

# Dogs VS Cats using CNN

## ABSTRACT :

Convolutional neural network (CNN), a class of artificial neural networks that has become dominant in various computer vision tasks, is attracting interest across a variety of domains, including radiology. CNN is designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. This review article offers a perspective on the basic concepts of CNN and its application to various radiological tasks, and discusses its challenges and future directions in the field of radiology. Two challenges in applying CNN to radiological tasks, small dataset and overfitting, will also be covered in this program, as well as techniques to minimize them. Being familiar with the concepts and advantages, as well as limitations, of CNN is essential to leverage its potential in diagnostic radiology, with the goal of augmenting the performance of radiologists and improving patient care.

## OBJECTIVE :

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data. 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.

CNN is designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers.

## INTRODUCTION :

The Dogs vs. Cats dataset is a standard computer vision dataset that involves classifying photos as either containing a dog or cat.

Although the problem sounds simple, it was only effectively addressed in the last few years using deep learning convolutional neural networks. While the dataset is effectively solved, it can be used as the basis for learning and practicing how to develop, evaluate, and use convolutional deep learning neural networks for image classification from scratch.

The above are the training and testing datasets of dogs and cats are :



Important libraries for this project are as follows :

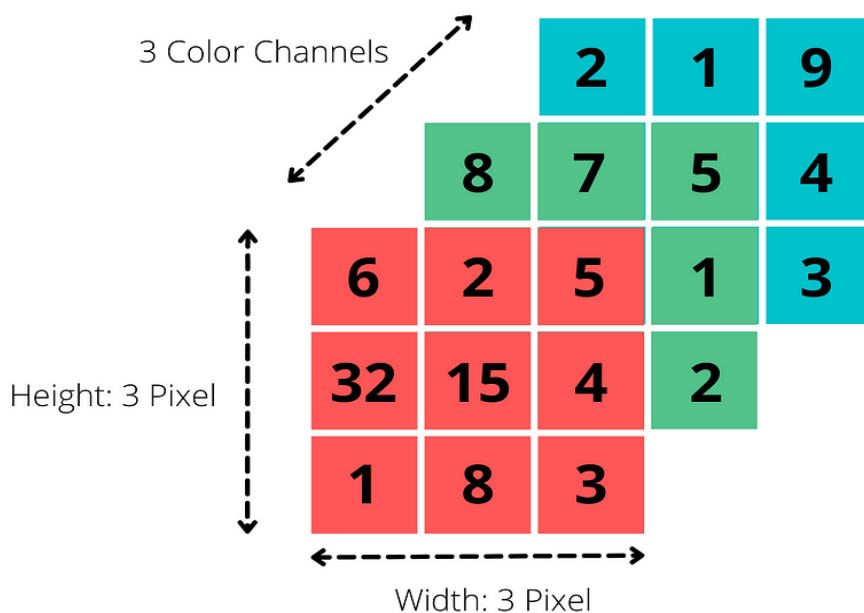
1. **NumPy** - For working with arrays, linear algebra.
2. **Pandas** – For reading/writing data
3. **Matplotlib** – to display images
4. **TensorFlow Keras models** – Need a model to predict right !!
5. **TensorFlow Keras layers** – Every NN needs layers and CNN needs well a couple of layers

## METHODOLOGY:

The Convolutional Neural Network (CNN or ConvNet) is a subtype of Neural Networks that is mainly used for applications in image and speech recognition. Its built-in convolutional layer reduces the high dimensionality of images without losing its information. That is why CNNs are especially suited for this use case.

### Image Processing Problems

If we want to use a fully-connected neural network for image processing, we quickly discover that it does not scale very well. For the computer, an image in RGB notation is the summary of three different matrices. For each pixel of the image, it describes what color that pixel displays. We do this by defining the red component in the first matrix, the green component in the second, and then the blue component in the last. So for an image with the size 3 on 3 pixels, we get three different 3x3 matrices.



