CNN-Image-Classification

Are Pandas more like Cats orlike Dogs?

INTRODUCTION

In this project I am developing the two image classifiers using Convolution Neural Networks, the two image classifiers are Dogs-Vs-Pandas and Cats-Vs-Pandas.

The architecture I am going to follow in the CNN models are with four pairs of Conv2D + Maxpool layer followed by one more layer of Conv2D and a Dense layer.

In this project I will work with a Kaggle dataset that contains pictures of PANDAS, CATS and DOGS (1000 each). In Colab, I obtained ("downloaded") the dataset with the below command,

!kaggle datasets download -d ashishsaxena2209/animal-image-datasetdog-cat-and-panda

Then unzipping the through the following command

!unzip -qq animal-image-datasetdog-cat-and-panda.zip

After doing that, you will have the following directory structure

	cats	1000 files named cats_xxxxx.jpg
./animals	dogs	1000 files named dogs_xxxxx.jpg
	panda	1000 files named panda_xxxxx.jpg
	… other	

In the tree above xxxxx are 5-digit sequences that go from 00001 to 01000

The accuracy expected from a Random Classifier for the Pandas-vs-Dogs & Pandas-vs-Cats classification on the corresponding Test datasets is 12.5%

Technology Used:

- Python

Libraries Used:

- Tensorflow
- Keras
- Matplotlib

[PART I]: Initial Reorganization of the data

Directories, Train-Validation-Test Splits

You must copy the image files into the following director structure:

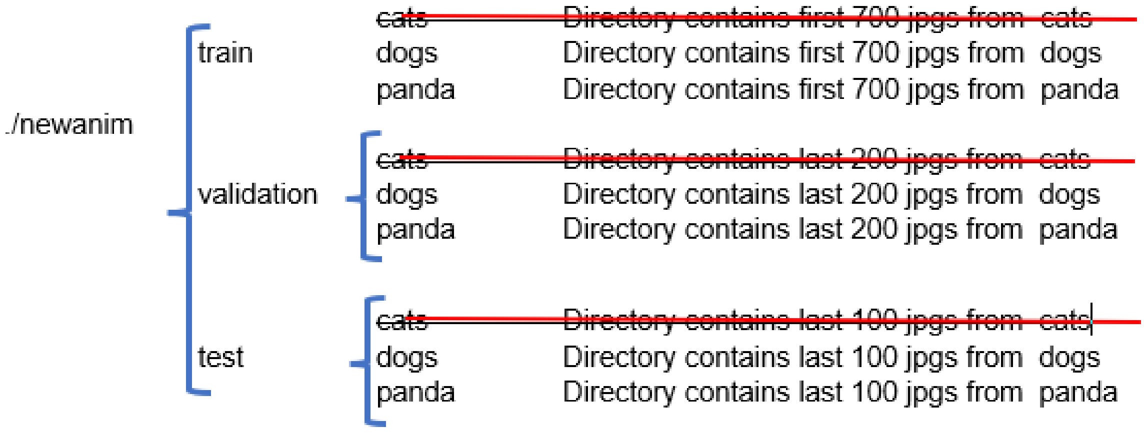
[PART II]: The Panda -vs- Dog Classifier

II.0 Creation of the TF Datasets (train, validation, test) for Panda-vs-Dog classification

For the 2-way classification of images between PANDAS and DOGS, you will create the necessary TensorFlow Dataset objects for training, validation and testing. This is easily accomplished using the image_dataset_from_directory method, as shown in Listing 8.9 of the book. However, to create datasets that only contain pandas and cats for the purpose of two-way classification of these classes, we need to first remove the cat's subdirectories in the train, validation and test subdirectories of newanim, to get:

```
!rm -r ./newanim/train/cats/
!rm -r ./newanim/validation/cats/
!rm -r ./newanim/test/cats/
```

I have used the above command to remove the cats from the subdirectories specified



After doing that, the datasets are created by:

train_dataset = image_dataset_from_directory(

new_base_dir / 'train',

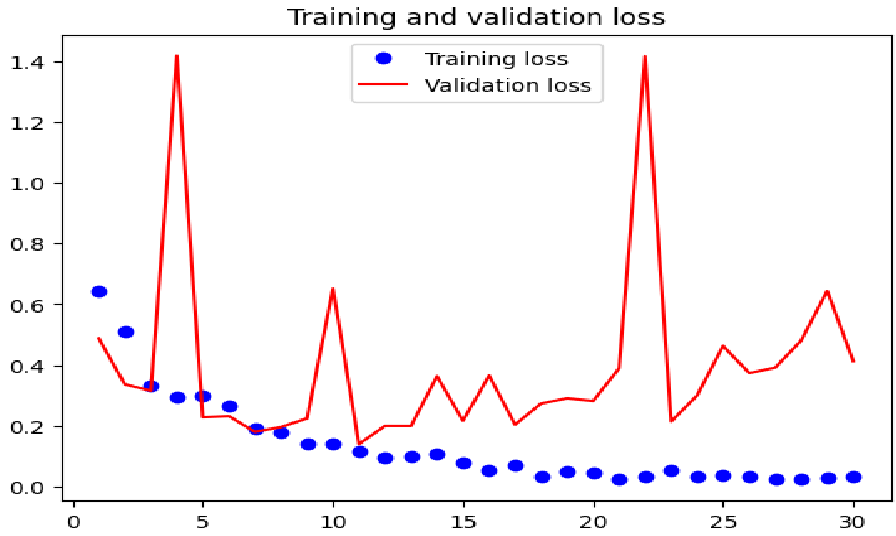
image_size=(180, 180),

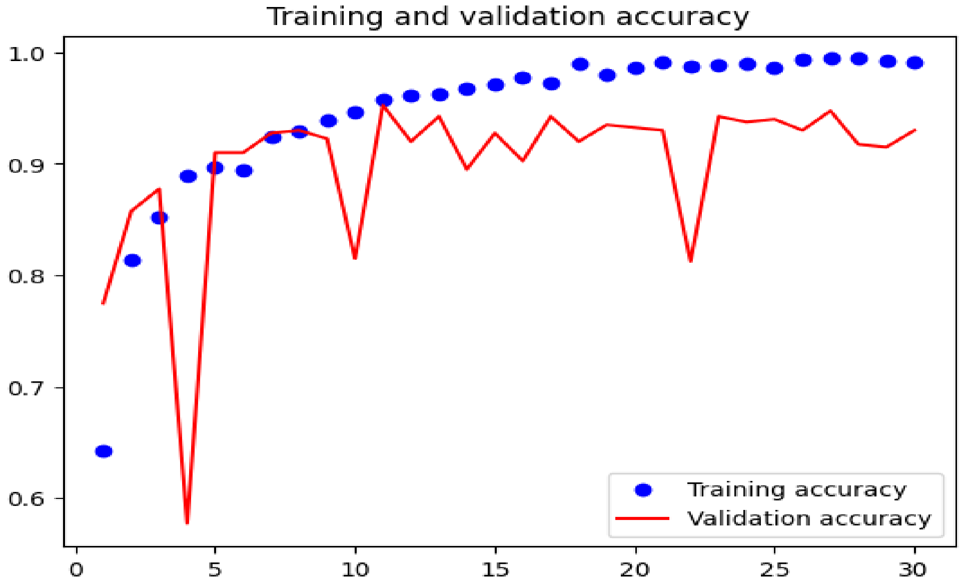
batch_size=32)

and similarly for the validation_dataset and the test_dataset.

II.1 - Develop a first, relatively simple CNN model named pvdm1 to classify Panda vs. Dog. – Do NOT use dropout, parameter regularization (L1, L2 norm) or data augmentation in this first model.

In this model first I am implementing without using callbacks that is called preliminary model so that I can set the base line of the accuracy to further in the next CNN models.

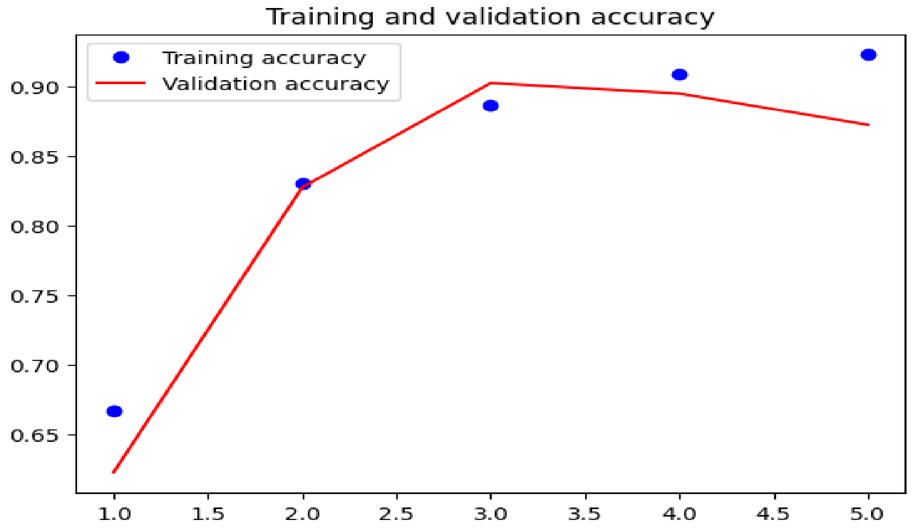


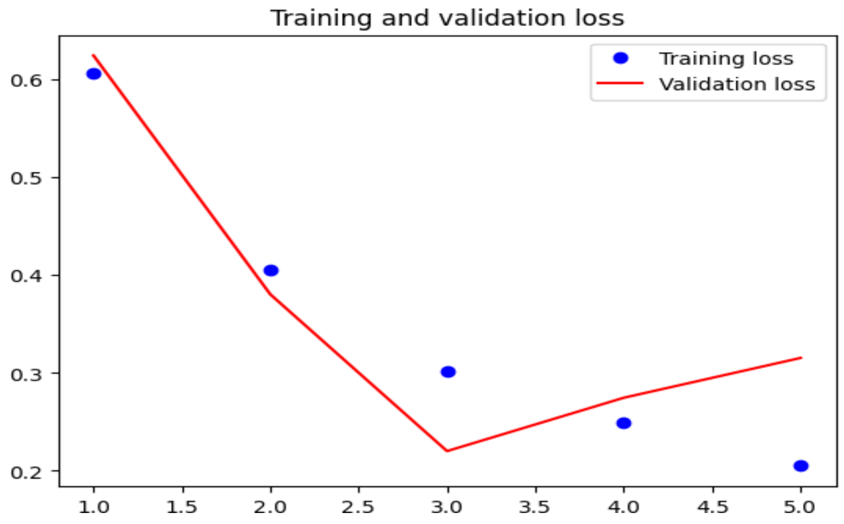



```
loss: 0.0352 - accuracy: 0.9914 - val_loss: 0.4131 - val_accuracy:
0.9300
```

After that I added callbacks to the above model to stop after an advantageous number of epochs, the plots are attached below.

