# CAT-DOG CLASSIFICATION

## Problem :

## Train a model that can classify between a cat and dog

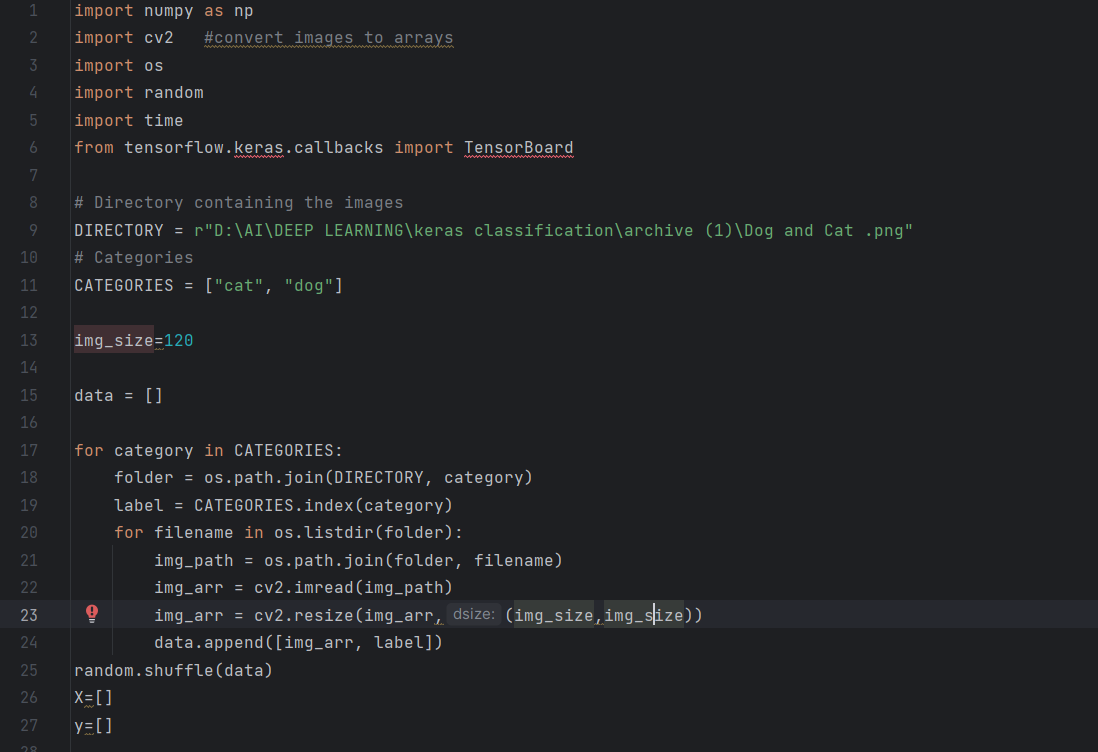
## We have to build a model for classifying whether the given image is a cat or a dog and determine the accuracy.

