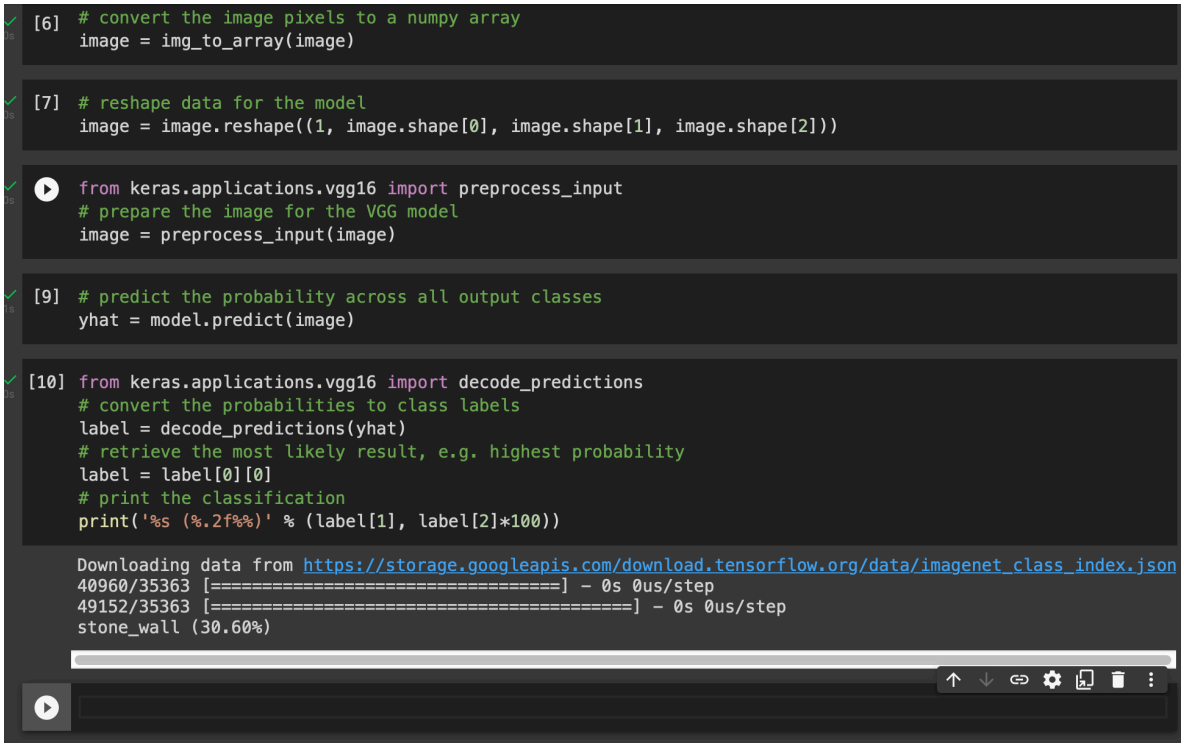
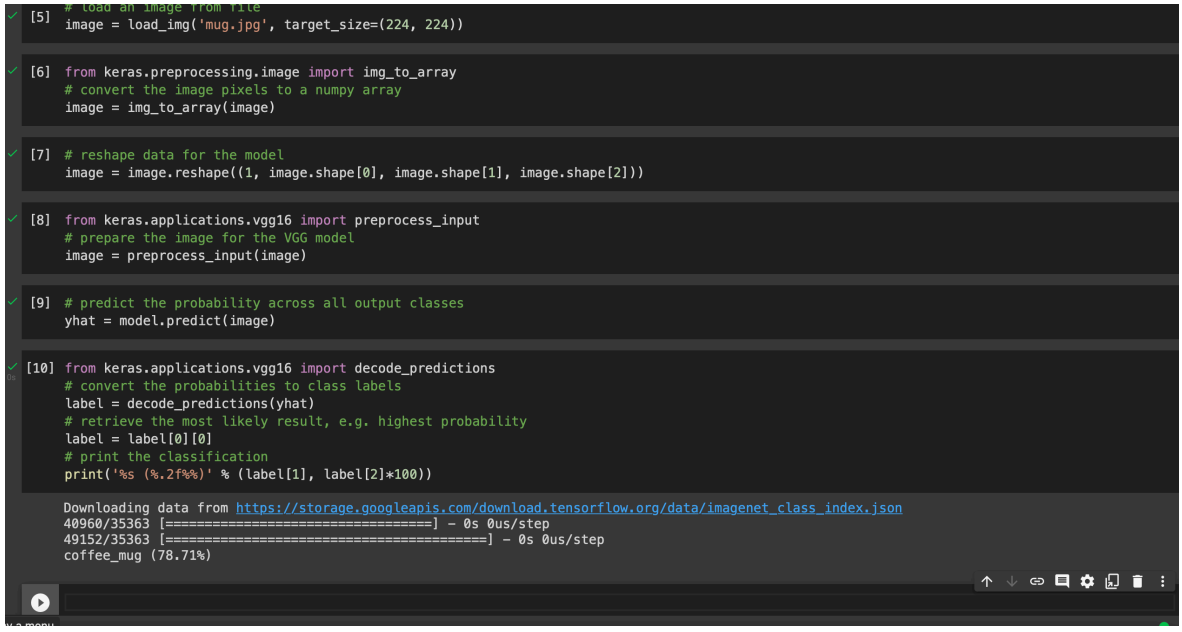
NOTES 3: DATATHON COMPETITION

