Image Recognition

Instructor: Vision Wang

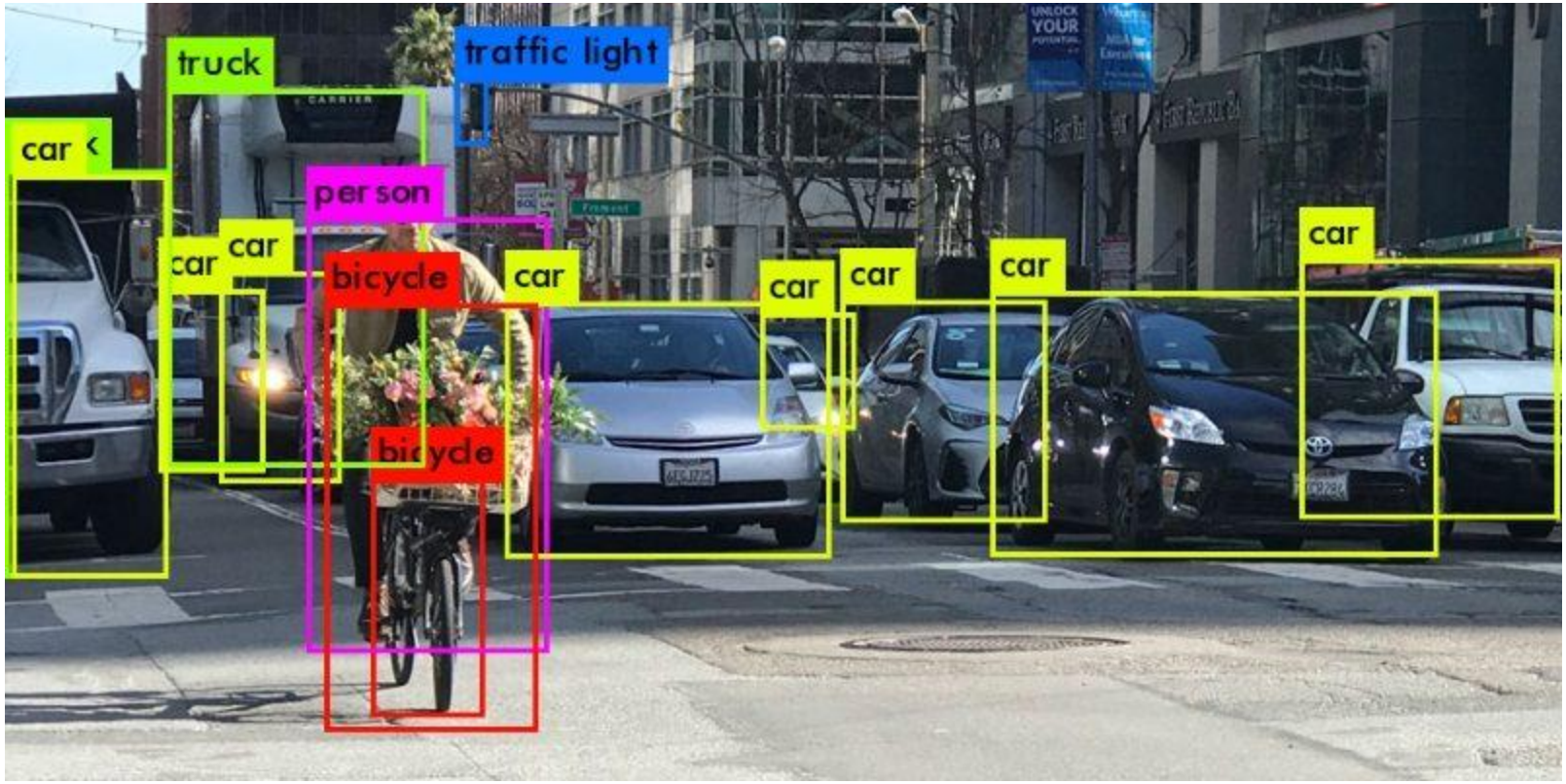
Email: xinwang35314@gmail.com

Image Recognition

- Definition and Applications
- Model
- Practice using Python

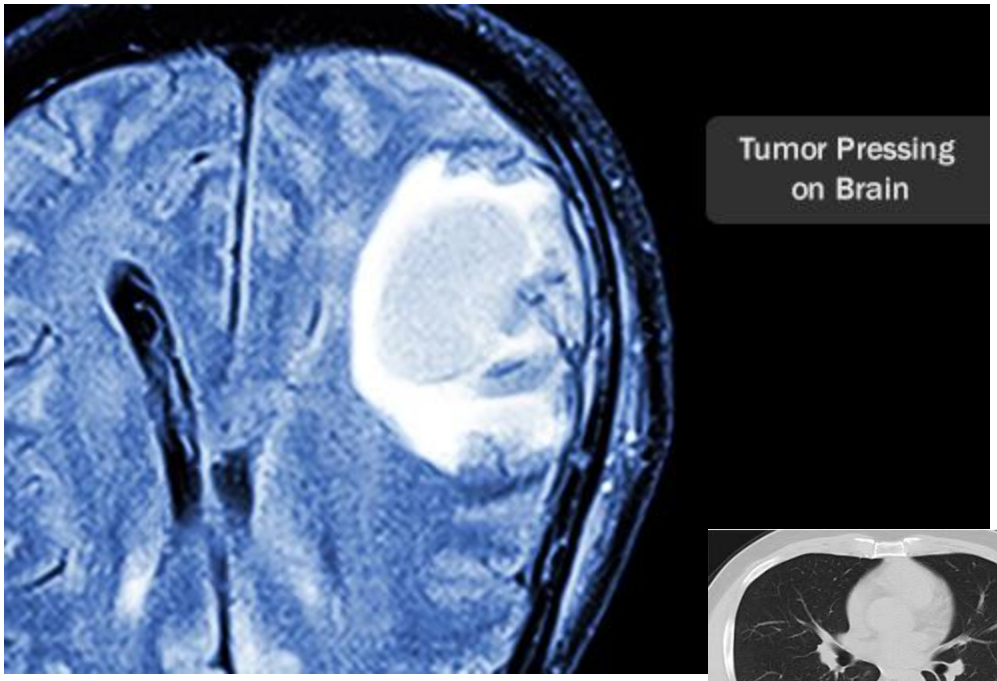
Definition

• **Image recognition** is a term for computer technologies that can recognize certain people, animals, objects or other targeted subjects through the use of algorithms and machine learning concepts.