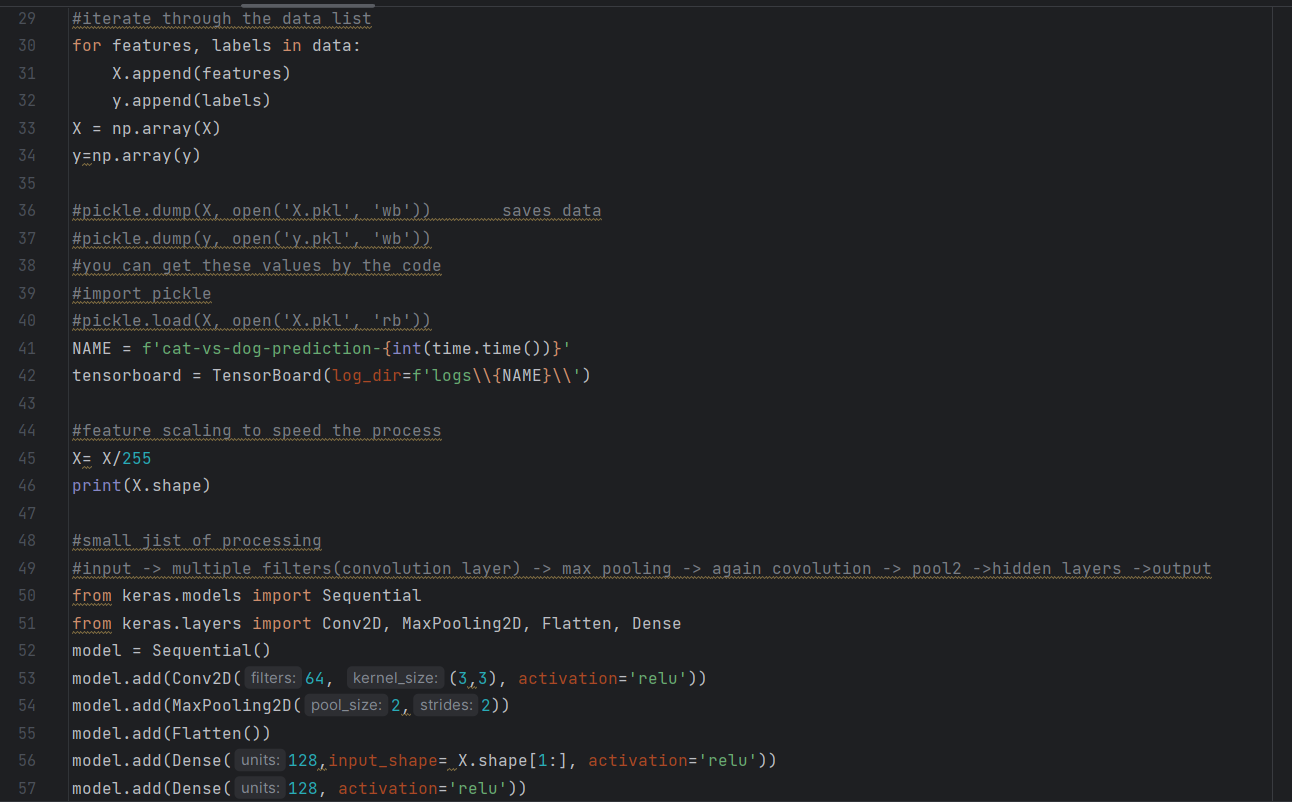
Solution:

Data loading , two different datasets are stored in a folder, here we have taken image of 100 dogs and cats , we have stored it in two folders

CODE :

[https://github.com/Abinayasen/DEEP-LEARNING/blob/main/cat\_dog\_classi.py](https://github.com/Abinayasen/DEEP-LEARNING/blob/main/cat_dog_classi.py%20)

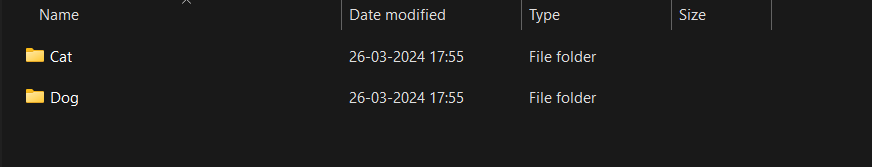






STEP 1 :

Involves in data loading, here we use a dataset of 100 images of cat and dog stored in two different folders



LIBRARY USED :

1. os -operating system
2. CV2 - reads image and converts it to arrays
3. Tensorflow

First we import the directory containing the images, the path to the directory is provided, two categories is created as strings as CATERGORIES,

We also create a list to store the arrays of images, as data = []

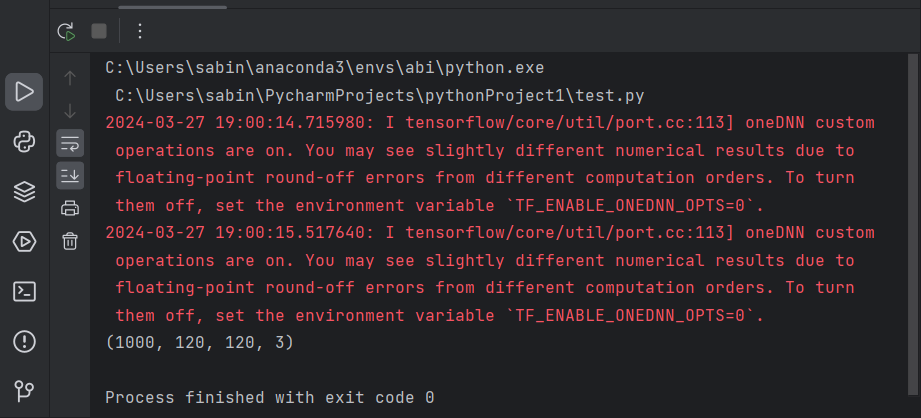
Next loop is created where each folder is joined using os to the strings saved as CATEGORIES earlier. We also assign a label to each label, now each image is opened and saved as array and is saved in the list “data”.

