IBM Machine Learning Professional Certificate

- Deep Learning -
(CNN)

Choe C.S

October 2021

Main Objective

Objective

- This report aims to perform categorization of images through **Deep Learning** algorithm on data set images of **Cats** and **Dogs**.

Data Set

- The data set used for this analysis was obtained from Kaggle.com.

Steps Involved

1) EDA – To explore the data sets used in this report.

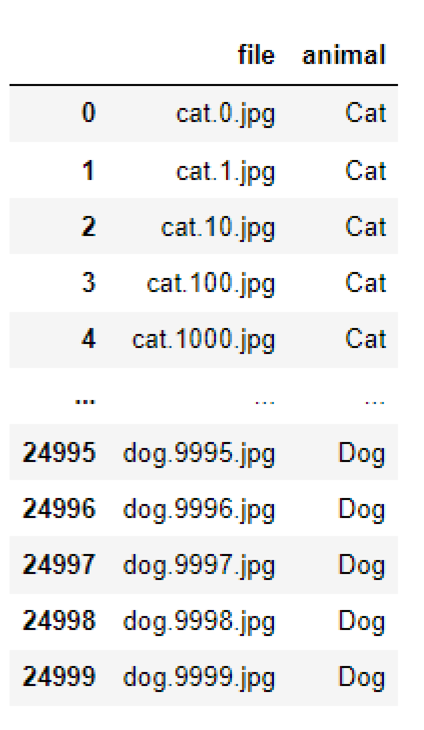
2) Perform **Deep Learning** on the data sets using algorithm

   - (**CNN** and **Transfer Learning**).

3) Discuss the results obtained from the Deep Learning models built.

Import Data

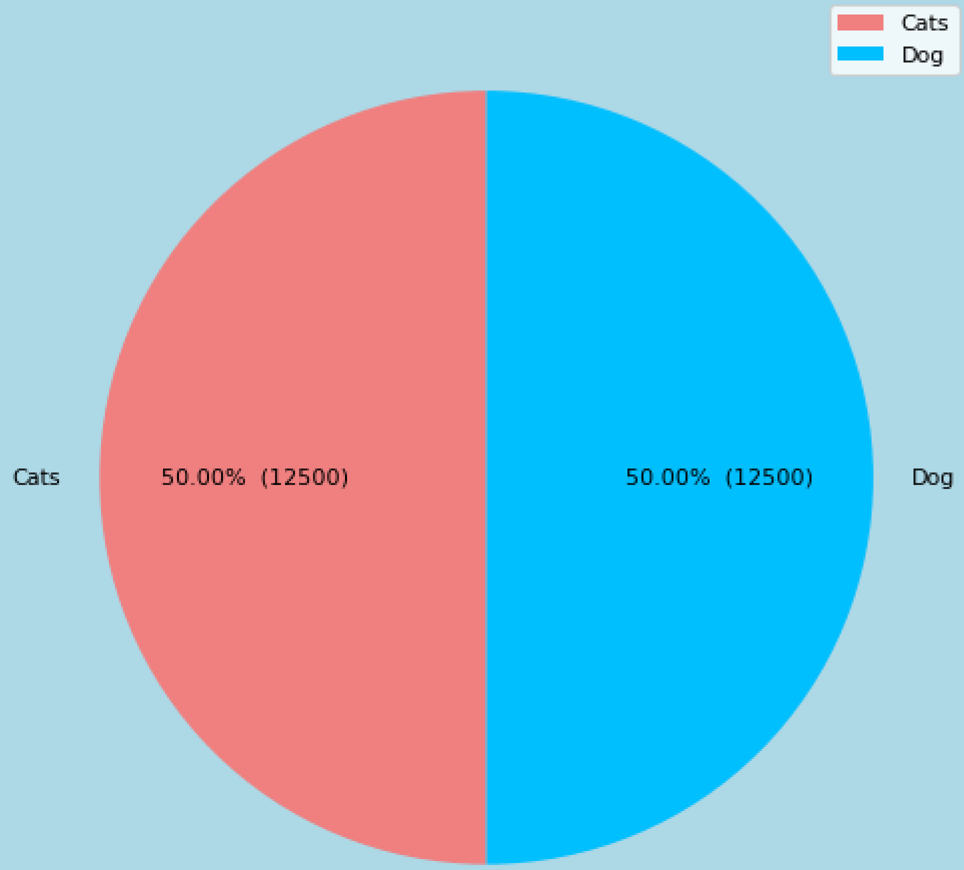
- The data set was import from Kaggle.com.
- The table at the bottom shows the top 5 and last 5 rows for the data set