- Started playing with various models that can potentially be our go to thing for the purpose of this competition
- Firstly, VGG16, using the already built model in the Keras library, predicted the presence of a coffee mug in a picture with 78% likelihood. This accuracy is good enough apparently because ‘coffee mug’ is one of the 1000 classes of the ImageNet dataset
- Point to be noted; although “STONE WALL” is also a class of the ImageNet dataset, the model was only able to predict one (taken from the MINC dataset uploaded on teams) with 30% likelihood, which is certainly less than what we want
- Codes for the same;

This was the image I gave the model for prediction



This code is for the above image

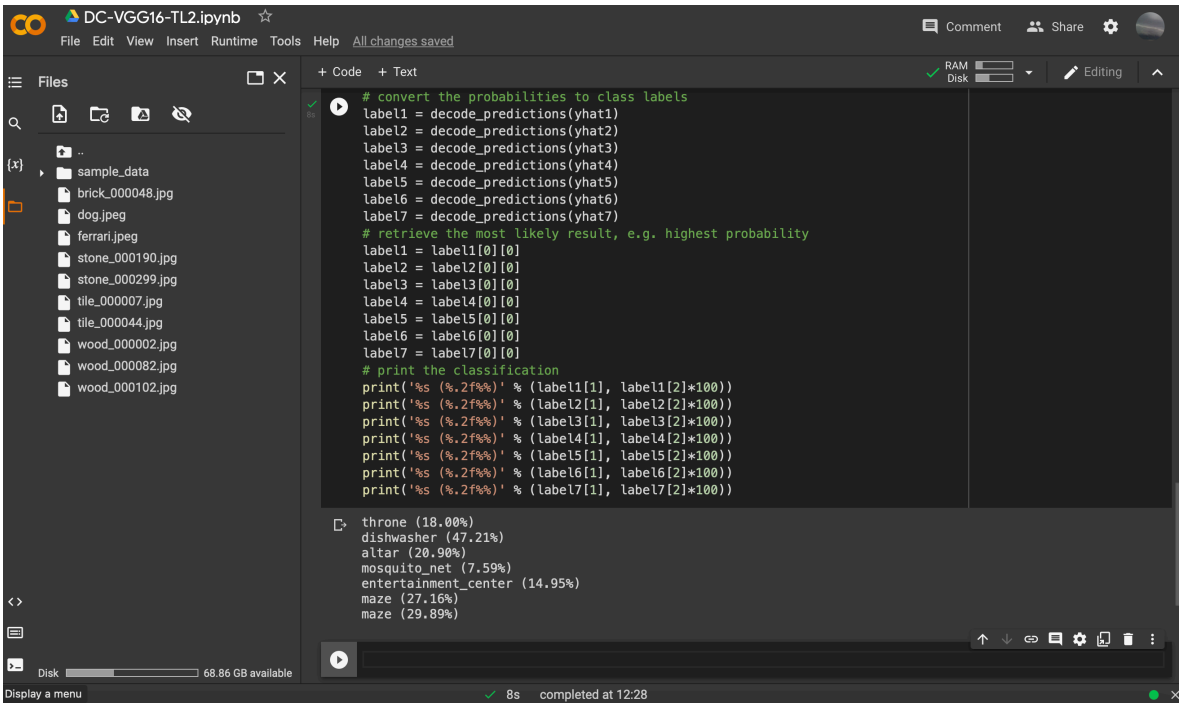


This code is for the prediction of Coffee Mug

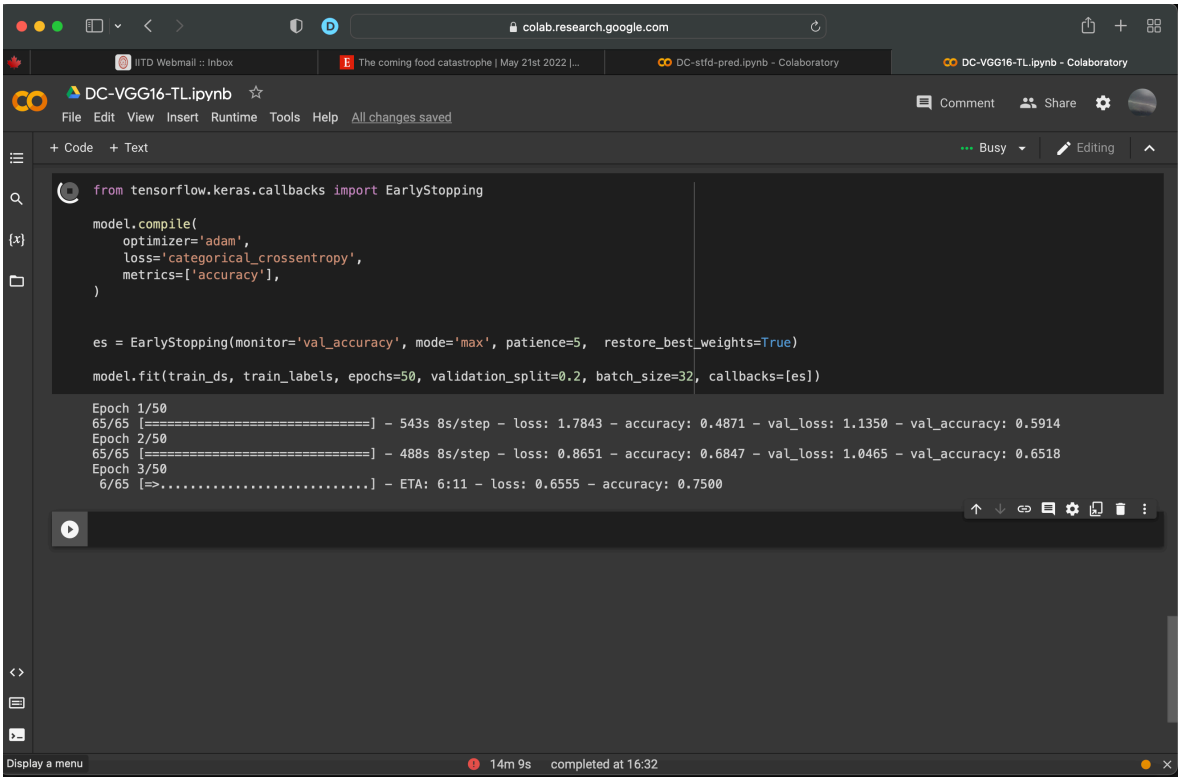
- The stark difference in the likelihood with which the model predicts these objects / classes (even when both of them are present in the class list of ImageNet, which will apparently not be the case with PVC pipes, Rebars and Cementitious Debris) is indicative of the open waters that we will have to venture into since the classes that we've to predict (5 labels given in the competition instruction sheet) are not so standard / regular / typical in appearance & occurrence. This is the reason (probably!) why we have a

