ANA 540: Pet Image Classification Project

1.1. Abstract

In this project, we were able to train a model that helped identify and classify images of dogs or a cat. Our model correctly identified 11 cats and 6 dogs. However, the model identified 1 dog as a cat; we are not entirely sure why we had this error, but our guess is that this picture has a human in the background and could have distorted the model's algorithm from performing its analysis accurately.

1.2. Introduction

The main goal in this project will be to develop a model that can distinguish and identify the images of cats and dogs. The model will be trained with distinct images of dogs and cats (training data) which will later be tested with random images to test the model's predictable power. I downloaded the Dogs vs Cats dataset from the Kaggle website. The dataset contains a set of images of cats and dogs.

Our main aim here is for the model to learn various distinctive features of cat and dog. Once the training of the model is done it will be able to differentiate images of cat and dog.

We are given a set of dog and cat images. The task is to build a model to predict the category of an animal: dog or cat.

1.3. Literature Review

Convolutional Neural Network (CNN) is an algorithm that is commonly used to build image classification models. It works by getting an image, designating it some weightage based on the different objects of the image, and then distinguishing them from each other. It is quite simple compared to other deep learning algorithms because it requires little to no pre-processed data and can handle large amounts of data.

CNNs were first developed and used around the 1980s. Initially the CNN algorithm could only recognize handwritten digits. It was used by the postal service sectors to read zip codes, pin codes, and simplistic digits. Since it required large data to better identify images and that was a major challenge in the 1980s. However, in 2012, Alex Krizhevsky realized that it was time to bring back this algorithm for deep learning but this time it had to use multi-layered neural networks. With the availability of large data sets training models with CNNs was a great comeback.

Convolutional Neural Network (CNN) is an algorithm taking an image as input then assigning weights and biases to all the aspects of an image and thus differentiates one from the other. Neural networks can be trained by using batches of images, each of them having a label to identify the real nature of the image (cat or dog here). A batch can contain few tenths to hundreds of images. For each image, the network prediction is compared with the corresponding existing label, and the distance between network prediction and the truth is evaluated for the

whole batch. Then, the network parameters are modified to minimize the distance and thus the prediction capability of the network is increased. The training process continues for every batch similarly.

1.4. Methodology

The first part of processing the data was the data cleaning. I downloaded the data from Kaggle and the data was generally clean. Cleaning data is usually the less exciting piece of data analysis.

We imported the Keras ImageDataGenerator is used to take the inputs of the original data and then transform it on a random basis, returning the output resultant containing solely the newly changed data. Keras ImageDataGenerator is used for data augmentation to generate batches comprising data from tensor pictures. In data augmentation, actions such as translations, rotations, scale changes, and vertical flips are performed at random utilizing an image data generator.

When all was completed, we proceeded with fitting the model. To ensure that the model is properly trained, an epoch of 30 was specified to ensure sufficient training iterations. Each sample from our training dataset will be used once per epoch, whether it is for training or validation because the more epochs, the more the model is trained. The key is to identify the number of epochs that fits the model to the data without overfitting. Based on our visualized model training plot, the model's accuracy was stable between 21 to 27 iterations.

The test generator was created to test the model and proceeded to making predictions with the model and visualize the results.

1.5 Data Analysis

Convolution is a linear operation involving the multiplication of weights with the input. The multiplication is performed between the layers of a network and an array of input data. A 2D array of weights, known as filter or kernel, is applied to the network. The filter is always smaller than input data and the dot product is performed between input and filter array.

The output from the final Pooling layer which is flattened is the input of the fully connected layer.

The Full Connection process practically works as follows:

The neurons present in the fully connected layer detect a certain feature and preserves its value then communicates the value to both the dog and cat classes who then check out the feature and decide if the feature is relevant to them. Will explain clearly when we get to model building.