	file	animal
0	cat.0.jpg	Cat
1	cat.1.jpg	Cat
2	cat.10.jpg	Cat
3	cat.100.jpg	Cat
4	cat.1000.jpg	Cat
...	...	...
24995	dog.9995.jpg	Dog
24996	dog.9996.jpg	Dog
24997	dog.9997.jpg	Dog
24998	dog.9998.jpg	Dog
24999	dog.9999.jpg	Dog

EDA – Visual exploration

- As we are interested in analyzing groups of Animals (Cats & Dogs), it is important to know the quantities of both species.
- The pie chart on the right shows that the quantity of both species occupy 50% of the overall data.



Deep Learning

Deep Learning (image classification)

- A total of **3 models** were build to analyze the categories for both species cats and dogs.
- The first **2 models** are built using **CNN** and the 3rd model is a **Transfer Learning** algorithm.
- The results obtained were compared and discussed in the next section.

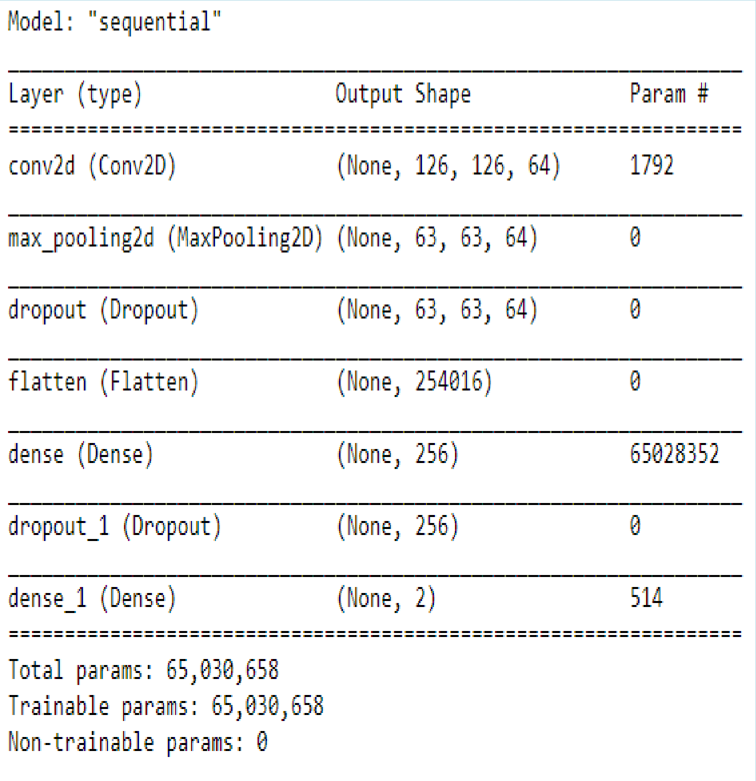
CNN

- There were in total 2 Models built using **CNN** algorithm.
- The 1st Model is **CNN** with 3 layers including the output layer.
- The 2nd Model is **CNN** with 5 layers including the output layer.

Transfer Learning

- There is only 1 model build for **Transfer Learning**.
- The 3rd Model is a **Transfer Learning** model using **VGG 16**.

