VIDYAVARDHAKA COLLEGE OF ENGINEERING

DEPARTMENT OF

CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

Report On

Animal Classification and Prediction

By:

AKASH BASKAR S

Aim of the Model:

The aim of this project is to develop a Convolutional Neural Network (CNN) for classifying images of cats and dogs. The model will be trained on the cat and dog dataset, which contains images of cats and dogs. The goal is to accurately distinguish between the two classes and create a robust classifier capable of generalizing to unseen images.

Objective:

The main objective of this project is to build a machine learning model that can accurately classify images as either cats or dogs. This binary classification task will be achieved using a CNN architecture, which is well-suited for image recognition tasks. The model will be trained to learn and identify patterns in the images that differentiate between cats and dogs.

Data Set:

The dataset used in this project is the AFHQ dataset, which consists of images of cats and dogs. The images are of varying sizes and dimensions. To ensure uniformity and efficient processing, the images are preprocessed by resizing them to a standardized 30x30 pixel resolution and converting them to grayscale. Additionally, the pixel values are normalized to the range [0, 1] to facilitate convergence during training.

Model Architecture:

The CNN model architecture is designed to capture and learn spatial features from the images. It consists of two convolutional layers, each followed by a ReLU activation function to introduce non-linearity. Max-pooling layers are added after each convolutional layer to reduce the spatial dimensions of the feature maps. The flattened output from the convolutional layers is passed through a fully connected layer with 64 neurons and a ReLU activation. Finally, the output layer consists of a single neuron with a Sigmoid activation function, enabling binary classification.

Training Process:

The model is trained on the preprocessed dataset using the Adam optimizer and binary cross-entropy loss function. The dataset is split into training and validation sets, with 70% used for training and 30% for validation. During the training process, the model learns to minimize the binary cross-entropy loss and improve its accuracy in distinguishing between cats and dogs.

Model Evaluation:

The model's performance is evaluated using the validation set. As this initial implementation is trained for only one epoch, the evaluation may not be fully representative of the model's potential. Future iterations can explore different hyperparameters, increase the number of epochs, and incorporate data augmentation techniques to improve the model's accuracy and robustness.

Model Prediction:

The trained model is saved and used to predict the class of new images. To make a prediction, the new image is preprocessed in the same manner as the training data, and the model outputs a probability score, indicating the likelihood of the image being a cat or a dog. The class with the highest probability is considered the predicted class.

Conclusion:

In conclusion, this project demonstrates the development of a simple CNN model for classifying images of cats and dogs. While the initial implementation shows promising results, there is room for further refinement and improvement. By exploring various strategies such as hyperparameter tuning, data augmentation, and utilizing more advanced CNN architectures, the model's accuracy and generalization capabilities can be enhanced. With continued development and optimization, the model could be deployed in various applications, such as image recognition systems, pet identification, and more.

Working of Model

1. **Dataset**: The model is trained and validated on a dataset containing images of cats and dogs from the AFHQ dataset.
2. **Data Preprocessing**: The images are resized to 30x30 pixels and converted to grayscale. Pixel values are scaled to the range [0, 1].
3. **Training Parameters**:
    - Batch size: 32
    - Epochs: 1
    - Loss function: Binary Crossentropy

    - Optimizer: Adam

4. **Training Process**: The model is trained on the dataset with a validation split of 30% (70% training, 30% validation). The model is saved as '64x3CNN.model' after training.
5. **Model Evaluation**: Since the model is trained on a small number of epochs (1 epoch), it may not achieve optimal performance. For a more accurate evaluation, training for more epochs may be necessary.
6. **Model Prediction**: The model is used to predict the class (cat or dog) of a sample image.
7. **Prediction Result**: The model's prediction for the provided image is displayed.