We shuffle the data , and two more lists are created, where “X” list consists of dependent variables and “y” consists of independent variables.

DATA PRE-PROCESSING :

We create to labels features and labels, where the “X” list is appended with features since it is a dependent variable, and “y” is appended with labels and it is the independent variable,

For easier access we convert the list to arrays. And divide the value of X by 255, to fasten the process(), we print the shape of X and we get



The output shows that 1000 X consists of 1000 values with each image of size 120x120. Here the 3 represents the RGB (Red, Green, Blue).

:

THEORY:

The input is given as images, for processing the input image is converted into multiple filters as convolution layers, these layers are max pooled, and again converted to convolution layers and another layer of max pooling. The last layer is sent as hidden layers and thus the output.

CODE:

From keras we import sequential and conv2D, MaxPooling2D,Flatten,Dense.

Convolutional Layers: You have a single convolutional layer with 64 filters/kernels, each of size (3, 3), and ReLU activation.

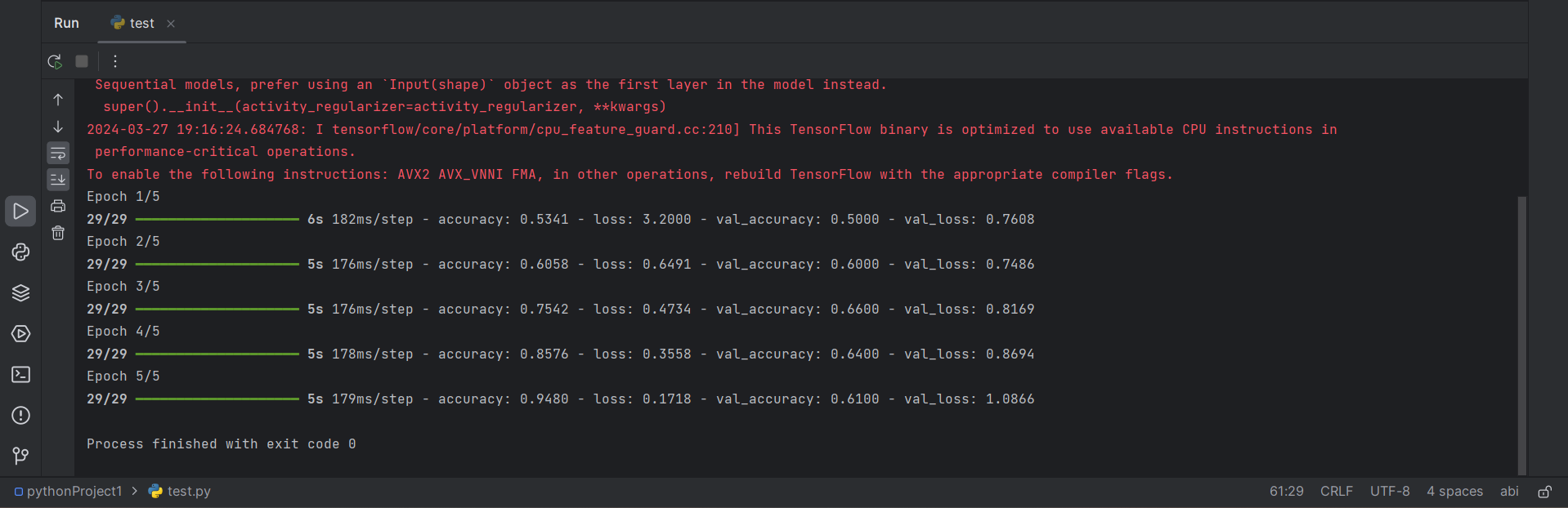
Max Pooling Layer: After the convolutional layer, there's a max-pooling layer with pool size (2, 2). This layer helps in reducing the spatial dimensions.

Flatten Layer: This layer is used to flatten the output of the previous layer into a 1D array, which can be fed into the Dense layers.

Dense Layers: You have two hidden Dense layers, each with 128 units and ReLU activation.

