Question 1

Part1:

For part one I tried different LR's of 0.1, 0.01, 0.001. 0.1 and 0.01, did not perform well at all with any of the three optimizers, Adam, SGD, and RMSProp. However I did manage to get the model to train at 0.001, with batch size at 128 and 25 epoches.

Adam
accuracy: 0.9965
Test loss: 0.057167619466781616
Test accuracy: 0.9882000088691711

SGD:
accuracy: 0.9930
Test loss: 0.048220593482255936
Test accuracy: 0.9846000075340271

RMSProp:
accuracy: 0.9943
Test loss: 0.08929948508739471
Test accuracy: 0.9871000051498413

Since the accuracy was at 99% I did not explore different batch sizes, also since the model was not increasing in accuracy from the 20th epoch to the 25th epoch, I didn't change the epoches to more epoches.

Part 1.2

Part 2 had very similar results, and again it seems like the batch size and amount of epoches I have picked worked out just fine.

Adam
accuracy: 0.9949
Test loss: 0.038366544991731644
Test accuracy: 0.9883000254631042

SGD: 0.9856
Test loss: 0.04696971923112869
Test accuracy: 0.984499990940094

RMSProp:
accuracy: 0.9918
Test loss: 0.06024818867444992
Test accuracy: 0.9876999855041504

Part 1.3
Part 3 has the same conclusion as part 1 and part 2.

Adam:
accuracy: 0.9952
Test loss: 0.04177796095609665
Test accuracy: 0.9890000224113464

SGD:
accuracy: 0.9844
Test loss: 0.08808410912752151
Test accuracy: 0.9700000286102295

RMSProp:
accuracy: 0.9946
Test loss: 0.06202826648950577
Test accuracy: 0.9878000020980835

Question 2

Part2.1: What is the effect of learning rate on the training process? Which performed best?

A LR of 0.01 was to much of a learning rate for the model to train with results of:

accuracy : 0.2486
Test loss: 129.45310974121094
Test accuracy: 0.2443999946117401

(note: The parameters for this is, batch size of 128 and 100 epoches, also I only used adam for this part since it seemed to perform better when testing at lr of 0.001.)

A lr of 0.001 seemed to do well, at 25 epoches and batch size of 128:
Adam: 0.5887
Test loss: 1.2418910264968872
Test accuracy: 0.5742999911308289

SGD: 0.1000
Test loss: nan
Test accuracy: 0.10000000149011612

RMSProp:0.5705
Test loss: 1.2877159118652344
Test accuracy: 0.5509999990463257

100 epoches, batch size of 128, lr 0.001, using adam:

accuracy: 0.6459
Test loss: 1.1973625421524048
Test accuracy: 0.598800003528595

1000 epoches, batch size of 128, lr 0.001, using adam:
accuracy: 0.6702
Test loss: 1.029354245464565
Test accuracy: 0.6225636562454

Any LR lower than 0.001 seemed to take too long to train, as it was not learning fast enough.

Part2.2: What is the effect of batch size on the training process? Which performed best? A smaller batch size takes longer to train. The bigger the batch size here the better the accuracy.

Lr 0.001, adam, 100 epoches:
Batch size 2:
accuracy: 0.3634
Test loss: 1.9981257915496826
Test accuracy: 0.3172999918460846

Batch size 4:
accuracy: 0.4553
Test loss: 1.6594843864440918
Test accuracy: 0.433499991893768

Batch size 8:
accuracy: 0.5266
Test loss: 1.51930570602417
Test accuracy: 0.4875999987125397

Batch size 16:
accuracy: 0.5771
Test loss: 1.2947492599487305
Test accuracy: 0.5590999722480774

Batch size 32:
accuracy: 0.6125
Test loss: 1.2311395406723022
Test accuracy: 0.5827999711036682

Batch size 64:
accuracy: 0.6345
Test loss: 1.2209553718566895
Test accuracy: 0.5906000137329102

Batch size 128:
accuracy: 0.6445
Test loss: 1.1978529691696167
Test accuracy: 0.5909000039100647

Part2.3
accuracy: 0.92%
train loss: 0.038
Test accuracy: 70%
Parameters are in note book

Part2.4
accuracy: 0.7031
Test loss: 2.2949001789093018
Test accuracy: 0.4402000010013580

LeNet:
(cov =)
Conv1: 1520
Conv2: 20040
Conv3: 60060
FC1: (300*84)+1*120 = 5124
FC2: (84*10)+1*10 = 850
Total params: 87,594

Fully connected:
Fc1: 24
Fc2: 112
Fc3: 2040
Fc4: 645204
Fc5: 850
Total params: 648,230
(got with print(model.summary()))

No, the model started to over fit and the test accuracy is worse.

Question3:

Part3.1: What are the dimensions of the input and the kernel (or filter)? How many parameters are there in the kernel f?

Input: 6x6
Kernal: 3x3
# of parameters are 9

Part3.2: What is the output activation map when you apply the convolutional operation using the filter f on the input X without padding?

The output will be:
C = ((n-f+2p)/s+1
C=((6-3+2x0)/1)+1=4
Used online matrix calculator:

18	9	-4	-18
17	-3	-10	-12
11	-9	-17	2
9	-1	-15	16

Part3.3: What is the output when you apply a max-pooling operation on the output from the previous question?

Used online matrix calculator:

18	9
17	16