<u>Plots and results for Pvdm1</u>





loss: 0.3019 - accuracy: 0.8864 - val_loss: 0.2201 - val_accuracy: 0.9025

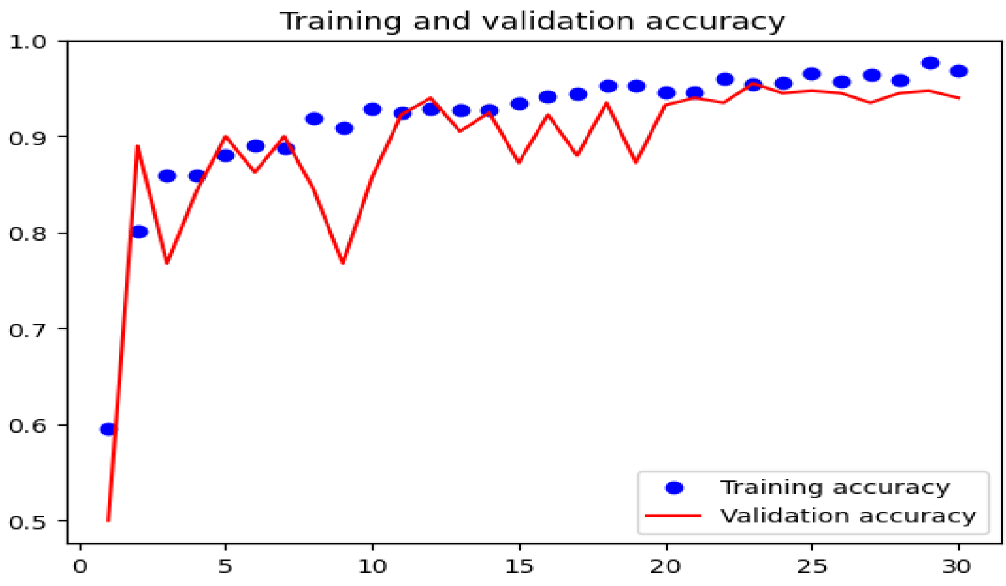
Test Accuracy for pvdm1 model: 93.5

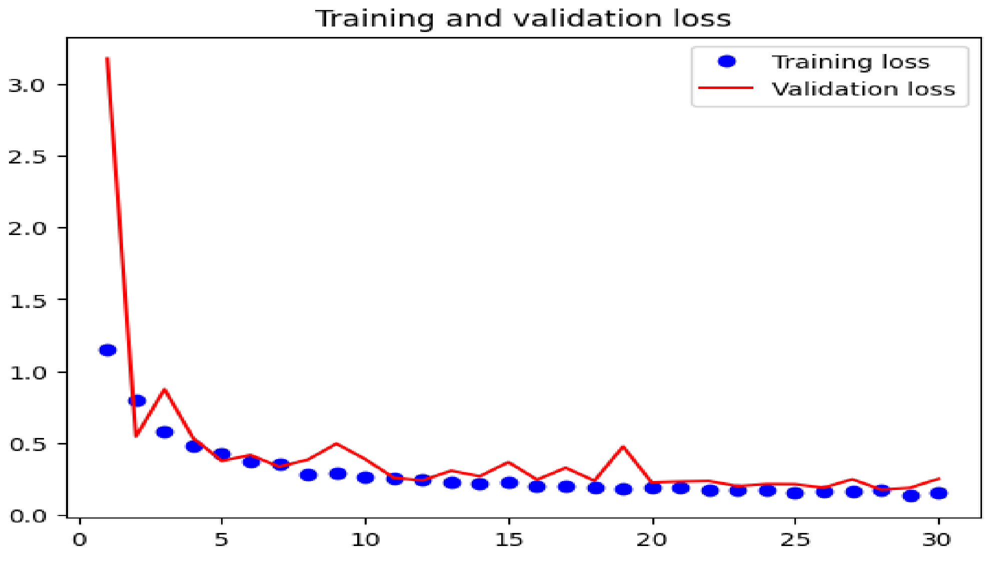
Summary of the pvdm1 model:

```
Model: "model_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_12 (InputLayer)       [(None, 180, 180, 3)]     0

 rescaling_11 (Rescaling)    (None, 180, 180, 3)       0

 conv2d_55 (Conv2D)          (None, 178, 178, 32)      896

 max_pooling2d_44 (MaxPoolin  (None, 89, 89, 32)       0
 g2D)

 conv2d_56 (Conv2D)          (None, 87, 87, 64)        18496

 max_pooling2d_45 (MaxPoolin  (None, 43, 43, 64)       0
 g2D)

 conv2d_57 (Conv2D)          (None, 41, 41, 128)       73856

 max_pooling2d_46 (MaxPoolin  (None, 20, 20, 128)      0
 g2D)

 conv2d_58 (Conv2D)          (None, 18, 18, 256)       295168

 max_pooling2d_47 (MaxPoolin  (None, 9, 9, 256)        0
 g2D)

 conv2d_59 (Conv2D)          (None, 7, 7, 256)         590080

 flatten_11 (Flatten)        (None, 12544)             0

 dense_11 (Dense)            (None, 1)                 12545

=================================================================
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0
```

II.2 -Develop a second, further improved CNN model named pvdm2 to classify Panda vs.Dog. Include the use of dropout and/or parameter regularization (L1, L2 norm) as part of the improvements, but DO NOT USE data augmentation.

