CS5242 Assignment 2

Zhu Jin
E0047338

Part 1. Mathematical explanations

Convolution forward: Let kernels be an matrix K of size F, C*KH*KW and input M be a matrix with size C*KH*KW, OH*OW*N reshaped from input matrix X based on information such as kernel size, padding and stride. The intermediate output is out' = K x M. Final output is obtained by transposing and reshaping out' into a shape of N, F, OH, OW.

Convolution backward:
1. Let shape of dout be N, F, OH, OW. As the bias is added to each of our filter, db is calculated by accumulating the gradients to the dimension that represent of the number of filters, the second dimension of dout. Hence the sum is operated on all axis except the second.
2. Let dout' be a matrix with size F, OH*OW*N obtained by reshaping dout. Then dw = dout' x transpose(M) where M is of size C*KH*KW, OH*OW*N as described in forward operation.
3. dx is obtained by doing a full convolution on dout with flipped kernel and then remapping the result to a matrix with the same shape as input matrix X.

Maxpooling forward:
Input matrix X can be reshaped into a matrix P of size C, PH*PW, OH*OW*N. The max pooling result can be obtained by taking the max along axis 1 and reshaping the matrix into a shape of N, C, OH, OW.

Maxpooling backward: There are two cases for dx. For x that is a max neuron during max pooling operation, dx = dout, since x = out during forward operation. For x that is not max, dx = 0, since a small change in such x won't affect max value and hence won't affect out.

Dropout forward: During testing phases, no dropout is performed. During training phases, nodes will be active with probability 1-p. For nodes that remain active, their values is scaled up by 1/(1-p), which is expressed by out = x/(1-p)

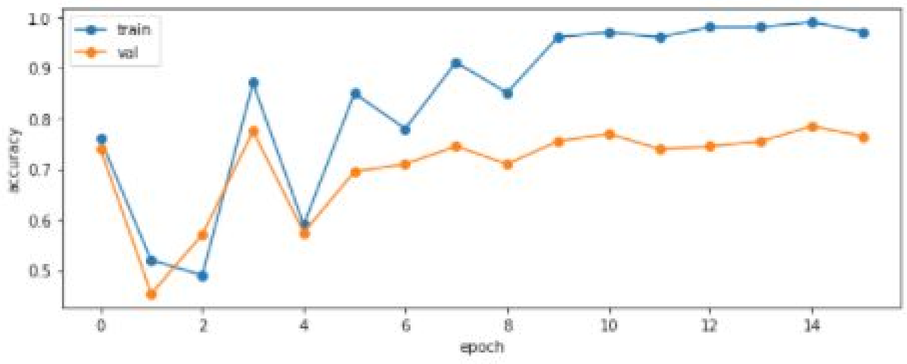
Dropout backward: During testing phases, dx = dout since no dropout was performed earlier. During training phases, for nodes that remained active in forward operation, dx = dout/(1-p). For nodes that got dropped, dx = 0, as there's no information flow through these nodes.

Regularisation:
1. Regularisation adds a regularisation term to final loss. Let original cost be c. Let l2-norm regularisation term be r = 0.5* lamda * $||W||^2$. Total lost L becomes c + r.
2. As changes in b or x won't affect regularisation term, db and dx will not get affected by regularisation. In the meantime, since dL/dw = d(c+r)/dw = dc/dw + dr/dw, we need to update dw by adding dr/dw = lamda * w.

Part 2. Sanity Check

Sanity check loss: Initial loss without regularization is 2.30258789799. With regularization where lamda = 0.5, the loss increases to 2.50854087057 as expected.

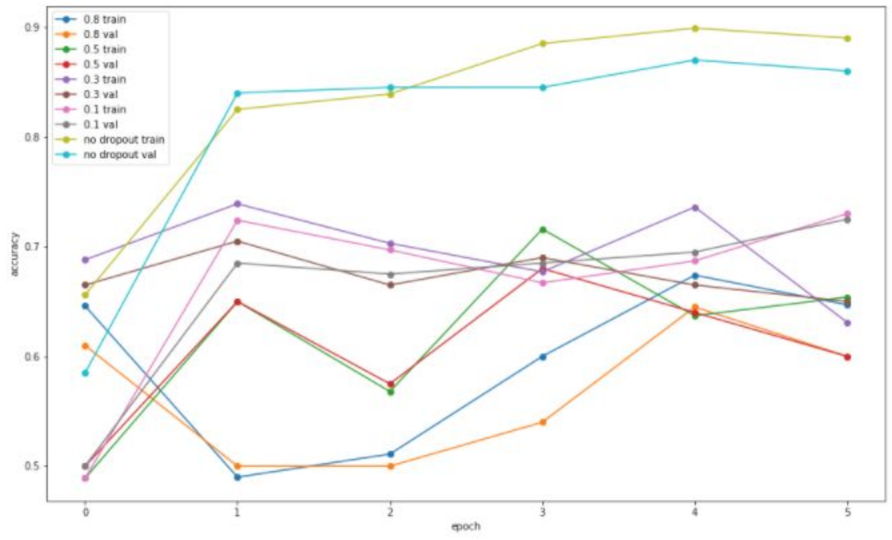
Overfit small data: The model overfits small data as shown by the accuracy gap shown below.