Deep Learning – (CNN 1st Model)



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 64)      1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 63, 63, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 254016)            0
_________________________________________________________________
dense (Dense)                (None, 256)               65028352
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
=================================================================
Total params: 65,030,658
Trainable params: 65,030,658
Non-trainable params: 0
```

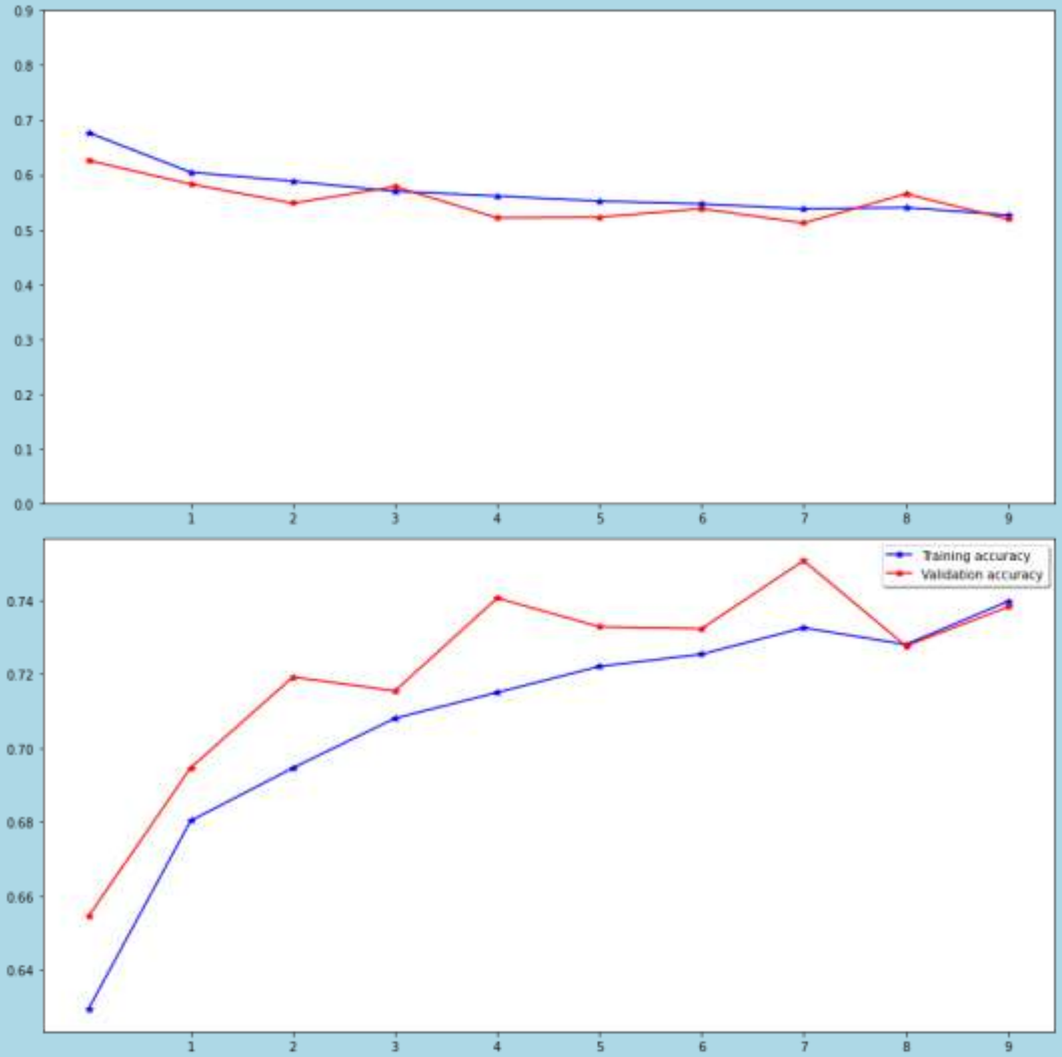
1st CNN Model

- The first model contains **3 layers** including output layer.
- The input shape is (**128 x 128 x 3)**.
- The 1st **layer** consist of **64 Nodes** of (**3 x 3**) **kernel** size with **Max pooling** of (**2 x 2**) and **Drop out** value of **0.5**.
- The 2nd **layer** has a Dense of **256** Node and **Drop out** of value **0.5**.
- Both **1st** and **2nd** layer uses **activation** function '**relu**'.
- The output layer consist of Dense of **2** Nodes with **activation** function "**Sigmoid**'.
- This model uses an "**adam"** optimizer with loss of "**binary_crossentropy"** and metrics "**accuracy**".

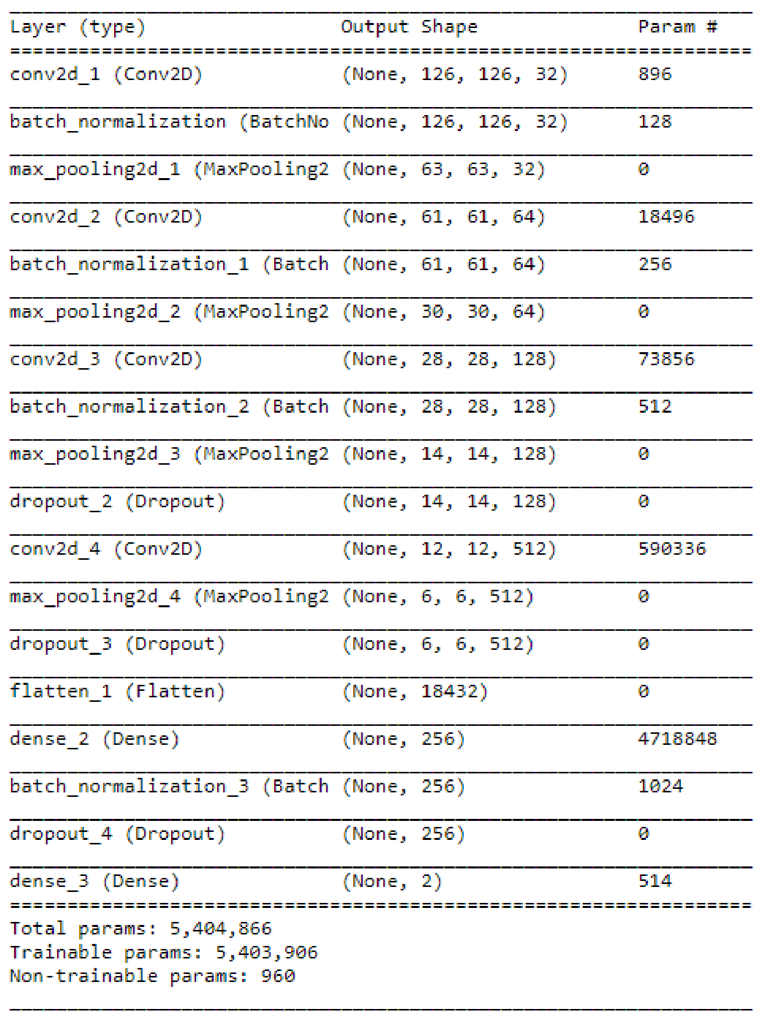
Results – (CNN 1st Model)

1st CNN Model

- A total of 10 epochs was used with a batch size of 10.
- The first model took 6983 seconds to trained.
- The final loss is at 0.526 and accuracy of 0.7397.
- The final validation loss is at 0.5191 and validation accuracy of 0.7381.
- The graph on the right shows the results plotted for both loss and accuracy of this model.



Deep Learning – (CNN 2nd Model)



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization (BatchNo (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 64)        18496
_________________________________________________________________
batch_normalization_1 (Batch (None, 61, 61, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 128)       73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 28, 28, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 12, 12, 512)       590336
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 6, 6, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               4718848
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 514
=================================================================
Total params: 5,404,866
Trainable params: 5,403,906
Non-trainable params: 960
_________________________________________________________________
```