<u>Plots and results for Pvdm2</u>





loss: 0.1763 - accuracy: 0.9543 - val_loss: 0.2005 - val_accuracy: 0.9550

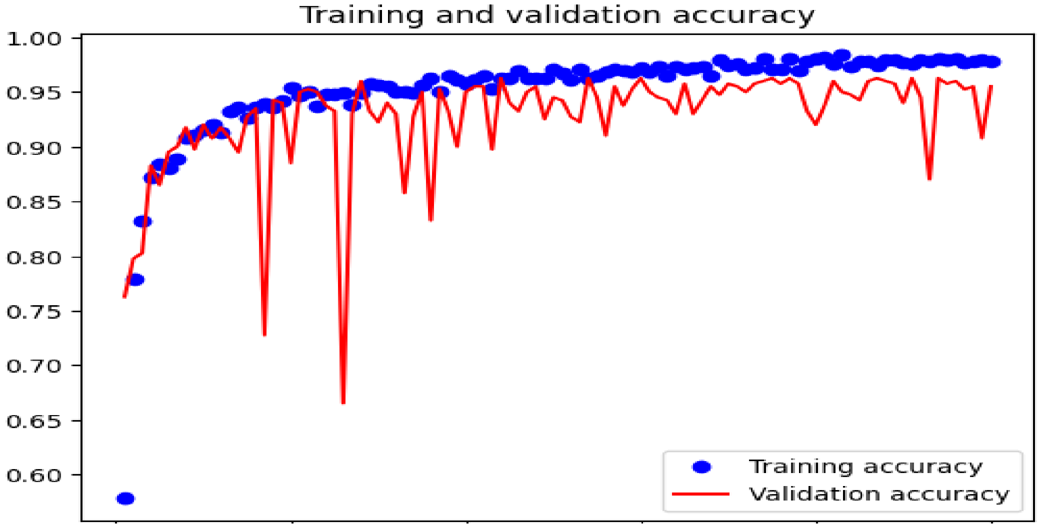
Test Accuracy for pvdm2 model: 95.0

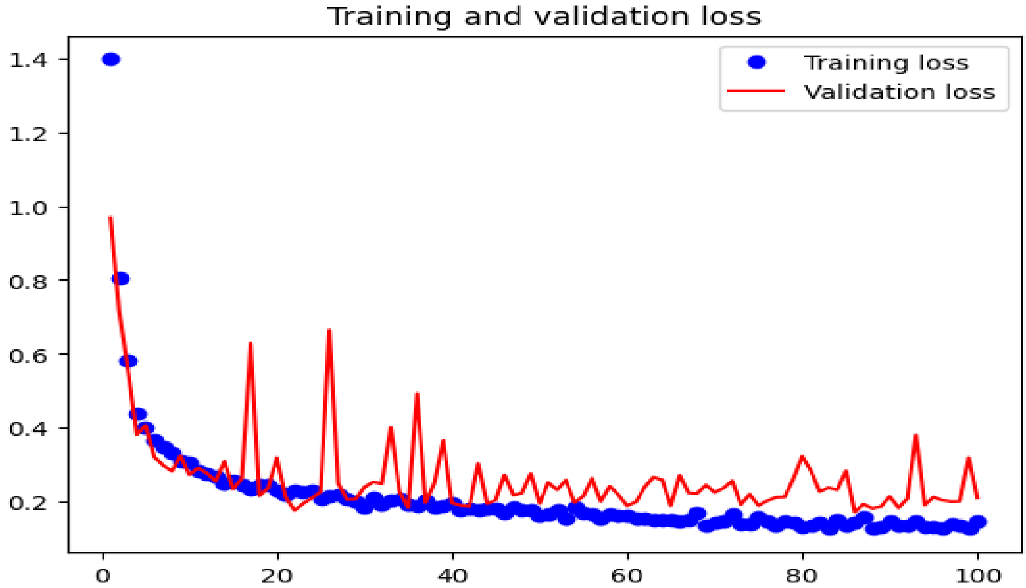
Summary of the pvdm2 model:

```
Model: "model_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_14 (InputLayer)       [(None, 180, 180, 3)]     0

 rescaling_13 (Rescaling)    (None, 180, 180, 3)       0

 conv2d_65 (Conv2D)          (None, 178, 178, 32)      896

 max_pooling2d_52 (MaxPoolin  (None, 89, 89, 32)       0
 g2D)

 conv2d_66 (Conv2D)          (None, 87, 87, 64)        18496

 max_pooling2d_53 (MaxPoolin  (None, 43, 43, 64)       0
 g2D)

 conv2d_67 (Conv2D)          (None, 41, 41, 128)       73856

 max_pooling2d_54 (MaxPoolin  (None, 20, 20, 128)      0
 g2D)

 conv2d_68 (Conv2D)          (None, 18, 18, 256)       295168

 max_pooling2d_55 (MaxPoolin  (None, 9, 9, 256)        0
 g2D)

 conv2d_69 (Conv2D)          (None, 7, 7, 256)         590080

 flatten_13 (Flatten)        (None, 12544)             0

 dropout_4 (Dropout)         (None, 12544)             0

 dense_13 (Dense)            (None, 1)                 12545

=================================================================
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0
_________________________________________________________________
```

II.3 -Developing a third CNN model, named pvdm3 USING DATA AUGMENTATION FOR ITS TRAINING.