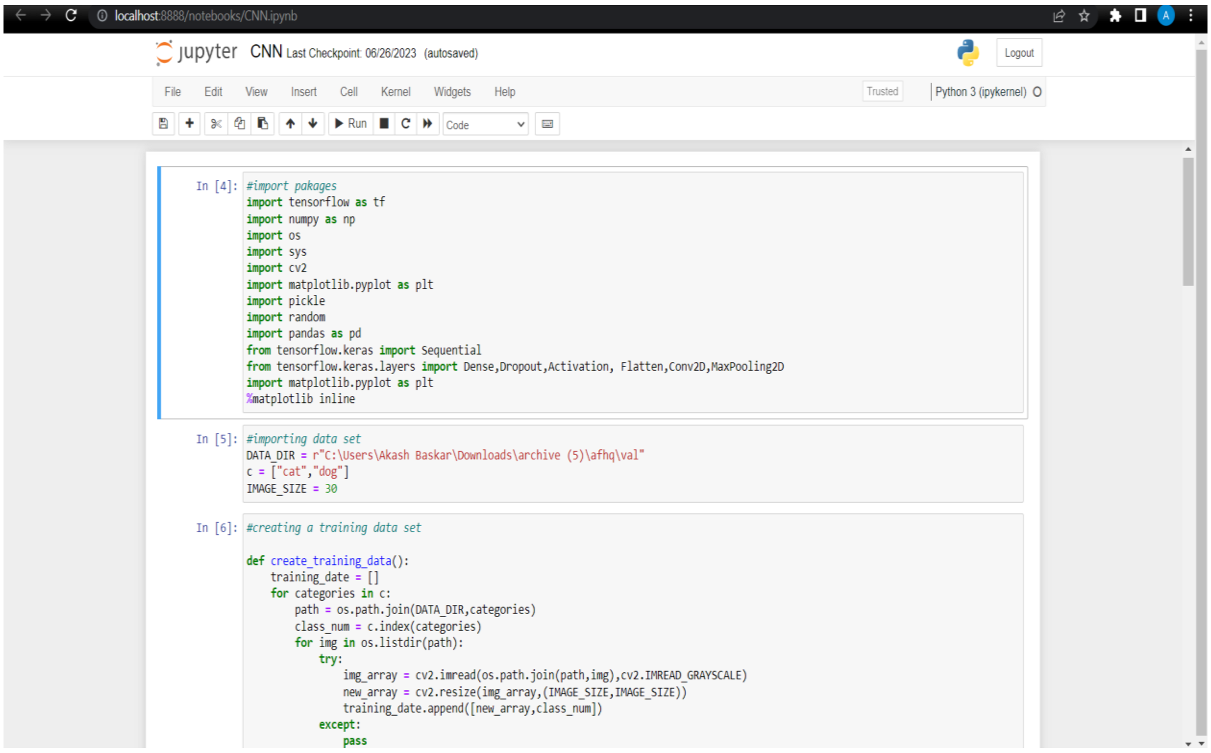
Source Code:

```
#!/usr/bin/env python

# coding: utf-8


# In[4]:


#import pakages

import tensorflow as tf

import numpy as np

import os

import sys
import cv2

import matplotlib.pyplot as plt

import pickle

import random

import pandas as pd

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Dense,Dropout,Activation,
Flatten,Conv2D,MaxPooling2D

import matplotlib.pyplot as plt

get_ipython().run_line_magic('matplotlib', 'inline')


# In[5]:


#importing data set

DATA_DIR = r"C:\Users\Akash Baskar\Downloads\archive
(5)\afhq\val"

c = ["cat","dog"]

IMAGE_SIZE = 30


# In[6]:


#creating a training data set


def create_training_data():
    training_date = []
    for categories in c:
        path = os.path.join(DATA_DIR,categories)
        class_num = c.index(categories)
        for img in os.listdir(path):
            try:
                img_array =
cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                new_array =
cv2.resize(img_array,(IMAGE_SIZE,IMAGE_SIZE))
                training_date.append([new_array,class_num])
            except:
                pass
    return training_date


# In[7]:


data = np.asarray(create_training_data())


# In[8]:


x_data = []
y_data = []
```