2. Exploratory Data Analysis

2.1. Import and Load Libraries

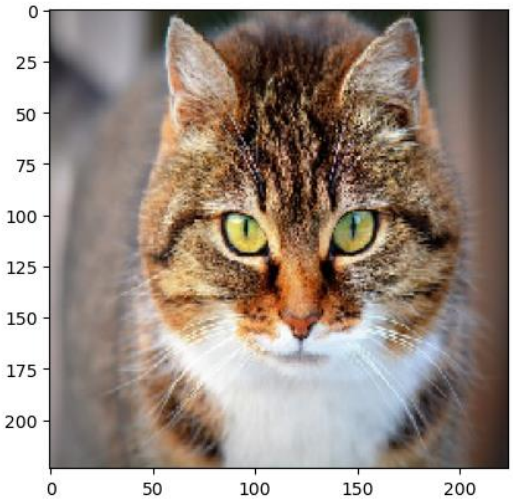
Our analysis was completed in Python. We started off by importing the important libraries and set up directories accordingly. Few of which include; tensorflow, numpy, pandas, random keras.utils, sklearn.model and matplotlib.

2.2. Explore Data

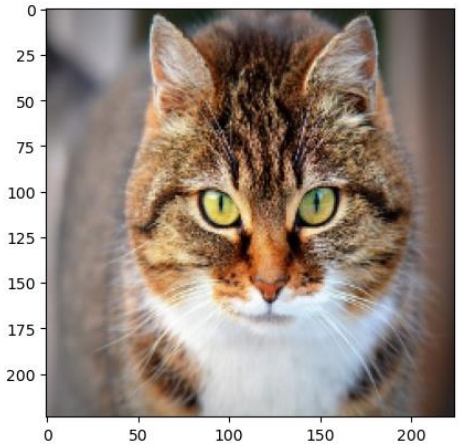
Methods of loading images:

Before jumping into complex analysis, i wanted to explore different methods of manipulating and uploading images on python. We will go ahead and sample one image file cat-300572.jpg for our analysis.

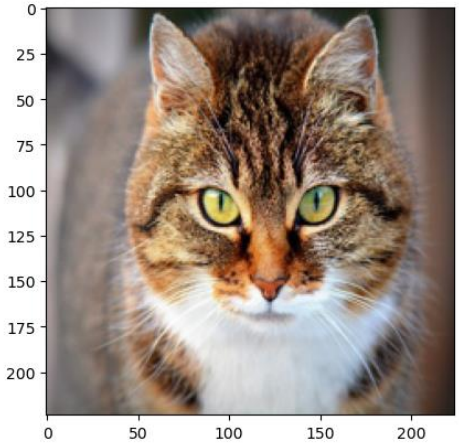
Method 1: Using tensorflow



Method 2: Using open cv



Method 3: Using Python imaging library (PIL)



Define Constants

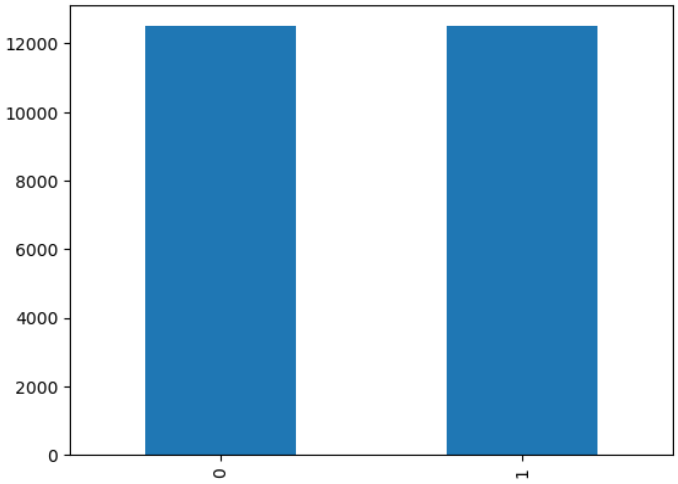
FAST_RUN = False
IMAGE_WIDTH=128
IMAGE_HEIGHT=128
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3

Prepare Training Data

	filename	category
0	cat.0.jpg	0
1	cat.1.jpg	0
2	cat.10.jpg	0
3	cat.100.jpg	0
4	cat.1000.jpg	0

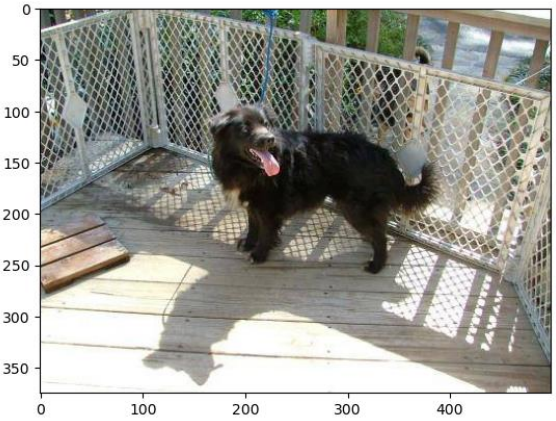
	filename	category
24995	dog.9995.jpg	1
24996	dog.9996.jpg	1
24997	dog.9997.jpg	1
24998	dog.9998.jpg	1
24999	dog.9999.jpg	1