Plots and results for Pvdm3






```
loss: 0.1341 - accuracy: 0.9807 - val_loss: 0.1916 - val_accuracy: 0.9625

Test Accuracy for pvdm3 model: 96.0


Summary of the pvdm3 model:

Model: "model_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_16 (InputLayer)       [(None, 180, 180, 3)]     0

 rescaling_15 (Rescaling)    (None, 180, 180, 3)       0
```

```
 conv2d_75 (Conv2D)          (None, 178, 178, 32)      896

 max_pooling2d_60 (MaxPoolin  (None, 89, 89, 32)       0
 g2D)

 conv2d_76 (Conv2D)          (None, 87, 87, 64)        18496

 max_pooling2d_61 (MaxPoolin  (None, 43, 43, 64)       0
 g2D)

 conv2d_77 (Conv2D)          (None, 41, 41, 128)       73856

 max_pooling2d_62 (MaxPoolin  (None, 20, 20, 128)      0
 g2D)

 conv2d_78 (Conv2D)          (None, 18, 18, 256)       295168

 max_pooling2d_63 (MaxPoolin  (None, 9, 9, 256)        0
 g2D)

 conv2d_79 (Conv2D)          (None, 7, 7, 256)         590080

 flatten_15 (Flatten)        (None, 12544)             0

 dropout_6 (Dropout)         (None, 12544)             0

 dense_15 (Dense)            (None, 1)                 12545

=================================================================
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0
_________________________________________________________________
```

[PART III]: The Panda -vs- Cat Classifier

For part III also I have used the jupyter notebook with out any disturbance to the Part II code and directories. In order not to crash the directories for the two parts I have created the new directory called /root/content1 and I have copied all the files from the directory /content which is called the original directory and I have used the files from the new directory for part III.

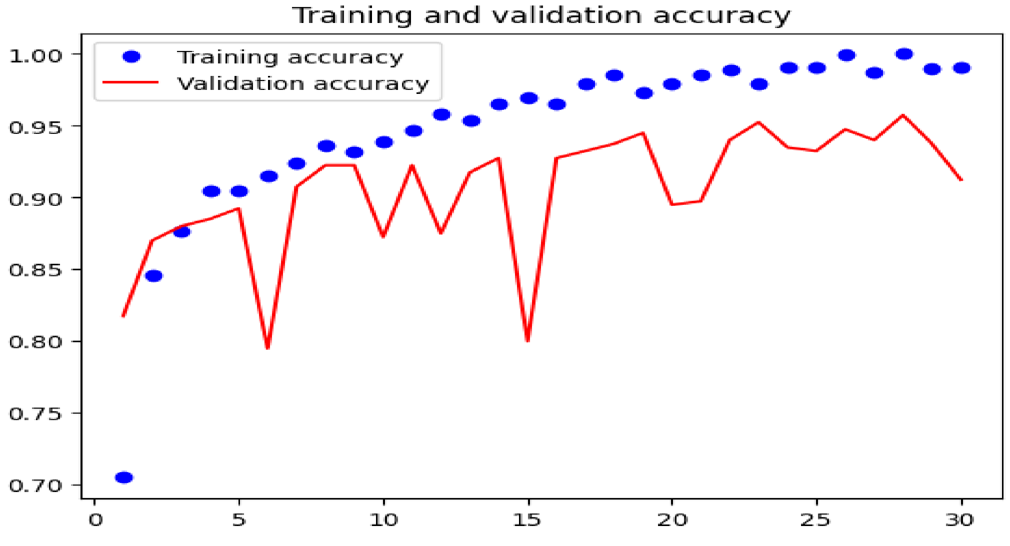
III.0 The below three commands are used for the subdirectories to remove dogs dataset.

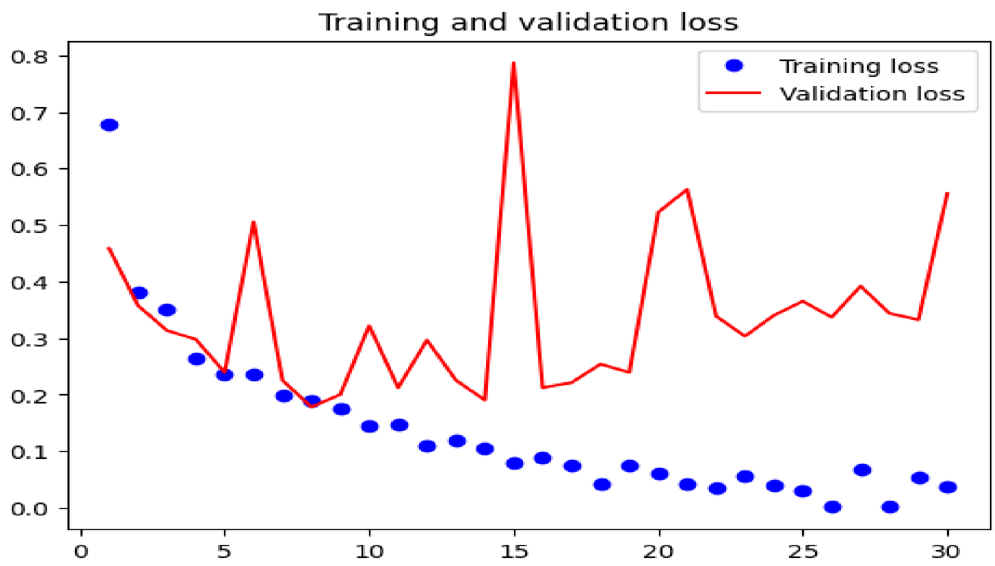
```
!rm -r ./newanim/train/dogs/
!rm -r ./newanim/validation/dogs/
!rm -r ./newanim/test/dogs/
```

III.1 - Developing a first, relatively simple CNN model named pvcm1 to classify Panda vs. Cat. – without dropout, parameter regularization (L1, L2 norm) or data augmentation in this first model.

<u>Plots and results for preliminary model:</u>

In this model first I am implementing without using callbacks that is called preliminary model so that I can set the base line of the accuracy to further in the next CNN models.