Part 3. Design decisions

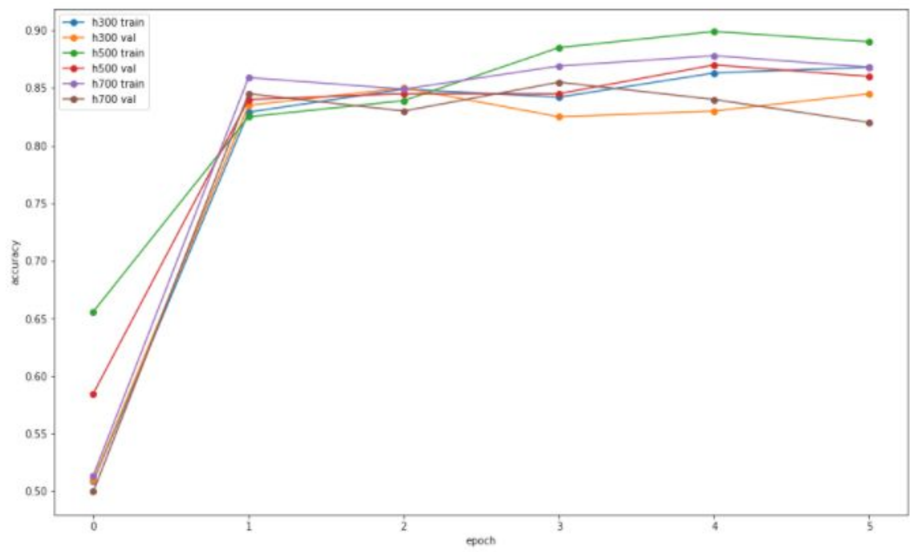
In this part, we compare various models against our base model, which has num_filters = 32, filter_size = 7, hidden_dim = 500 and reg=0.001 and achieves about 80% accuracy on the validation set after one epoch.

Dropout rate (0.1-0.3-0.5-0.8): There are two dropout layers, one after conv_pooling_relu layer and the other after hidden-affine-relu layer. It is obvious that when dropout rate is too high, the result tends to be bad. This is expected as useful information may get lost after going through two dropout layers with high dropout rate. In general, dropout does not improve accuracy compared to the base model. This may be explained by the fact that the base model without dropout does not suffer a lot from overfitting as shown in the plot. Therefore, introducing dropout to our base model may not add too much value and potentially drop out important information during training.

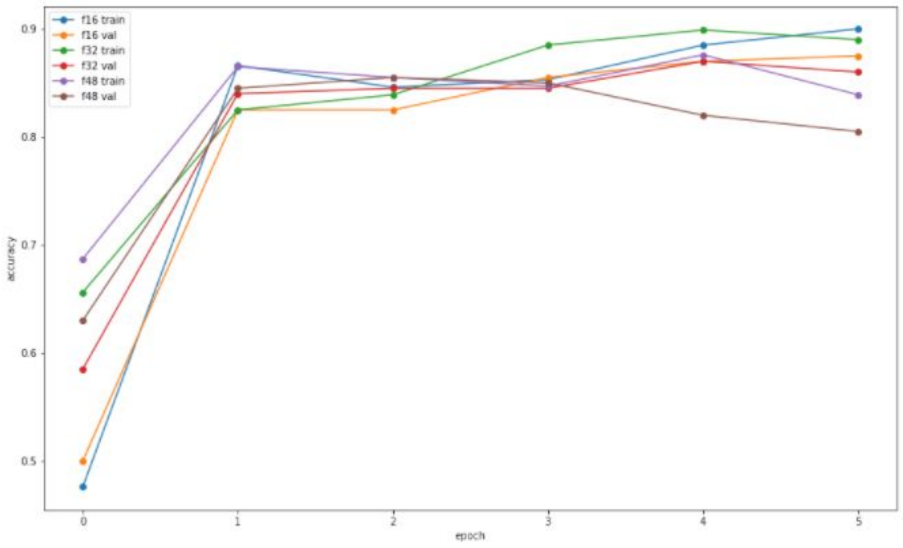


Num of hidden nodes (300, 500, 700):