See total count (How many cat vs dog pictures)

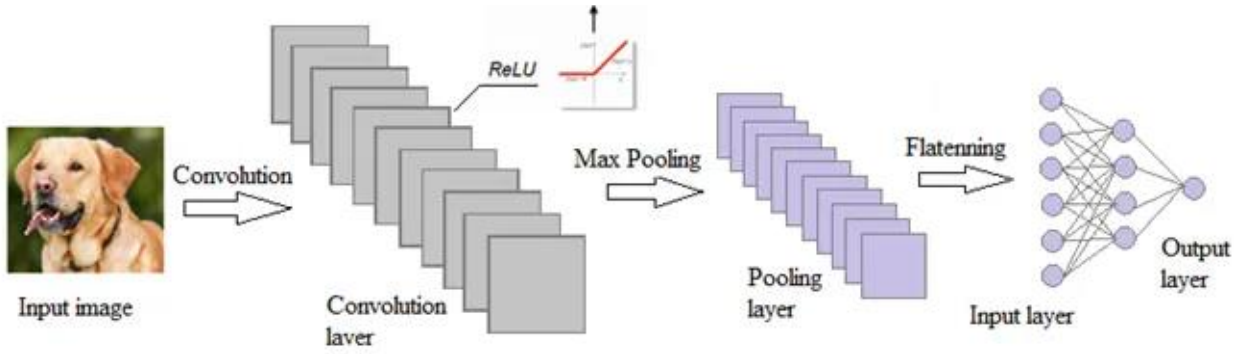


From our data we have 12000 cats and 12000 dogs

Query and display sample image from training data.



Build Model



Input Layer: It represent input image data. It will reshape image into single diminsion array. Example your image is 64x64 = 4096, it will convert to (4096,1) array.

Convolution Layer: This layer will extract features from image.

Pooling Layer: This layer reduces the spatial volume of input image after convolution.

Fully Connected Layer: It connect the network from a layer to another layer.

Output Layer: It is the predicted values layer.

Early Stop

To prevent over fitting we will stop the learning after 10 epochs and val_loss value not decreased.

