**Analysis**:

**Case 1. Basic Convnet from Scratch with small data**:

Graphical user interface, text, application

Description automatically generated

* From the given volume of the dataset, we took subset of it and divide the images into three folders i.e train, validate and test. Using make\_subset function.

Graphical user interface, text, application

Description automatically generated

* This piece of code demonstrates the CNN architecture, we stack up the series of layers, initially with input layer and the rescaled the features, finally with 2-Dimensional convolution layer.
* We use Maxpooling and at the end we use Flatten and single Dense Layer with a sigmoid function in it.

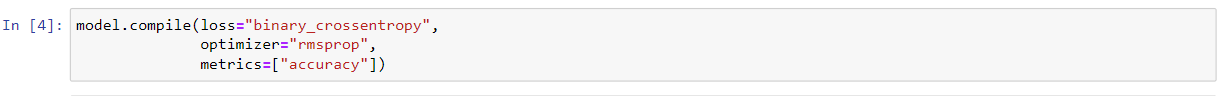
Why Maxpooling?

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map

Graphical user interface, text, application, email

Description automatically generated

* model.summary() gives the information about the architecture of the CNN.



* Here we used binary\_crossentropy loss factor and rmsprop as optimizer in the model.

Graphical user interface, application

Description automatically generated with medium confidence

We started the training here with 30 epochs and validate with validation set.

Chart, line chart, scatter chart

Description automatically generated

Chart, line chart

Description automatically generated

The above two graphs, 1. Training and Validation accuracy and 2. Training and Validation loss

**Note:**

From the graphs we observe that training has good accuracy and validation shows poor. We learned if a model works good on training data but performs poor on validation or test data, then it is clearly an **overfitting problem**.

Below methods can help us to solve overfitting problems.

* **Gaining access to more training data.**
* **Data augmentation**
* Cross validation
* **Addition of noise to the input data**
* **Making the network simple or tuning the capacity of the network** (the more capacity than required leads to a higher chance of overfitting).
* Regularization.
* **Adding dropouts**

**Case 2: Basic Convnet from Scratch with 2000 training samples and 500 validation and testing data**:

A picture containing chart

Description automatically generated

Chart, histogram

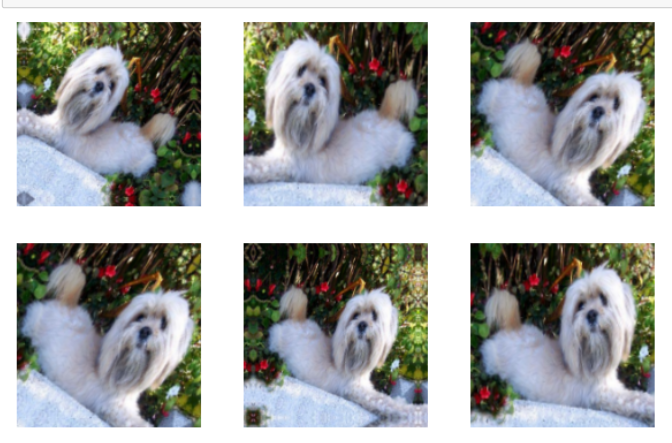
Description automatically generated

**Case 3: Basic Convnet from scratch with Dropout and Data Augmentation:**

Shape, rectangle

Description automatically generated

* In the first assignment, I experienced the importance of dropouts and regularization in a neural network, from little research I found “ADAM” is considered as best optimizer and used dropouts to avoid overfitting.
* In this case, I would like to see how data augmentation effects and solves the overfitting with dropout layers.





These are random images with flips and rotation using data augmentation.

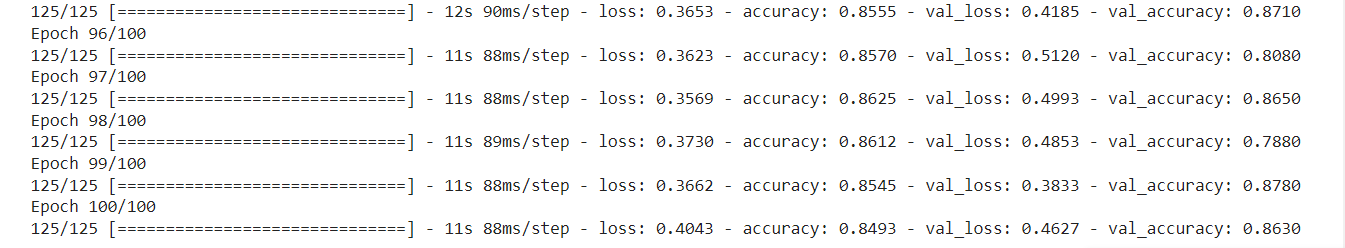
**Note**:

A picture containing table

Description automatically generated

From the above figure, we notice that there has been a drastic variation and we figured out this technique has solved our overfitting problem. We see it has train accuracy of 93.25 and validation accuracy of 83.60.

