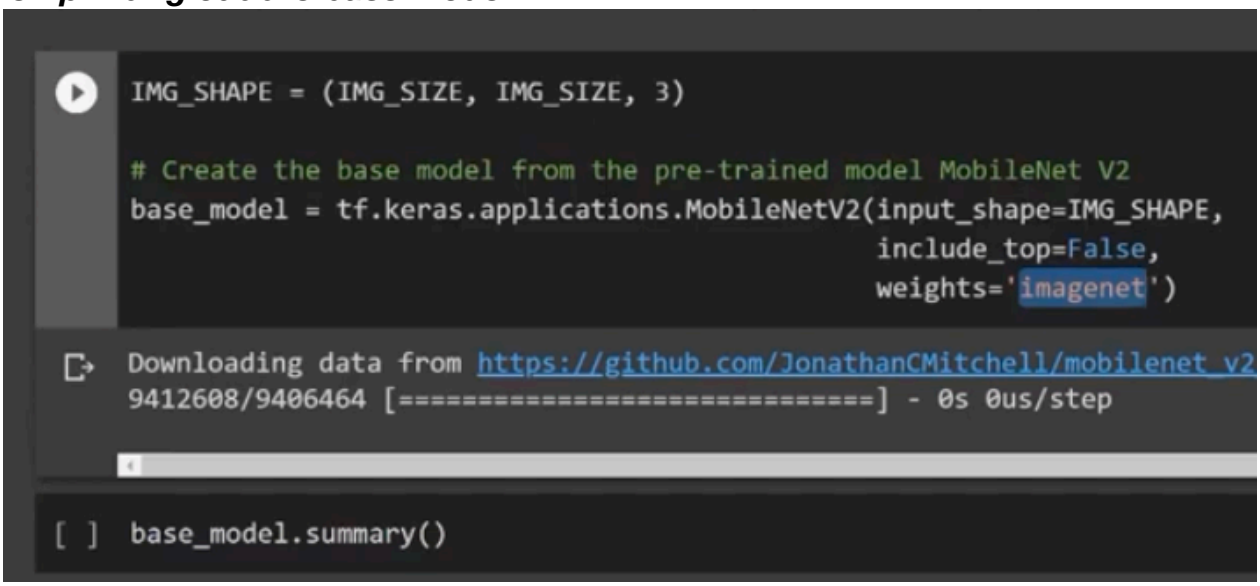
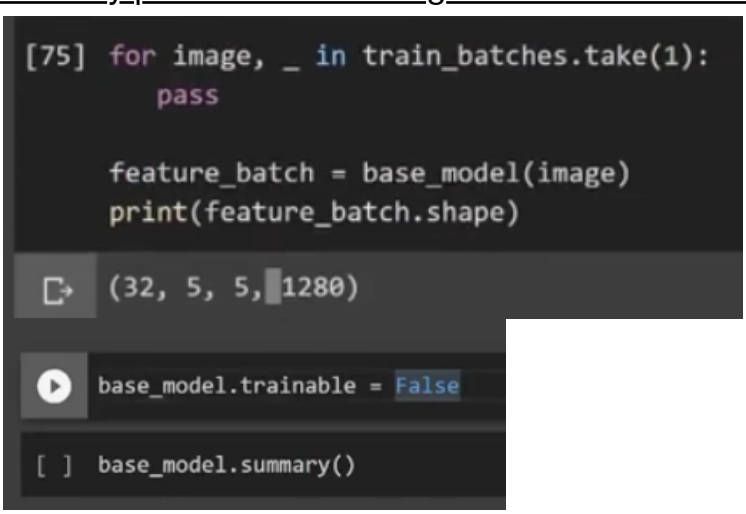
- ***-> picking a pre-trained model to reuse***
  - -> this is one from google which is called mobile net v2 which is built into tensor flow
  - ***-> he's defined the base model -> then the image shape which was defined above it***
    - -> the architecture of what we want
    - -> weather we include the classifier or not
    - -> i.e the model which we're importing works for 1,000 different classes of objects -> we're not including the top
    - -> we're just loading the weights for what we want
  - ***-> then printing out the base model***