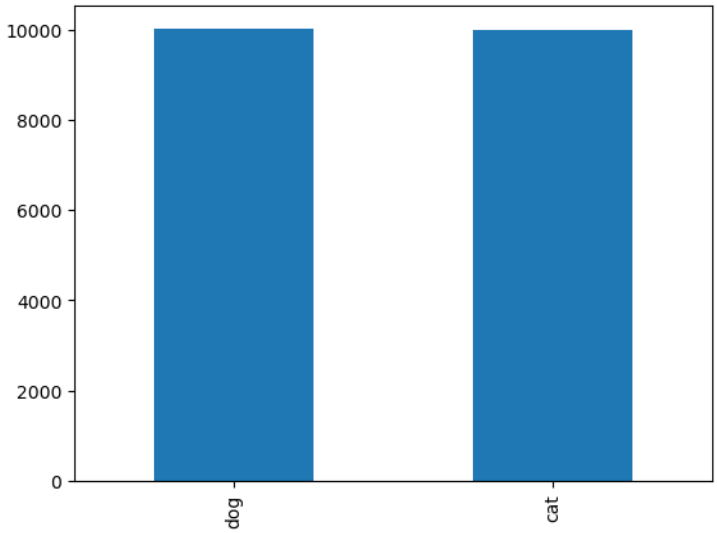
Learning Rate Reduction

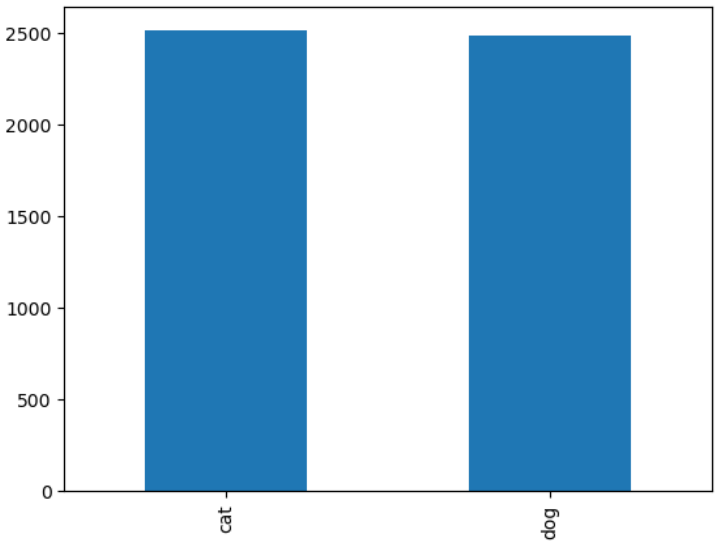
We will reduce the learning rate when then accuracy does not increase for 2 steps.

Prepare data.

Because we will use image genaretor with class_mode="categorical". We need to convert column category into string. Then imagenerator will convert it one-hot encoding which is good for our classification.

