Assessment
Pet Classification Model Using CNN

DESCRIPTION

Project Objective:

Build a CNN model that classifies the given pet images correctly into dog and cat images.
The project scope document specifies the requirements for the project “Pet Classification Model Using CNN.” Apart from specifying the functional and non-functional requirements for the project, it also serves as an input for project scoping.

Project Description and Scope:

You are provided with a collection of images of pets, that is, cats and dogs. These images are of different sizes with varied lighting conditions and they should be used as inputs for your model.

You are expected to write the code for CNN image classification model using TensorFlow that trains on the data and calculates the accuracy score on the test data.

Project Guidelines:

Begin by creating the ipynb file in the same parent folder where the downloaded data set is kept. The CNN model should have the following layers:

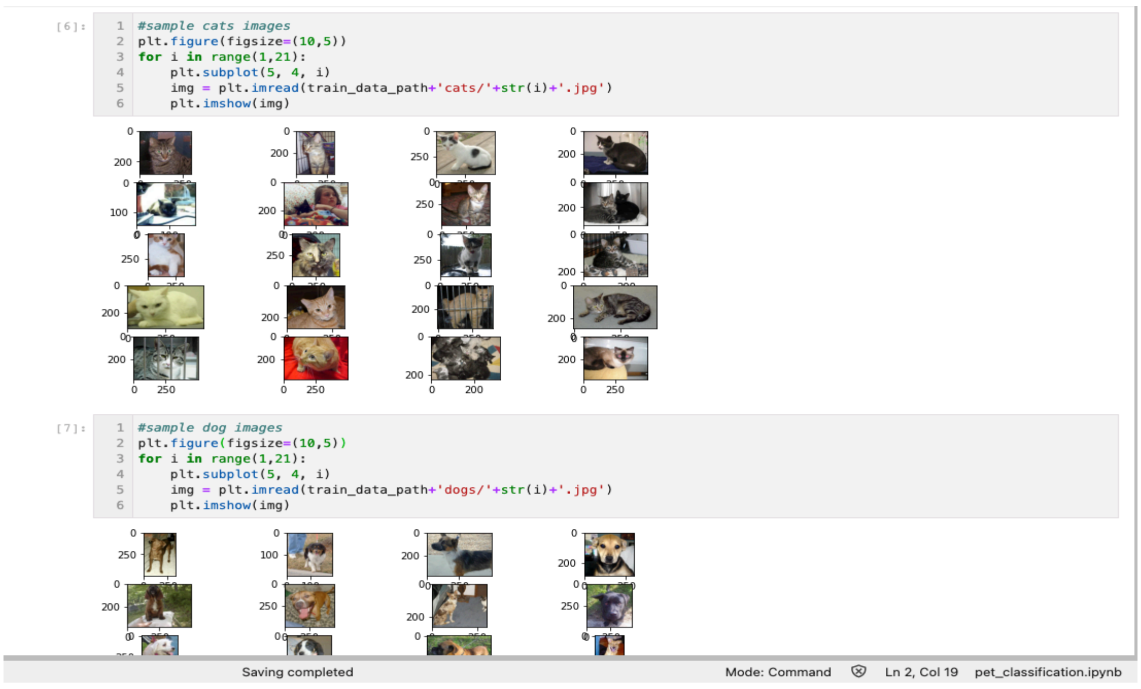
- Input layer
- Convolutional layer 1 with 32 filters of kernel size[5,5]
- Pooling layer 1 with pool size[2,2] and stride 2
- Convolutional layer 2 with 64 filters of kernel size[5,5]
- Pooling layer 2 with pool size[2,2] and stride 2
- Dense layer whose output size is fixed in the hyper parameter: fc_size=32
- Dropout layer with dropout probability 0.4

Predict the class by doing a softmax on the output of the dropout layers.
This should be followed by training and evaluation:

- For the training step, define the loss function and minimize it
- For the evaluation step, calculate the accuracy

Run the program for 100, 200, and 300 iterations, respectively. Follow this by a report on the final accuracy and loss on the evaluation data.

Sample Data



```
[6]:
#sample cats images
plt.figure(figsize=(10,5))
for i in range(1,21):
    plt.subplot(5, 4, i)
    img = plt.imread(train_data_path+'cats/'+str(i)+'.jpg')
    plt.imshow(img)
```

```
[7]:
#sample dog images
plt.figure(figsize=(10,5))
for i in range(1,21):
    plt.subplot(5, 4, i)
    img = plt.imread(train_data_path+'dogs/'+str(i)+'.jpg')
    plt.imshow(img)
```

Provided data consists of only 20 images of cats and dogs each for training and 10 images each for testing/validation.

