TP2:

Presents all obtained the results (graphs, accuracy, running time)

Part I:

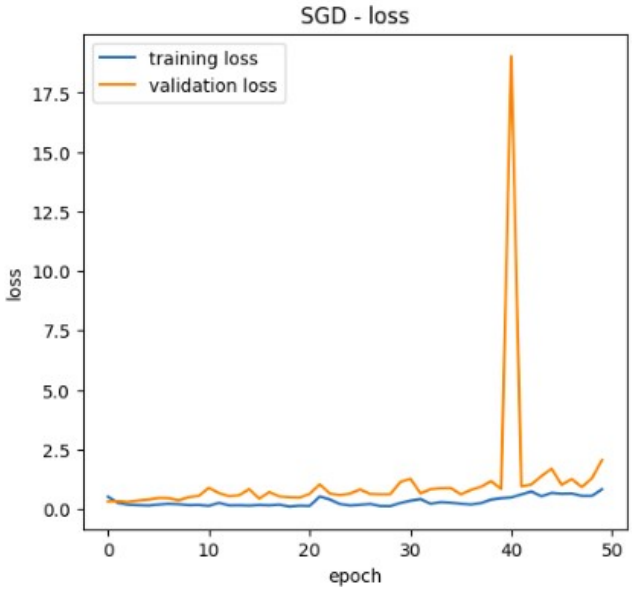
The used algorithms:
- SGD (Stochastic Gradient Descent): updates the model parameters using the gradient of the loss function, the size of the batch is 1
- batch SGD: updates the model parameters using the gradient of the loss function with respect to the entire training dataset, it has stable updates but computationally expensive.
- mini-batch SGD: updates the model parameters using the gradient of the loss function with 64 as batch size, its balances the stability of batch SGD and the efficiency of SGD.
- mini-batch SGD with decay: we use 10e-6 as learning rate, which decreases over time, the decease in learning rate can help in fine-tuning the model as it converges.
- SGD with decay and momentum: 10e-6 as learning rate which decreases over time and momentum to accelerate convergence, it helps in the relevant direction and dampens oscillations, this algorithm leads to faster convergence and reduced oscillations compared to standard SGD
- Adam (Adaptive Moment Estimation): combines the benefits of momentum and RMSProp, it adapts the learning rate for each parameter, with learning rate or 0.001 and batch size of 1 then 32, generally it converges faster than SGD and is less sensitive to hyperparameters.
- RmsProp (Root Mean Square Propagation): adapts the learning rate (0.001) by dividing it by an exponentially decaying average of squared gradients, we use it with batch size of 1 then 32, it’s good for non-stationary objectives and can handle sparse gradients well.

The execution time:

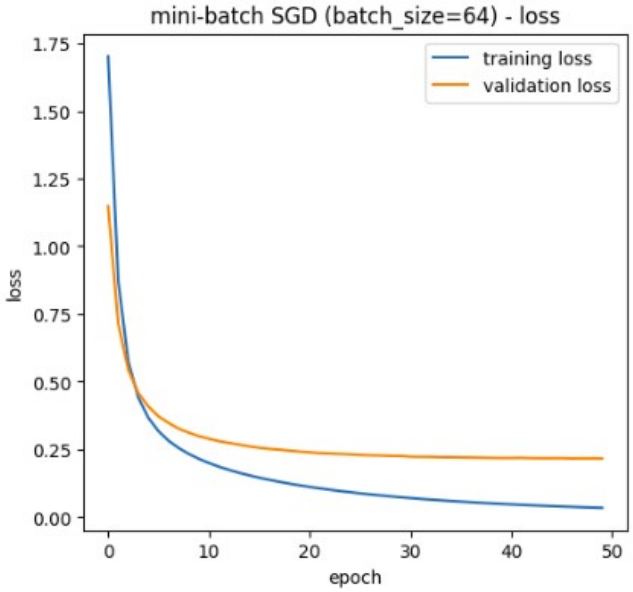
The algorithm	Batch size	Iterations	Execution time
SGD	1	6400	596 (10m)
Mini-batch	64	100	35 (0.5m)
Batch	6400	1	13 (0.2m)
Mini-batch with decay	64	100	37 (0.6m)
SGD with decay/momentum	1	6400	1081 (18m)
Adam with learning rate	1	6400	1436 (24m)
RmsProp with learning rate	1	6400	1488 (25m)
Mini-batch	32	200	65 (1m)
Adam	32	200	89 (1.5m)
RmsProp	32	200	95 (1.6m)

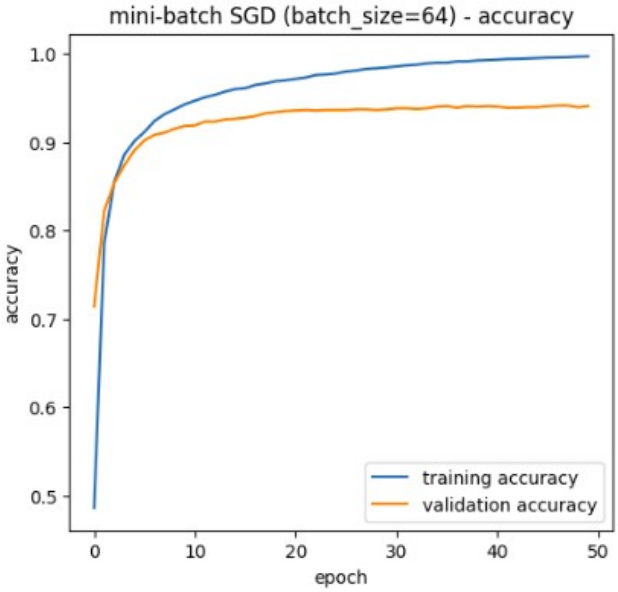
Accuracy and graphs:

The below graphs show training/validation loss and accuracy with 50 epochs

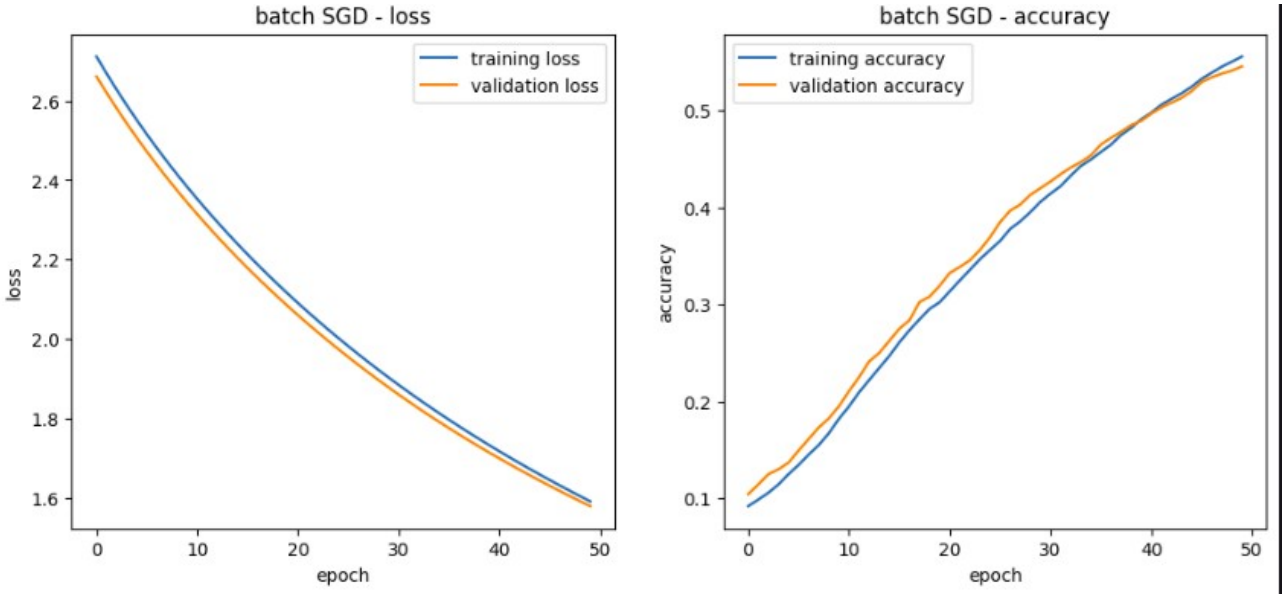


SGD with learning rate 0.01 starts with a loss around 1 but the it decreases to 0 then increases with unstable way, on the validation set we had loss arround 17.5 in the 40th epoch, the accuracy increases from 0.85 to 0.98 then deacreses to 0.7 when we achienve the 50th epoch.

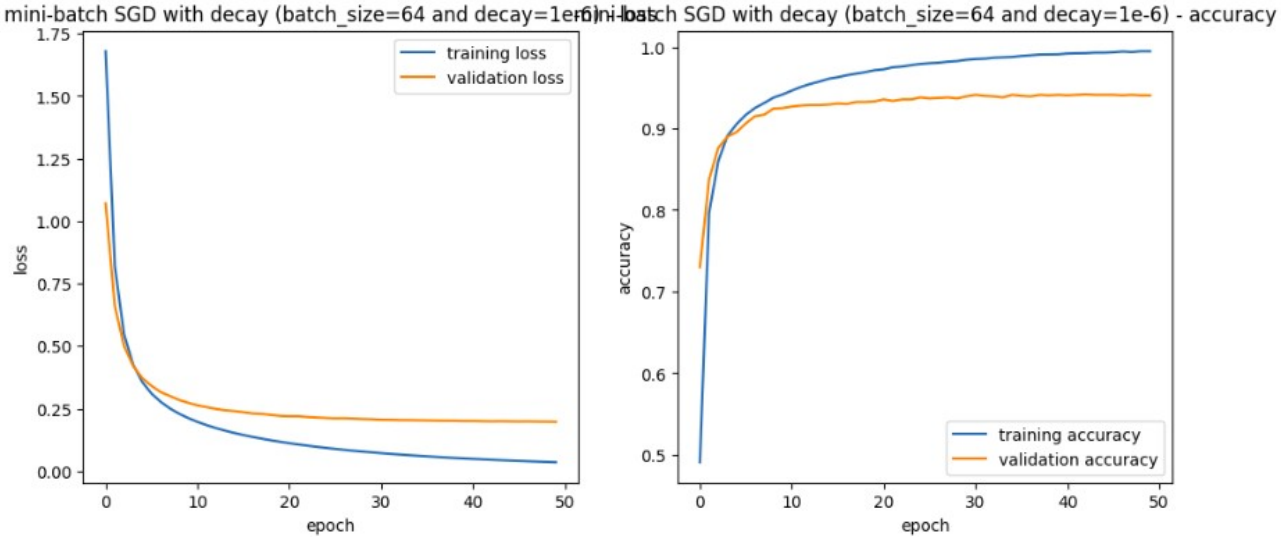




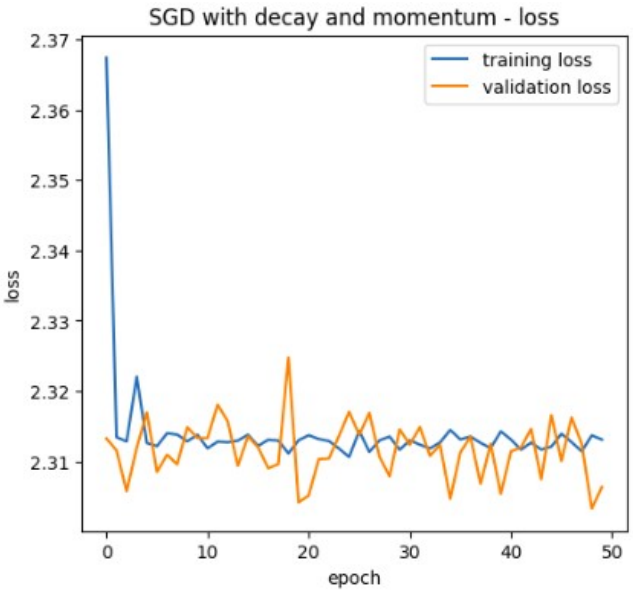
Mini-batch SGD (with batch size 64) has loss starting at 1.75 and going down to 0, the accuracy was from 0.5 to 0.9 on validation set and to ~1 for the training set, we had slower convergence highlighting noise from smaller batches.