So, we will convert 1 to dog and 0 to cat

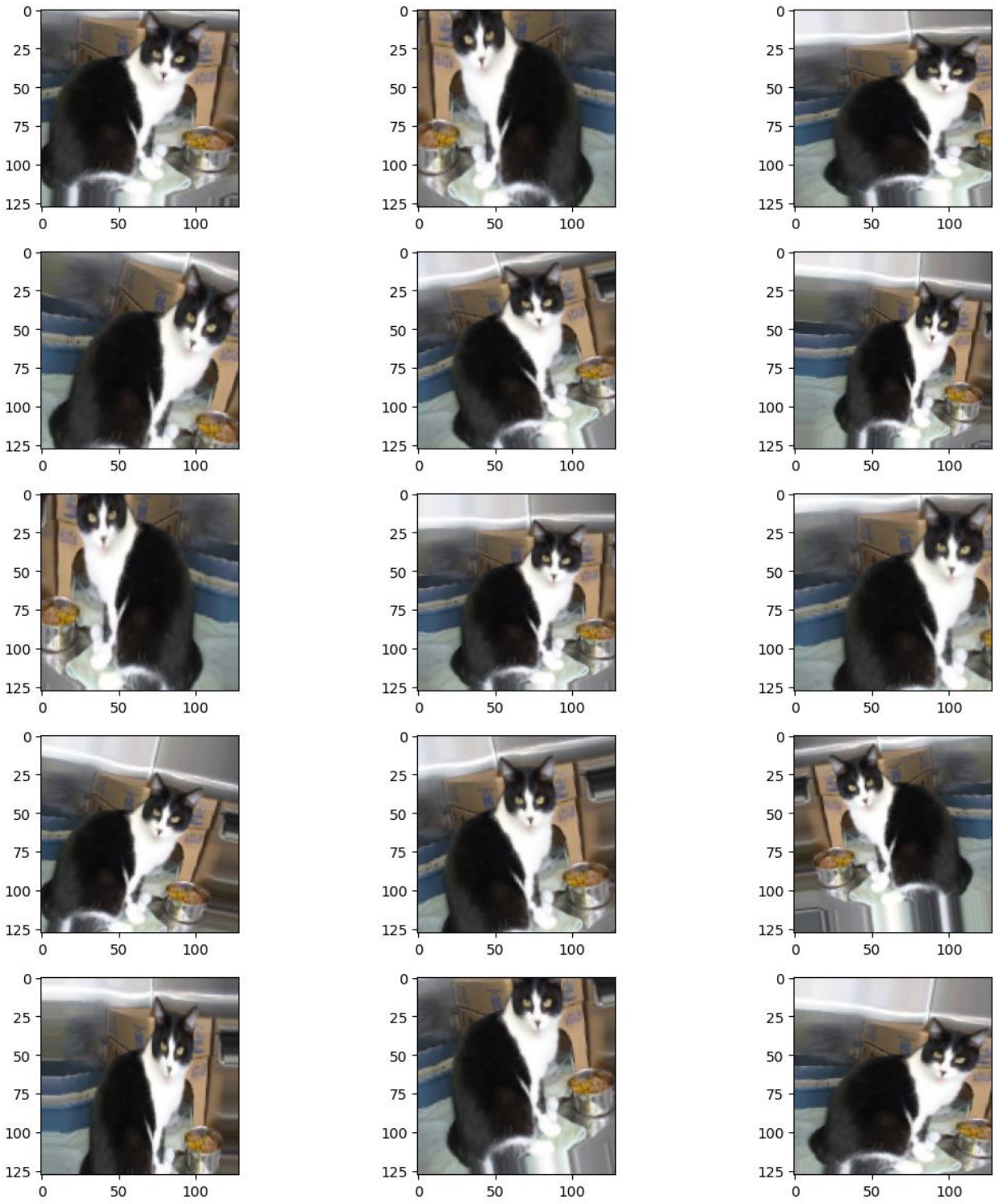




Training Generator
Found 20000 validated image filenames belonging to 2 classes.

Validation Generator
Found 5000 validated image filenames belonging to 2 classes.

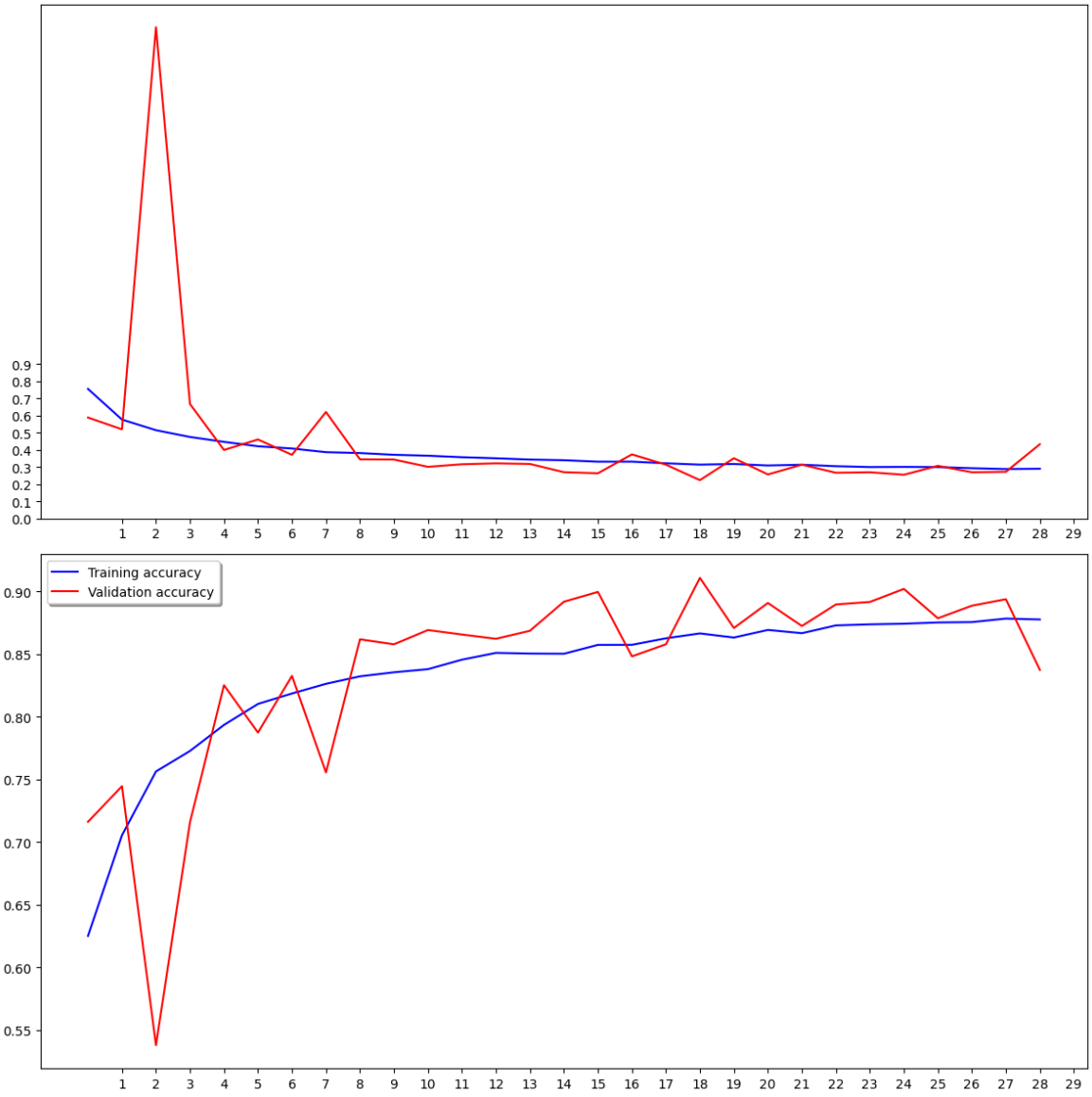
See how our generator work.



