Deep Learning and its Applications

CS671

Course Instructor : Dr. Aditya Nigam

Assignment 2 : Question 2

Multi-Head Classification

Group 13

Aman Khandelwal	B16007
Amrendra Singh	B16010
Bharat Lodhi	B16015

https://github.com/ashking13th/deepLearning

https://students.iitmandi.ac.in/~b16010/CS671_grp_13/

1 Problem Statement

Design a non-sequential convolutional neural network for classifying the line dataset(used in assignment-1). This network will have 4 outputs based on the 4 kind of variations(length, width, color, angle). You are required to divide your network architecture into two parts:

- Feature network
- Classification heads

The feature network will be responsible for extracting the required features from the input and attached to it would be the four classification heads one for each variation The first 3 classification heads are for 2 class problems namely length, width and color classification. In all these the final layer contains a single neuron with a sigmoid activation followed by binary crossentropy loss. The last classification head is a 12 class problem for each 12 angles of variation. In this the final layer contains 12 neurons with softmax activation and Categorical Cross entropy loss.

2 Details about Data

The dataset used for this problem consist of lines with 2 possible lengths, 2 possible widths, 2 possible colors and at 12 possible angles. So in total considering all variations there will be 96 classes. Now if considering variation in length only there will be 2 classes, similarly for width and color there will be 2 classes and for angle there will be 12 classes.
The images in the dataset had 28x28 pixels with 3 channels(RGB). For the training of the network in this problem we have, 70%-30% train-test split.

3 Some Acknowledgments

For this assignment as it was allowed to use high-level APIs, so we've used Keras Fuctional API. Also it was ensured that network was end-to-end trainable, for the proof I checked the number of total number of parameters involved in the architecture and it was coming according to the definition of end-to-end trainable network.

4 Network Architecture

Firstly we shall assign each classification head to some property of the line. So that is done as following-

- Classification head-1 – Classify on the basis of length

- Classification head-2 – Classify on the basis of width
- Classification head-3 – Classify on the basis of height
- Classification head-4 – Classify on the basis of angle

So now the basic idea about network architecture. We have common shared network for all four classification heads till we are extracting the feature map from the image and then separate FC layer network for all classification heads.

The network upto the feature map was kept same in all the variations tried, actually it was inspired from the VGG Net. Below is the architecture for the VGG Net.

So the key observations by us about this net which we implemented in our network too are following-

- We have alternate two 2-D Conv layer and then 2-D max-pooling.
- We have kept the feature map size always same in convolution by appropriate padding.
- We have used Batch Normalization before applying activation to the current feature map, in our case before applying Relu.
- While using Conv layers, the consecutive layers were same and the one after max pooling had double the number of filters then the previous Conv layers.

Now the variations tried-

- Figure 1 is the very first network architecture we implemented.