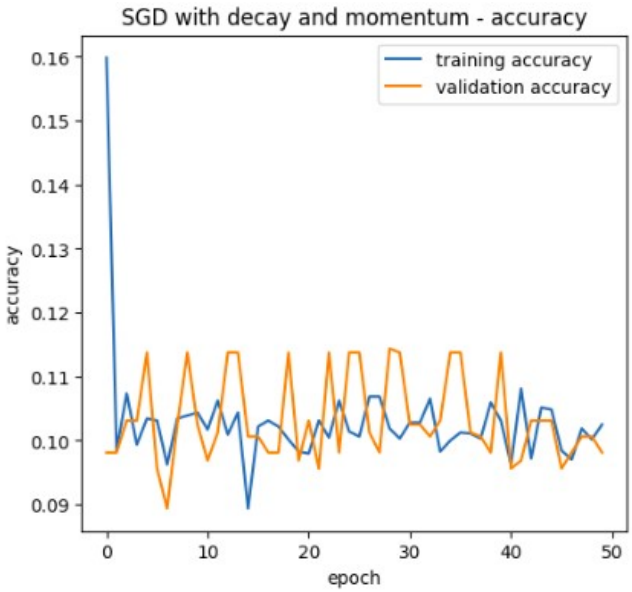


Batch sgd has loss starting from 2.7 decreasing to 1.6, the accuracy increases from 0.1 to ~0.55, it converges faster because we're using all the data in training step which lead to get global results on all the data.

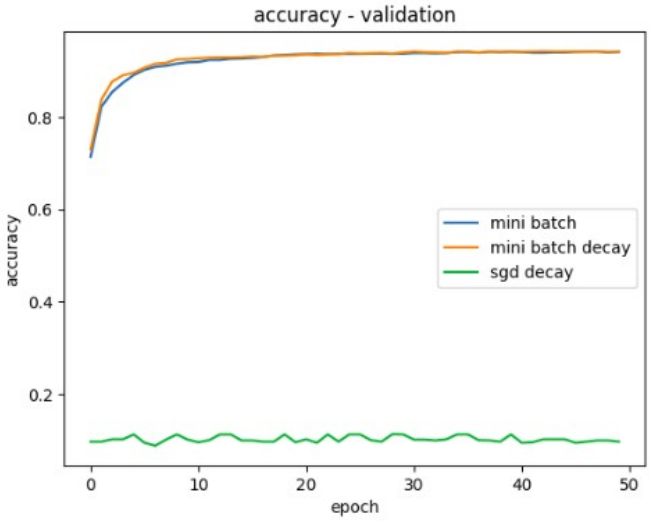


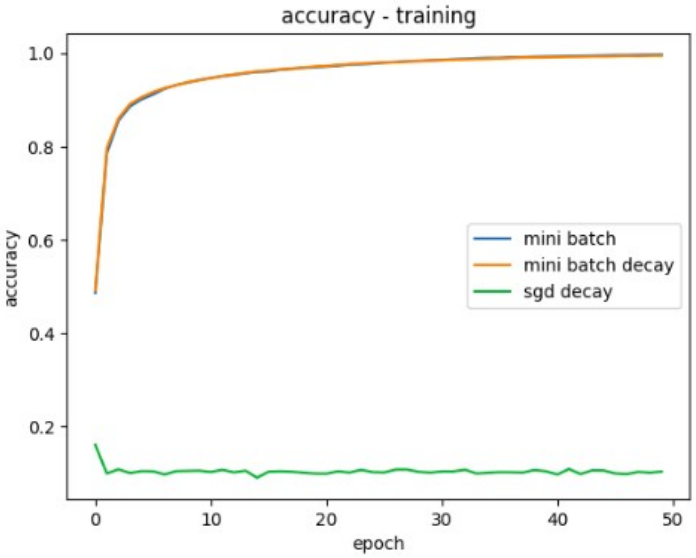
Mini-batch SGD (with batch size 64) with decay=10e-6 reached results as same as mini-batch SGD without decay, which wasn't expected because the decay helps in convergence.



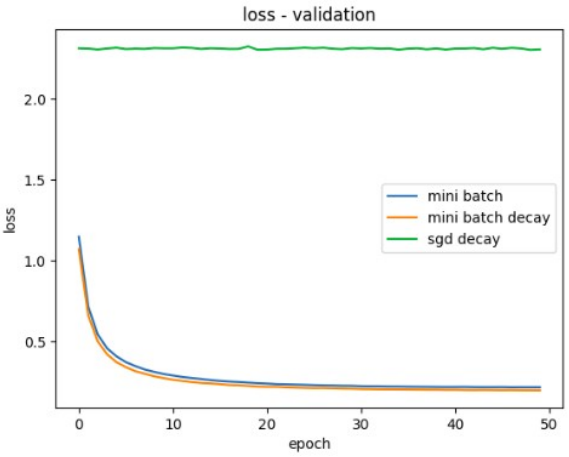


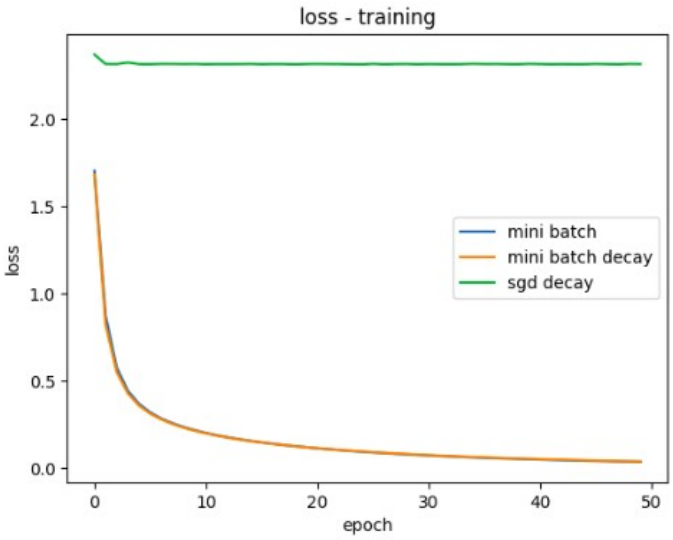
SGD with momentum and decay has loss 2.37~2.31 which is higher comparing to other algorithms, it has lower accuracy 0.09~0.16, also we have no convergence due the processing of each sample alone.