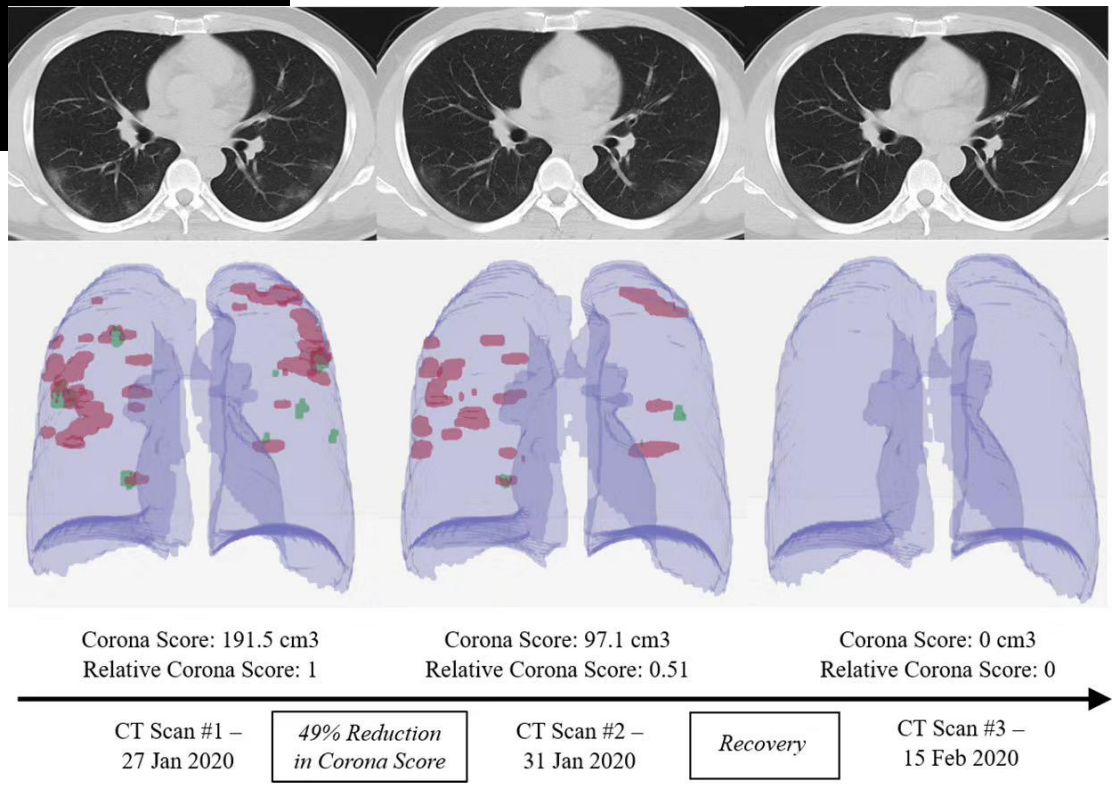


Applications

Self driving car, known as a robotic car, is a vehicle that is able to sense its environment and moving safely with little or no human input.



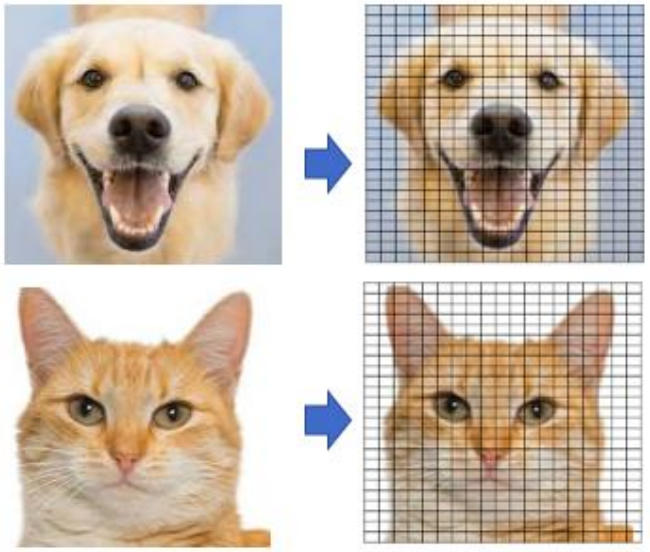
Medical Diagnostic imaging is a method of looking inside the body to help determine the cause of an injury or an illness, and to confirm a diagnosis.



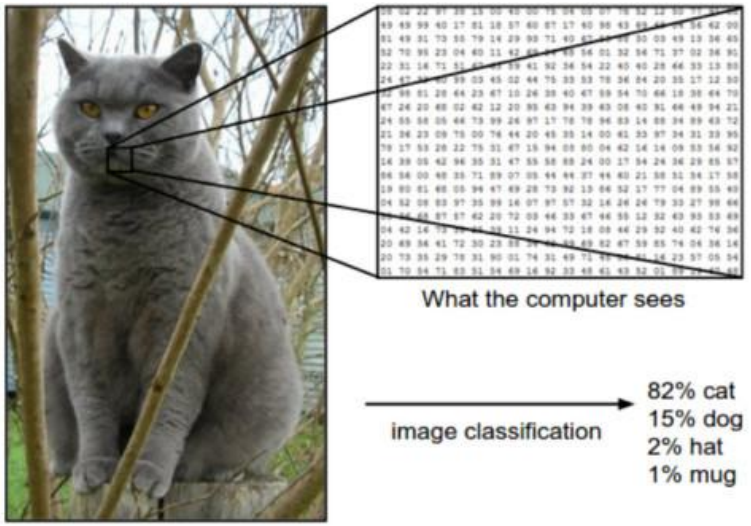
Model

- **Convolutional Neural Networks** (CNNs) is the most popular neural network model being used for image classification problem.
- In technical terms, convolutional neural networks make the image processing computationally manageable through filtering the connections by proximity.