Fit the Model

The model accuracy from training fluctuated between 70% to 90% accuracy. I believe this model will be a good fit because it will not suffer from over fitting or under fitting.

Virtualize Model Training Process



Model seems to have successfully completed its training process and ready to move to the testing phase.

Preparing Testing Data

We will go ahead and import the test data from our repository.

Create Testing Generator
Found 12500 validated image filenames.

The test generator seems to have worked out well and this will usher us into the next phase of prediction.

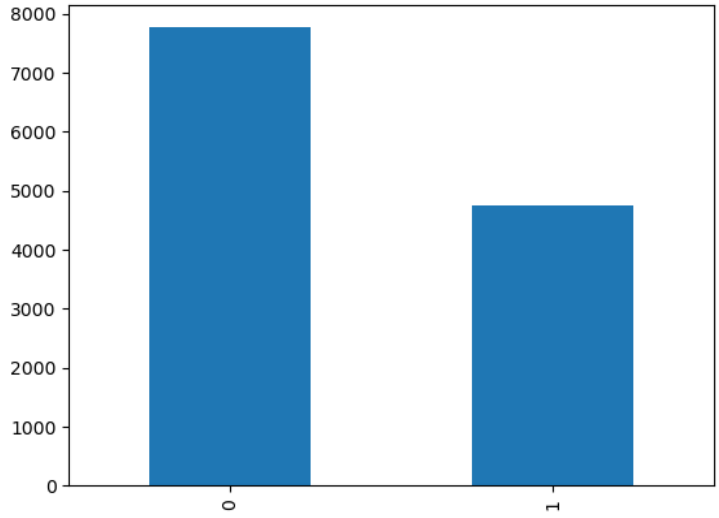
Making Predictions with Model

For categorical classification the prediction will come with probability of each category. So, we will pick the category that have the highest probability with numpy average max

We will convert the predict category back into our generator classes by using train_generator.class_indices. It is the classes that image generator map while converting data into computer vision.

From our prepare data part. We map data with {1: 'dog', 0: 'cat'}. Now we will map the result back to dog is 1 and cat is 0.

Virtualize Model Result



Our results indicate that the model identified about 5000 dogs and over 7500 cats.

See predicted result with images.

Results and conclusion

Key findings

Our predictions identified a sample of 11 cats and 6 dogs correctly. However, due to the model's accuracy level it did not correctly identify 1 dog. Our guess is that this picture has a human in the background and could have distorted our algorithm.

Recommendations for Future Research

- I will advise to train model moderately and not over train or under train model because the accuracy level may start fluctuating.
- Once you understand the concept of Image Classification, now you can try different classification like Lung Cancer Detection using CNN.

Conclusion

Generally, our model results indicate that the model identified about 5000 dogs and over 7500 cats. This is probably not our best result, but it is due to a shorter time training the model.

We can conclude that the model works well and has a high level of accuracy if a picture of a dog or cat has only the pet in that picture without third party images.

References

Aurelien, Geron. (June 2019) "Hands on Machine Learning with Scikit Learn Keras, and Tensorflow: Concepts, Tools and Techniques to Build Intelligent Systems" Second Edition, OReilly Media 2019. pp. 87-189

Khushi, Shah. (June 2021) "Beginner-friendly Project- Cat and Dog classification using CNN" Data Science Blogathon article.

Uysim. (June 2019) "Keras CNN Dog or Cat Classification" Kaggle.

Luke, Sun. (April 2020) "CNN Image Classification: Cat or Dog" Technical Walk-through on Convolution Neural Network using Keras for Image Classification.