To process an image, we enter each pixel as input into the network. So for an image of size 200x200x3 (i.e. 200 pixels on 200 pixels with 3 color channels, e.g. red, green and blue) we have to provide  $200 * 200 * 3 = 120,000$  input neurons. Then each matrix has a size of 200 by 200 pixels, so  $200 * 200$  entries in total. This matrix then finally exists three times, each for red, blue, and green. The problem then arises in the first hidden layer, because each of the neurons there would have 120,000 weights from the input layer. This means the number of parameters would increase very quickly as we increase the number of neurons in the Hidden Layer.

This challenge is exacerbated when we want to process larger images with more pixels and more color channels. Such a network with a huge number of parameters will most likely run into overfitting. This means that the model will give good predictions for the training set, but will not generalize well to new cases that it does not yet know. Additionally, due to a large number of parameters, the network would very likely stop attending to individual image details as they would be lost in sheer mass. However, if we want to classify an image, e.g. whether there is a dog in it or not, these details, such as the nose or the ears, can be the decisive factor for the correct result.

## Convolutional Neural Network

For these reasons, the Convolutional Neural Network takes a different approach, mimicking the way we perceive our environment with our eyes. When we see an image, we automatically divide it into many small sub-images and analyze them one by one. By assembling these sub-images, we process and interpret the image. How can this principle be implemented in a Convolutional Neural Network?

The work happens in the so-called **convolution layer**. To do this, we define a filter that determines how large the partial images we are looking at should be, and a step length that decides how many pixels we continue between calculations, i.e. how close the partial images are to each other.

By taking this step, we have greatly reduced the dimensionality of the image.

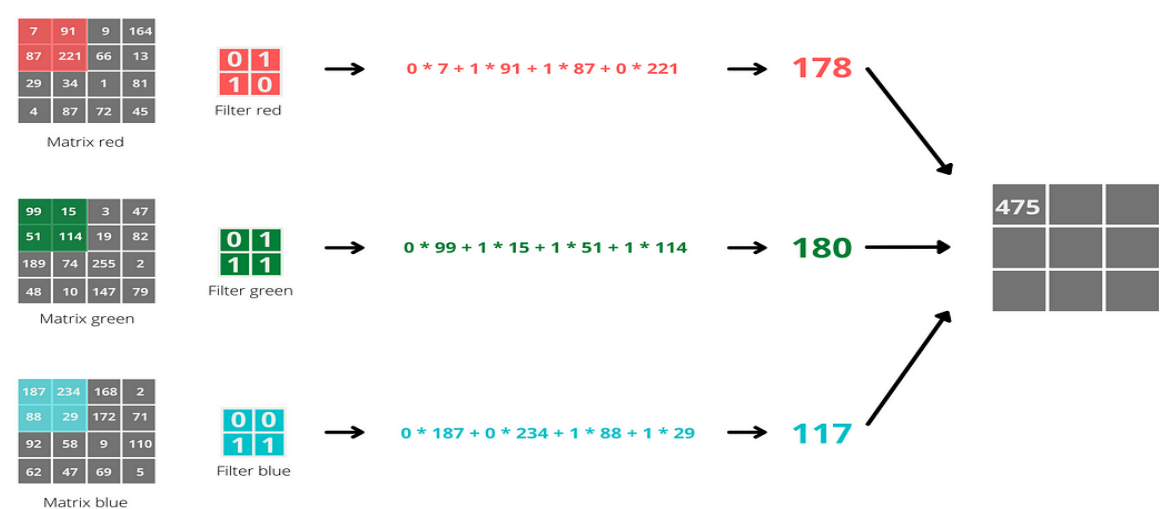
The next step is the **pooling layer**. From a purely computational point of view, the same thing happens here as in the convolution layer, with the difference that we only take either the average or maximum value from the result, depending on the application. This preserves small features in a few pixels that are crucial for the task solution.

Finally, there is a **fully-connected layer**, as we already know it from regular neural networks. Now that we have greatly reduced the dimensions of the image, we can use the tightly meshed layers. Here, the individual sub-images are linked again in order to recognize the connections and carry out the classification.