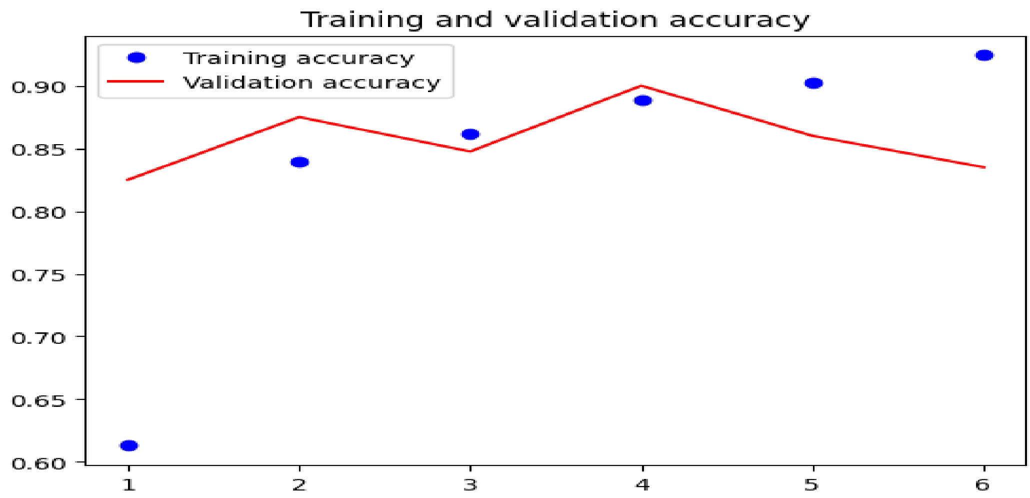


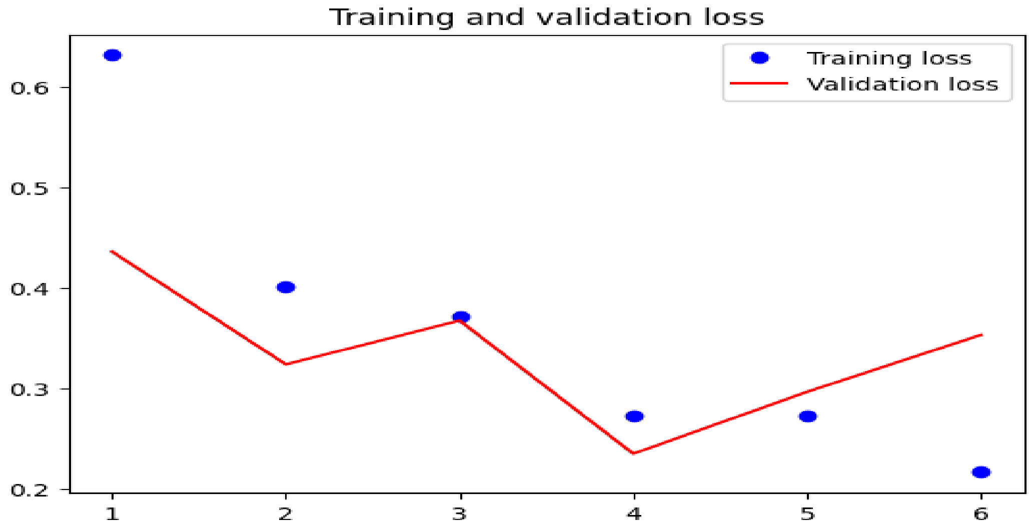



```
loss: 0.0371 - accuracy: 0.9907 - val_loss: 0.5560 - val_accuracy:
0.9125
```

After that I added callbacks to the above model to stop after an advantageous number of epochs, the plots are attached below.

Plots and results for Pvcm1:





loss: 0.2448 - accuracy: 0.9100 - val_loss: 0.1816 - val_accuracy: 0.9275

Test Accuracy for pvcm1 Model: 91.0

Summary for pvcm1 model:

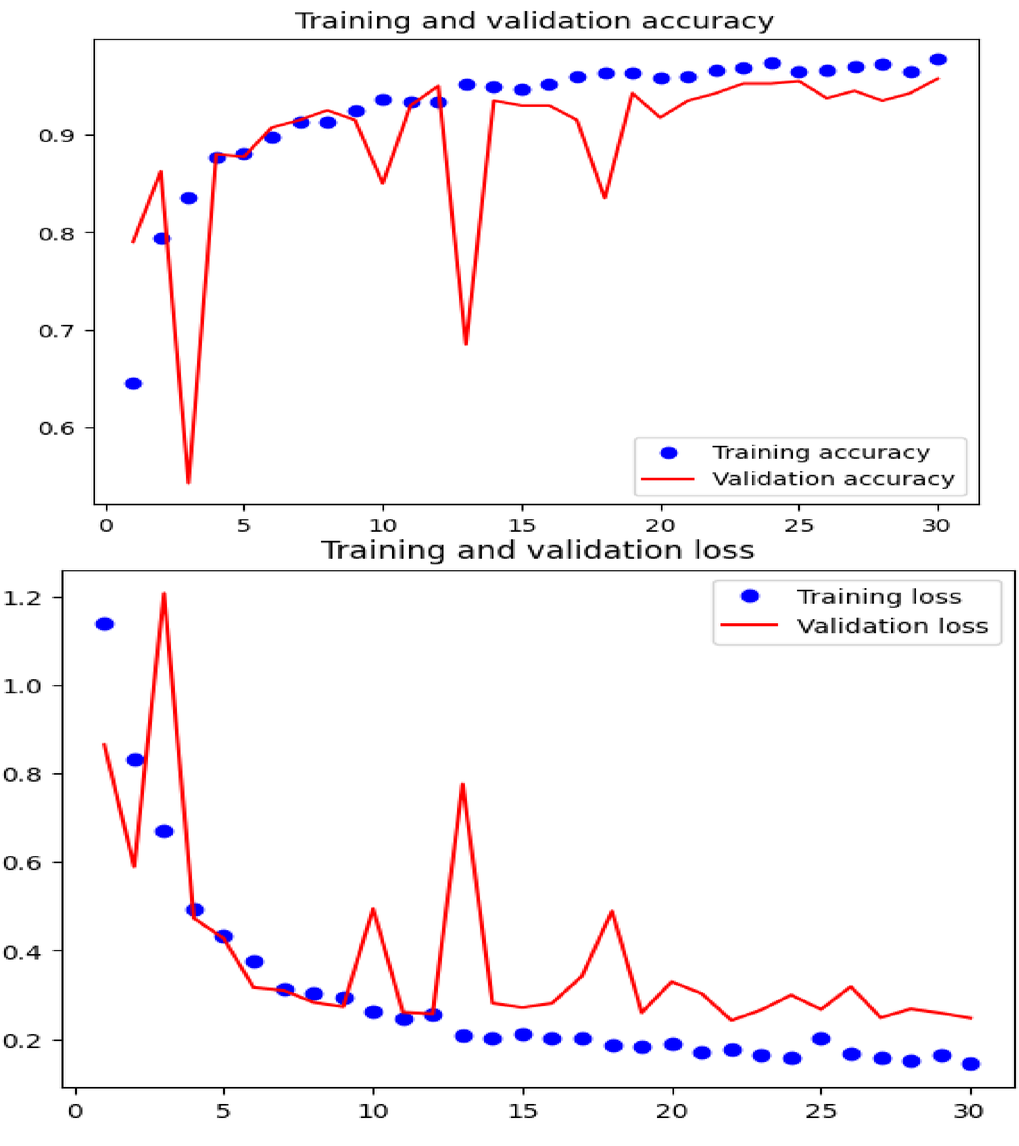
Model: "model_22"

