2022 Machine Learning Assignment 3

Readme.txt
1.For Question1, I test the learning rate, batch size with Adam optimizer, then I test the SGD, RMSProp optimizer, the 3 complete different models are at the end of question 1, before question2.
2.For question2, I test the learning rate, batch size, then implement an equivalent feed forward network
3.For question3, just the code for convolutional operation

Question 1:

1)3 different types of models

Learning rate:0.0001, batch size 128, Adam

Type	train accuracy	Val accuracy	Train loss	Val loss
1	0.9814	0.9768	0.0083	0.0215
2	0.9831	0.9733	0.0582	0.0899
3	0.9825	0.9593	0.0588	0.1306

Conclusion

- As you add more filters, it increases the depth of the output image
- The higher the number of filters, the higher the number of abstractions that your Network is able to extract from image data, so the first model generates the best result
- Raw data are always noisy,initial layers learn only primitive regularities in the data
- When downsample the image through the layers. Increasing units tries to tolerate this resolutional loss by increasing model capacity
- So the number of filters are incremented so as to be able to properly encode the increasingly richer and richer representations as the signal moves up the representational hierarchy in order to avoid the bottleneck effect

2)Learning rate & batch size

Experiment: Based Adam optimizer,

Learning rate	Batch size	train accuracy	Val accuracy	Train loss	Val loss
0.0001	128	0.9977	0.9938	0.0083	0.0215
0.001	128	0.9817	0.9546	0.0592	0.1715
0.01	128	0.9787	0.7572	0.0723	1.1452
0.001	512	0.9947	0.9926	0.0177	0.0240
0.001	32	0.9898	0.9881	0.0329	0.0471

Conclusion:

- Learning rate is small(0.0001 is good), could increase the result(accuracy), but takes more time to train
- Batch size is large(512), the model could learn more so as to increase the result(accuracy),
- We need to trade off the two parameters.
- **There is a high correlation between the learning rate and the batch size**, when the learning rates are high, the large batch size performs better than with small learning rates. We recommend choosing small batch size with low learning rate.

Learning rate

- Refers to the rate at which an algorithm converges to a solution
- Specifically, increasing the learning rate speeds up the learning of your model, yet risks overshooting its minimum loss. Reducing batch size means your model uses fewer samples to calculate the loss in each iteration of learning.

Batch size:

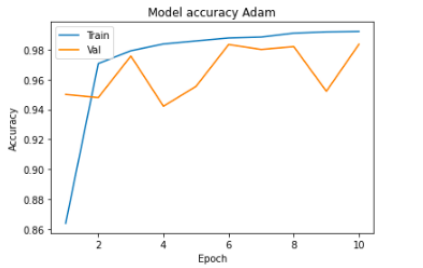
- Batch size defines the number of samples we use in one epoch to train a neural network.
- Batch normalization is a technique designed to automatically standardize the inputs to a layer in a deep learning neural network, accelerating the training process of a neural network,

3)Optimizer

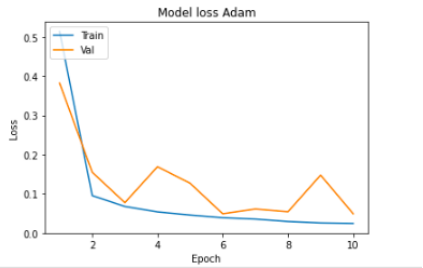
Experiment

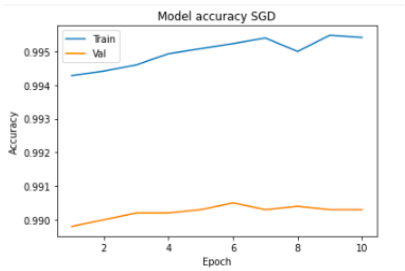
Learning rate:0.0001, batch size 128

Optimizer	train accuracy	Val accuracy	Train loss	Val loss
Adam	0.9977	0.9938	0.0083	0.0215
SGD	0.9942	0.9906	0.0182	0.0182
RMSProp	0.9969	0.9934	0.0103	0.0220

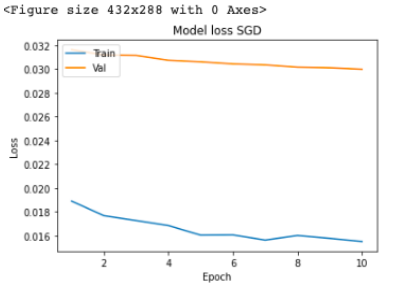


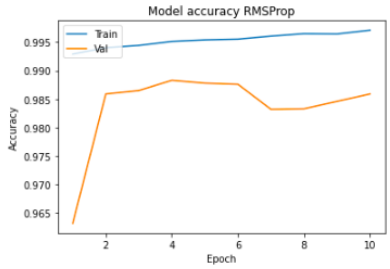
<Figure size 432x288 with 0 Axes>



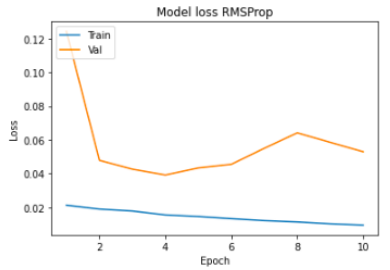


<Figure size 432x288 with 0 Axes>





<Figure size 432x288 with 0 Axes>



Conclusion:

- Adam optimizer combines the RMSProp and SGD, it would generate the best result(from the figure above)
- SGD is faster, but gets a lot of fluctuations in the cost

Adam (Adaptive Moment Estimation)

- Adam essentially combines RMSProp and momentum by storing both the individual learning rate of RMSProp and the weighted average of momentum.
- In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients

$$m_i = \beta_1 m_i + (1 - \beta_1) \frac{\partial L}{\partial \theta_i}$$

$$v_i = \beta_2 v_i + (1 - \beta) \left(\frac{\partial L}{\partial \theta_i}\right)^2$$