prediction with such a low likelihood for Stone Walls but the accuracy is quite decent for the Coffee Mug.

- It is interesting to see if the model is able to predict the images relevant to us. I took some images from the MINC dataset and gave them to this model, as is already done for the stone wall. However, since stones, tiles and wood are **NOT** there in the list of classes in ImageNet, it is quite redundant to predict them. For the sake of it, I've done it though —>



- Trained another VGG model through transfer learning on a dataset available through TensorFlow. Was taking too long to execute (2 out of 50 epochs in 15 mins), but the model was going in the right direction. We might as well want to proceed with transfer learning but will have to experiment with various already available models and see which one of those models corroborates our dataset and our needs the best.



- Transfer Learning can basically take place in either of these **two aspects:** We can either use a Pre-Trained Model as a "Classifier" (using the complete model) or as a "Feature Extractor" (using the initial part of the model)
- In the "Feature Extractor" case, we can customise according to our needs in the sense that we can choose the number of final layers that we wish to remove from the original model. This highly depends on the similarity between the task that we want to perform and the task that the model is capable of performing.

- Here's another couple of pics I gave to the VGG16 (trained on ImageNet), one of a doberman and another of a Ferrari and the model was able to give out correct predictions for both of them.

```
[1]  # example of using a pre-trained model as a classifier
     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
     # load an image from file
     image = load_img('dog.jpeg', target_size=(224, 224))
     # convert the image pixels to a numpy array
     image = img_to_array(image)
     # reshape data for the model
     image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
     # prepare the image for the VGG model
     image = preprocess_input(image)
     # load the model
     model = VGG16()
     # predict the probability across all output classes
     yhat = model.predict(image)
     # convert the probabilities to class labels
     label = decode_predictions(yhat)
     # retrieve the most likely result, e.g. highest probability
     label = label[0][0]
     # print the classification
     print('%s (%.2f%%)' % (label[1], label[2]*100))

     Downloading data from https://storage.googleapis.com/tensorflow/keras-applications
     553467904/553467096 [==============================] - 3s 0us/step
     553476096/553467096 [==============================] - 3s 0us/step
     Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/
     40960/35363 [==================================] - 0s 0us/step
     49152/35363 [==========================================] - 0s 0us/step
     Doberman (35.42%)
```

```
# example of using a pre-trained model as a classifier
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
# load an image from file
image = load_img('ferrari.jpeg', target_size=(224, 224))
# convert the image pixels to a numpy array
image = img_to_array(image)
# reshape data for the model
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
# prepare the image for the VGG model
image = preprocess_input(image)
# load the model
model = VGG16()
# predict the probability across all output classes
yhat = model.predict(image)
# convert the probabilities to class labels
label = decode_predictions(yhat)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
sports_car (83.53%)
```

From Jiarui Lin et.al.:

- This paper seems to be just the ideal for our referral since its contemporary (published just 3 months ago), has decent overlap with the image classes that we have to predict in the competition and also, the paper explains in detail the methodology that they use at almost every other step of the process

- Object Detection Datasets in Construction: they've highlighted the same problems that we already know existed before, which is why they've worked on building a dataset per se and tried to make it work

