulers to see which one will work best with the optimizer to reach a minimum faster, if at all it does aid the optimizers.

Experiment Design Strategy

The problem we have is that of Cardiac MRI Segmentation. The dataset contains around 4k images for training. We are dividing the dataset into 80% as training and 20% as validation sets.

We are creating a U-Net model to train the dataset as given in the paper. (Chuckyee, 2017)

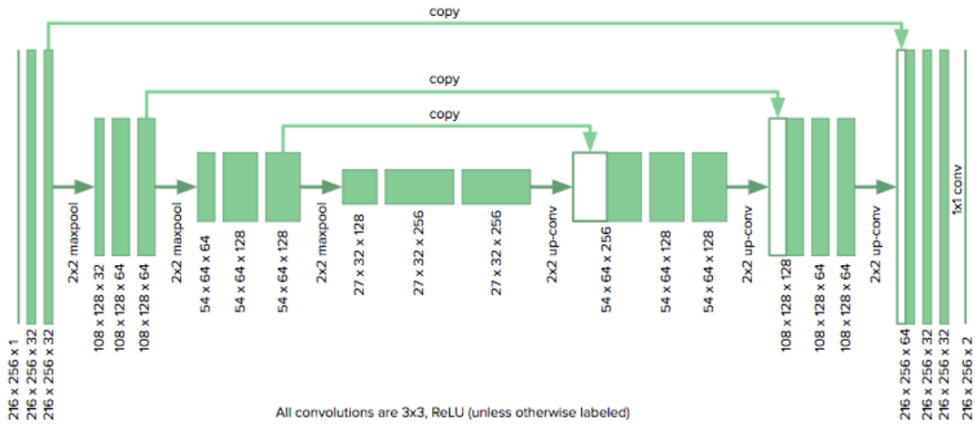


Figure 1 Unet Architecture (Yee, 2017)

We are not use data augmentation methods given in the paper for training.

We used Weighted categorical Cross entropy as our loss function which we call as pixel loss.

Mean Intersection-Over-Union was used along with Accuracy as a metric for evaluation as our task is semantic image segmentation

Dice loss was also computed as an evaluation metric.

Batch size = 32

We use the following 7 different optimizers for our experiment.

SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam

Experiment 1:

We are going to train the data using each of the above optimizers with 4 different learning rates

0.1, 0.01, 0.001 and 0.001 to find out how fast the loss converges to a minimum.

We find the minimum number of epochs needed for the validation error to converge to a minimum as measure of convergence.

The training was run for 100 epochs to check the convergence for high learning rates and to 700 epochs for low learning rates.

Experiment 2:

After finding the best learning rate for each of the optimizers from the results of the first experiment, we vary the data available for training and see how that affects the rate of convergence for each of the optimizers.

For this experiment, we are only training at the best learning rates we found for each optimizer. So, from the result we can know if availability of data affects the behaviour of the optimizer.

We train the model using 20%, 40%, 60%, 80% and 100% of available training data.

Results

Search learning rate for fast convergence, for different optimizers

The following table shows the results of experiment 1. It shows the learning rates which achieved fastest convergence of the validation error during training.

Optimizers	Learning rate for fastest convergence	No. of Epochs for convergence
SGD	0.1	50
RMSprop	0.01 – 0.1	2
Adagrad	0.2	2
Adadelta	-*	-
Adam	0.001 – 0.1	2
Adamax	0.001 – 0.1	2
Nadam	0.001 – 0.1	2

Table 1 Learning rates for fastest convergence

From the table it is clear that fastest convergence is achieved when the learning rates between 0.1 and 0.01 are used for most of the optimizers. The optimizers like Adam and its variants, Adamax and Nadam converges fast for a wide range of learning rates from 0.001-0.1. But a learning rate of 0.0001 is too low for fast convergence for all the optimizers.

Train model using same learning rate for different optimizers.

The following table shows the number of epochs needed by the 7 optimizers to converge to minimum loss.

	Learning Rates			
	0.1	**0.01**	**0.001**	**0.0001**
SGD	50	500+	Doesn't converge	Doesn't converge
RMSprop	2	2	2	Doesn't converge
Adagrad	2	500+	Doesn't converge	Doesn't converge
Adadelta	Doesn't converge	Doesn't converge	Doesn't converge	Doesn't converge
Adam	2	2	2	200+
Adamax	2	2	2	550+
Nadam	2	2	2	208+

Table-2. Number of Epochs to reach convergence for different learning rates.