    - -> aka like the head of the dataset
    - -> the architecture and the data
      - -> the architecture is like which rows and columns the matrix will have and the data is what fills it
    - -> converting that imported model (to reuse) into a tensor
  - ***-> freezing the base***
    - -> we are reusing a model and adding our points onto the top of it
    - -> the model we are reusing is the 'base' -> freezing it means it doesn't change
    - -> the only parts we are training are our data which we are adding over the top of it



    - -> then printing out another summary of the data
    - ***-> adding another classifier on top of it***
      - taking the average of all the layers and inputting them into a 1D tensor
      - -> this is the global average pooling
        - moving the points into a 1D tensor
        - -> converting the borrowed model into one dense node

- ***-> and then adding a prediction layer***
  - -> the prediction layer creates the final model
  - -> running another summary then returns the number of trainable data
- ***-> then combining it with our data to create a new model***

```
[ ]  global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
```

Finally we will add the predicition layer that will be a single dense neuron. We

```
prediction_layer = keras.layers.Dense(1)
```

Now we will combine these layers together in a model.

```
[ ]  model = tf.keras.Sequential([
       base_model,
       global_average_layer,
       prediction_layer
     ])
```

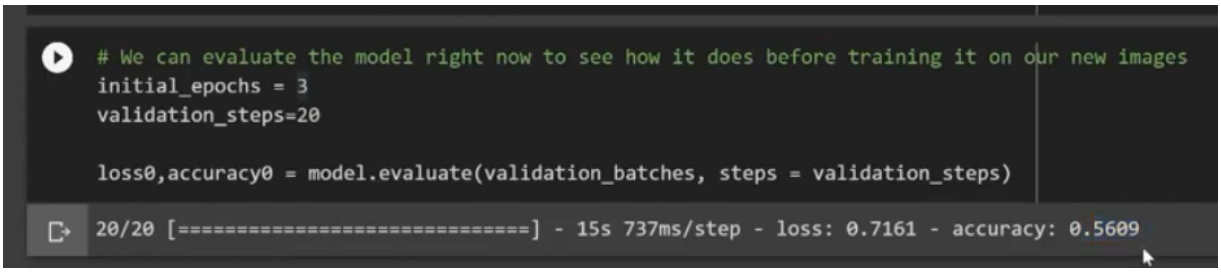
```
[ ]  model.summary()
```

- ***-> then he's ran another summary***
  - -> global average pooling

```
model.summary()

Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_160 (Model) (None, 5, 5, 1280)        2257984
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1280)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 1281
=================================================================
Total params: 2,259,265
Trainable params: 1,281
Non-trainable params: 2,257,984
_________________________________________________________________
```

  - -> the only data which is trainable is our data (not the model we've borrowed)
- ***-> training the model***
  - -> the learning rate
    - -> he's picked one which is slow
    - -> not making major changes to the model
  - -> then using binary cross entropy for the loss function because there are two input parameters
  - -> evaluating the model currently without training it on our validation data
  - -> it's returning the accuracy of the model as it trains it
  - -> then running the training with the base layer



        - -> it takes an hour to train with the base layer -> but has ~90% accuracy once it's been trained
        - -> keras can be used to load the model -> rather than tensor flow
    - ***-> using the model***
        - -> model.predict <- and it's suggesting the different parameters you could have
- ***Object detection***
    - -> you can use tensor flow to do object detection and recognition
    - -> there is an API for object detection
    - -> you can use this project using the documentation on its GitHub page from tensor flow
    - -> Python also has a facial recognition module
    - -> you need enough training data
        - -> then to create a model which uses mobile nets
        - -> mobile nets is the google training model which he's reused in this example
- ***Next***
    - -> recurrent neural networks