Layer (type)	Output Shape	Param #
input_23 (InputLayer)	[(None, 180, 180, 3)]	0
rescaling_22 (Rescaling)	(None, 180, 180, 3)	0
conv2d_110 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_88 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_111 (Conv2D)	(None, 87, 87, 64)	18496
max_pooling2d_89 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_112 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_90 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_113 (Conv2D)	(None, 18, 18, 256)	295168
max_pooling2d_91 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_114 (Conv2D)	(None, 7, 7, 256)	590080
flatten_22 (Flatten)	(None, 12544)	0
dense_22 (Dense)	(None, 1)	12545

```
=================================================================
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0
_________________________________________________________________
```

III.2 -Developing a second, further improved CNN model named pvcm2 to classify Panda vs Cat using parameter regularization and then adding dropout.

<u>Plots and results for Pvcm2:</u>



```
loss: 0.1503 - accuracy: 0.9693 - val_loss: 0.1647 - val_accuracy:
0.9600
```

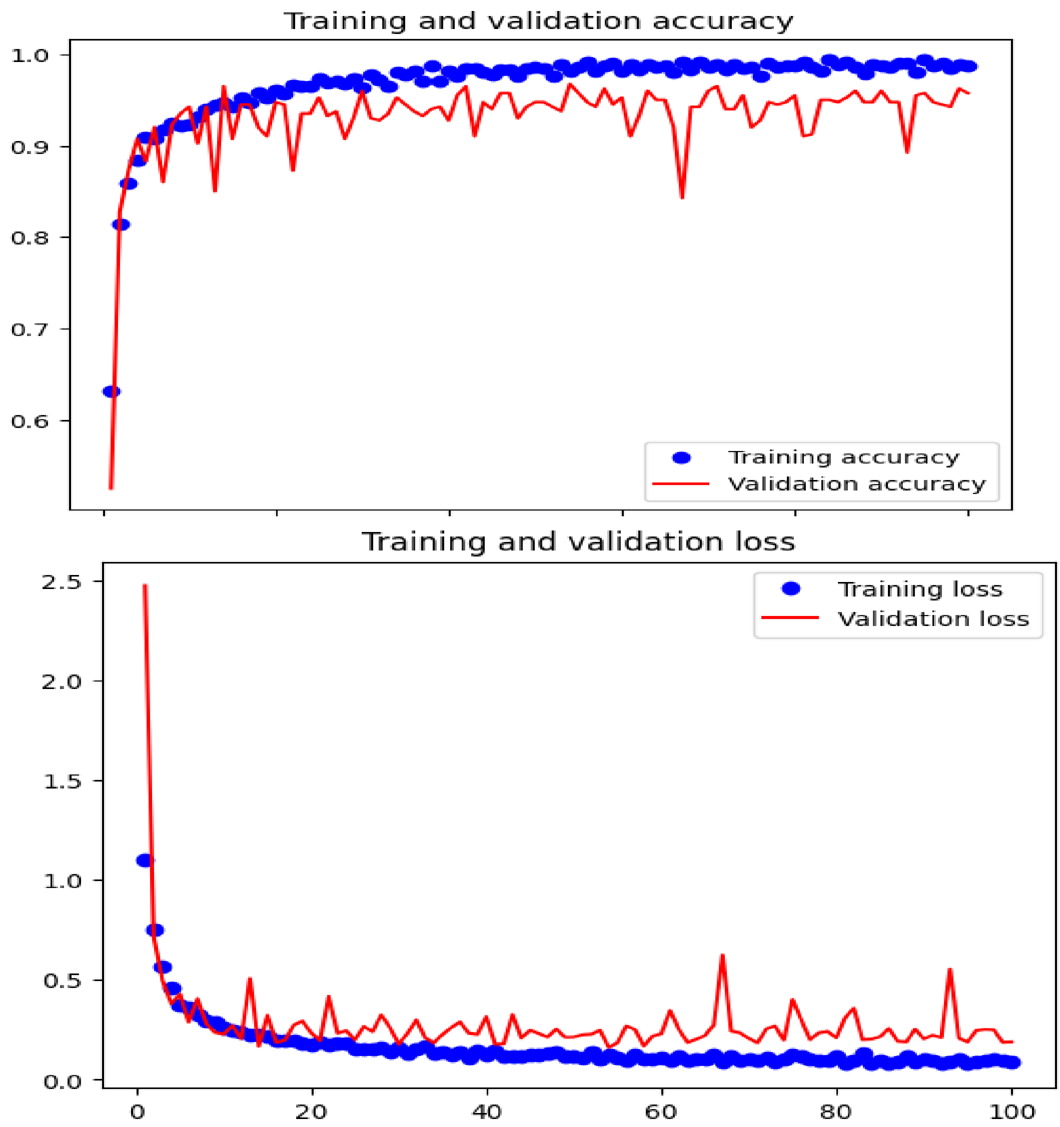
```
Test accuracy for pvcm2 model: 95.0
```

```
Summary for pvcm2 model:
```

```
Model: "model_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_25 (InputLayer)       [(None, 180, 180, 3)]     0

 rescaling_24 (Rescaling)    (None, 180, 180, 3)       0

 conv2d_120 (Conv2D)         (None, 178, 178, 32)      896

 max_pooling2d_96 (MaxPoolin  (None, 89, 89, 32)       0
 g2D)

 conv2d_121 (Conv2D)         (None, 87, 87, 64)        18496

 max_pooling2d_97 (MaxPoolin  (None, 43, 43, 64)       0
 g2D)

 conv2d_122 (Conv2D)         (None, 41, 41, 128)       73856

 max_pooling2d_98 (MaxPoolin  (None, 20, 20, 128)      0
 g2D)

 conv2d_123 (Conv2D)         (None, 18, 18, 256)       295168

 max_pooling2d_99 (MaxPoolin  (None, 9, 9, 256)        0
 g2D)

 conv2d_124 (Conv2D)         (None, 7, 7, 256)         590080

 flatten_24 (Flatten)        (None, 12544)             0

 dropout_9 (Dropout)         (None, 12544)             0

 dense_24 (Dense)            (None, 1)                 12545

=================================================================
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0
```