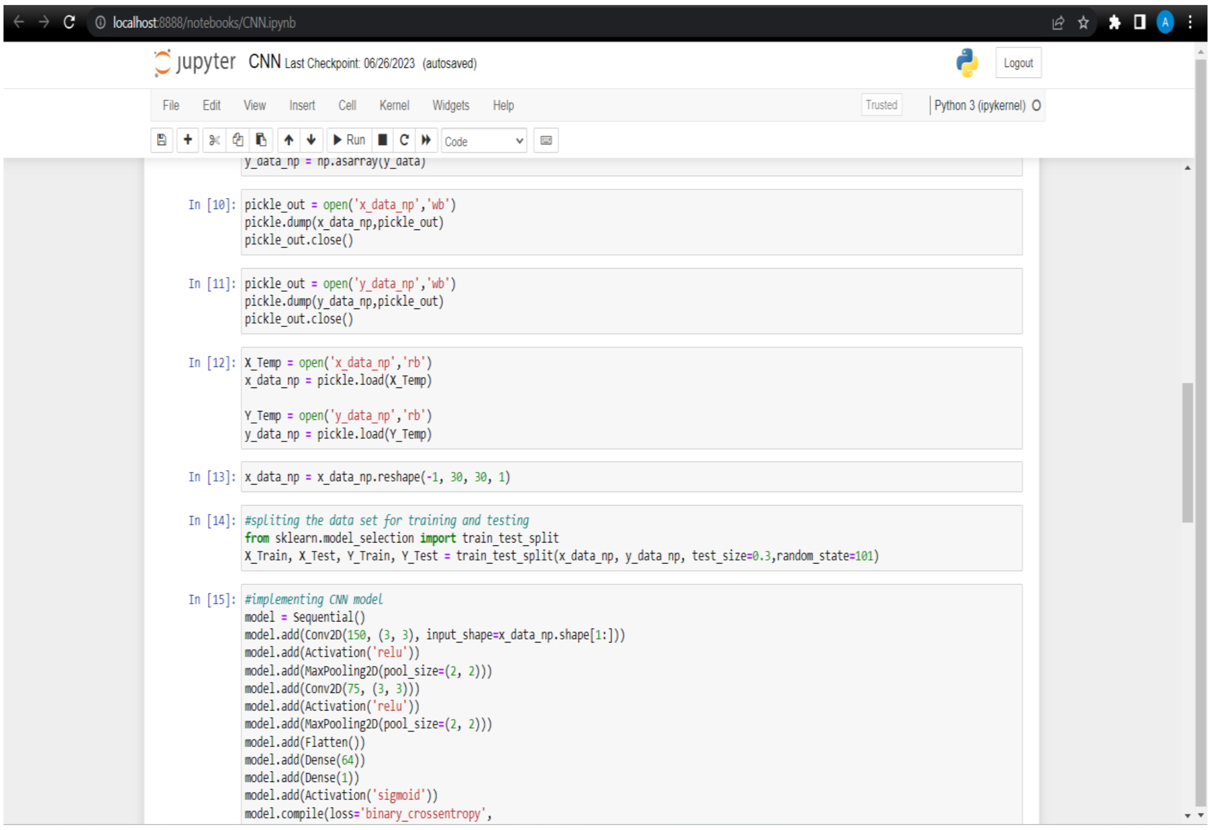
```
for x in data:
    x_data.append(x[0])
    y_data.append(x[1])

# In[9]:


x_data_np = np.asarray(x_data)/255.0
y_data_np = np.asarray(y_data)

# In[10]:


pickle_out = open('x_data_np','wb')
pickle.dump(x_data_np,pickle_out)
pickle_out.close()

# In[11]:


pickle_out = open('y_data_np','wb')
pickle.dump(y_data_np,pickle_out)
pickle_out.close()

# In[12]:


X_Temp = open('x_data_np','rb')
```

```
x_data_np = pickle.load(X_Temp)


Y_Temp = open('y_data_np','rb')
y_data_np = pickle.load(Y_Temp)


# In[13]:


x_data_np = x_data_np.reshape(-1, 30, 30, 1)


# In[14]:


#spliting the data set for training and testing
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(x_data_np,
y_data_np, test_size=0.3,random_state=101)


# In[15]:


#implementing CNN model
model = Sequential()
model.add(Conv2D(150, (3, 3),
input_shape=x_data_np.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(75, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(64))

model.add(Dense(1))

model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',

              optimizer='adam',

              metrics=['accuracy'])


# In[16]:


#training the data set

model.fit(x_data_np, y_data_np, batch_size=32, epochs=1,
validation_split=0.3)

model.save('64x3CNN.model')


# In[17]:


def prepare(filepath):

    training_date = []


    img_array = cv2.imread(filepath,cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array,(IMAGE_SIZE,IMAGE_SIZE))
    new_image =  new_array.reshape(-
1,IMAGE_SIZE,IMAGE_SIZE,1)
    return new_image


# In[18]:


model = tf.keras.models.load_model('64x3CNN.model')


# In[22]:


#importing the image to get the output
filepath = 'C:/Users/Akash Baskar/Desktop/cat.jpg'
img_array = cv2.imread(filepath,cv2.IMREAD_GRAYSCALE)

plt.imshow(img_array)


# In[20]:


test = model.predict([prepare(filepath=filepath )])


# In[21]:


print(c[int(test[0][0])])
```