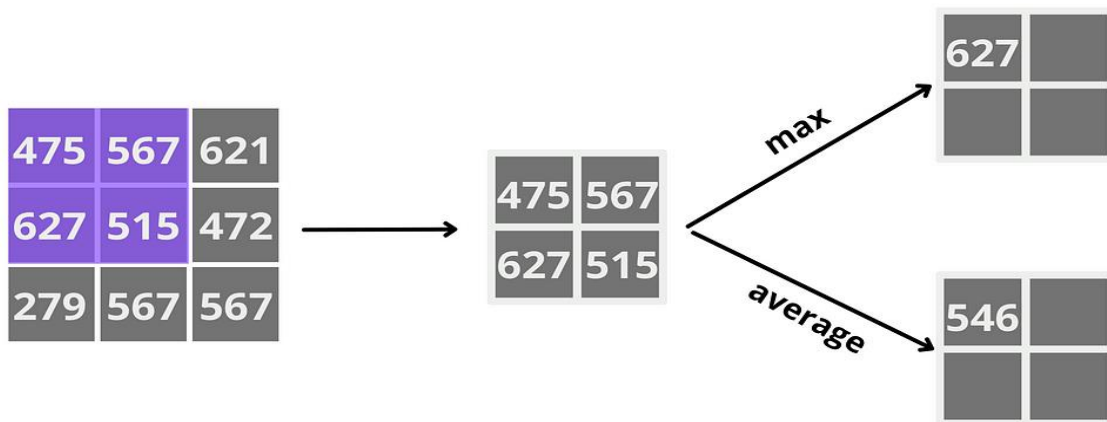
Now that we have a basic understanding of what the individual layers roughly do, we can look in detail at how an image becomes a classification. For this purpose, we try to recognize from a 4x4x3 image whether there is a dog in it.

### Detail: Convolution Layer

In the first step, we want to reduce the dimensions of the 4x4x3 image. For this purpose, we define a filter with the dimension 2x2 for each color. In addition, we want a step length of 1, i.e. after each calculation step, the filter should be moved forward by exactly one pixel. This will not reduce the dimension as much, but the details of the image will be preserved. If we migrate a 4x4 matrix with a 2x2 and advance one column or one row in each step, our Convolutional Layer will have a 3x3 matrix as output. The individual values of the matrix are calculated by taking the scalar product of the 2x2 matrices, as shown in the graphic.



### Detail: Pooling Layer



The (Max) Pooling Layer takes the 3x3 matrix of the convolution layer as input and tries to reduce the dimensionality further and additionally take the important features in the image. We want to generate a 2x2 matrix as the output of this layer, so we divide the input into all possible 2x2 partial matrices and search for the highest value in these fields. This will be the value in the field of the output matrix. If we were to use the average pooling layer instead of a max-pooling layer, we would calculate the average of the four fields instead.

The pooling layer also filters out noise from the image, i.e. elements of the image that do not contribute to the classification. For example, whether the dog is standing in front of a house or in front of a forest is not important at first.

### Detail: Fully-Connected Layer

The fully-connected layer now does exactly what we intended to do with the whole image at the beginning. We create a neuron for each entry in the smaller 2x2 matrix and connect them to all neurons in the next layer. This gives us significantly fewer dimensions and requires fewer resources in training.

This layer then finally learns which parts of the image are needed to make the classification dog or non-dog. If we have images that are much larger than our 5x5x3 example, it is of course also possible to set the convolution layer and pooling layer several times in a row before going into the fully-connected layer. This way you can reduce the dimensionality far enough to reduce the training effort.

### Dataset:

Datasets are classified as training and testing of dogs and cats separately to test and predict the data.