Modeling Step 1: Extract pixel features from an image

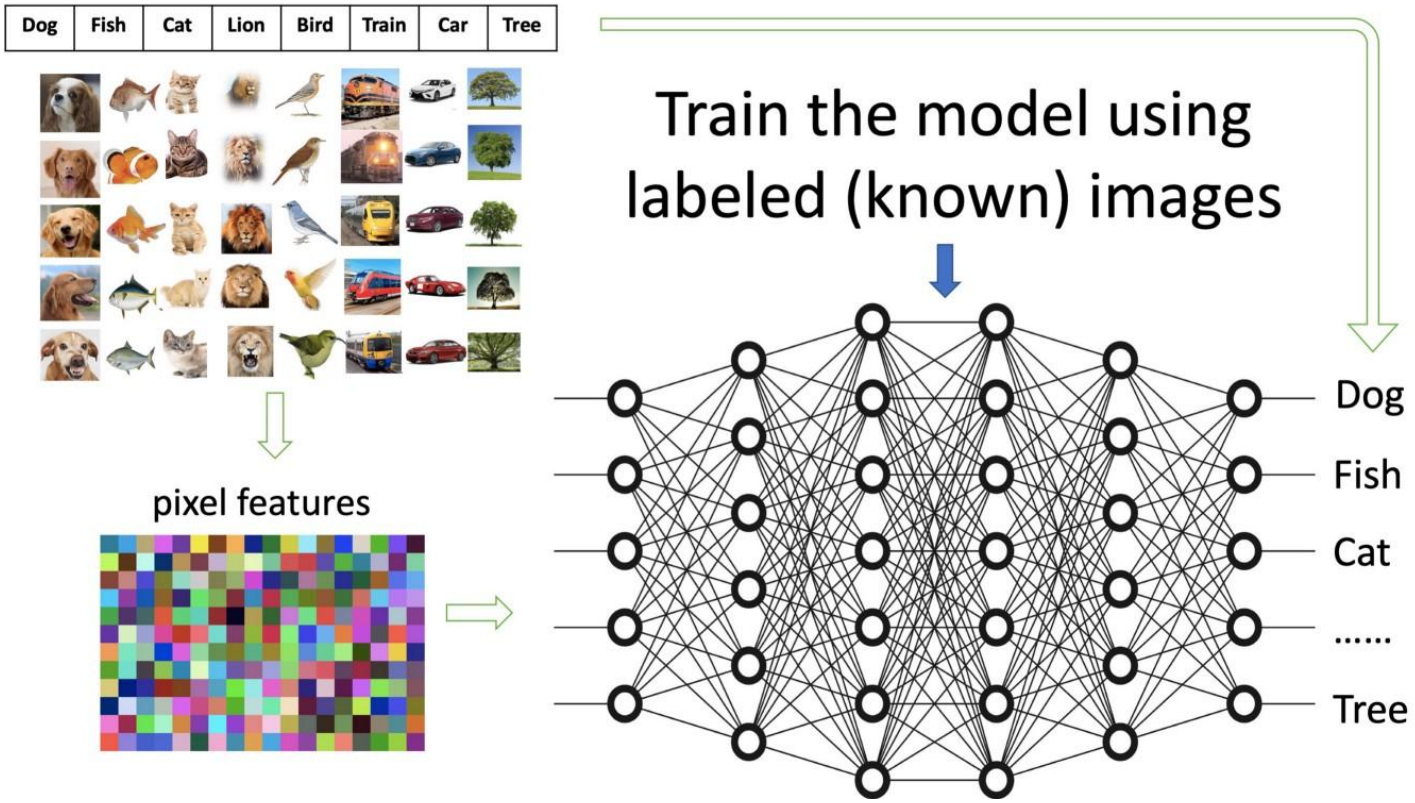


An image is actually made of “pixels”. Each pixel is represented by a number or a set of numbers — and the range of these numbers is called the color depth (or bit depth).