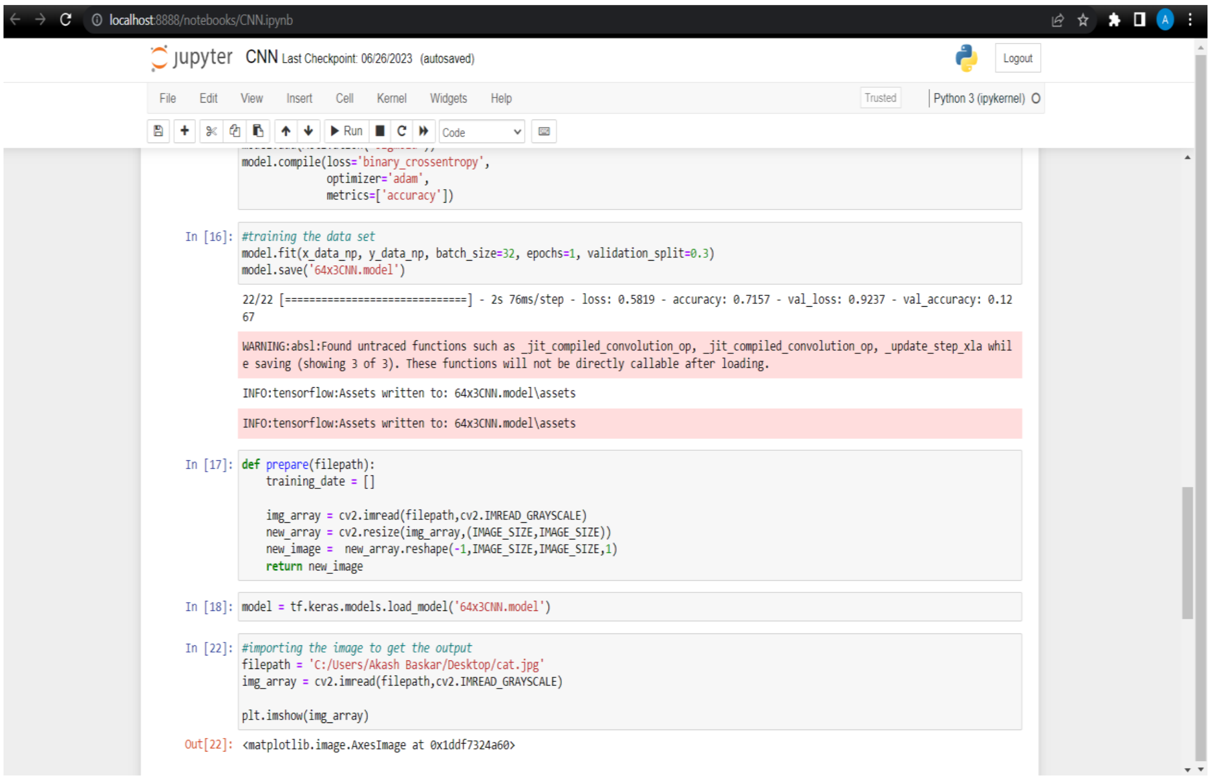
Code Images:


localhost:8888/notebooks/CNN.ipynb
jupyter CNN Last Checkpoint: 06/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [4]: #import pakages
        import tensorflow as tf
        import numpy as np
        import os
        import sys
        import cv2
        import matplotlib.pyplot as plt
        import pickle
        import random
        import pandas as pd
        from tensorflow.keras import Sequential
        from tensorflow.keras.layers import Dense,Dropout,Activation, Flatten,Conv2D,MaxPooling2D
        import matplotlib.pyplot as plt
        %matplotlib inline

In [5]: #importing data set
        DATA_DIR = r"C:\Users\Akash Baskar\Downloads\archive (5)\afhq\val"
        c = ["cat","dog"]
        IMAGE_SIZE = 30

In [6]: #creating a training data set

        def create_training_data():
            training_date = []
            for categories in c:
                path = os.path.join(DATA_DIR,categories)
                class_num = c.index(categories)
                for img in os.listdir(path):
                    try:
                        img_array = cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                        new_array = cv2.resize(img_array,(IMAGE_SIZE,IMAGE_SIZE))
                        training_date.append([new_array,class_num])
                    except:
                        pass
```


localhost:8888/notebooks/CNN.ipynb
jupyter CNN Last Checkpoint: 06/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
        y_data_np = np.asarray(y_data)

In [10]: pickle_out = open('x_data_np','wb')
         pickle.dump(x_data_np,pickle_out)
         pickle_out.close()

In [11]: pickle_out = open('y_data_np','wb')
         pickle.dump(y_data_np,pickle_out)
         pickle_out.close()

In [12]: X_Temp = open('x_data_np','rb')
         x_data_np = pickle.load(X_Temp)

         Y_Temp = open('y_data_np','rb')
         y_data_np = pickle.load(Y_Temp)

In [13]: x_data_np = x_data_np.reshape(-1, 30, 30, 1)

In [14]: #spliting the data set for training and testing
         from sklearn.model_selection import train_test_split
         X_Train, X_Test, Y_Train, Y_Test = train_test_split(x_data_np, y_data_np, test_size=0.3,random_state=101)

In [15]: #implementing CNN model
         model = Sequential()
         model.add(Conv2D(150, (3, 3), input_shape=x_data_np.shape[1:]))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Conv2D(75, (3, 3)))
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Flatten())
         model.add(Dense(64))
         model.add(Dense(1))
         model.add(Activation('sigmoid'))
         model.compile(loss='binary_crossentropy',
```


localhost:8888/notebooks/CNN.ipynb
jupyter CNN Last Checkpoint: 06/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [16]:
```
#training the data set
model.fit(x_data_np, y_data_np, batch_size=32, epochs=1, validation_split=0.3)
model.save('64x3CNN.model')
```