CODE :

Home Page - S x | Untitled8 - Jup x | Untitled19 - Ju x | Home Page - S x | Untitled20 - Ju x | dog photos - S x | how to take sc x | +

localhost:8892/notebooks/Untitled20.ipynb?kernel\_name=python3

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Untitled20 Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3 (pykernel)

In [3]:

import numpy as np  
import random  
import matplotlib.pyplot as plt  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Conv2D,Dense,Flatten,MaxPooling2D

In [4]:

X\_train = np.loadtxt('C:/Users/lenovo/Desktop/input.csv',delimiter=',')  
Y\_train = np.loadtxt('C:/Users/lenovo/Desktop/labels.csv',delimiter=',')  
X\_test = np.loadtxt('C:/Users/lenovo/Desktop/input\_test.csv',delimiter=',')  
Y\_test = np.loadtxt('C:/Users/lenovo/Desktop/labels\_test.csv',delimiter=',')  
  
X\_train = X\_train.reshape(len(X\_train),100,100,3)  
Y\_train = Y\_train.reshape(len(Y\_train),1)  
X\_test = X\_test.reshape(len(X\_test),100,100,3)  
Y\_test = Y\_test.reshape(len(Y\_test),1)

In [5]:

X\_train = X\_train/255.0  
X\_test = X\_test/255.0

Home Page - S x | Untitled8 - Jup x | Untitled19 - Ju x | Home Page - S x | Untitled20 - Ju x | dog photos - S x | how to take sc x | +

localhost:8892/notebooks/Untitled20.ipynb?kernel\_name=python3

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter Untitled20 Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3 (pykernel)

In [5]:

print("shape of X\_train : ",X\_train.shape)  
print("shape of Y\_train : ",Y\_train.shape)  
print("shape of X\_test : ",X\_test.shape)  
print("shape of Y\_train : ",Y\_test.shape)  
  
shape of X\_train : (2000, 100, 100, 3)  
shape of Y\_train : (2000, 1)  
shape of X\_test : (400, 100, 100, 3)  
shape of Y\_train : (400, 1)

In [6]:

X\_train[1,:]

Out[6]:

array([[0.51372549, 0.50196078, 0.52941176],  
 [0.62745098, 0.61568627, 0.64313725],  
 [0.77647059, 0.75294118, 0.8 ],  
 ...,  
 [0.98039216, 0.97647059, 0.96862745],  
 [1., 1., 0.99215686],  
 [0.98039216, 0.97647059, 0.96078431]],  
  
 [[0.54901961, 0.5372549 , 0.56470588],  
 [0.49803922, 0.48627451, 0.51372549],  
 [0.47058824, 0.44705882, 0.48627451],



A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8892/notebooks/Untitled20.ipynb?kernel\_name=python3'. The notebook title is 'Untitled20' with a status 'Last Checkpoint: an hour ago (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and code execution. The main area displays a list of 100 random numbers, grouped into three visible sections. The first section contains 10 numbers, the second contains 10 numbers, and the third contains 10 numbers. The numbers are displayed in a monospace font. The bottom status bar shows the system clock as 09:23 on 17-05-2023.

localhost:8892/notebooks/Untitled20.ipynb?kernel\_name=python3

**UPDATE** Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. [Don't show anymore](#)

**Jupyter** Untitled20 Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel)

```
In [5]: idx = random.randint(0, len(X_train))
plt.imshow(X_train[idx,:])
plt.show()
```

Home Page - S xUntitled8 - Jup xUntitled19 - Jup xHome Page - S xUntitled20 - Jup xdog photos - S xhow to take sc x

localhost:8892/notebooks/Untitled20.ipynb?kernel\_name=python3

Sign in

UPDATE

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore

jupyter

Untitled20

Last Checkpoint: an hour ago (autosaved)

Python 3 (ipykernel)

Logout


FileEditViewInsertCellKernelHelp

Trusted

Python 3 (ipykernel)

In [7]:

```
idx = random.randint(0,len(X_train))
plt.imshow(X_train[idx,:])
plt.show()
```



Type here to search

31°C Mostly cloudy

09:23 17-05-2023

Home Page - S xUntitled8 - Jup xUntitled19 - Jup xHome Page - S xUntitled20 - Jup xdog photos - S xhow to take sc x

localhost:8892/notebooks/Untitled20.ipynb?kernel\_name=python3

Sign in

UPDATE

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore

jupyter

Untitled20

Last Checkpoint: an hour ago (autosaved)

Python 3 (ipykernel)