Output Layer: Finally, there's an output Dense layer with 2 units (assuming it's a binary classification problem) and softmax activation, which is suitable for multi-class classification tasks.

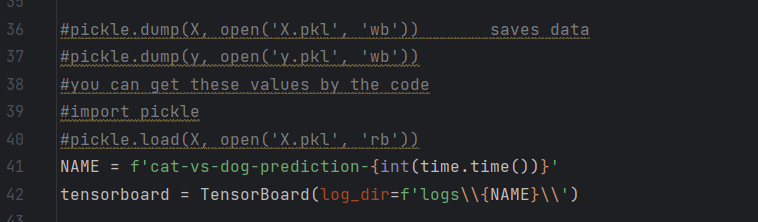
OUTPUT:



CALLBACKS:

TensorBoard callbacks in Keras are used to visualize and monitor the training process of your neural network model. TensorBoard is a visualization toolkit that comes with TensorFlow, which can display various aspects of your model's performance and structure in a web interface.

“from tensorflow.keras.callbacks import TensorBoard”

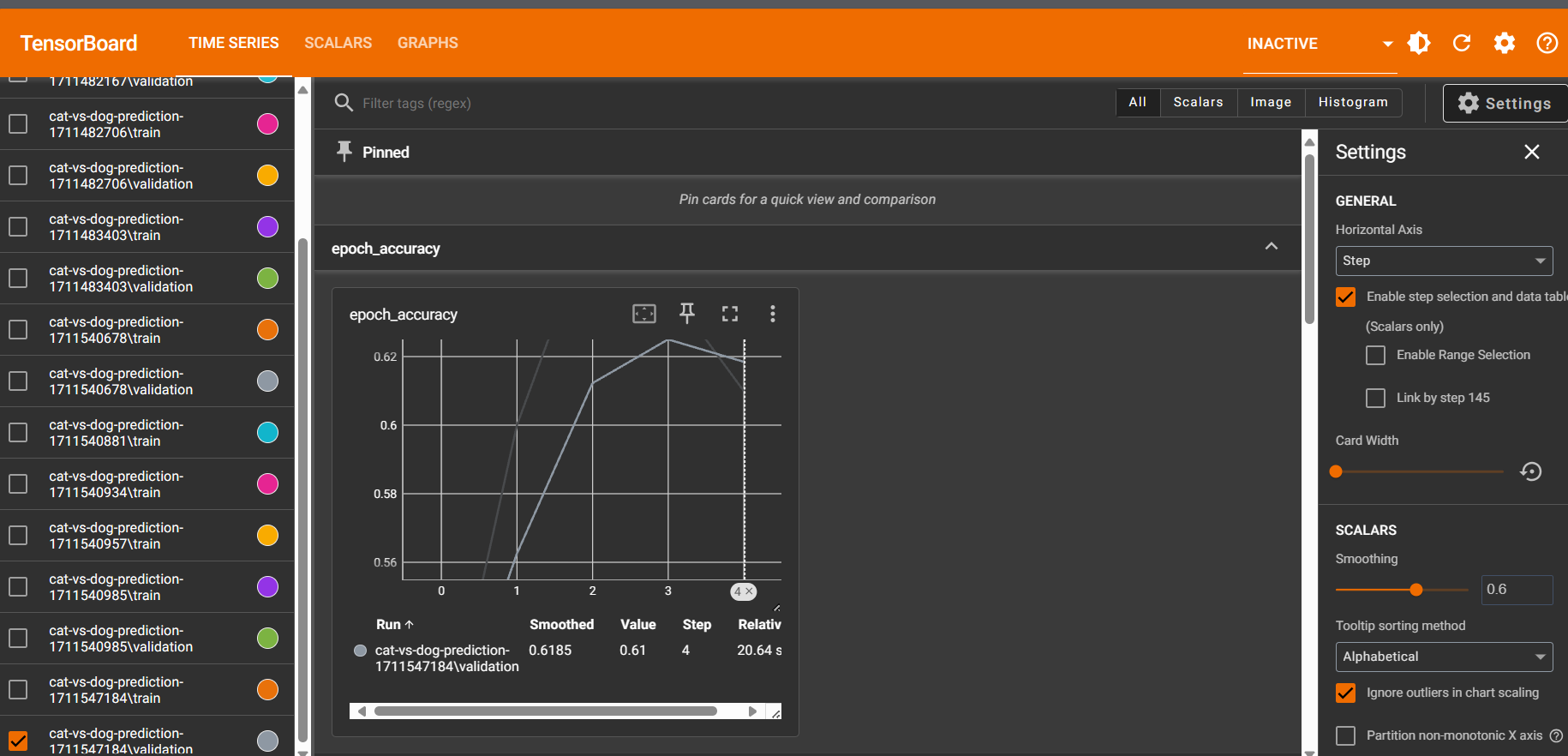


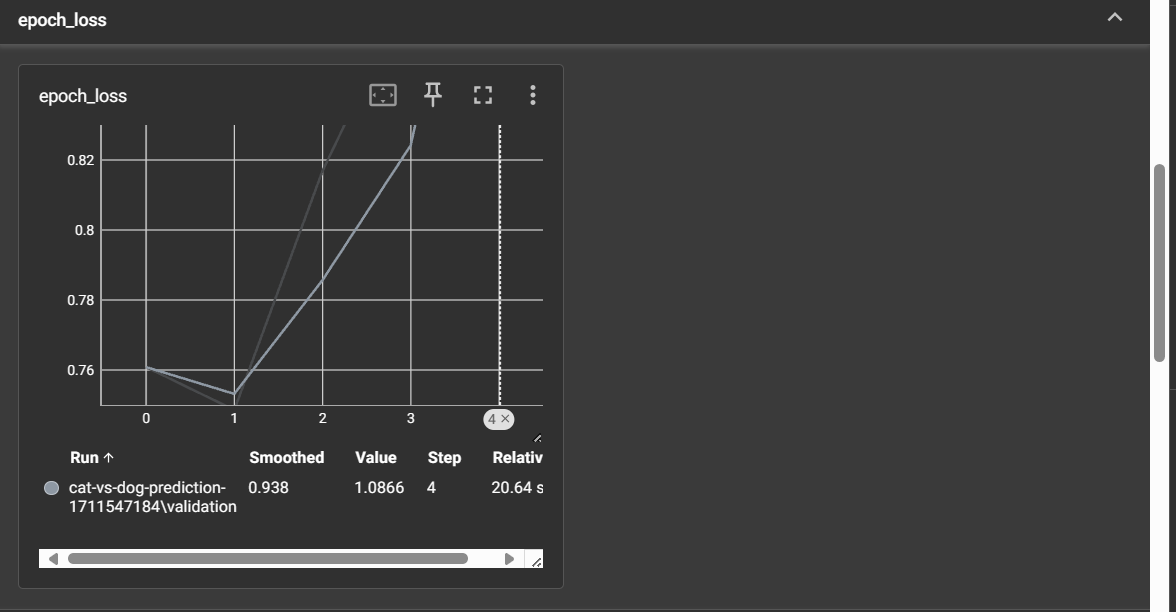
We create new files for each run using the code.

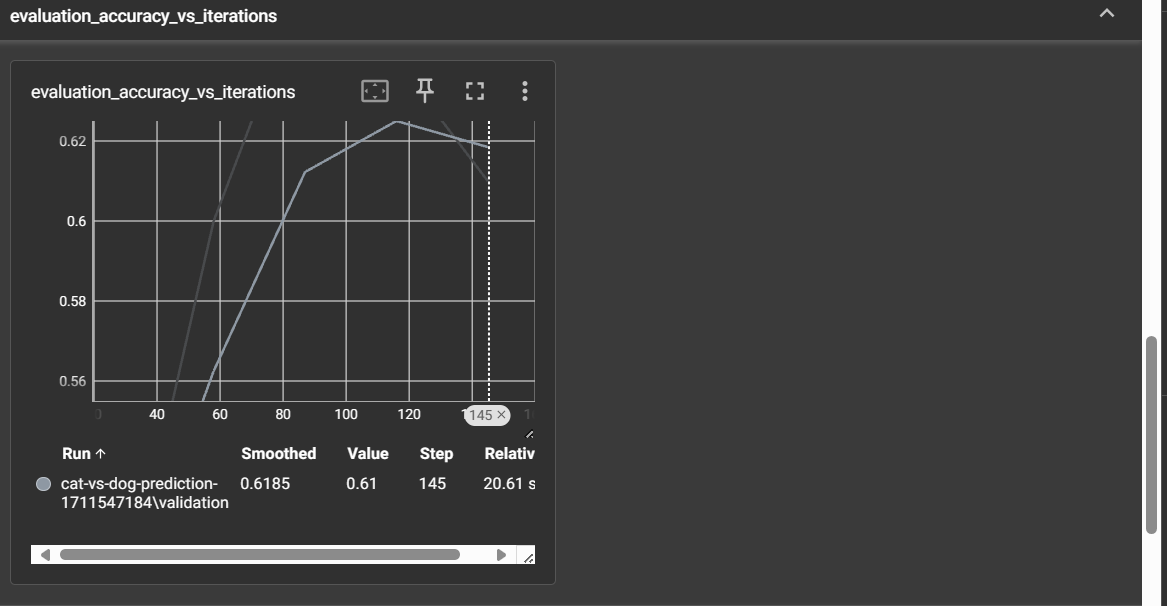
To view the output, enter the code in terminal