SGD(stochastic gradient descent)

- subtracts the gradient multiplied by the learning rate from the weights.

$$\theta_i = \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$$

RMSProp(root mean square prop)

1. If the gradients are consistently large, the values of v_i will increase, and the learning rate will decrease. This adaptively adjusts the learning rate for each parameter and enables the usage of larger learning rates

$$v_i = \beta v_i + (1 - \beta) \left(\frac{\partial L}{\partial \theta_i}\right)^2$$

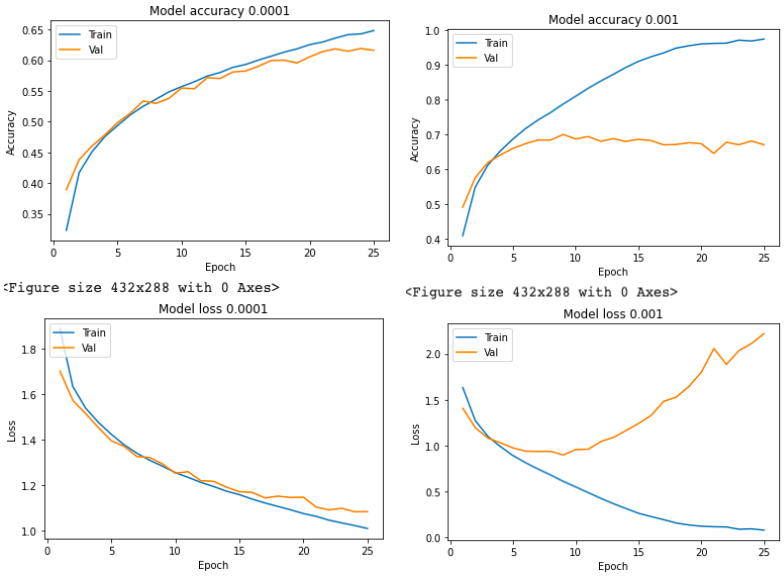
$$\theta_i = \theta_i - \alpha \frac{\left(\frac{\partial L}{\partial \theta_i}\right)}{\sqrt{v_i} + \epsilon}$$

Question 2:

Experiment: Based Adam optimizer,

Learning rate	Batch size	train accuracy	Val accuracy	Time cost(each epoch)	Issue
0.00005	128	0.6068	0.5877	99	
0.0001	128	0.6481	0.6161	63	
0.001	128	0.9727	0.6695	95	Overfitting
0.01	128	0.6519	0.4648	93	Overfitting
0.001	512	0.5802	0.5627	88	Slow
0.001	32	0.7907	0.6760	74	Overfitting

1. What is the effect of learning rate on the training process? Which performed best?
   - Affect the result accuracy, 0.01 is with 0.4638, 0.0001 with 0.6161, also 0.01 and 0.001 are overfitting
   - Affect the training time, usually smaller learning rate need more time to train, 00005 needs 99s for each epoch



2. What is the effect of batch size on the training process? Which performed best?
   - Larger batch size need more time to train, size 512 needs 88s, size 32 needs 74s
   - a higher batch size does not usually achieve high accuracy

3. Try different hyperparameters to obtain the best accuracy on the test set. What is your best performance and what were the hyperparameters?
   - Learning rate: 0.0001
   - Batch size :128
   - Accuracy:0.6161
   - **There is a high correlation between the learning rate and the batch size**, when the learning rates are high, the large batch size performs better than with small learning rates. We recommend choosing small batch size with low learning rate.

4.Feed forward neural networks are artificial neural networks in which nodes do not form loops. In this network, the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any) and to the output nodes.

Based on the experiment, the conclusion are listed below:
   - It is unnecessary, the fully connected layers may learn the background noise, the CNN could learn the useful information from the image using filter

- It requires large amounts of data in order to function properly, as well as a high level of computational power that is required to facilitate such functionality
- It has `10,325,030` parameter, takes longer time `137ms/step`, and the results are not good(Accuracy 0.053)
- Since the NN finds a general approximation of a solution, there is a small error usually associated with all the NN outputs

Additionally ,CNN is a feed forward neural network that is generally used for Image recognition and object classification.

Question 3:

1.What are the dimensions of the input and the kernel (or filter)? How many parameters are there in the kernel f? [2 points]

Input: 6x6
Filter: 3x3
Parameters: (3x3x1) +1 = 10, based on ((w * h * d)+1)* k), where: width w, height h, previous layer's filters depth d, k is the filter count

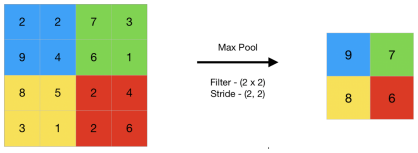
2. What is the output activation map when you apply the convolutional operation using the filter f on the input X without padding? [4 points]

The code is on the .ipynb

```
X*f output(With out Padding):
[[ 16   9  -4 -18]
 [ 17  -5 -10 -12]
 [ 11  -9 -17   2]
 [  9  -1 -15  16]]
```

3. What is the output when you apply a max-pooling operation on the output from the previous question? [4 points]
Pooling layer filter 2x2, stride 2

```
[[7,-4],
[11,16]]
```



Reference:

1) https://ruder.io/optimizing-gradient-descent/
2) https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008
3) https://towardsdatascience.com/exploit-your-hyperparameters-batch-size-and-learning-rate-as-regularization-9094c1c99b55#:~:text=When%20learning%20gradient%20descent%2C%20we,in%20each%20iteration%20of%20learning.
4) https://www.baeldung.com/cs/learning-rate-batch-size
5) https://keras.io/api/optimizers/