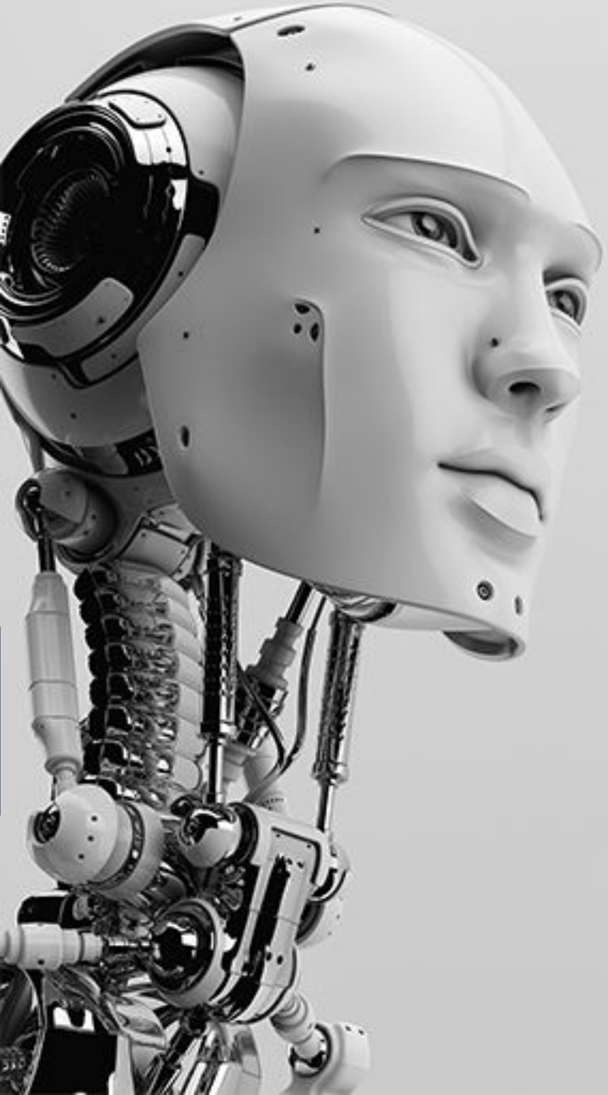
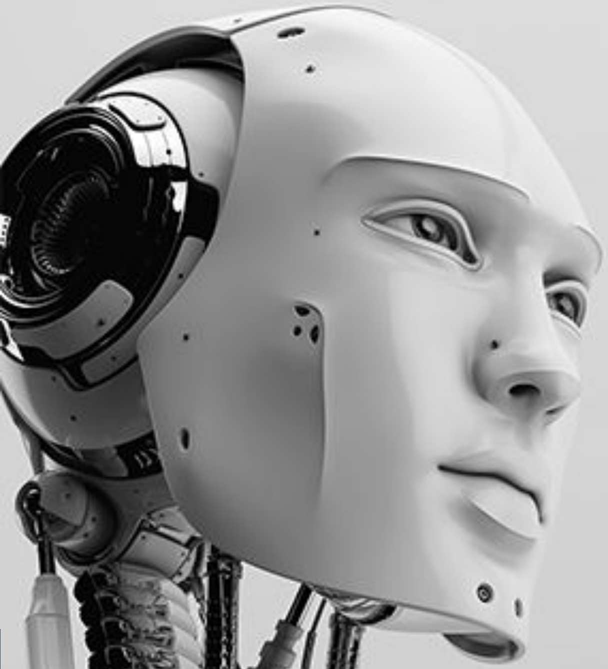