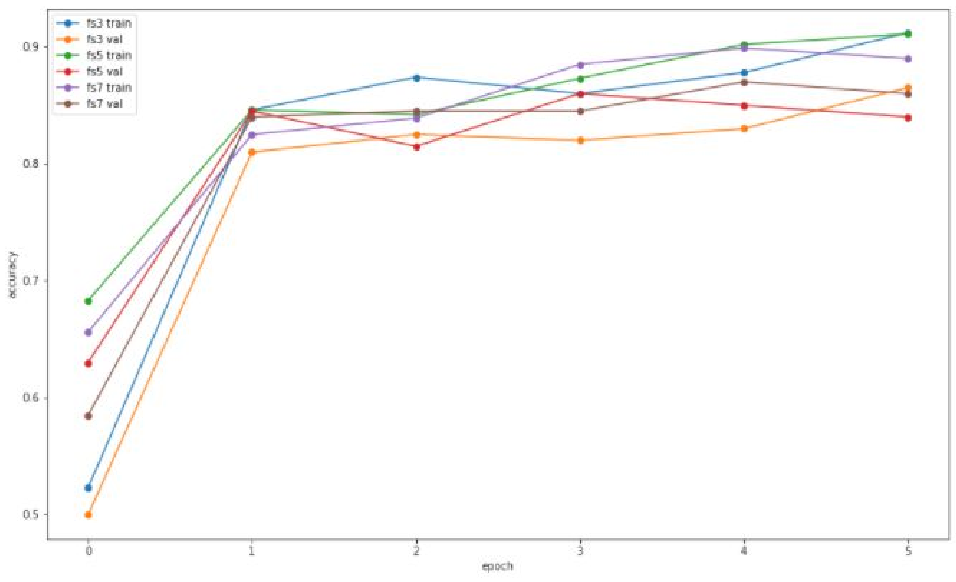
The base model which has hidden_dim = 500 gives the best result in this experiment. When hidden_dim = 700, we can observe that the network is overfitting from the gap between purple line and brown line. When hidden_dim = 300, the network does not have high accuracies in both training and valuation phases.



Number of filters (16-32-48):
From the plot below, we can see that when there are two many filters such as 48 filters, the network tends to suffer from overfitting. Based on the plot below, it seems reasonable to reduce the number of filters compared to the base model as it will give similar or higher accuracy but much less calculation complexity.

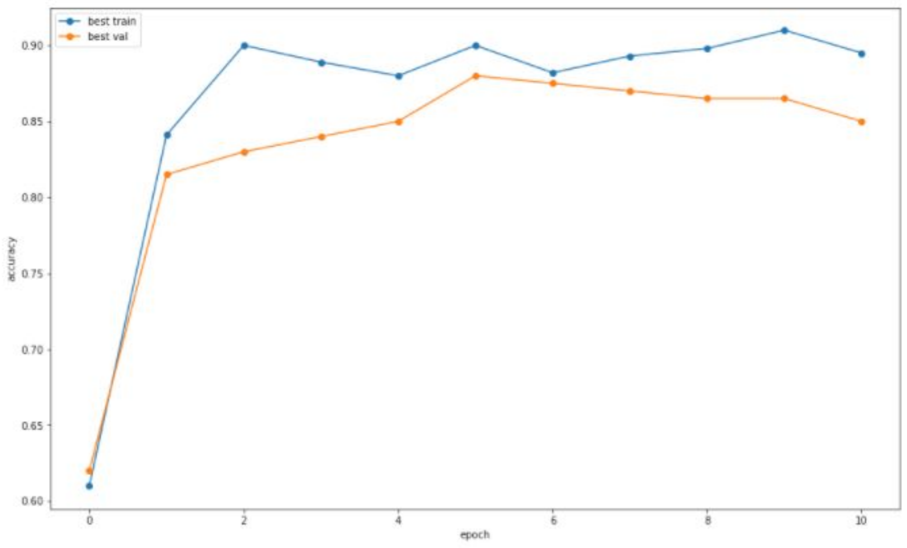


Filter size (3-5-7):
Filter sizes 3 and 5 results in good training accuracy but not good test accuracy. This is because a smaller filter size is able to capture more information and hence more vulnerable to overfitting.



Best net:

Based on the above experiments, we keep most of the parameters in our base model the same except that we update num_filters from 32 to 16. This gives a test accuracy of 85% after running 10 epochs.


best train
best val
accuracy
epoch