III.3 -Developing a third CNN model, named pvcm3 USING DATA AUGMENTATION FOR ITS TRAINING.

Plots and results for Pvcm3:




```
loss: 0.1247 - accuracy: 0.9814 - val_loss: 0.1612 - val_accuracy: 0.9675
```

```
Test Accuracy of pvcm3 model: 95.5
```

```
Summary of pvcm3 model:
```

```
Model: "model_26"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_25 (InputLayer)       [(None, 180, 180, 3)]     0

 rescaling_24 (Rescaling)    (None, 180, 180, 3)       0
```

```
 conv2d_120 (Conv2D)         (None, 178, 178, 32)      896

 max_pooling2d_96 (MaxPoolin  (None, 89, 89, 32)       0
 g2D)

 conv2d_121 (Conv2D)         (None, 87, 87, 64)        18496

 max_pooling2d_97 (MaxPoolin  (None, 43, 43, 64)       0
 g2D)

 conv2d_122 (Conv2D)         (None, 41, 41, 128)       73856

 max_pooling2d_98 (MaxPoolin  (None, 20, 20, 128)      0
 g2D)

 conv2d_123 (Conv2D)         (None, 18, 18, 256)       295168

 max_pooling2d_99 (MaxPoolin  (None, 9, 9, 256)        0
 g2D)

 conv2d_124 (Conv2D)         (None, 7, 7, 256)         590080

 flatten_24 (Flatten)        (None, 12544)             0

 dropout_9 (Dropout)         (None, 12544)             0

 dense_24 (Dense)            (None, 1)                 12545

=================================================================
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0
```

CONCLUSIONS:

The improvements and methods I followed in the above models are using callbacks for the model1 which was not used in the preliminary model. For model 2 I have added L2 regularizer for both the classifiers. I have chosen the different reg_strength values to best fit and generalize ,what I have observed in the process of selecting the reg_strength value is if the value of reg_strength is too high, the model may become too simple and underfit the data. And if the value of reg_strength is too low, the model may overfit the data. And in the next step for the both models I have added the drop out method with L2 regularizer which yielded the better results.

For the final best model of two classifiers I have augmented the data by different manipulations of rotation, flip, GaussianNoise, Translation, crop, zoom.

I feel that for the Cat-Vs-Panda classifier seems to be harder and it took a lot of time for getting the best model after data augmentation and achieved after so many tunings, though I am able to achieve a minor difference for Pvcm2 and Pvcm3, I am still trying to improve the accuracy of Pvcm3 by tuning the parameters and adding more manipulations.

I can definitely say that I can improve the accuracies of the models by using VGG16 architecture , it was a pre-trained older model which was trained on 1.4 million labelled images and 1000 different classes. I can improve by using feature extraction or by fine tuning the model.

All my models that is PVDM1, PVDM2, PVDM3 and PVCM1, PVCM2, PVCM3 did better than as random classifiers on the test set.

	Accuracy on the TEST dataset			
Panda-vs-Dog	Random Classifier	Pvdm1	Pvdm2	Pvdm3
	50 (12.5-hit ratio)	93.5	95.0	96.0
Panda-vs-Cat	Random Classifier	Pvcm1	Pvcm2	Pvcm3
	50 12.5-hit ratio)	91.0	95.0	95.5