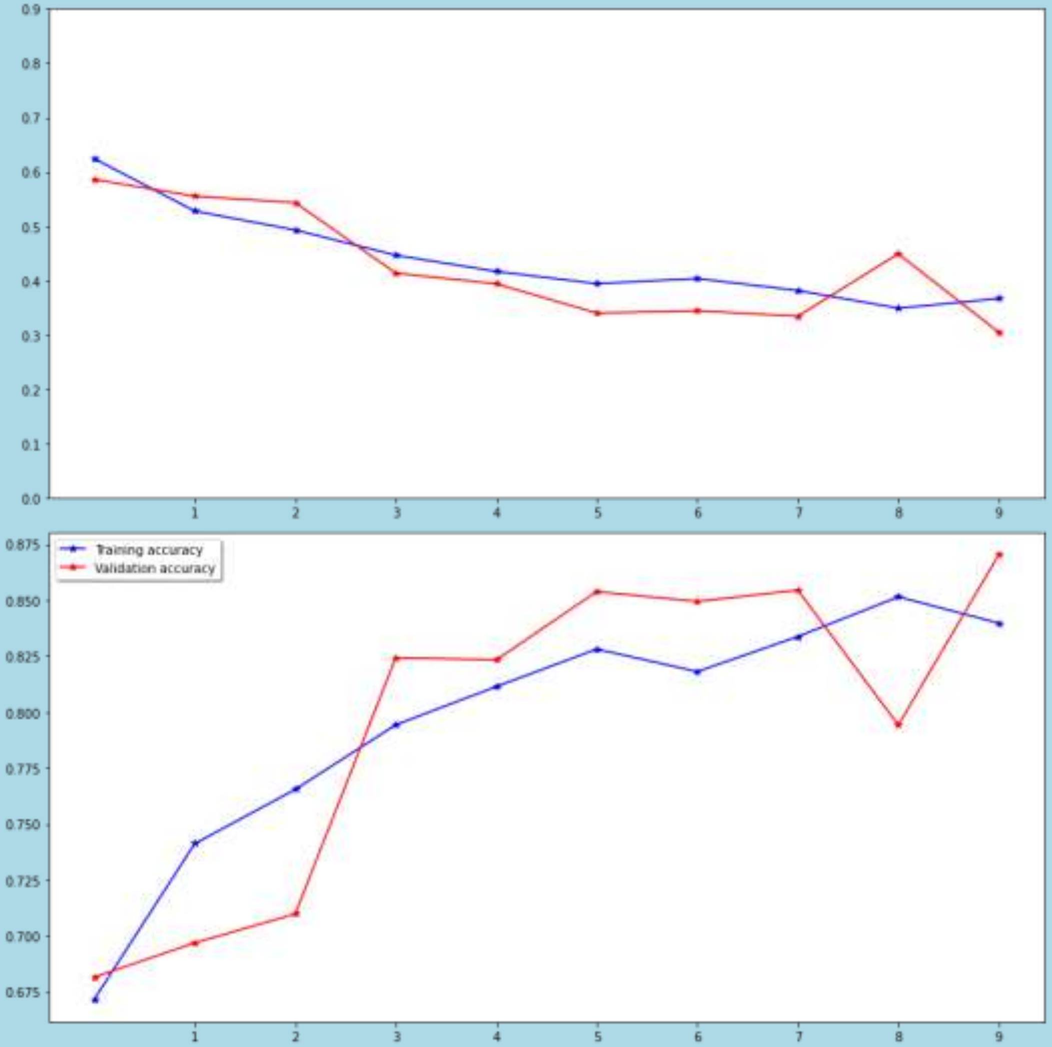
2nd CNN Model

- The second model is an improvement of the first model with additional layers included.
- The input shape is **(128 x 128 x 3)**.
- The 1st **layer** consist of **32 Nodes** of (**3 x 3**) **kernel** size with **Max pooling** of (**2 x 2**).
- The 2nd **layer** consist of **64 Nodes** of (**3 x 3**) **kernel** size with **Max pooling** of (**2 x 2**).
- The 3rd **layer** consist of **128 Nodes** of (**3 x 3**) **kernel** size with **Max pooling** of (**2 x 2**) and **Drop out** with value of 0.5.
- The 4th **layer** consist of **128 Nodes** of (**3 x 3**) **kernel** size with **Max pooling** of (**2 x 2**) and **Drop out** with value of 0.5.
- The 5th **layer** consist of **512 Nodes** of (**3 x 3**) **kernel** size with **Max pooling** of (**2 x 2**) and **Drop out** with value of 0.5.
- The 6th **layer** consist of **256 Nodes** of (**3 x 3**) **kernel** size with **Max pooling** of (**2 x 2**).
- All layers **activation** function ‘**relu**’.
- The output layer consist of Dense of **2** Nodes with **activation** function “**Sigmoid**’.
- This model uses an “**adam”** optimizer with loss of “**binary_crossentropy”** and metrics “**accuracy**”.