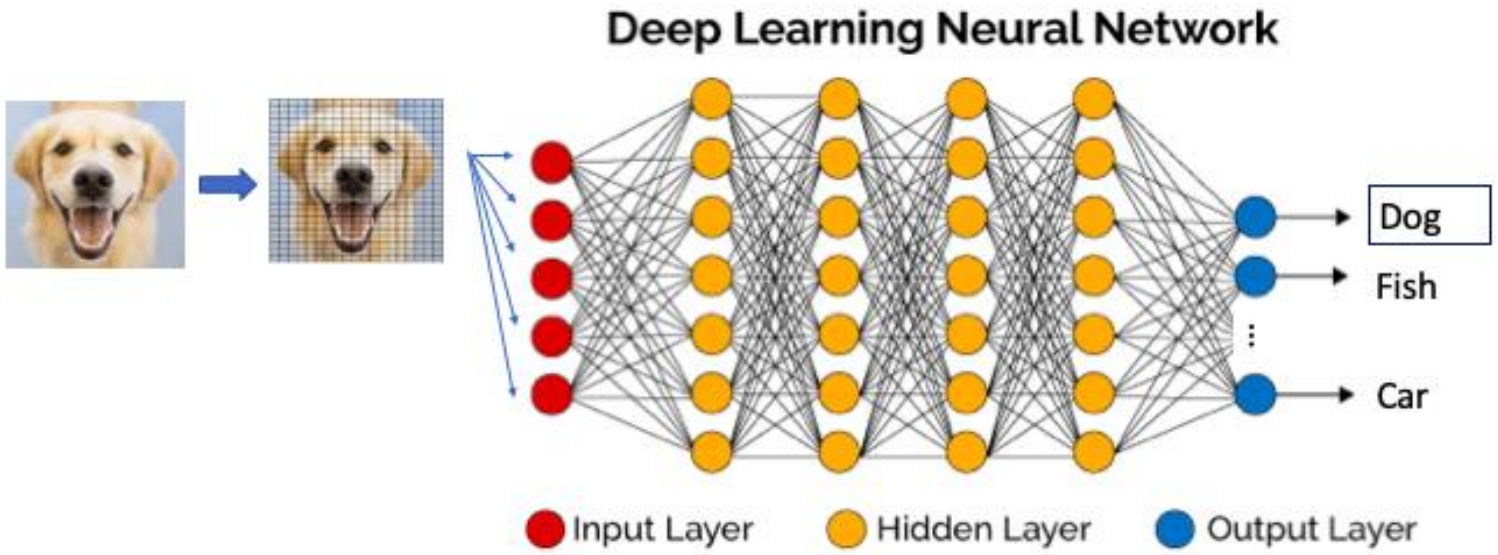
Model

Modeling Step 2: Train the model to be able to categorize images



Model

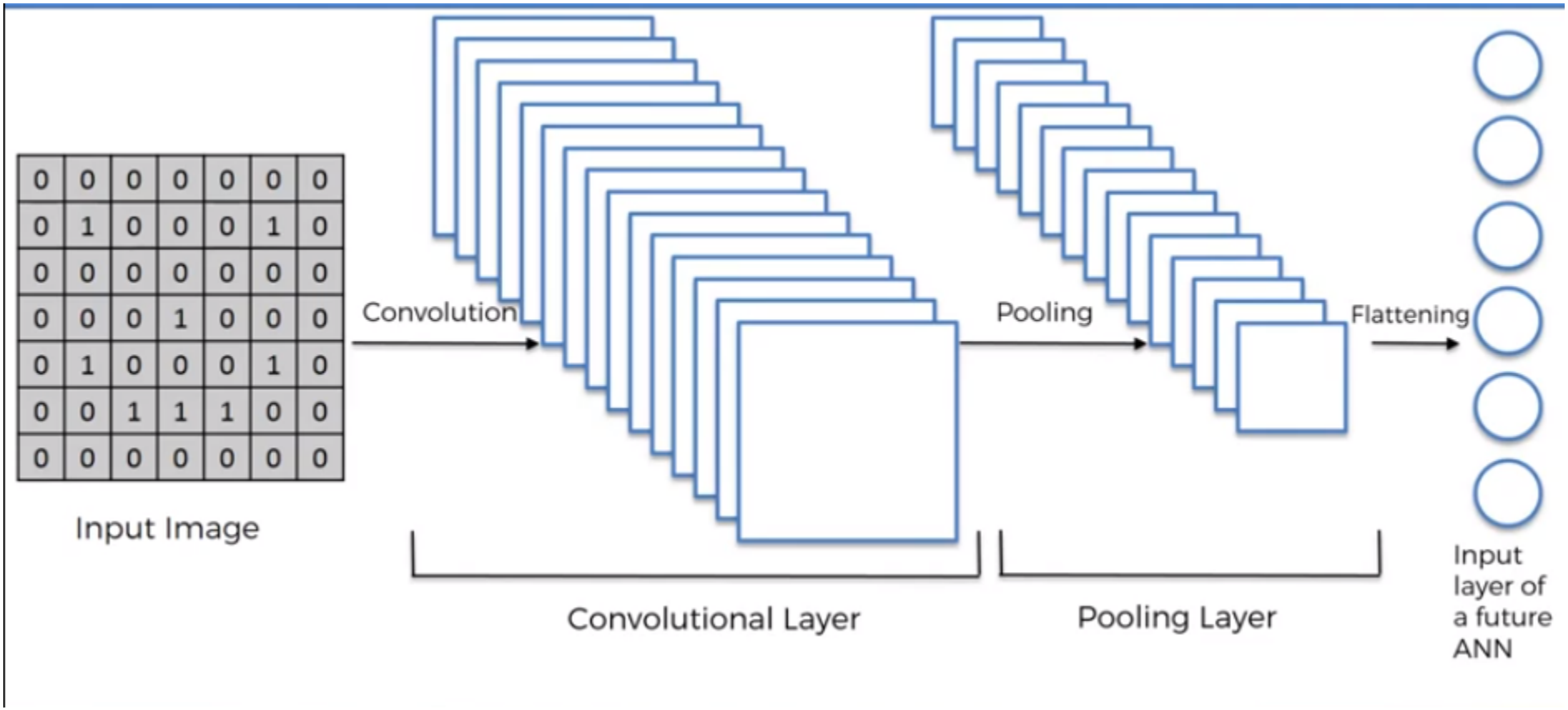
Modeling Step 3: Recognize a new image from which category



CNN Model

Building the **CNN** – There are three crucial parts in whole CNN stucture.

1. **Convolutional** – Extract features from the input image
2. **Polling** – Reduce the dimensionality of each feature map but retain the most import information.
3. **Flattening** – Convert the matrix into a linear array to input it into the nods of the neural network.



Practice using Python

- The **Keras** library in Python makes it pretty simple to build a CNN.
- Keras is an open-source neural-network library written in Python. It is designed to enable fast experimentation with deep neural networks

Import all the modules

```
>>>from keras.preprocessing.image import load_img
>>>from keras.preprocessing.image import img_to_array
>>>from keras.applications.vgg16 import preprocess_input
>>>from keras.applications.vgg16 import decode_predictions
>>>from keras.applications.vgg16 import VGG16
```

Function load_img() in keras.preprocessing.image loads and resizes the image the to the required size of 224*224 pixels (the input for the 1st convolutional layer is of fixed size 224*224).

```
>>>image = load_img(file, target_size=(224, 224))
```

Then we are going to convert the pixels to a Numpy array, using function img_to_array() in keras

```
>>>image = img_to_array(image)
```

The network expects one or more images as input; that means the input array will need to be 4-dimensional: samples, rows, columns, and channels. As we only have 1 sample here, the first number should be 1. Therefore, we shall have a (1, 224, 224, 3) shape array.

```
>>>image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
```

The image pixels need to be prepared by subtracting the mean RGB value from each pixel using preprocess_input() from keras.

```
>>>image = preprocess_input(image)
```

Predict the probability of the target image belonging to each of the targets across the 1000 known object types (as pre-trained). The result is a flatten matrix – an array.

```
>>>y = VGG16().predict(image)
```

The model has made its prediction. The next step is to interpret the result in a more comprehensive way. The Keras' function decode_predictions() serves the purpose, producing a list with class and its corresponding probability.

```
>>>label = decode_predictions(y)
>>>label = label[0][0]
```

Display the highest score in a more fashionable way,

```
>>>print('%s (%.2f%%)' % (label[1], label[2]*100))
```