Data Augmentation

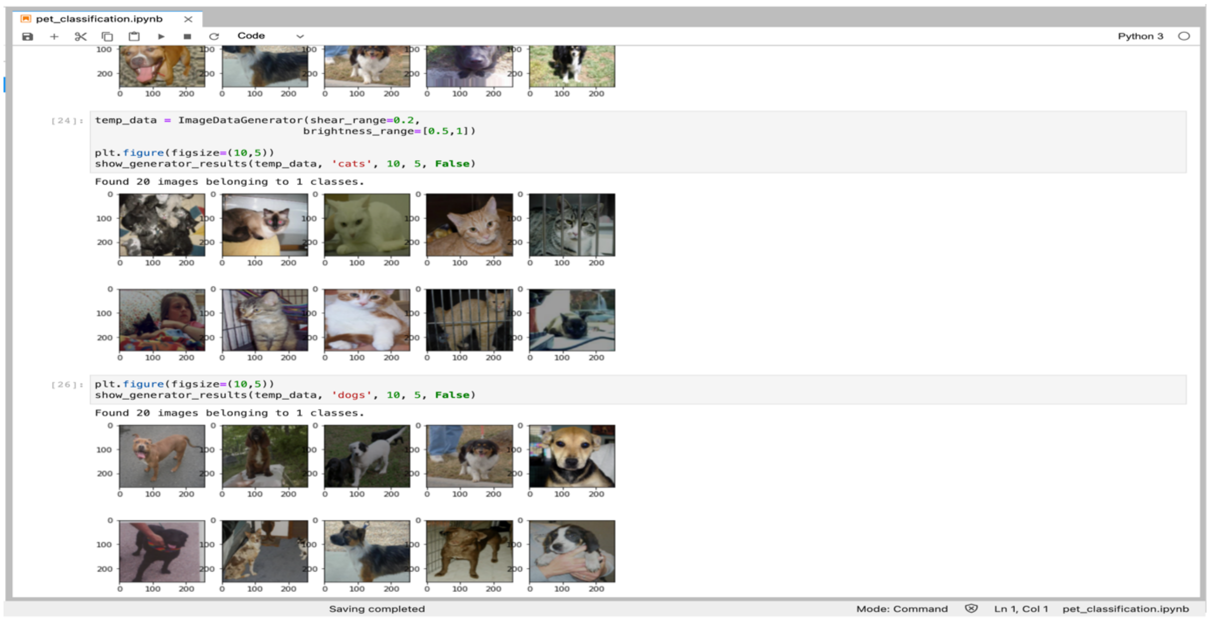
As the data is very low, we need to perform Data Augmentation.

Data augmentation is a technique to artificially create new training data from existing training data.

In this project augmentation technique used are

- Rescale
- Rotation
- Shear Transformation
- Zoom
- Height and Width Shift
- Brightness

Sample Results of Data Augmentation using Tensorflow Keras API ImageDataGenerator



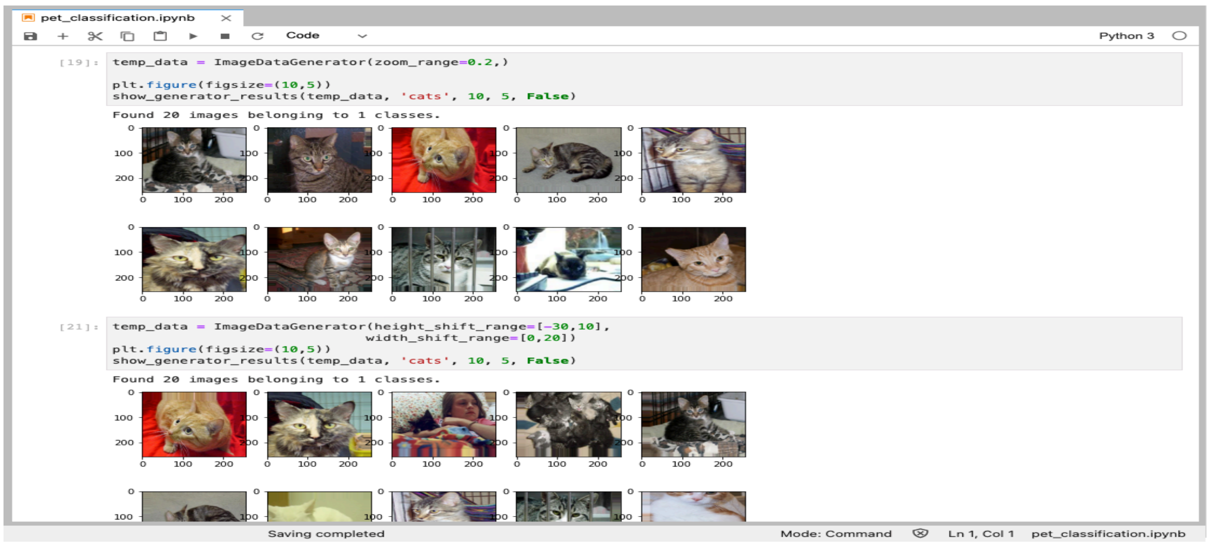
```
[24]: temp_data = ImageDataGenerator(shear_range=0.2,
                                     brightness_range=[0.5,1])

      plt.figure(figsize=(10,5))
      show_generator_results(temp_data, 'cats', 10, 5, False)

      Found 20 images belonging to 1 classes.
```

```
[26]: plt.figure(figsize=(10,5))
      show_generator_results(temp_data, 'dogs', 10, 5, False)

      Found 20 images belonging to 1 classes.
```