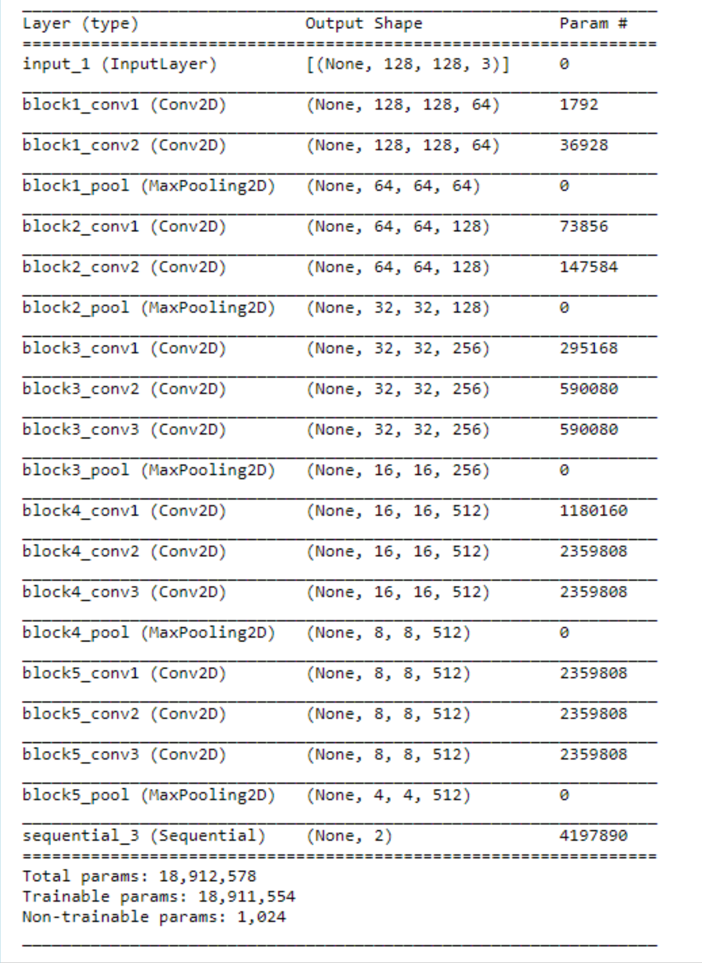
Results – (CNN 2nd Model)

2nd CNN Model

- A total of **10 epochs** was used with a **batch size** of **10**.
- The second model took **5106 seconds** to trained.
- The final **loss** is at **0.3669** and **accuracy** of **0.8398**.
- The final **validation loss** is at **0.3048** and **validation accuracy** of **0.8704**.
- The graph on the right shows the results plotted for both loss and accuracy of this model.



Deep Learning – (VGG 16 3rd Model)