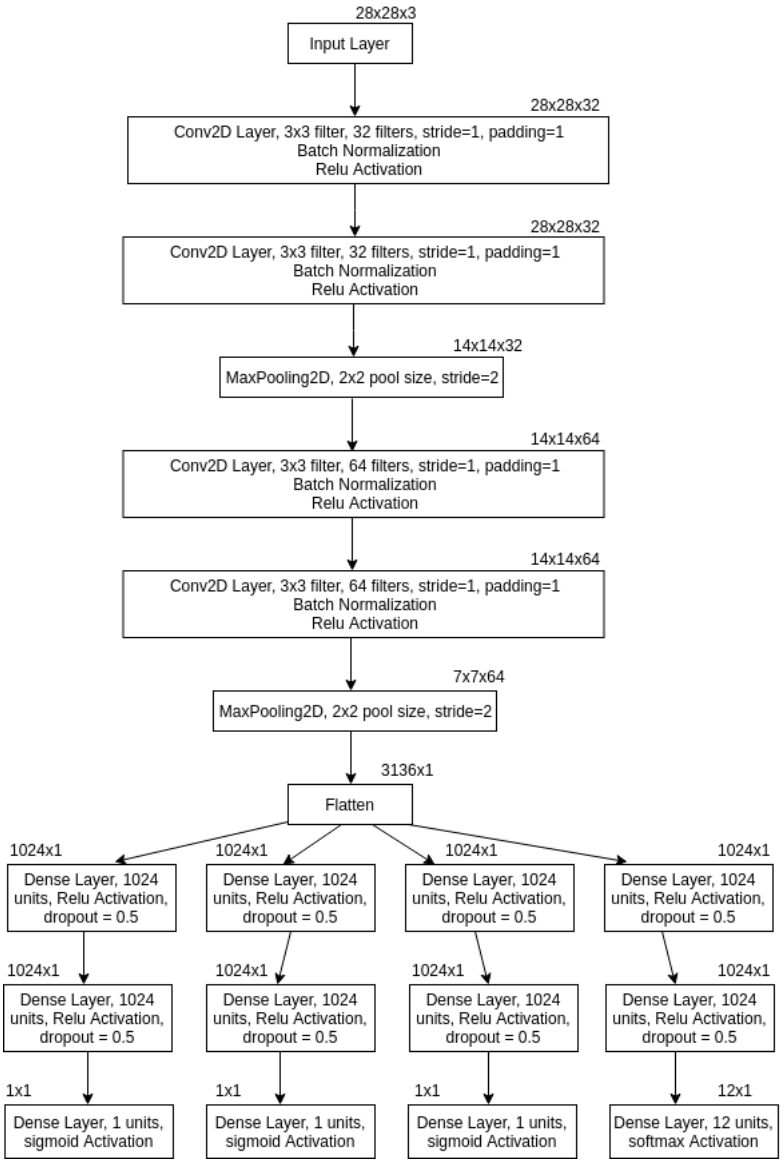

Figure 1. Network(Model) Architecture for the first variation tried

In this, the architecture for the FC network is same for all 4 classification heads, also we involved the dropout factor for introducing regularization in the network and prevent it from over-fitting. The number of epochs were kept 5 with batch size of 64 for training with Adam optimizer. But this network didn't work well. Accuracies obtained are following:

- Classification head-1 – 89%
- Classification head-2 – 95%
- Classification head-3 – 96%
- Classification head-4 – 45%

The low accuracy in the last classification head can't be attributed to overfitting because if there was overfitting then the first it will affect the above three classification heads. The latter argument is given on the basis of the fact that we've more variation in the samples in angle-based classification than the other three. So the reasoning for this low accuracy can be that the dropout involved was large in respect to the classification head-4 and it was not able to capture all the variations.

- In the second Network Architecture variation we tweaked the first one by reducing the dropout for the 4th classification head and there was not any significant improvement in the performance.
- Now, as there were only two classes for the first 3 classification heads so we removed the last hidden layer from the 3 classification heads keeping rest of the architecture same, thinking that it will not impact the performance much. But actually there was significant improvement in performance for the first three classes the accuracies jumped to 99%, 98% and 100% repectively. So this implies that the inference drawn for first architecture was wrong. Actually the two FC layes were causing overfitting in the network. This motivated us to try the similar thing for the 4th classification head.
- Now, the second hidden layer in FC network of 4th classification head was removed from the above architecture and this gave the performance boost for the same. The accuracy increased from somewhere close to 50% to 85%.
- On changing the filter size of one of the two consecutive Conv2D layer the accuracy decreased.
- Now, as there was dropout of 0.5 for all four classification heads and accuracy for the three classification heads was pretty good and they were having similarity of capturing information about two classes and the 4th classification head was capturing variation for 12 classes so it was obvious to reduce the dropout for the 4th classification head. So, we removed the dropout for the hidden layer in the 4th classification head. The number of epochs over whole training data for this architecture were varied from 2 to 5 with batch size variation of 64 and 128. We achieved best results for the combination of epochs=3 and batch size = 128. Intuition behind using the batch size of 128 was there were in total 96 variations(classes) so the batch size of 128 captures them in one go and then updates the parameters. Final accuracies obtained for this architecture were as follows
  - Classification head-1 – 100%
  - Classification head-2 – 100%
  - Classification head-3 – 100%
  - Classification head-4 – 100%