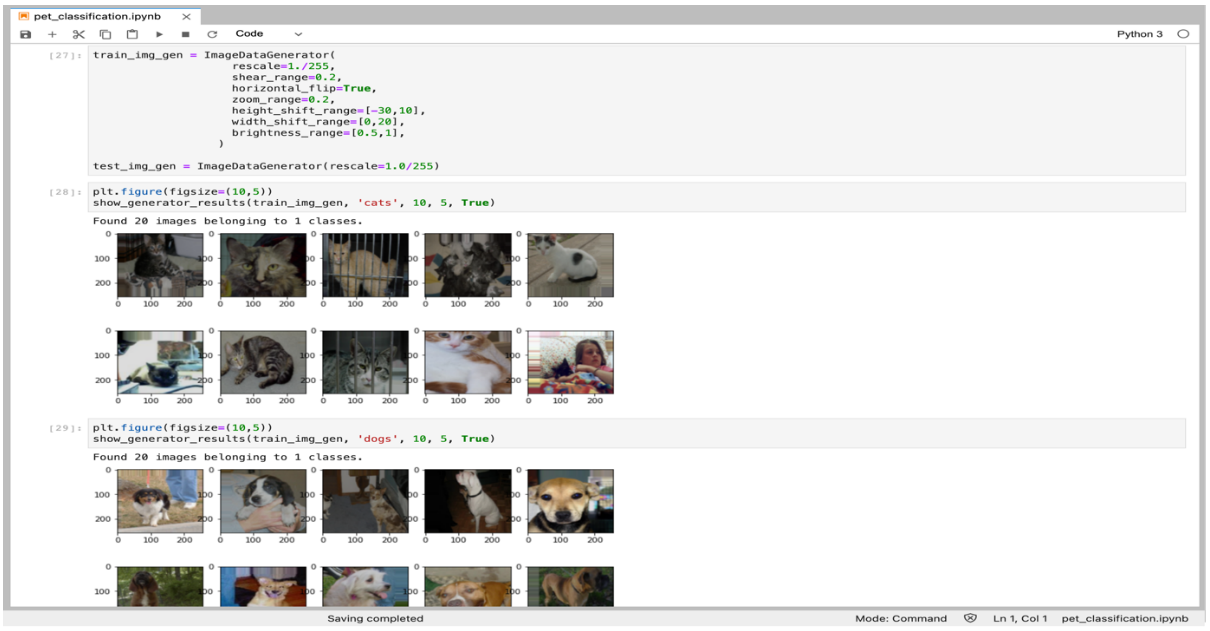
```
[19]: temp_data = ImageDataGenerator(zoom_range=0.2,)

      plt.figure(figsize=(10,5))
      show_generator_results(temp_data, 'cats', 10, 5, False)

      Found 20 images belonging to 1 classes.
```

```
[21]: temp_data = ImageDataGenerator(height_shift_range=[-30,10],
                                     width_shift_range=[0,20])
      plt.figure(figsize=(10,5))
      show_generator_results(temp_data, 'cats', 10, 5, False)

      Found 20 images belonging to 1 classes.
```



```
[27]: train_img_gen = ImageDataGenerator(
                          rescale=1./255,
                          shear_range=0.2,
                          horizontal_flip=True,
                          zoom_range=0.2,
                          height_shift_range=[-30,10],
                          width_shift_range=[0,20],
                          brightness_range=[0.5,1],
                      )

      test_img_gen = ImageDataGenerator(rescale=1.0/255)
```

```
[28]: plt.figure(figsize=(10,5))
      show_generator_results(train_img_gen, 'cats', 10, 5, True)

      Found 20 images belonging to 1 classes.
```

```
[29]: plt.figure(figsize=(10,5))
      show_generator_results(train_img_gen, 'dogs', 10, 5, True)

      Found 20 images belonging to 1 classes.
```