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128, 128, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
sequential_3 (Sequential)    (None, 2)                 4197890
=================================================================
Total params: 18,912,578
Trainable params: 18,911,554
Non-trainable params: 1,024
_________________________________________________________________
```

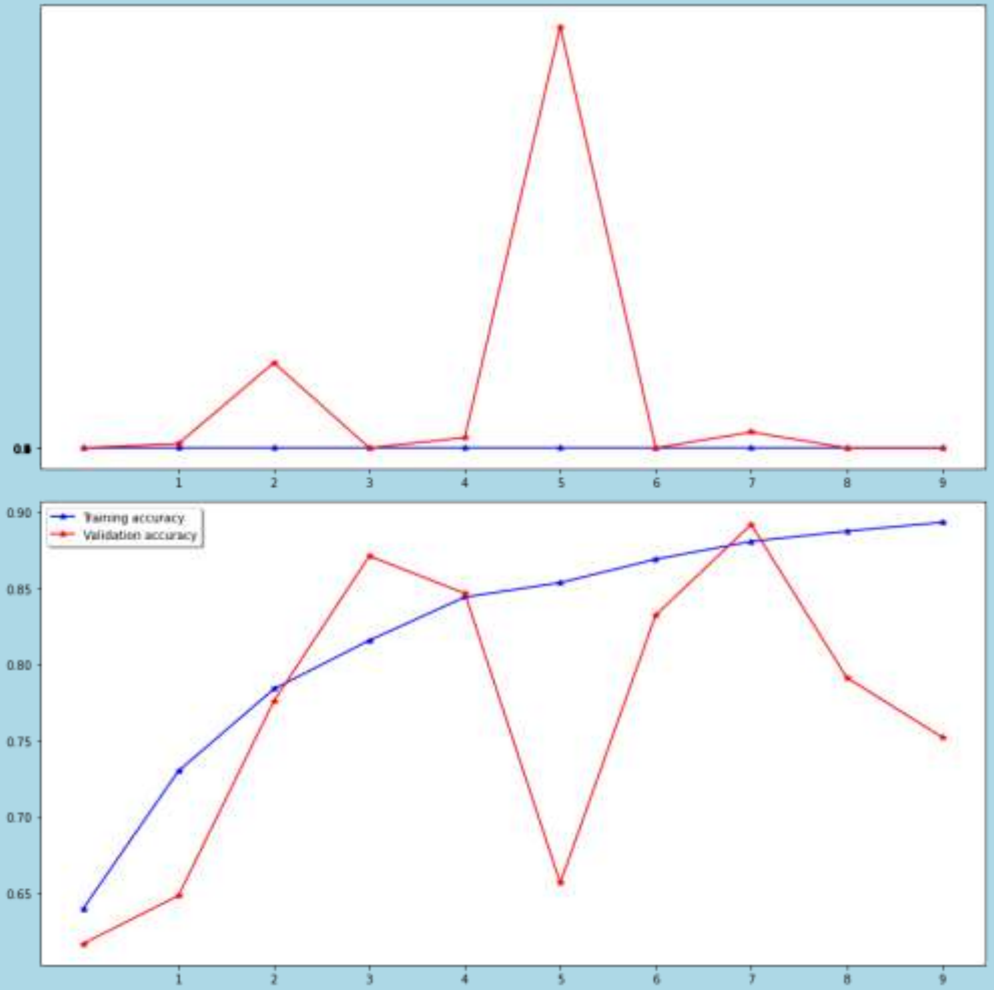
3rd CNN Model

- The second model is a Transfer Learning model used (VGG 16).
- VGG 16 is a pre-built model used to win the ILSVR (imagnet) competion in 2014.
- The Layers involved is as the figure shown on the right.
- The VGG 16 algorithm Is used but the final layer was freezed.
- The final layer included is Dense of **2** Nodes with **activation** function “**Sigmoid**’.

Results – (VGG 16 3rd Model)

3rd VGG 16 Model

- A total of **10 epochs** was used with a **batch size** of **10**.
- The third model took **30611 seconds** to trained.
- The final **loss** is at **0.2674** and **accuracy** of **0.8931**.
- The final **validation** loss is at **0.6949** and validation **accuracy** of **0.7524**.
- The graph on the right shows the results plotted for both loss and accuracy of this model.



Results & Discussion

Results

Model	Time taken(s)	Loss	Validation Loss	Accuracy	Validation Accuracy
1st Model (CNN)	6983	0.526	0.5191	0.7397	0.7381
2nd Model (CNN)	5106	0.3669	0.3048	0.8398	0.8704
3rd Model (VGG 16)	30611	0.2674	0.6949	0.8931	0.7524

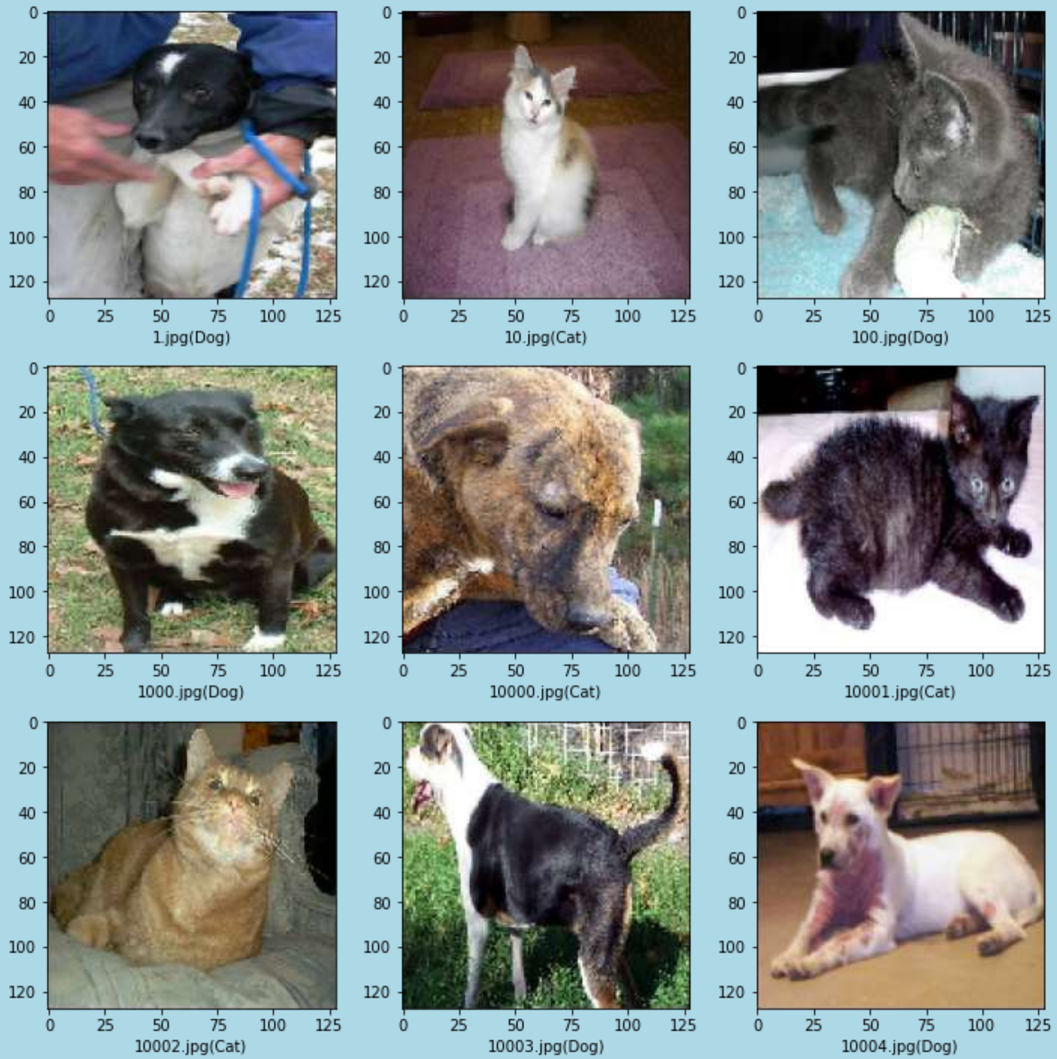
Discussion

- The Model 1 took the shortest time to train at seconds, Model 2 took 5106 seconds and Model 3 took the longest time to train at 30611 seconds.