5 Final Network Architecture

Now in the final architecture we worked forward from the last discussed architecture and interchanged the position of the batch normalization and relu activation in the architecture. This resulted in slight improvement in the performance. Figure 2 is the final architecture.

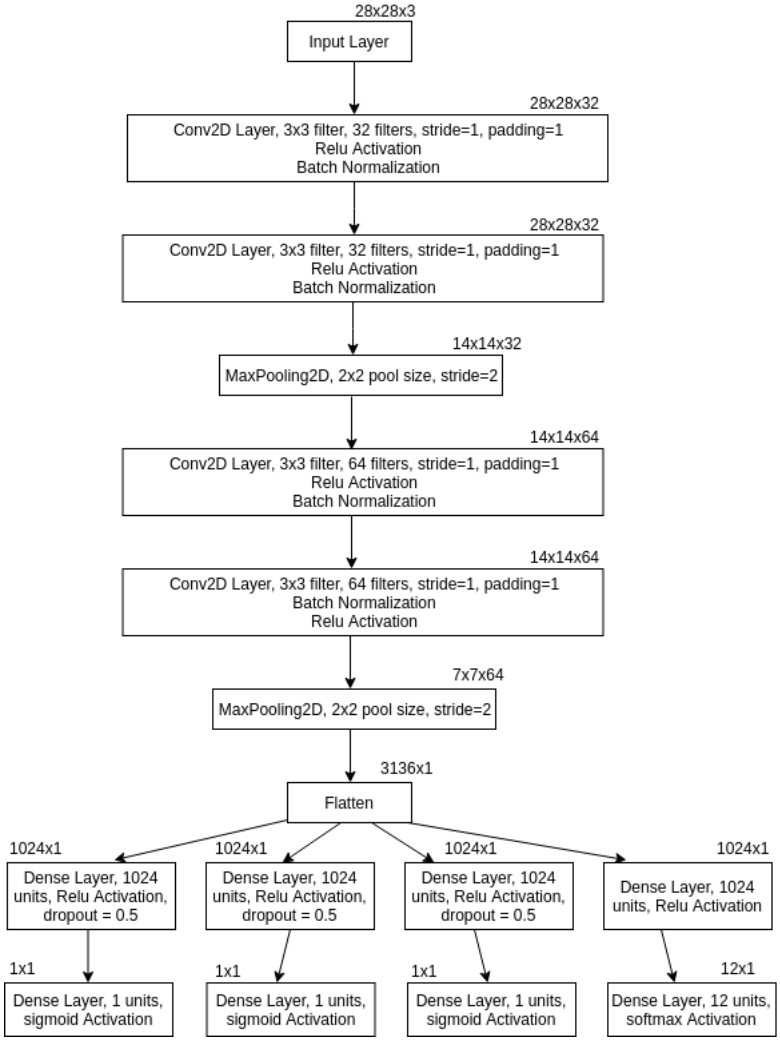

Figure 2. Final Network(Model) Architecture

5.1 Learning curves

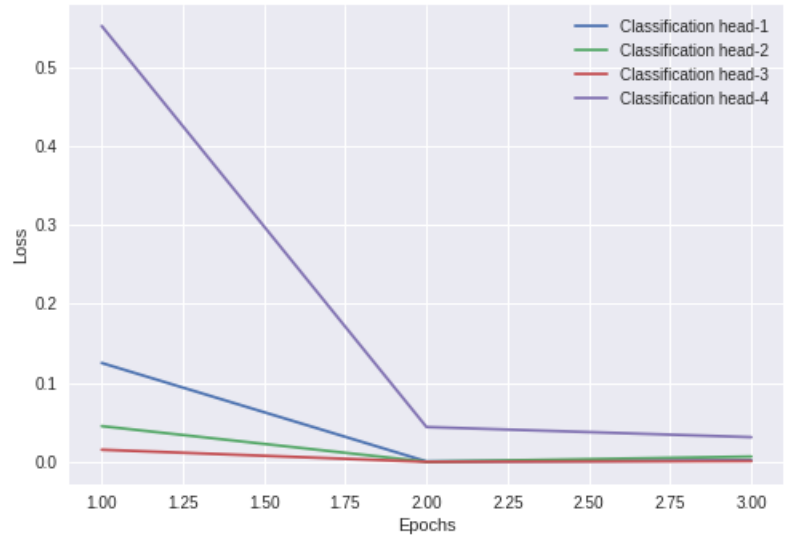

Figure 3. Loss vs Epochs curve for all the classification heads

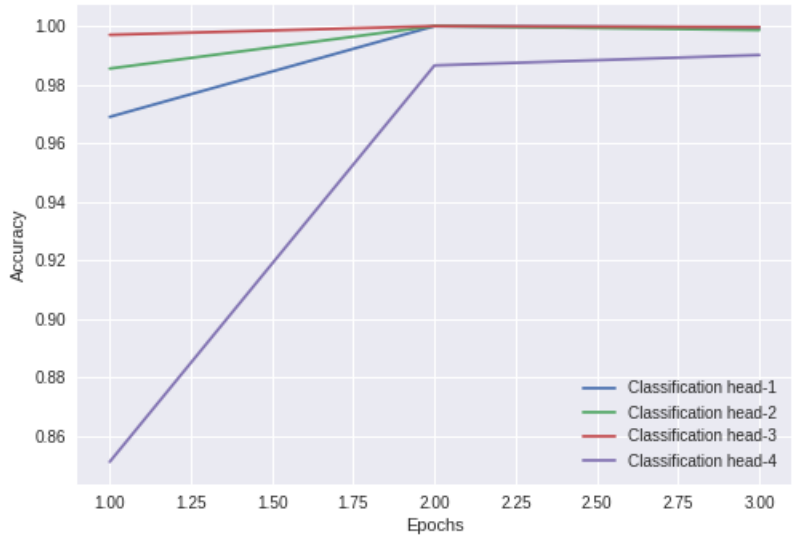

Figure 4. Accuracy vs Epochs curve for all the classification heads

5.2 F-scores and Confusion Matrices

Class	Precision	Recall	F-1 score
0(Length=7)	1.00	1.00	1.00
1(Length=15)	1.00	1.00	1.00

Table 1. F1-score for the classification head-1

Class	0	1
0(Length=7)	14378	0
1(Length=15)	0	14422