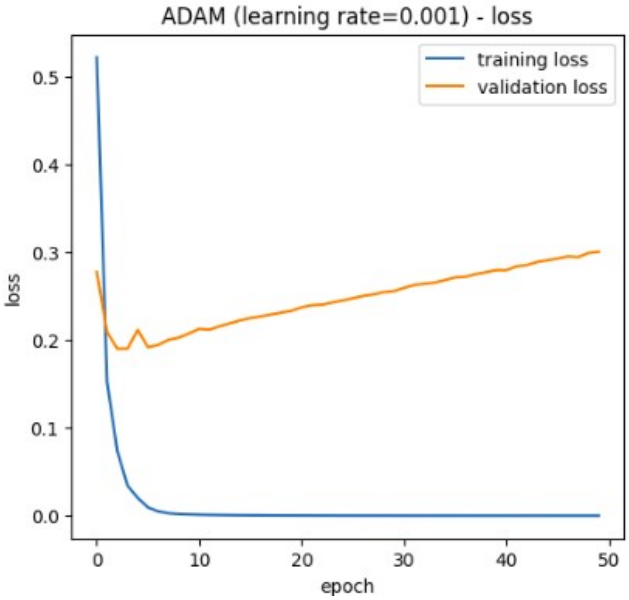


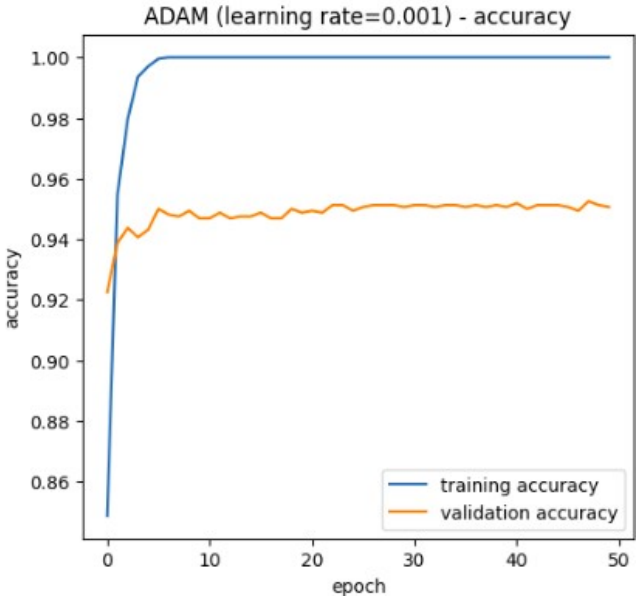
Comparing the accuracy result for the three algorithms SGD with decay, mini-batch SGD with and without decay, we get that both mini-batch algorithms acheive almost same result, 0.7~0.9 on validation set, for the SGD with decay it has values around 0, so it has the lower accuracy, maybe due the processing of each individual so it handles noises and update the weight based on them.



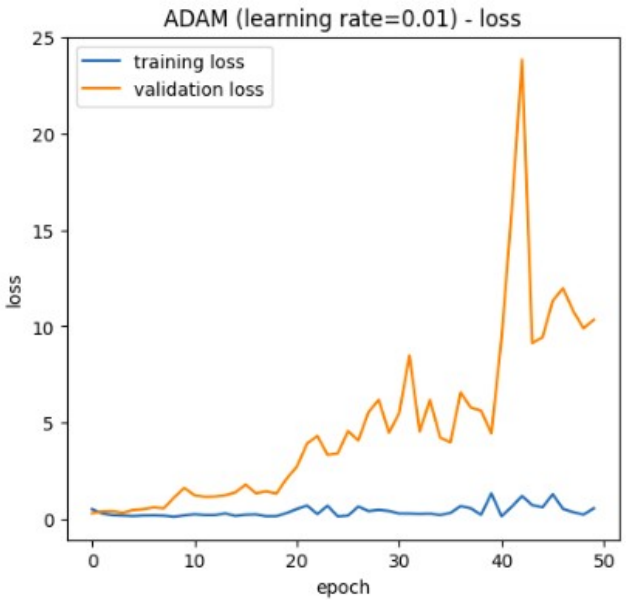


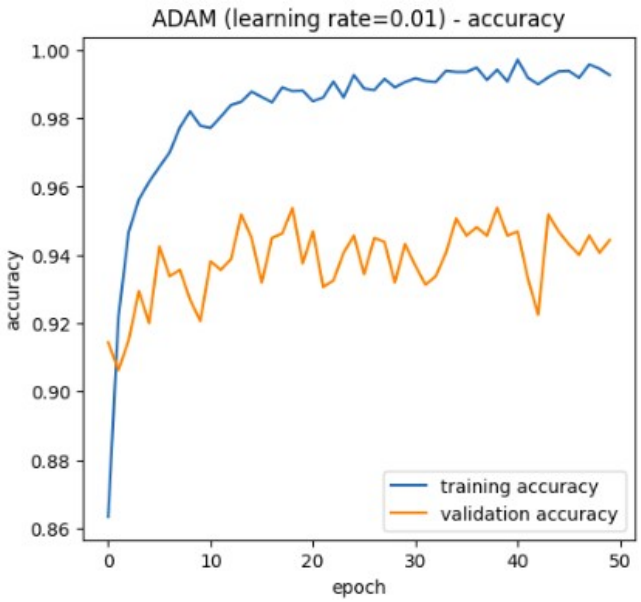
When it comes to the loss, SGD with decay has the higher loss (it's expected because it acheives the lower accuracy), the same note as the previous for mini-batch with/without decay, they have almost the same result, we have the mini-batch with decay algorithm converges faster but with small difference with mini-batch around epoch number 5.