- The loss and validation loss for Model 2 is roughly the same at 0.3669 and 0.3048.
- The loss and validation loss for Model 3 shows a large amount of difference at 0.2674 and 0.6949.
- The accuracy and validation accuracy for Model 2 is not far apart at 0.8398 and 0.8704.
- However, the accuracy and validation accuracy for Model 3 has large difference at 0.8931 and 0.7524 due to overfitting.
- From this analysis model 2 has the best overall results obtained as both the training and validation results does not have large significant difference compared to Model 3 and much better results compared to Model 1 with lower loss and higher accuracy.
- Therefore, Model 2 is the most suitable Model for this

Improvements

- The next step to take for better prediction is to include other pre-trained model such as ResNet, Inception, Xception etc.

Results - Prediction with 2nd Model



Prediction results for test images using 2nd Model

- A total of 9 images were printed with labels of **file name** and predicted results of either (**Cat** or **Dog**) was listed at the bottom of each images

- The images shown on the left consist of **5 dogs** and **4 cats**.

- The predicted output made **7 correct** predictions and **2 wrong** predictions out of **9** in total.

- The model has wrongly predicted the 1st roll 2nd column as **Dog** but the actual result was **Cat**.

- The model has wrongly predicted the 2nd roll 2nd column as **Cat** but the actual result was **Dog**.

- The overall percentage of correct prediction was **77.78%** accurate.

Conclusion & Improvements

Link to Code

https://github.com/cs-robot-collab/IBM-ML-DL/blob/master/IBM%20Machine%20Learning%20DL%20report.ipynb