*- means training didn't converge

Now the following tables show the accuracy achieved by the evaluation metrics for each learning rates when it reaches convergence

Learning Rate 0.1			
Optimizers	Accuracy	Mean IoU	No. of Epochs
SGD	0.95	0.85	50
RMSprop	0.95	0.92	2
Adagrad	0.95	0.88	116
Adadelta	-*	-	-
Adam	0.95	0.92	2
Adamax	0.95	0.92	2
Nadam	0.95	0.92	2

Table 3 Training results for learning rate of 0.1

Learning Rate 0.01			
Optimizers	Accuracy	Mean IoU	No. of Epochs
SGD	0.95	0.80	500
RMSprop	0.95	0.92	2
Adagrad	0.95	0.85	500
Adadelta	-	-	-
Adam	0.95	0.92	2
Adamax	0.95	0.92	2
Nadam	0.95	0.92	2

Table 4 Training results for learning rate of 0.01

Learning Rate 0.001			
Optimizers	Accuracy	Mean IoU	No. of Epochs
SGD	-	-	-
RMSprop	0.95	0.92	8
Adagrad	-	-	-
Adadelta	-	-	-
Adam	0.95	0.92	2
Adamax	0.95	0.92	2
Nadam	0.95	0.92	2

Table 5 Training results for learning rate of 0.001

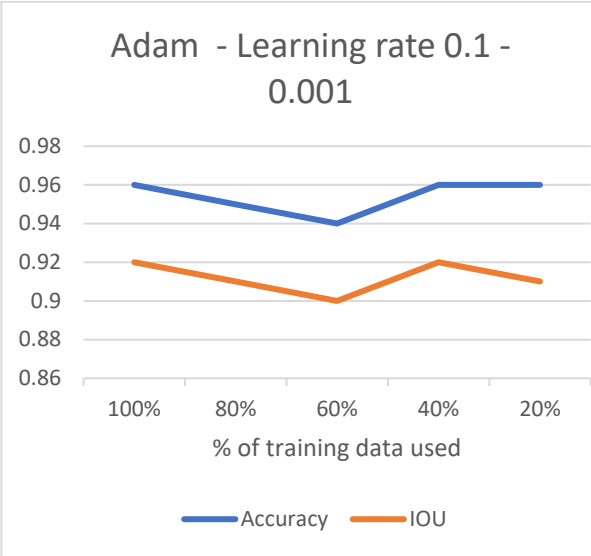
Learning Rate 0.0001			
Optimizers	Accuracy	Mean IoU	No. of Epochs
SGD	-	-	-
RMSprop	0.92	0.65	65
Adagrad	-	-	-
Adadelta	-	-	-
Adam	0.95	0.90	200
Adamax	0.97	0.87	550
Nadam	0.95	0.92	208

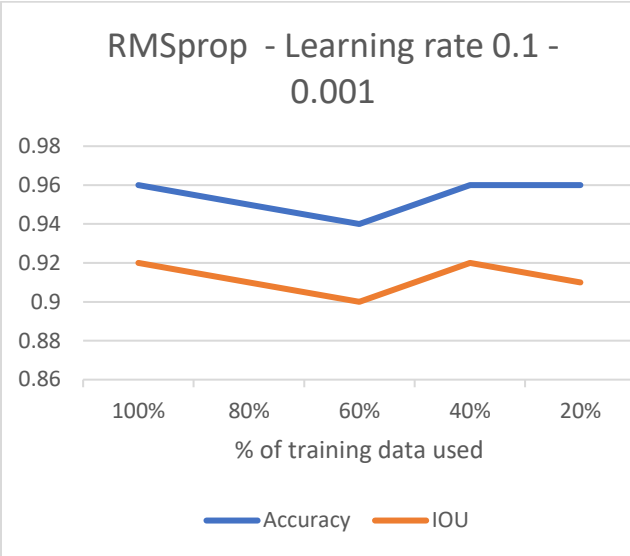
Table 6 Training results for learning rate of 0.0001