- Object Detection Algorithms in Construction: can be divided into mainly two parts; **Two Stage Algorithms** (R-CNN, Fast R-CNN, Faster R-CNN) aka Region Based Method because of the two stage processing. First it extracts a series of checkboxes and then uses CNN's for classification. **One Stage Algorithms** (YOLO) (finally, the full name of the algorithm makes sense!!) directly obtains the prediction results from the image and is also called the end-to-end method

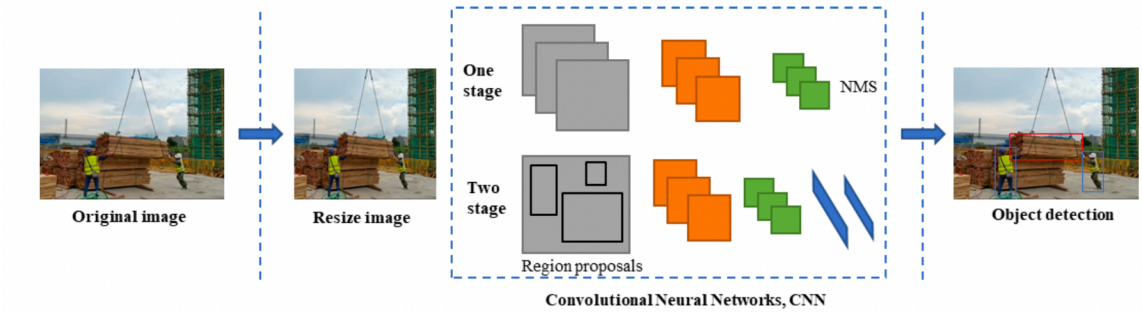


Fig. 1. Flowchart of the object detection algorithm

- The above figure shows a pictographically representation of the difference in methodology of the two algorithms

- The **quantifiable differences** between the two algorithms are: Two-Stage has higher precision but relatively low speed

- Methodology: Site selection —> image collection / acquisition —> Data cleaning (4 steps) —> Image annotation

Table 2. Categories and each class of the object label

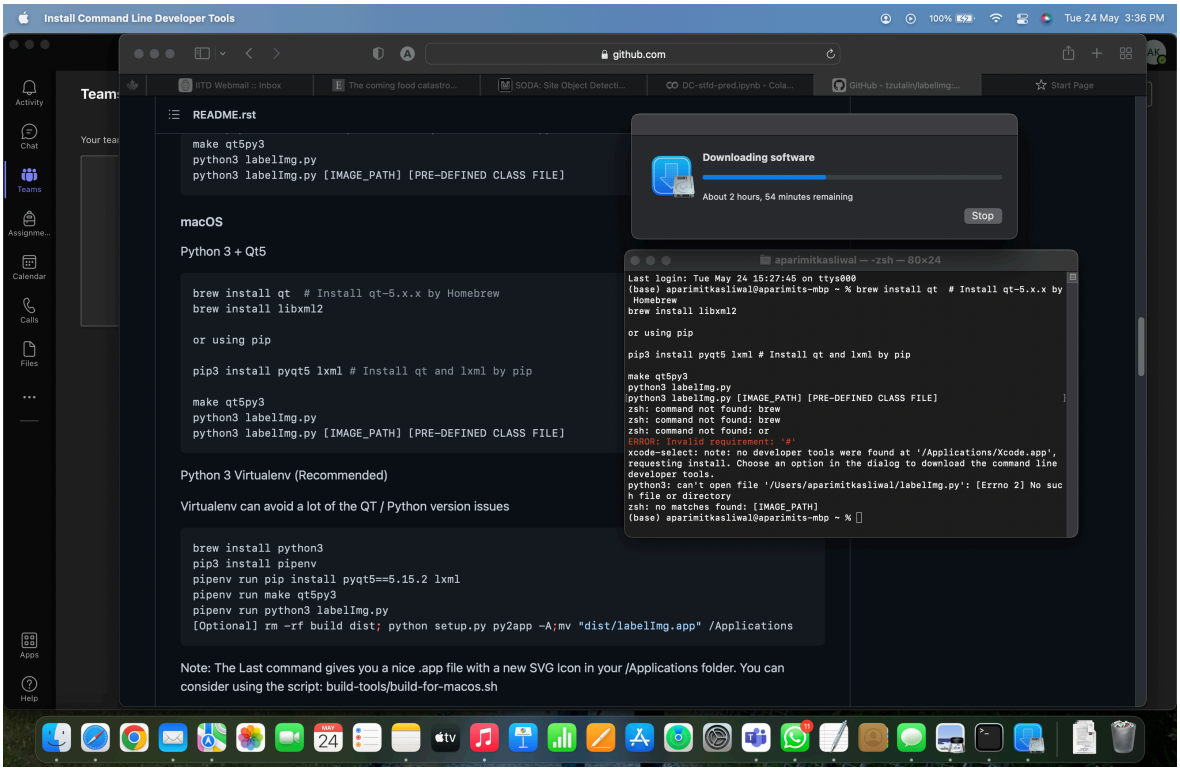
category	**label**				
Person	person	helmet	vest		
Material	board	wood	rebar	brick	scaffold
Machine	handcart	cutter	ebox	hopper	hook
Layout	fence	slogan			