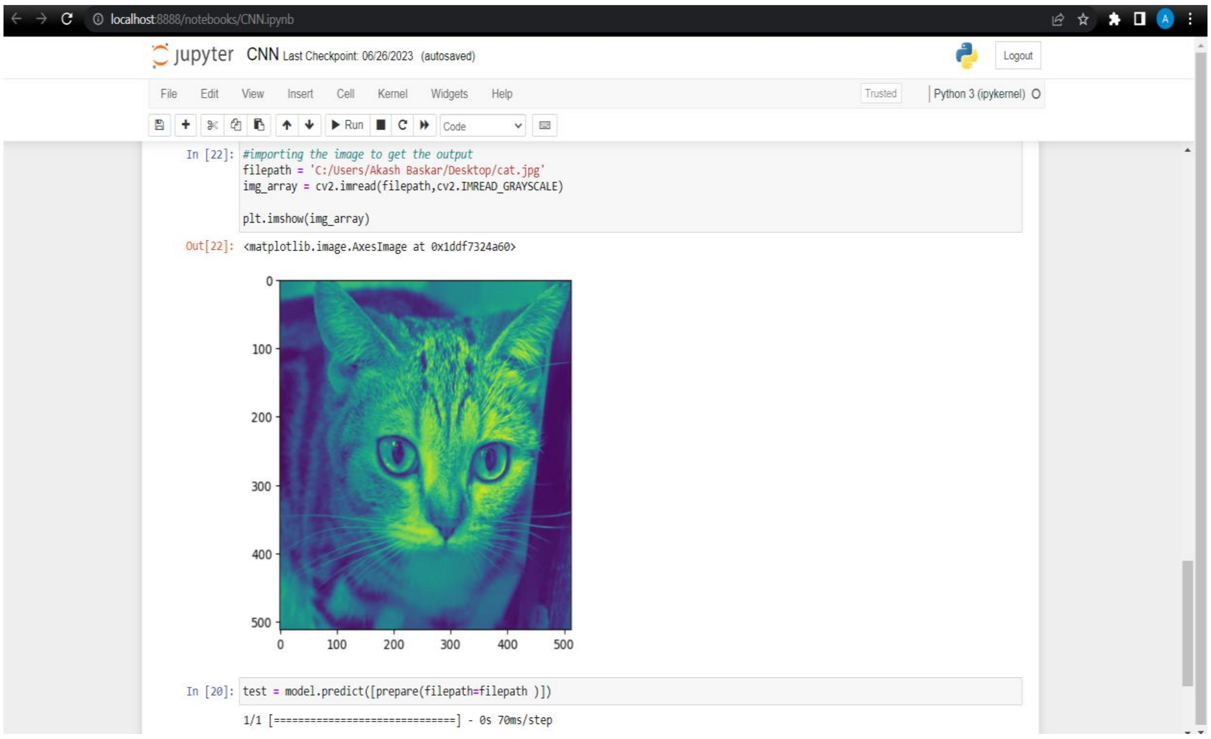
```
22/22 [==============================] - 2s 76ms/step - loss: 0.5819 - accuracy: 0.7157 - val_loss: 0.9237 - val_accuracy: 0.12
67
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _update_step_xla whil
e saving (showing 3 of 3). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: 64x3CNN.model\assets
INFO:tensorflow:Assets written to: 64x3CNN.model\assets
```

In [17]:
```
def prepare(filepath):
    training_date = []

    img_array = cv2.imread(filepath,cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array,(IMAGE_SIZE,IMAGE_SIZE))
    new_image =  new_array.reshape(-1,IMAGE_SIZE,IMAGE_SIZE,1)
    return new_image
```

In [18]:
```
model = tf.keras.models.load_model('64x3CNN.model')
```

In [22]:
```
#importing the image to get the output
filepath = 'C:/Users/Akash Baskar/Desktop/cat.jpg'
img_array = cv2.imread(filepath,cv2.IMREAD_GRAYSCALE)

plt.imshow(img_array)
```