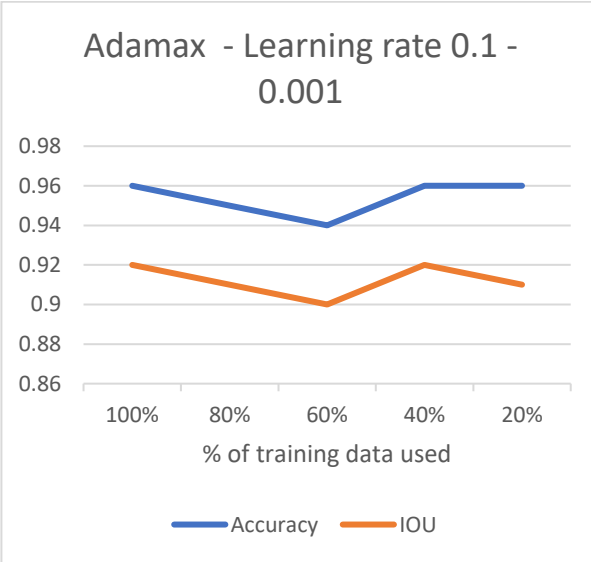
From the data above, it is clear that maximum evaluation accuracy is obtained using higher learning rates. Amoung the various optimizers, SGD is most sensitive to change in learning rates and Adam and its variants, Adamax and Nadam are least sensitive to change in learning rates. For very low learning rates, SGD, Adadelta and Adagrad doesn't even converge. Amoung all the optimizers, Nadam fares best as it performs the best in the entire span of learning rates tested and gives fastest convergence.

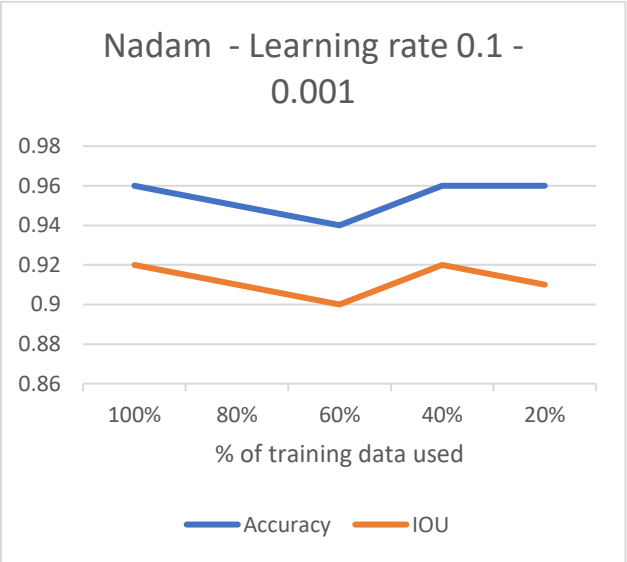
Train models using 20%, 40%, 60%, 80% and 100% of available training data.

Below are the results showing the variation in validation accuracy when we use different percentages of the training data.