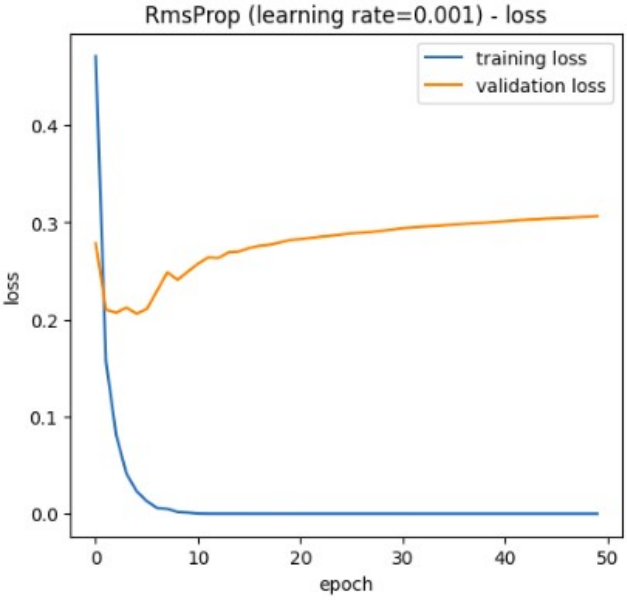


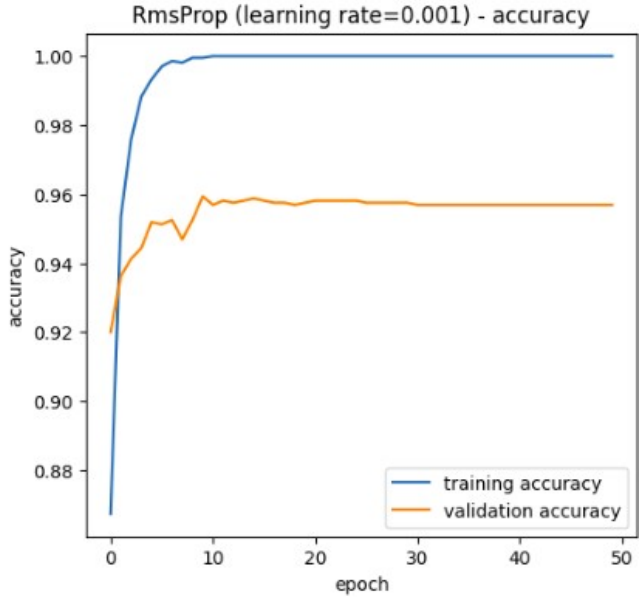
Adam with learning rate of 0.001, its training/validation loss are both going down from 0.5 to ~0 while the accuracy increases from 0.86 to ~1, for the validation set it hasn't as much as good result because the accuracy doesn't pass 0.95 and the loss suddenly inceases which may indicate to overfitting on the training set.



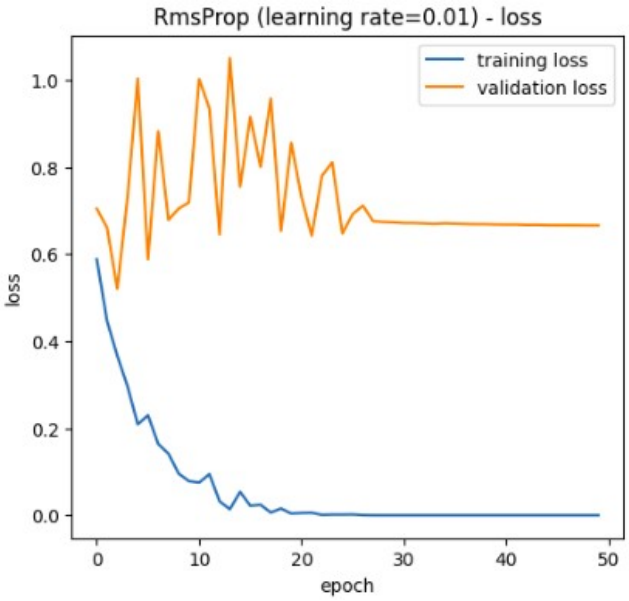


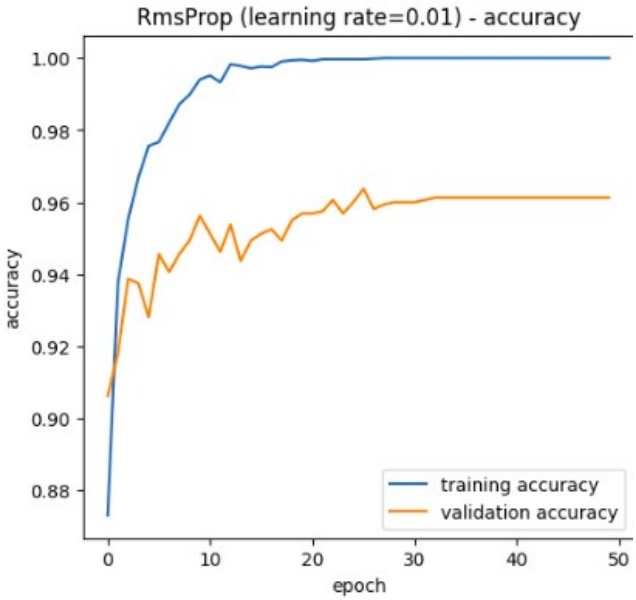
We almost have the same result for Adam with batch size = 32, except that we have oscilliation due the use of mini-batch with clear overfitting on training set where the loss was around 0 but it reaches 25 for the validation set.





RMSProp with learning rate 0.001 shows loss decreasing from around 0.4 to near 0.0 over the 50 epochs with validation loss around 0.25 which suddenly increases, it may indicates to overfitting, the accuracy starts at around 0.88 and goes up to ~1.



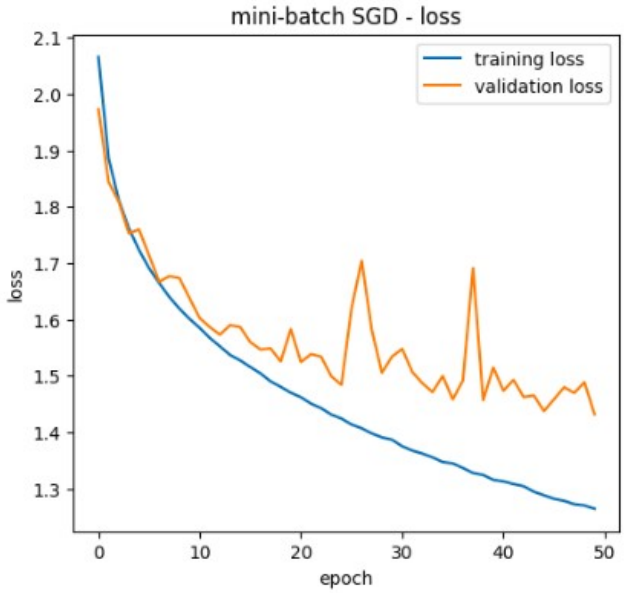


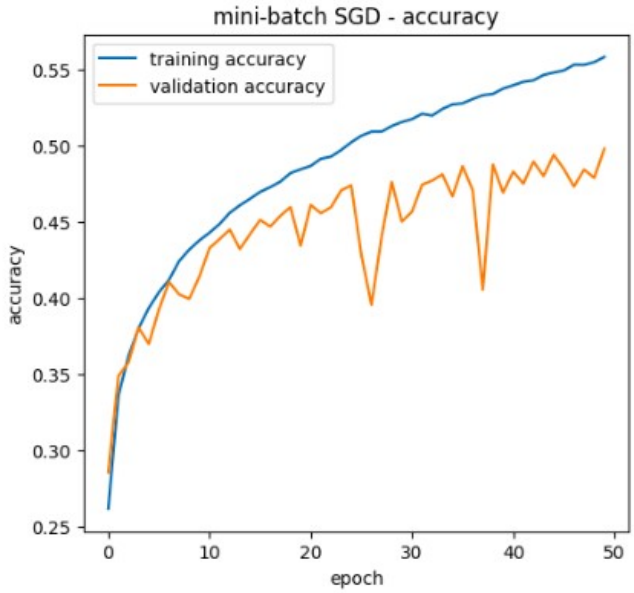
RmsProp with batch size = 32 shows clear overfitting, the both loss start around 0.65, the training one decreases until ~0 but the validation one was going up and down from 0.6 to 1 then has stable value 0.7, which indicates to oscillisation problem.

