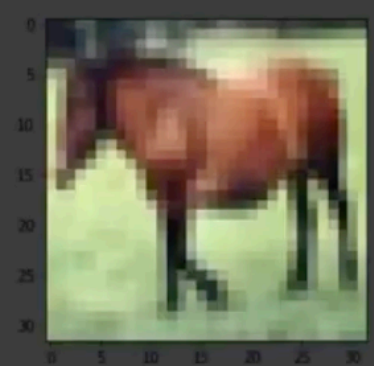
- ***Creating a convolutional neural network***
  - -> this is using keras
  - -> also a dataset of images -> which contains matrices
  - -> looking up what you need in the documentation
  - -> there are different objects in the dataset and 60, 000 images
  - ***Loading in the data***

```
TensorFlow is already loaded. Please restart the runtime to change versions.

#  LOAD AND SPLIT DATASET
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
```

    - -> then normalising the data into training and test images
      - -> dividing them by 255 to make sure that they are normalised and the algorithms run properly
      - -> then the names of the classes
    - -> then he is changing the indices of different values in the cell -> which returns different images in the dataset

```
# Let's look at a one image
IMG_INDEX = 7  # change this to look at other images

plt.imshow(train_images[IMG_INDEX] ,cmap=plt.cm.binary)
plt.xlabel(class_names[train_labels[IMG_INDEX][0]])
plt.show()
```



  - ***Making the convolutional neural network***

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

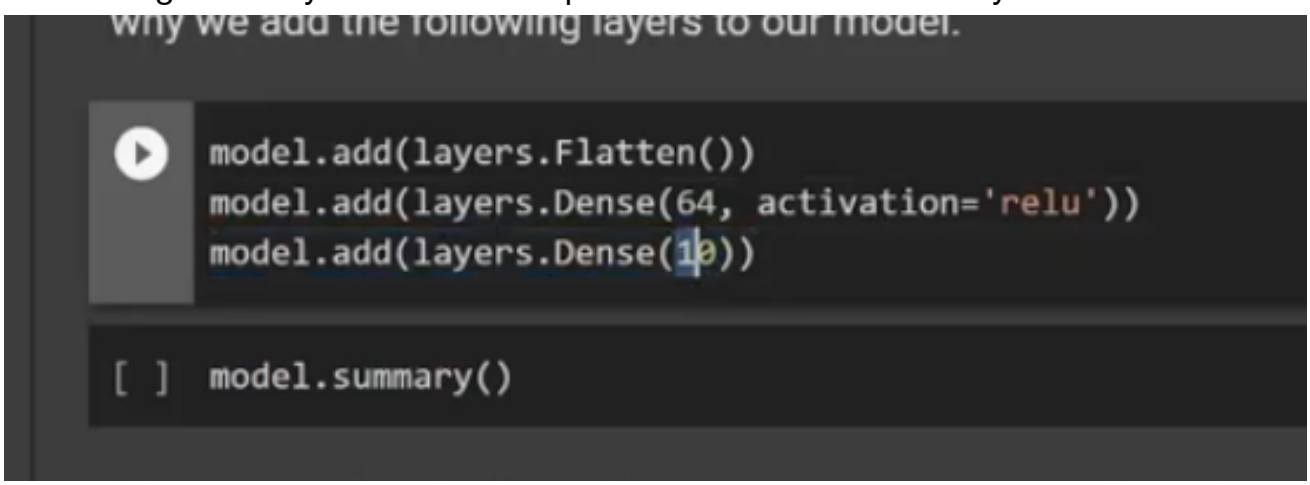
    - -> cnn <- convolutional neural network
    - -> stacking convolutional layers / max pooling / min pooling layers
    - -> each of the lines of code are a layer in the network
      - -> max pooling layers
      - -> layers to reduce the dimensionality
      - -> you define the amount of filters, the elements per filter, the activation function ->

to normalise the dot products after running the filter, then the input shape -> the rows by columns of the input to the model

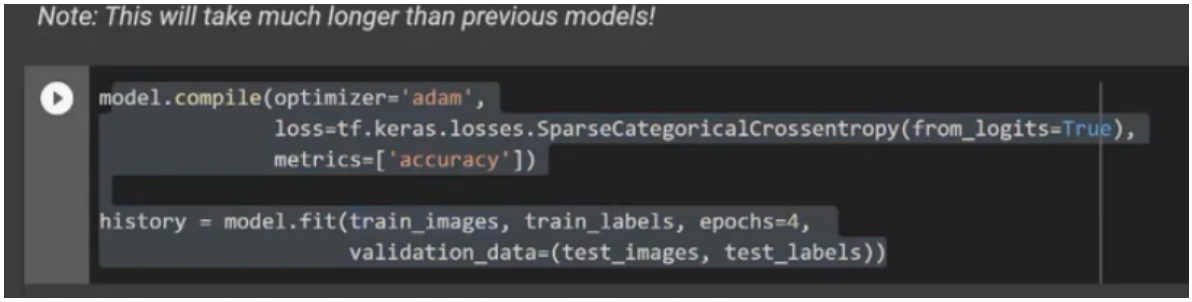
```
model.summary()  # let's have a look at our model so far

Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 64)          36928
=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
```

- -> it's 30x30x32 -> because there are two pixels of padding
- -> shrinking the shape by a factor of 2
- -> taking 64 filters
- -> max pooling
- -> this is the stack of convolution and max pooling layers

- ***Adding the dense layers***
  - -> this is a convolution base which extracts the features from the image
  - -> then we add the dense layers which take those different features and combine them
    - -> these combination of features can be used to classify the object
  - -> to add the dense layers
    - -> model.add(layers.Flatten()) <- moving them into one dimension
    - -> a 64 dense layer
  - -> after adding these layers he has then printed out another summary of the model



    - -> 10 is the amount of classes which we are classifying it into
  - -> the convolutional base / classifier
- ***Training the model***
  - -> he's reducing the epochs to 4 -> it should be on 10 but this takes too long
  - -> when the model is trained its accuracy is returned
  - -> the loss function computes the cross entropy loss
  - -> use atom / a categorical cross entropy loss function -> there are a lot of different loss



functions and you can look them up depending on the context