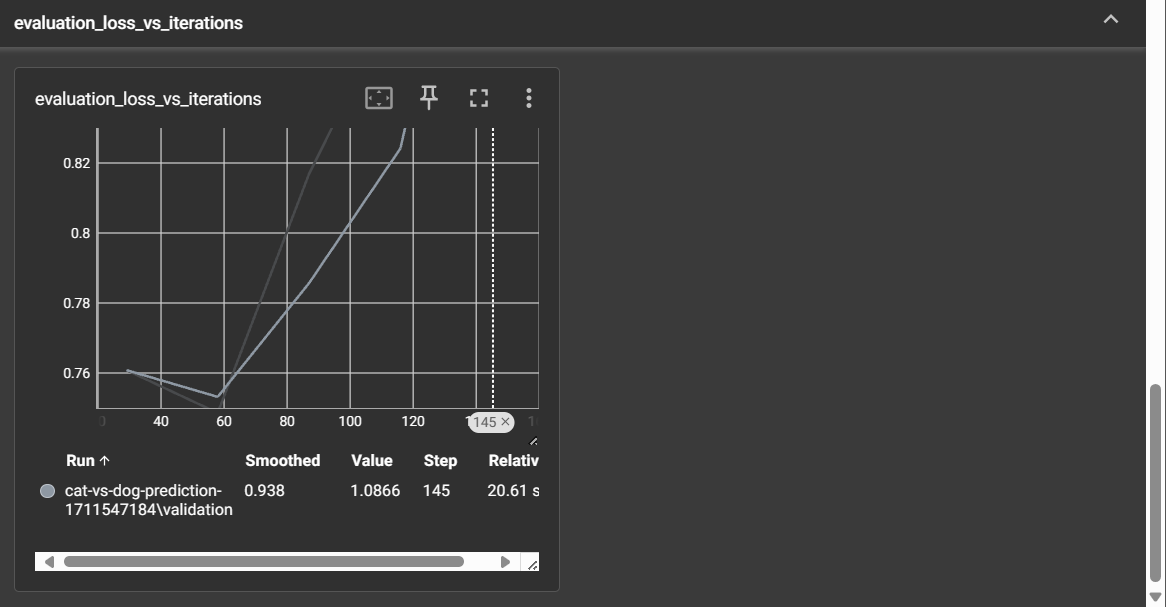
*tensorboard --logdir=logs/*

paste the code in a browser, we get the following output for our model.