So, comparing all those models, the mini-batch with decay = 10e-6 and mini batch = 64 was the best, it has low execution time and good performance.

Part II:

-Mini-batch SGD (batch_size = 128, learning_rate = 0.01, epochs = 50):



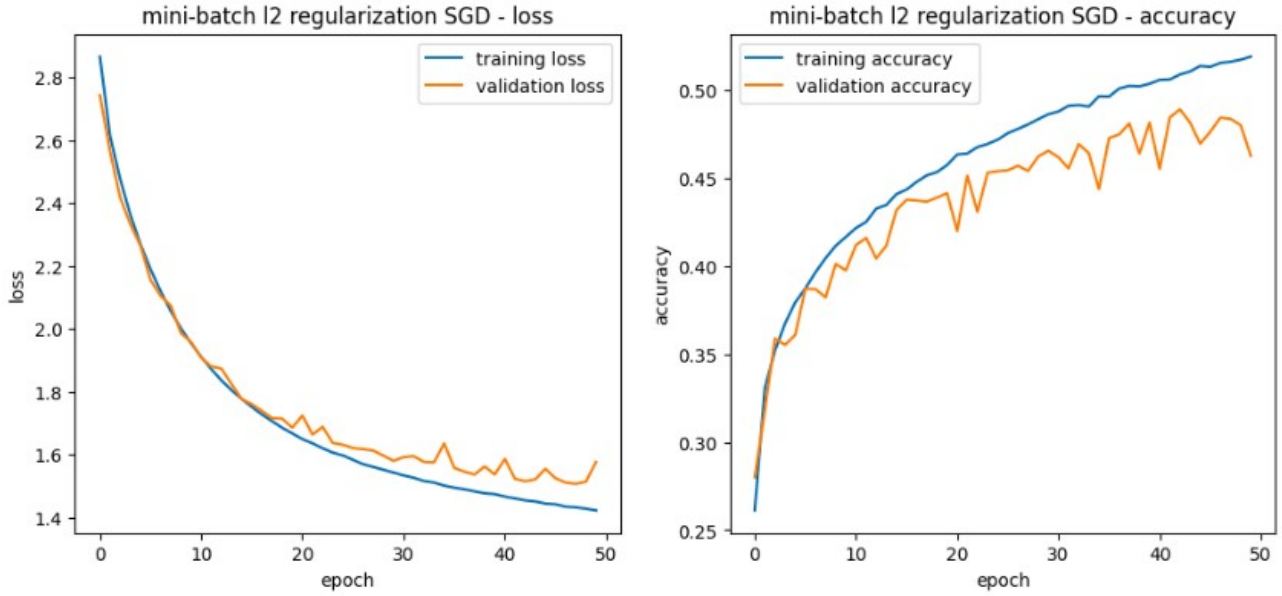


The execution time: 150 seconds (2.5min)

The loss dencreases from 2.1 to 1.3, and the accuracy increases from 0.26 to 0.55, we notice that we have overfitting because the validation set accuracy didn’t pass 0.5 with high loss (from 2 to 1.5)

-Mini-batch SGD with L2 norm regularization on the second fully connected layer:

L2 regularization (weight decay) is a technique used to prevent overfitting by penalizing large weights, it helps the model generalize better to unseen data (improves generalization), it reduces the model complexity by using small weights which ensure the model remains simple while large weight leads to overfitting



The execution time: 149 seconds (2.48min)

The loss of both validatio/training sets dencrease from 2.8 to 1.5, and the accuracy increases from 0.25 to 0.5

-Compare (model with/without L2 norm regularization):

with the L2 regularization, we have less gap between training/validation accuracy, the validation accuracy is more stable, it uses small weights which leands to better generalization.

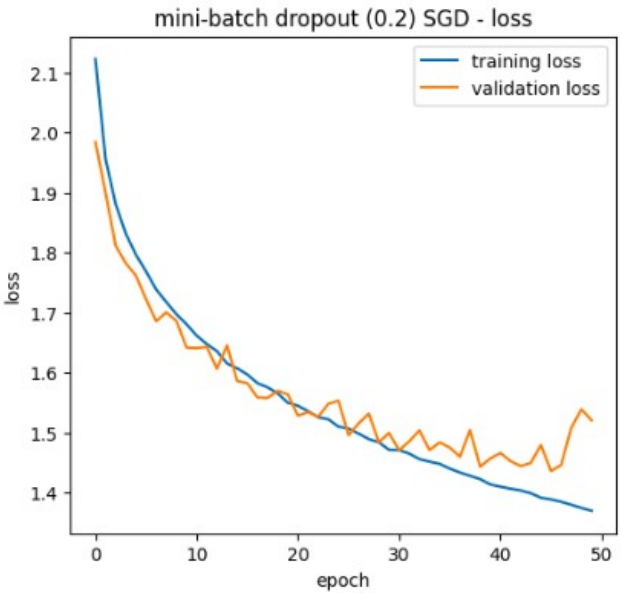
-Mini-batch SGD with dropout layer:

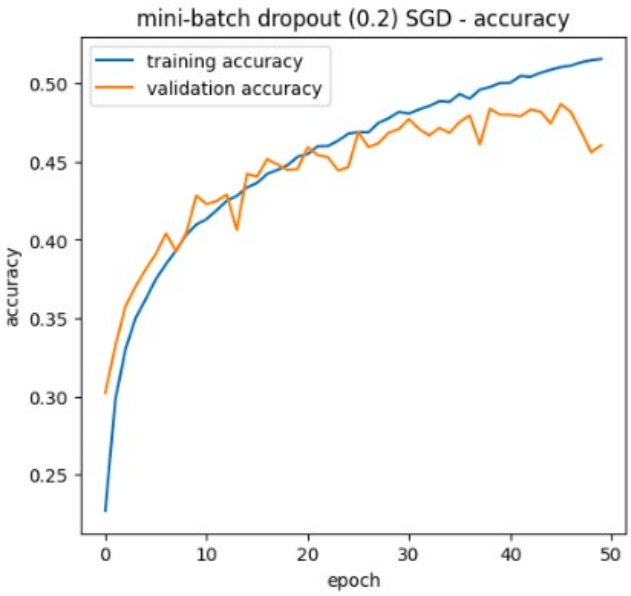
Dropout is regularization technique that helps prevent overfitting bu randomly disabling neurons during the training, means in each training steps it will remove connections, this technique will also improve generalization. The dropout rate means how many neurons will randomly dropped in each forward pass

-rate = 0.2:

The execution time: 158 seconds (2.63min)

the gap between the training and validation accuracy/loss is less, the loss decreases from 2 to 1.5 then increase to 1.6 around the 50th epoch without converging for the validation set and the accuracy increases starting from 0.3 but it did’t pass 0.5, but it improves compared to no dropout.