Out[22]:
```
<matplotlib.image.AxesImage at 0x1ddf7324a60>
```

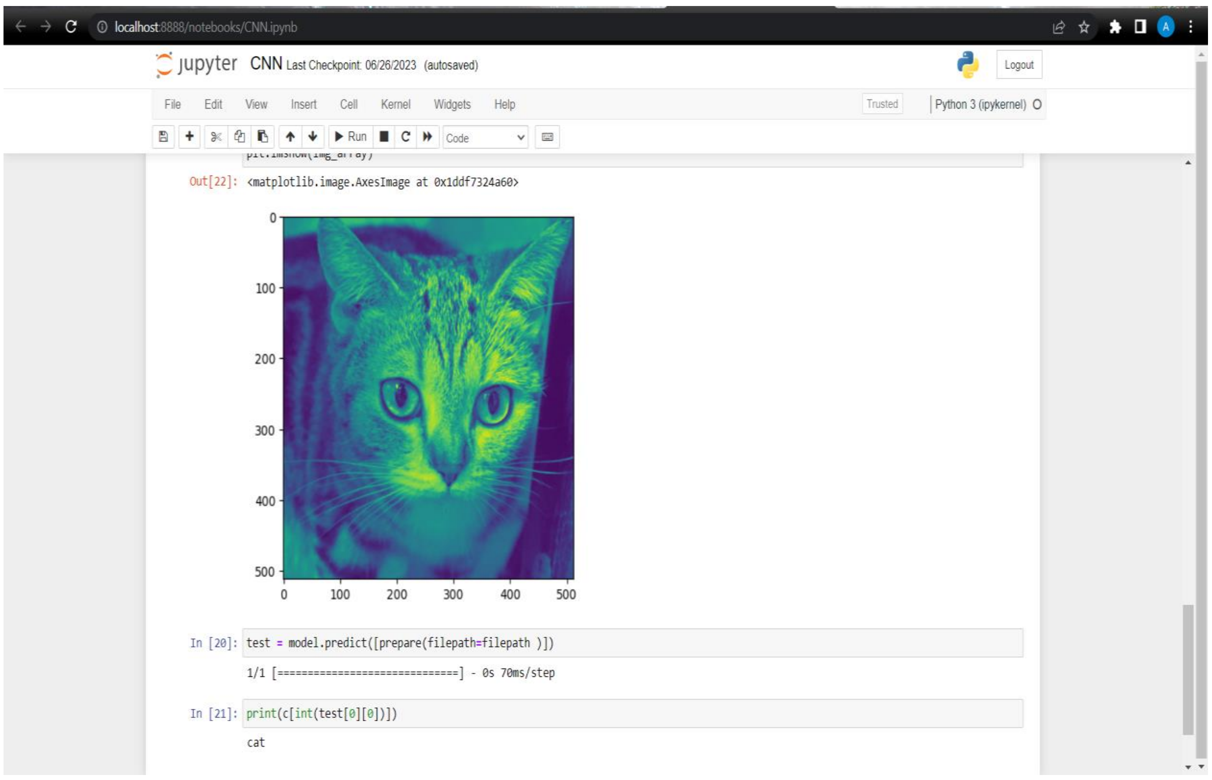

localhost:8888/notebooks/CNN.ipynb
jupyter CNN Last Checkpoint: 06/26/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
0
100
200
300
400
500
0
100
200
300
400
500

In [22]:
```
#importing the image to get the output
filepath = 'C:/Users/Akash Baskar/Desktop/cat.jpg'
img_array = cv2.imread(filepath,cv2.IMREAD_GRAYSCALE)

plt.imshow(img_array)
```

Out[22]:
```
<matplotlib.image.AxesImage at 0x1ddf7324a60>
```

In [20]:
```
test = model.predict([prepare(filepath=filepath )])
```

```
1/1 [==============================] - 0s 70ms/step
```



localhost:8888/notebooks/CNN.ipynb
jupyter CNN Last Checkpoint: 06/26/2023 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code

```
plt.imshow(img_array)
Out[22]: <matplotlib.image.AxesImage at 0x1ddf7324a60>
```

```
In [20]: test = model.predict([prepare(filepath=filepath )])

         1/1 [==============================] - 0s 70ms/step

In [21]: print(c[int(test[0][0])])

         cat
```