**Case 4: Basic Convnet from scratch with Dropout and Data Augmentation with more training, validation, and test samples**



Graphical user interface, chart

Description automatically generated

**Let’s now compare the cases for the network that is created from scratch.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cases | Training Accuracy | Validation Accuracy | Training Loss | Validation Loss | Test accuracy | Observations |
| Basic Convnet from scratch (no dropout, data augmentation) | 98.80 | 70.50 | 0.05 | 3.05 | 68.80 | Overfitting problem |
| Basic Convnet with data augmentation and dropouts | 93.25 | 83.60 | 0.18 | 0.49 | 81.4 | Here we clearly notice that our model is consistent in with training and validation loss and accuracy. Though our training accuracy is reduced to 93. We eliminated some overfitting problem. |
| Convnet with more training samples, validation, and test samples | 99.08 | 77.20 | 0.03 | 1.85 | 78.3 | Here we can see that overfitting problem can be overcome by adding more data. |
| With Data Augmentation and dropout with more training, validation and test samples | 84.93 | 86.30 | 0.4 | 0.46 | 83.3 | In this case, it is weird that our results accuracy has dropped and its is more consistent. |

**With Pretrained Network:**

**Using pretrained VGG16 network**.

The below link explains 10 different architectures

[Illustrated: 10 CNN Architectures | by Raimi Karim | Towards Data Science](https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d)

**Case: Using VGG16 as Base**

Architecture of VGG16:

Graphical user interface, application, table

Description automatically generated

Graphical user interface

Description automatically generated

**Case: Using VGG16 as base with data augmentation and dropout layer**

**A picture containing table

Description automatically generated**

**Case: Using VGG16 as base with more training, validation and test samples**

**A picture containing table

Description automatically generated**

**Chart, line chart

Description automatically generated**

**Basic convnet with ADAM**:

A picture containing table

Description automatically generated**Basic convnet with data augmentation:**

A picture containing table

Description automatically generated

Graphical user interface, chart, histogram

Description automatically generated

Case: VGG16 with ADAM:

A picture containing table

Description automatically generated

Graphical user interface, chart

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Overall Summary:**

**Let’s now compare the cases for the network that is created from scratch.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cases | Training Accuracy | Validation Accuracy | Training Loss | Validation Loss | Test accuracy | Observations |
| Basic Convnet from scratch (no dropout, data augmentation) | 98.80 | 70.50 | 0.05 | 3.05 | 68.80 | Clearly from the results we notice that our model works on training data and behaves poorly on validation and test data. (Overfitting problem) |
| Basic Convnet with data augmentation and dropouts | 93.25 | 83.60 | 0.18 | 0.49 | 81.4 | Here we notice that our model is consistent in with training and validation loss and accuracy. Though our training accuracy is reduced to 93. We eliminated some overfitting problem. |
| Convnet with more training samples, validation, and test samples | 99.08 | 77.20 | 0.03 | 1.85 | 78.3 | Here we can see that overfitting problem can be avoided by adding more data samples. |
| With Data Augmentation and dropout with more training, validation and test samples | 84.93 | 86.30 | 0.4 | 0.46 | 83.3 | In this case, it is weird that our results accuracy has dropped and its is more consistent. |
| Optimizer changed to ADAM, dropouts and data augmentation, regularizer | 99.37 | 89.47 | 0.11 | 0.32 | 89.34 | This case has showed best performance results when we all the optimizations and regularizations techniques. |

**Pretrained Network - VGG16 Cases:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cases | Training Accuracy | Validation Accuracy | Training Loss | Validation Loss | Test accuracy | Observations |
| Using VGG16 as base | 98.90 | 97.90 | 0.5266 | 2.46 | 97.60 | Overall, the result was decent using the VGG16 and I noticed a validation loss is more which can be reduced with some optimizations |
| Using VGG16 with data augmentation and dropouts | 99.80 | 98.20 | 0.06 | 1.6 you clear customs whatever delete | 97.80 | We noticed that accuracy has increased, and validation loss is reduced. |
| Using VGG16 as base with more training, validation, and test | 98.90 | 97.90 | 0.5 | 2.4 | 97.6 | It also shows the decent result but it has validation loss |
| With Data Augmentation & Dropouts | 99.65 | 98.20 | 0.04 | 0.8 | 97.7 | Best result so far with optimizations & Data Augmentation techniques. |

**Conclusion:**

In this task/Assignment, I compared the basic convnet with pretrained VGG16 network. Overall, the initial performance of the basic network is poor and then later we significantly improved the results with data augmentation, adding some random noise to the data. Later I tried to use some advanced optimizers additionally to see the results. With all the modifications, we improved accuracy and reduced loss.

Later I started the model with VGG16 as base and trained the model, initial results were good, and we can notice that the accuracy is above 95 percent and it significantly improved with the other techniques.