Table 2. Confusion matrix for the classification head-1

Class	Precision	Recall	F-1 score
0(Width=1)	1.00	1.00	1.00
1(Width=3)	1.00	1.00	1.00

Table 3. F1-score for the classification head-2

Class	0	1
0(Width=1)	14368	0
1(Width=3)	0	14432

Table 4. Confusion matrix for the classification head-2

Class	Precision	Recall	F-1 score
0(Color=Red)	1.00	1.00	1.00
1(Color=Blue)	1.00	1.00	1.00

Table 5. F1-score for the classification head-3

Class	0	1
0(Color=Red)	14387	0
1(Color=Blue)	0	14413

Table 6. Confusion matrix for the classification head-3

Class	precision	recall	F1-score
0(Angle=0)	1.00	1.00	1.00
1(Angle=15)	1.00	1.00	1.00
2(Angle=30)	1.00	1.00	1.00
3(Angle=45)	1.00	1.00	1.00
4(Angle=60)	1.00	1.00	1.00
5(Angle=75)	1.00	1.00	1.00
6(Angle=90)	1.00	1.00	1.00
7(Angle=105)	1.00	1.00	1.00
8(Angle=120)	1.00	1.00	1.00
9(Angle=135)	1.00	1.00	1.00
10(Angle=150)	1.00	1.00	1.00
11(Angle=165)	1.00	1.00	1.00