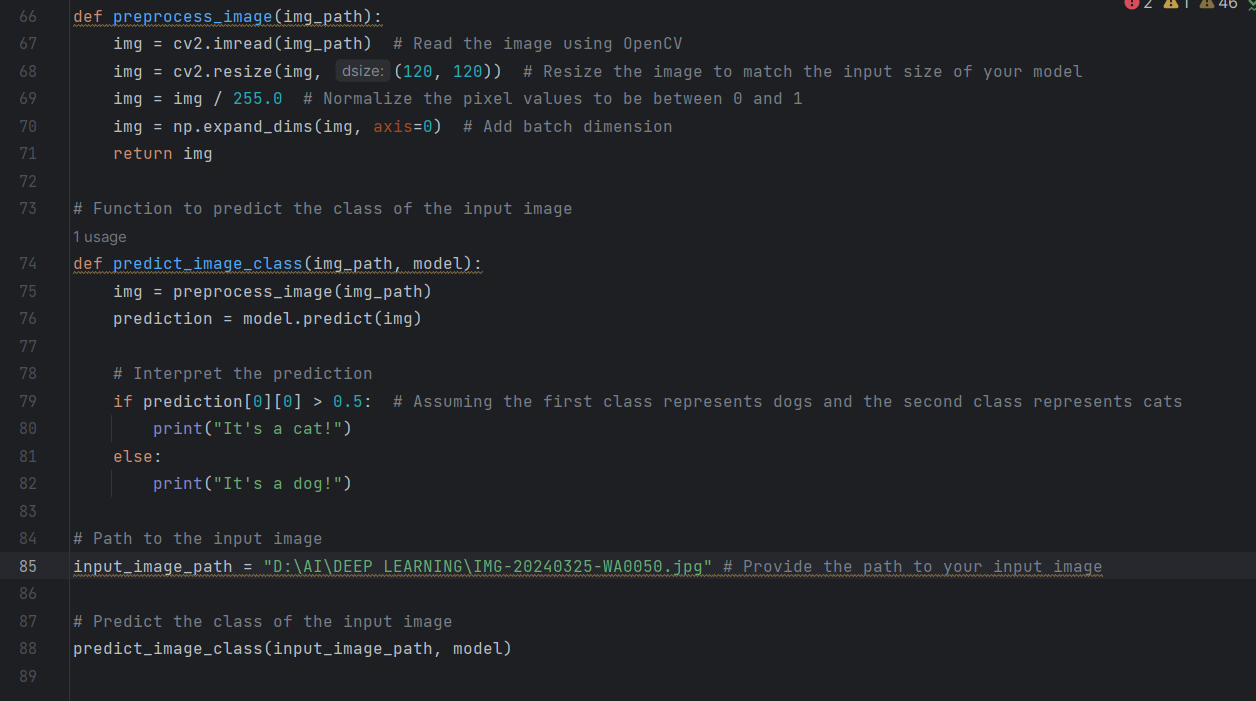


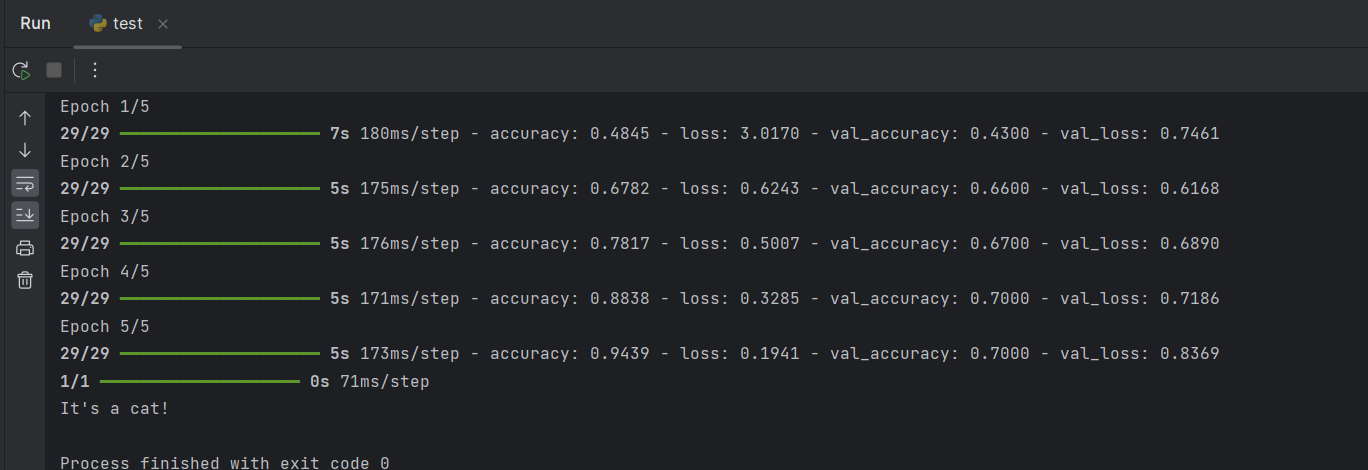
TEST:

We test our model by adding a sample image of a cat and check if the model can figure out if it is a cat or dog,

We add this image of cat



CODE:



Thus the model can recognize it as CAT.