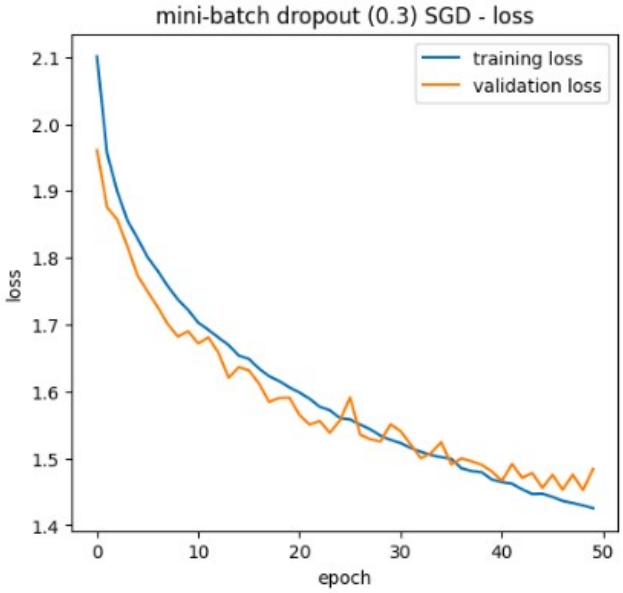


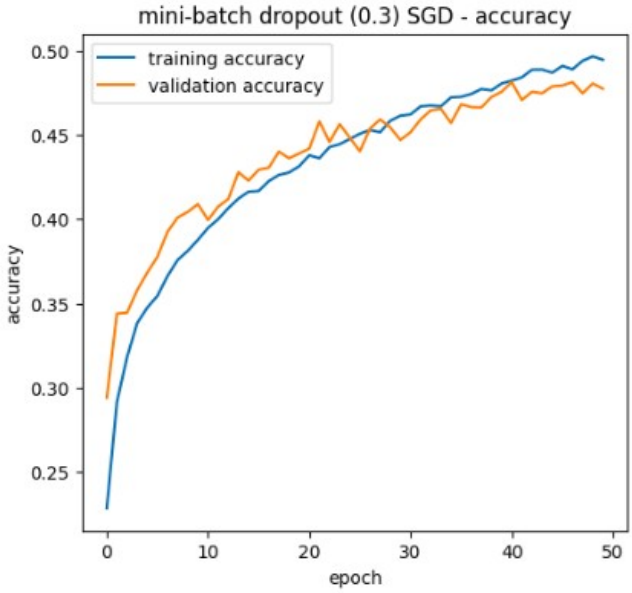


-rate = 0.3:

The execution time: 169 seconds (2.81min)

the gap between them is less, the loss decreases until 1.5 and the accuracy is closer to 0.5 for the training set

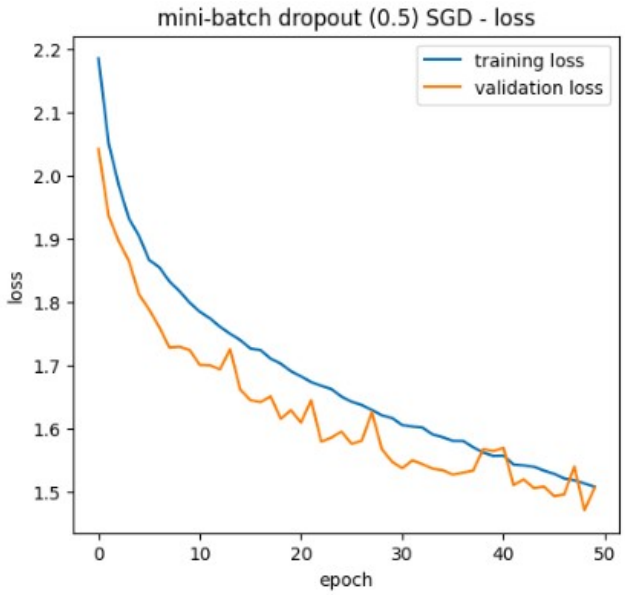


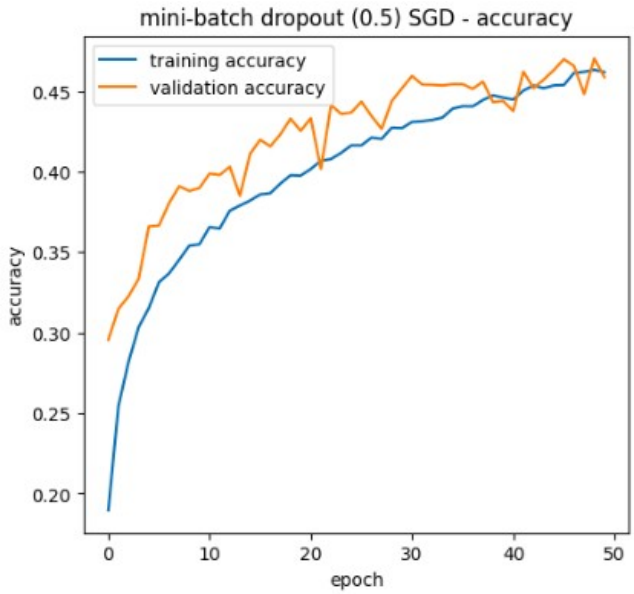


-rate = 0.5:

The execution time: 163 seconds (2.71min)