Table 7. F1-score for the classification head-4

Class	0	1	2	3	4	5	6	7	8	9	10	11
0(Angle=0)	2417	0	0	0	0	0	0	0	0	0	0	0
1(Angle=15)	0	2406	0	0	0	0	0	0	0	0	0	0
2(Angle=30)	0	9	2399	0	0	0	0	0	0	0	0	0
3(Angle=45)	0	0	0	2346	0	0	0	0	0	0	0	0
4(Angle=60)	0	0	0	0	2386	1	0	0	0	0	0	0
5(Angle=75)	0	0	0	0	0	2388	9	0	0	0	0	0
6(Angle=90)	0	0	0	0	0	0	2417	0	0	0	0	0
7(Angle=105)	0	0	0	0	0	0	0	2417	0	0	0	0
8(Angle=120)	0	0	0	0	0	0	0	2	2386	0	0	0
9(Angle=135)	0	0	0	0	0	0	0	0	0	2416	0	0
10(Angle=150)	0	0	0	0	0	0	0	0	0	0	2382	0
11(Angle=165)	0	0	0	0	0	0	0	0	0	0	0	2437

Table 8. Confusion matrix for the classification head-4

5.3 Accuracy

- For the classification head-1 $\rightarrow$ 100%
- For the classification head-2 $\rightarrow$ 100%
- For the classification head-3 $\rightarrow$ 100%
- For the classification head-4 $\rightarrow$ 99.98958%
- Aggregated Accuracy $\rightarrow$ 99.98958%
  Aggregated accuracy calculation - For each input sample we check, if it was classified correctly by all four classification heads. If yes, then we increase the number of correctly classified samples by one otherwise not.

6 Inferences

- The network architecture before the FC layer network works well may be because the two consecutive Conv2D layer tries to learn the combinations of the smaller features present in the image and the next two with higher number of filters tries to learn the aggregated features in the image.
- In our network, Batch Normalization after activation layer seems to work better than before activation. This is usually specific to network structure, much like the hyperparameters.It is topic of debate that we should use BN gefore activation or after activation for good accuracy. It depends on our network strucure.
- Introduing more FC layers didn't help and results in overfitting and poor test accuracy. The FC layers should be added with respect to the total output classes involved.
- Keeping the FC network (network after feature map) same for all 4 classification heads didn't work well, this may be due to the fact that number of classes involved in last classification head was different. Also the last classification head needs more complicated network than the other three to work well and indeed this can be seen from our network.
- Low number of epochs leads to under-fitting of data, whereas too many epochs leads to over-fitting. With too many epochs in training, the network, instead of learning the data, starts memorizing the training data and hence isn't able to perform well on test data.

Bibliography

[1] Keras Functional API tutorial
https://machinelearningmastery.com/keras-functional-api-deep-learning/

[2] Keras Documentation of Model
https://keras.io/models/model/#predict

[3] Keras Functional API guide
https://keras.io/getting-started/functional-api-guide/

[4] Keras: Multiple output and multiple losses tutorial
https://www.pyimagesearch.com/2018/06/04/
keras-multiple-outputs-and-multiple-losses/

[5] Epoch vs Batch
https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-\
4dfb9c7ce9c9
https://datascience.stackexchange.com/questions/27561/
can-the-number\-of-epochs-influence-overfitting

[6] Batch Normalization
https://forums.fast.ai/t/lesson-2-using-batch-normalization-after-non\
-linearity-or-before-non-linearity/4817