Model Building

As mentioned in the project guidelines, Model consists of 2 convolution layers with kernel size as 5x5 and filters 32 and 64 respectively. Each convolution layer is followed by a activation function 'relu' and max_pooling layer with filter size as 2x2 and strides as 2.

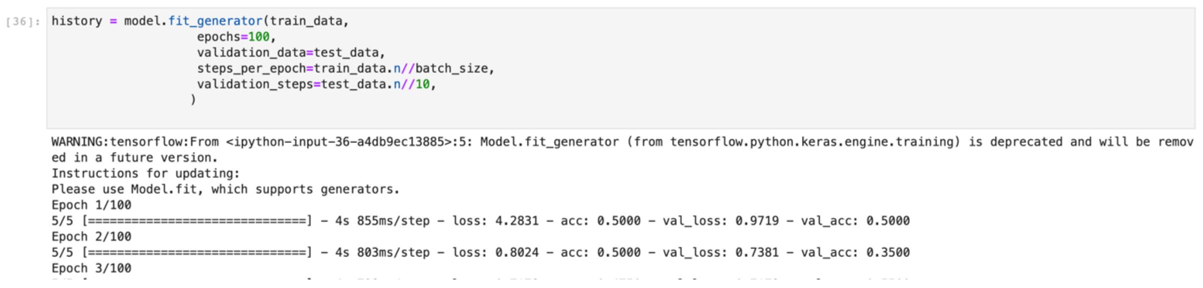
After 2nd Pooling layer image is flatten and passed to Fully connected 32 neurons.

Output layer consists of 2 neurons with activation function as 'softmax' which outputs the probability of each class i.e cat and dog.

Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 220, 220, 32)      2432
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 110, 110, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 106, 106, 64)      51264
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 53, 53, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 179776)            0
_________________________________________________________________
dense (Dense)                (None, 32)                5752864
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 66
=================================================================
Total params: 5,806,626
Trainable params: 5,806,626
Non-trainable params: 0
_________________________________________________________________
```

Model Training and Evaluation

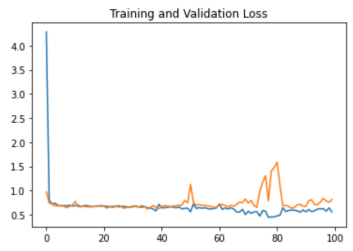


```
[36]: history = model.fit_generator(train_data,
                    epochs=100,
                    validation_data=test_data,
                    steps_per_epoch=train_data.n//batch_size,
                    validation_steps=test_data.n//10,
                   )
```

```
WARNING:tensorflow:From <ipython-input-36-a4db9ec13885>:5: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be remov
ed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/100
5/5 [==============================] - 4s 855ms/step - loss: 4.2831 - acc: 0.5000 - val_loss: 0.9719 - val_acc: 0.5000
Epoch 2/100
5/5 [==============================] - 4s 803ms/step - loss: 0.8024 - acc: 0.5000 - val_loss: 0.7381 - val_acc: 0.3500
Epoch 3/100
```

```
[39]: plt.plot(epochs, loss)
      plt.plot(epochs, val_loss)
      plt.title('Training and Validation Loss')
```

```
[39]: Text(0.5, 1.0, 'Training and Validation Loss')
```



```
[40]: _, acc = model.evaluate_generator(test_data, steps=test_data.n, verbose=0)
      print('> %.3f' % (acc * 100.0))
```

```
WARNING:tensorflow:From <ipython-input-40-ab90825a0fac>:1: Model.evaluate_generator (from tensorflow.python.keras.engine.training) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use Model.evaluate, which supports generators.
> 40.000
```

Conclusion

Model is trained for 100, 200 and 300 epochs and evaluation results are as follow,

Number of Epochs	100	200	300
Accuracy	40%	51%	60%