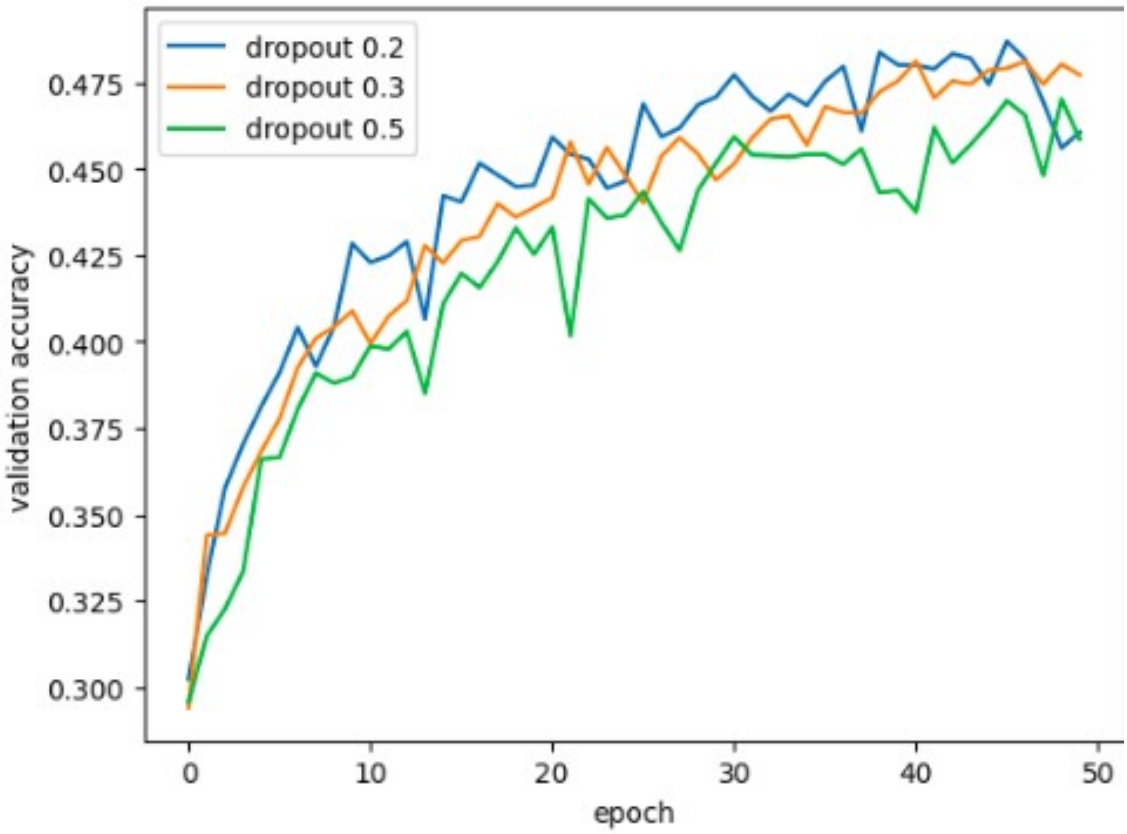
the accuracy of the validation set is higher than the training one, regulization effects can cause this because we’re using





-Compare the three models with dropout and different rate:

the validation accuracy improves using the dropout technique but using high rate like 0.5 leads to regularization effects such as validation accuracy higher than the training one, because dropping 50% of the neurons on training makes the model less reliant on specific neurons and forcing it to generalize better but turned off the dropout during the validation means we're using the full network so the best rate for our dataset is 0.3.

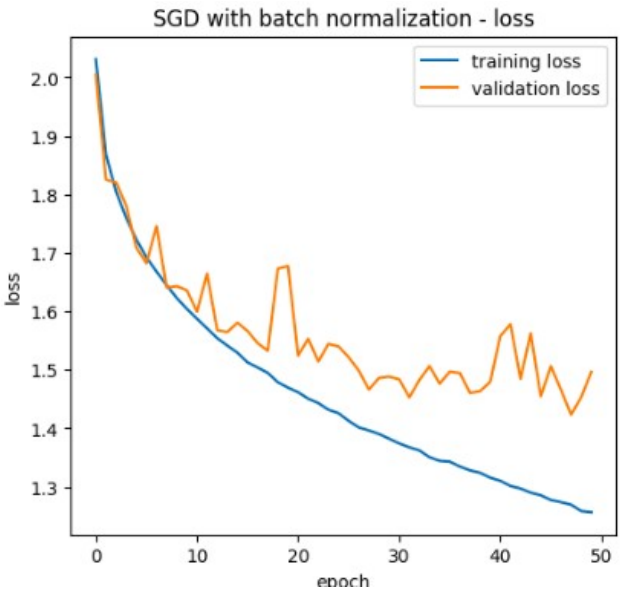


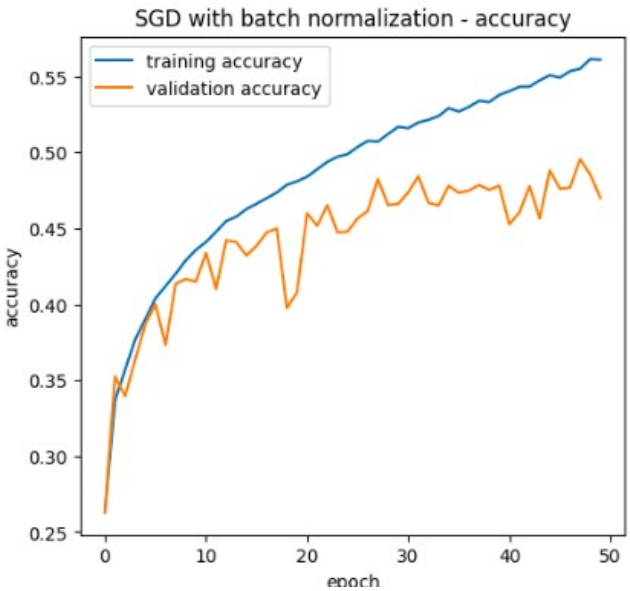
-The early stopping technique:

the training stopped at epoch 21

-Mini-batch SGD with batch normalization layer after the first hidden layer:

Batch normalization used to normalize activations making training more stable and efficient by normalizing the output of a layer across the mini-batch, not the whole dataset, then it rescale/shift the normalized values (with gamma and beta parameters)





-Random search:

from

```

*learning_rates = [0.001, 0.005, 0.01, 0.05, 0.1]*
*dropout_rates = [0.1, 0.2, 0.3, 0.4, 0.5]*
*batch_sizes = [16, 32, 64, 128]*

```

the best hyperparameters are

learning_rate = 0.005

dropout_rate = 0.4

batch_size = 32