Appendix

Project Python Syntax:

```
import tensorflow as tf
import numpy as np
import pandas as pd
import random
import os # importing directory command
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
os.getcwd() # identifying/display current directory
os.chdir('C:\\Users\\LENOVO\\Desktop\\Python\\image_data') # Changing current directory
filename = 'cat-300572.jpg'
from tensorflow.keras.preprocessing import image
img = image.load_img(filename, target_size = (224,224))
plt.imshow(img)

%pip install opencv-python
import cv2
img2 = cv2.imread(filename)
plt.imshow(img2)
plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))
img2 = cv2.resize(img2,(224,224))
plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))

# %pip install pillow
from PIL import Image
im = Image.open(filename)
im = im.resize((224, 224))
plt.imshow(im)

FAST_RUN = False
IMAGE_WIDTH=128
IMAGE_HEIGHT=128
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3
filenames = os.listdir('C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\train_data')
categories = []
for filename in filenames:
    category = filename.split('.')[0]
    if category == 'dog':
        categories.append(1)
    else:
```

```
    categories.append(0)

df = pd.DataFrame({
   'filename': filenames,
   'category': categories
})
df['category'].value_counts().plot.bar()
from tensorflow.keras.utils import load_img
sample = random.choice(filenames)
image = load_img('C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\train_data\\'+sample)
plt.imshow(image)
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation,
BatchNormalization

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT,
IMAGE_CHANNELS)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax')) # 2 because we have cat and dog classes

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

model.summary()

from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

```
earlystop = EarlyStopping(patience=10)
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=2,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
callbacks = [earlystop, learning_rate_reduction]
df["category"] = df["category"].replace({0: 'cat', 1: 'dog'})
train_df, validate_df = train_test_split(df, test_size=0.20, random_state=42)
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
train_df['category'].value_counts().plot.bar()
validate_df['category'].value_counts().plot.bar()
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
batch_size=15

from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    'C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\train_data\\',
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = validation_datagen.flow_from_dataframe(
    validate_df,
    'C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\train_data\\',
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
```

```
    class_mode='categorical',
    batch_size=batch_size
)
example_df = train_df.sample(n=1).reset_index(drop=True)
example_generator = train_datagen.flow_from_dataframe(
    example_df,
    'C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\train_data\\',
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical'
)
plt.figure(figsize=(12, 12))
for i in range(0, 15):
    plt.subplot(5, 3, i+1)
    for X_batch, Y_batch in example_generator:
        image = X_batch[0]
        plt.imshow(image)
        break
plt.tight_layout()
plt.show()

epochs=3 if FAST_RUN else 30
history = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=total_validate//batch_size,
    steps_per_epoch=total_train//batch_size,
    callbacks=callbacks
)
model.save_weights("model.h5")

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 12))
ax1.plot(history.history['loss'], color='b', label="Training loss")
ax1.plot(history.history['val_loss'], color='r', label="validation loss")
ax1.set_xticks(np.arange(1, epochs, 1))
ax1.set_yticks(np.arange(0, 1, 0.1))

ax2.plot(history.history['accuracy'], color='b', label="Training accuracy")
ax2.plot(history.history['val_accuracy'], color='r',label="Validation accuracy")
ax2.set_xticks(np.arange(1, epochs, 1))

legend = plt.legend(loc='best', shadow=True)
```

```
plt.tight_layout()
plt.show()
test_filenames = os.listdir("C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\test_data")
test_df = pd.DataFrame({
   'filename': test_filenames
})
nb_samples = test_df.shape[0]
test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(
   test_df,
   "C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\test_data",
   x_col='filename',
   y_col=None,
   class_mode=None,
   target_size=IMAGE_SIZE,
   batch_size=batch_size,
   shuffle=False
)
predict = model.predict_generator(test_generator, steps=np.ceil(nb_samples/batch_size))
test_df['category'] = np.argmax(predict, axis=-1)
label_map = dict((v,k) for k,v in train_generator.class_indices.items())
test_df['category'] = test_df['category'].replace(label_map)
test_df['category'] = test_df['category'].replace({ 'dog': 1, 'cat': 0 })
test_df['category'].value_counts().plot.bar()

sample_test = test_df.head(18)
sample_test.head()
plt.figure(figsize=(12, 24))
for index, row in sample_test.iterrows():
   filename = row['filename']
   category = row['category']
   img = load_img("C:\\Users\\LENOVO\\Desktop\\Python\\image_data\\test_data\\"+filename,
target_size=IMAGE_SIZE)
   plt.subplot(6, 3, index+1)
   plt.imshow(img)
   plt.xlabel(filename + '(' + "{}".format(category) + ')' )
plt.tight_layout()
plt.show()
```