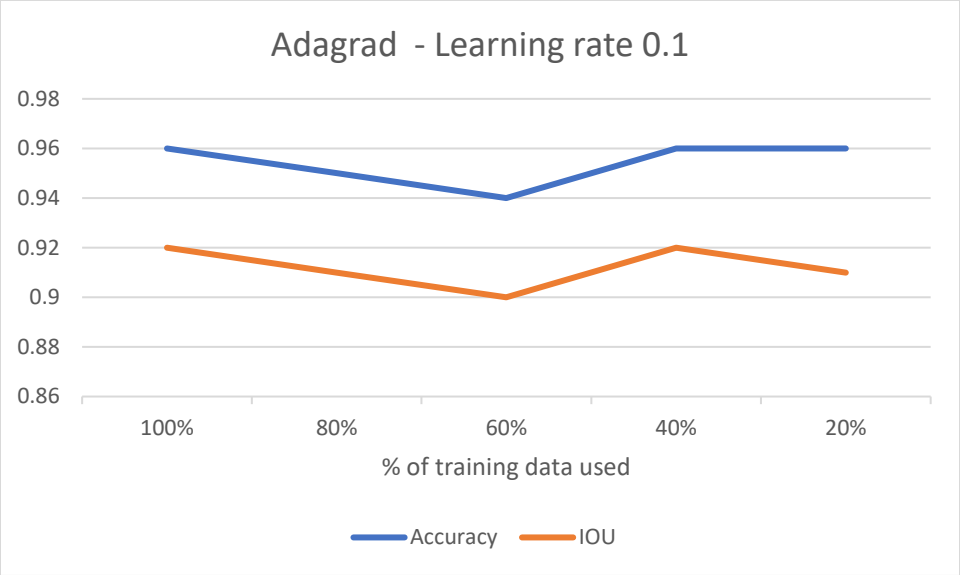


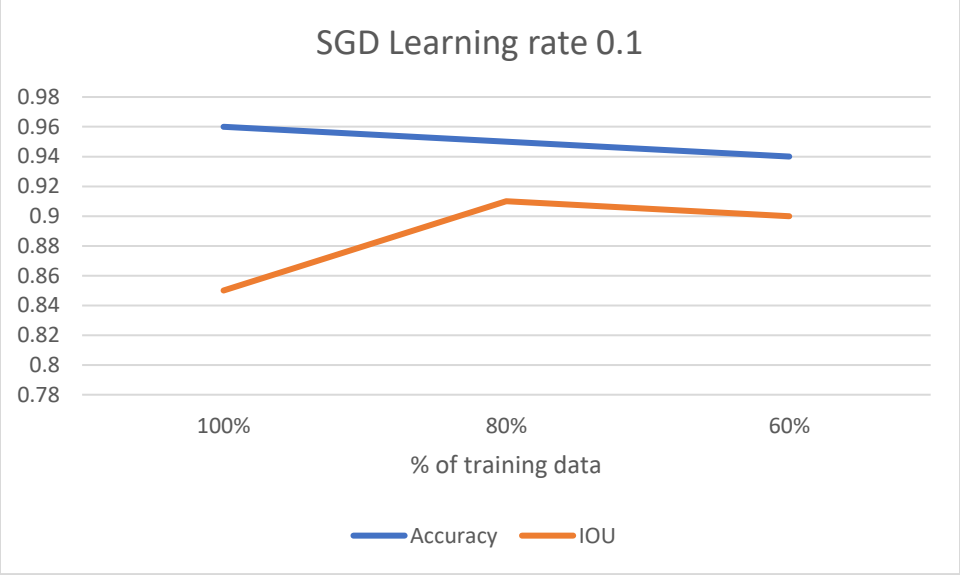




These optimizers show the same performance across the range of learning rates from 0.1 to 0.001

The following data was for Adagrad and SGD was obtained using learning rate of 0.1.





Adadelta didn't converge for any learning rate. We used only one learning rate for Adagrad and SGD because they were not converging for other learning rates for reduced datasets.

From the data, we can see that the evaluation accuracy upon convergence almost remains constant even though the amount of data is reduced. But the possibility of convergence reduces as the data available for training is reduced.

Adagrad converged to minimum in 2 epochs for all reductions in percentage of training data.

The tables below show the number of epochs needed for convergence for different learning rates for the optimizers

Adam	Learning Rates		
Percentage of Training data	0.1	0.01	0.001
100	2	2	2
80	2	2	2
60	2	2	50
40	2	2	40
20	2	2	250

Nadam	Learning Rates		
Percentage of Training data	0.1	0.01	0.001
100	2	2	2
80	2	2	4
60	2	2	2
40	2	2	4
20	2	2	5

Adamax	Learning Rates		
Percentage of Training data	0.1	0.01	0.001
100	2	2	2
80	2	2	2
60	2	2	2
40	2	2	2
20	2	2	2

RMSprop	Learning Rates		
Percentage of Training data	0.1	0.01	0.001
100	2	2	8
80	2	2	2
60	2	2	2
40	2	2	2
20	2	2	10

From the data, we can infer that for higher learning rates, Adam, Adamax, Nadam and RMS prop remains robust to reduction in training data. As the learning rate decreases, the speed of convergence reduces for all these optimizers.

Amoung the optimizers Adamax was most robust to change in availability of training data.

Since most of the optimizers which converged while training converged very fast for high values for learning rate (2 epochs for learning rates of 0.1 – 0.01), using learning rate schedulers which reduce the learning rate as training epochs increases doesn't affect the performance of these optimizers. Most of the optimizers didn't converge only for very low values of learning rates, and using a learning rate scheduler which further lowers the learning rate is only going to adversely impact the training process and further delay convergence.

Future Work

For this experiment only segmentation of the inner membrane was considered. If we had more time, we would have tried running the model for segmentation of outer membrane and compared the results with varying the learning rates on these optimizers. We would have tried using multiple learning rate schedulers in Adadelta to find if using the learning rate scheduler will help in achieving convergence. We would have tried these hyperparameter experiments on models other that Unet, like dilated Unet, dilated Densenet, and also tried to find the impact of these parameters in transfer learning using pretrained models like ResNet and VGGnet. We could use different learning rate in each layer and analyze the results. We can also find the impact of learning rate turning when we fine tune the decoder part of the Unet in transfer learning. The little variations in the hyperparameter tuning might been due to the scarcity of data. We couldn't do data augmentation to increase the size of training set as loading and augmenting was taking too much time. If we get more time, proper preprocessing and data augmentation will be done.

References

Chuckyee. (2017, November 27). *cardiac-segmentation*. Retrieved from Github: https://github.com/chuckyee/cardiac-segmentation

Yee, C.-H. (2017, August 28). *Cardiac MRI Segmentation*. Retrieved from Github: https://chuckyee.github.io/cardiac-segmentation/