Logout

FileEditViewInsertCellKernelHelp

Trusted

Python 3 (ipykernel)

In [7]:

```
model = Sequential([
    Conv2D(32, (3,3), activation = 'relu', input_shape = (100,100,3)),
    MaxPooling2D(2,2),

    Conv2D(32, (3,3), activation = 'relu'),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(64, activation = 'relu'),
    Dense(1, activation = 'sigmoid')
])

In [8]: model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

In [9]: model.fit(X_train,Y_train,epochs=5,batch_size = 64)
```

Epoch 1/5

Type here to search

31°C Mostly cloudy

09:23 17-05-2023

Home Page - S xUntitled8 - Jup xUntitled19 - Jup xHome Page - S xUntitled20 - Jup xdog photos - S xhow to take sc x

localhost:8892/notebooks/Untitled20.ipynb?kernel\_name=python3

Sign in

UPDATE

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore

jupyter

Untitled20

Last Checkpoint: an hour ago (autosaved)

Python 3 (ipykernel)

Logout

FileEditViewInsertCellKernelHelp

Trusted

Python 3 (ipykernel)

In [8]:

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

In [9]:

```
model.fit(X_train,Y_train,epochs=5,batch_size = 64)
```

Epoch 1/5

32/32 [=====] - 4s 114ms/step - loss: 0.7884 - accuracy: 0.5335

Epoch 2/5

32/32 [=====] - 4s 139ms/step - loss: 0.6761 - accuracy: 0.5745

Epoch 3/5

32/32 [=====] - 5s 142ms/step - loss: 0.6371 - accuracy: 0.6400

Epoch 4/5

32/32 [=====] - 4s 113ms/step - loss: 0.5800 - accuracy: 0.6930

Epoch 5/5

32/32 [=====] - 4s 114ms/step - loss: 0.5007 - accuracy: 0.7590

Out[9]:

```
<keras.callbacks.History at 0x1ecb0361d0>
```

In [10]:

```
model.evaluate(X_test,Y_test)
```

13/13 [=====] - 1s 25ms/step - loss: 0.6230 - accuracy: 0.6750

Out[10]:

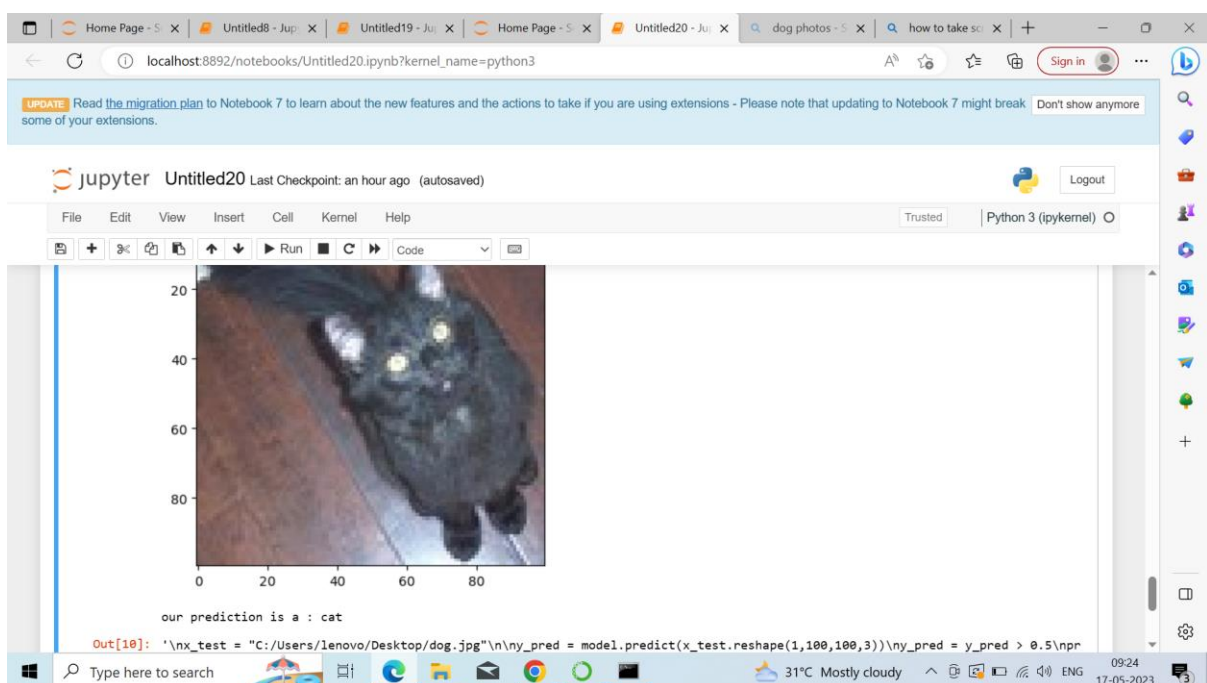
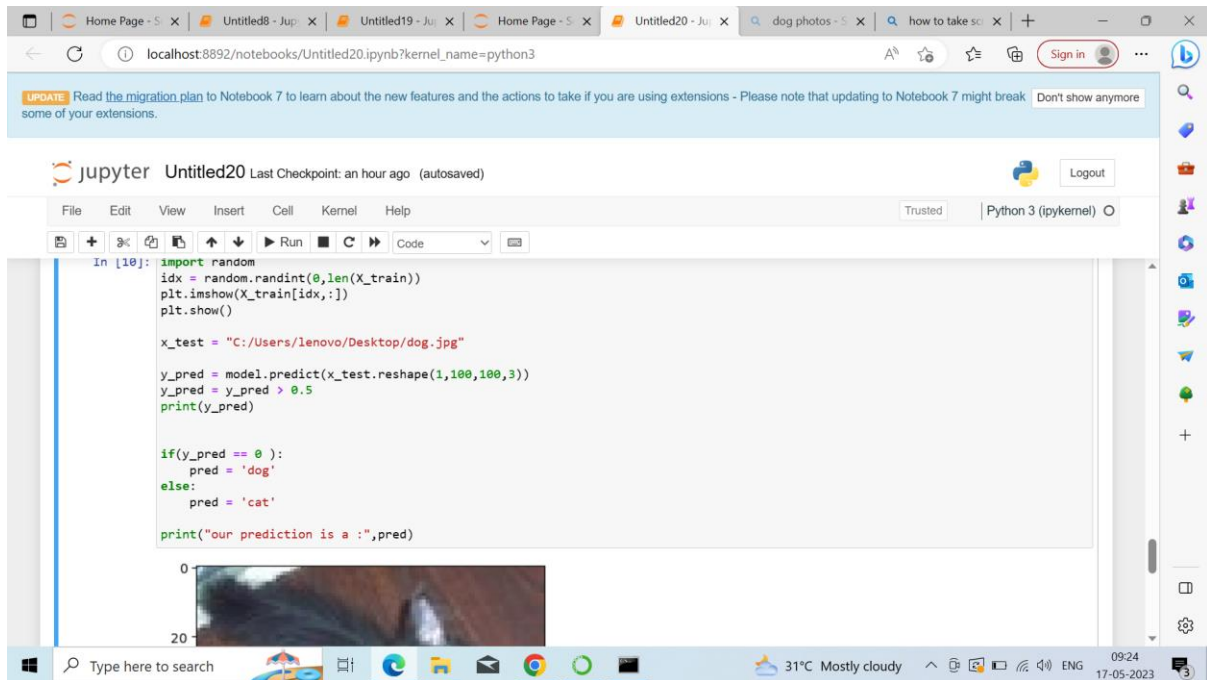
```
[0.6229591369628906, 0.675000011920929]
```

Type here to search

31°C Mostly cloudy

09:24 17-05-2023





## CONCLUSION :

This project is for the programmers which introduces you to ,how to build an image classifier. I hope at the end you will have a basic understanding of Convolutional Neural networks and can classify images of cats and dogs. As shown in the example, This project deals with the random image, and it classifies the image whether it is a dog/cat.