- Classes relevant to us would probably only be brick and rebar
- First two parts of the methodology are quite straightforward and are not that relevant to us at the moment
- **Data cleaning**: they suggest four steps as is evident from the following image

Fig. 5. Example of images that need to be removed and processed.

- **Data annotation**: They suggest the use of VOC (Visual Object Classes) which is perhaps the most widely used, accepted and standard dataset format. They initially point out a few basic things regarding the annotating process and and common errors that one can fall for which must be avoided.

- They use the API / software labelImg to annotate the 15 image categories and generate the corresponding XML files (containing the coded information pertaining to the annotation). It should be noted that LabelMe generated .json files while labelImg seems to be generating .xml file format. The installation for labelImg took too long, but the UI and API seem quite similar to that of LabelMe



- They then discuss and elaborate on the common spelling and other errors that can possibly occur during labelling.

Table 4. Comparison of SODA dataset with other datasets of building construction

Dataset	Image	Object	Description	Size
SODA	19846	286201	The 15 classes of objects in the construction site including four categories of workers, materials, machines, layout are annotated.	1920*1080 and other
MOCS	41668	222861	The 13 classes of moving objects (worker and mobile machine) in construction sites are annotated.	1200*--
ACID	10000	15,767	Collect 10000 images of 10 construction machines and annotate machine types and their corresponding positions on the images.	>608*608
CHV	1330	9209	Special dataset of 1330 images for multiple PPE classes considering actual construction site background.	608*608

- The comparison (attached above) of the SODA dataset with other competitive datasets in the construction industry gives us a helpful peek into the datasets that are available and which might be of our use.

- **Experimentation on the dataset:** They've primarily utilised the YOLOv3 and YOLOv4 for testing the model. Another learning from their experimental methodology is the division of the 100 epochs of training into two parts, with the main model parameters being frozen in the first 50 and the tuning of more intricate hyper-parameters being the focus during the last 50 epochs.

- In the **Conclusion** section of the paper, they mention that manual annotation can be ditched for better techniques that either use automation or are annotated using internet based crowdsourcing. Since this paper seems to be quite recent and pretty much state-of-the-art, we might have to proceed with manual annotating for our purpose during the competition as well —>

construction industry. 4. This dataset construction still uses manual annotation. Annotators are students majoring in engineering management. In the future, more annotation methods such as crowdsourcing annotation [44] and automatic annotation can be explored to construct datasets. 5. The data acquisition

- Commonly used labelling softwares include the following —>

Image Annotation Tools

Here is a list of tools that you can use for annotating images:

1. MakeSense.AI
2. LabelImg
3. VGG image annotator
4. LabelMe
5. Scalable
6. RectLabel

(From “Towards Data Science”)