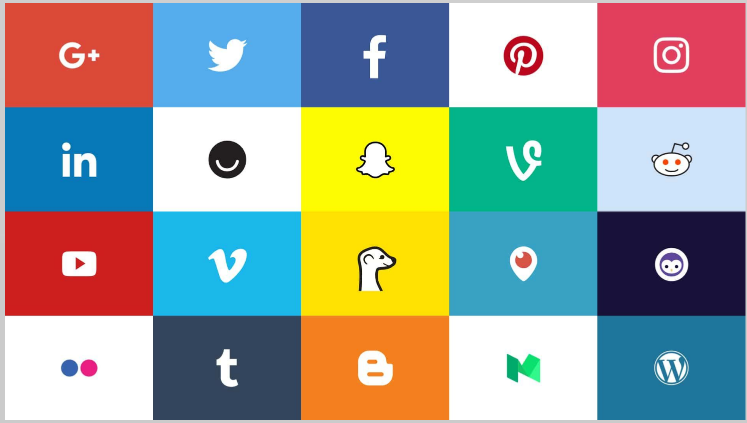
GEM 2.0

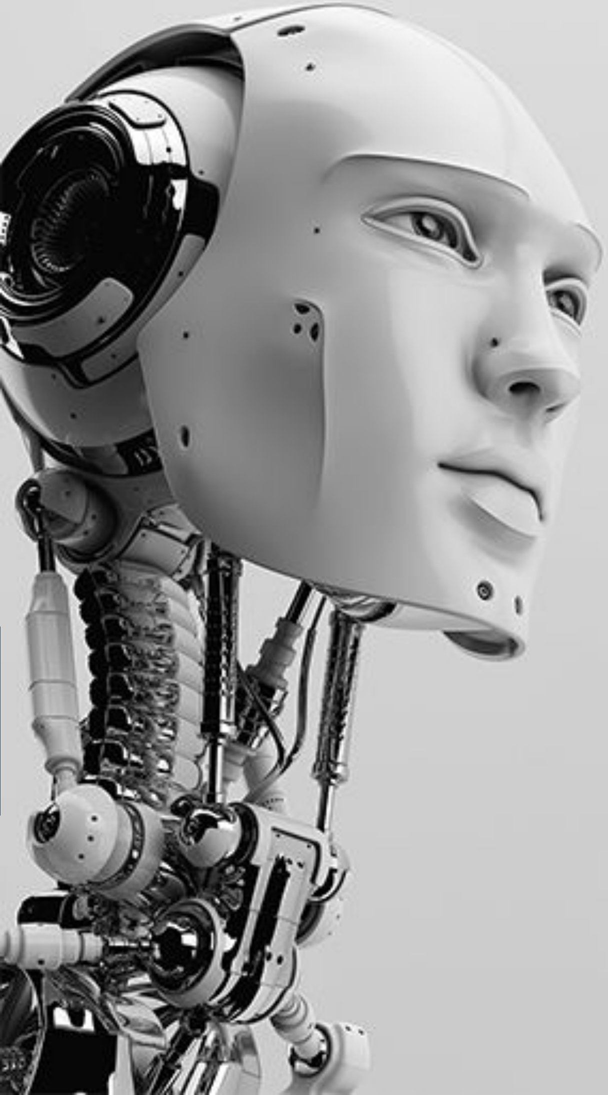
Logo Relevance - Akhil and Tarun

Background - Logo



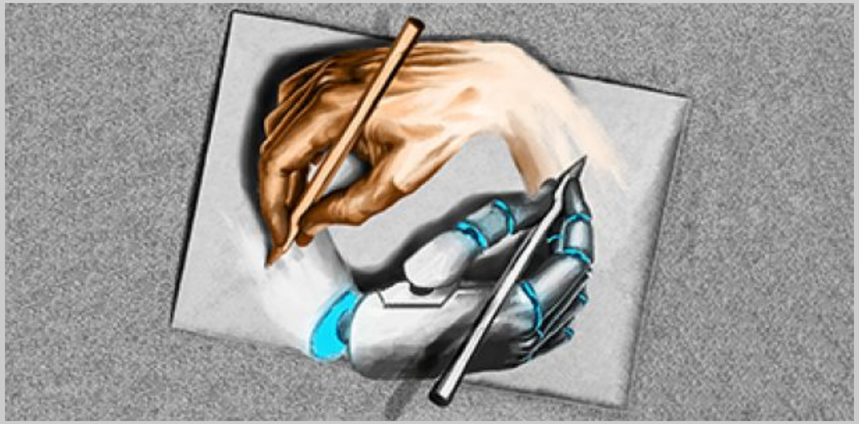
- Logo is the visual entity **signifying an organization**
- As soon as you begin **advertising your product** with your logo, your logo is technically **trademarked** in the eyes of the law.
- In the United States, trademark rights begin when the trademark is put into **commercial use**.

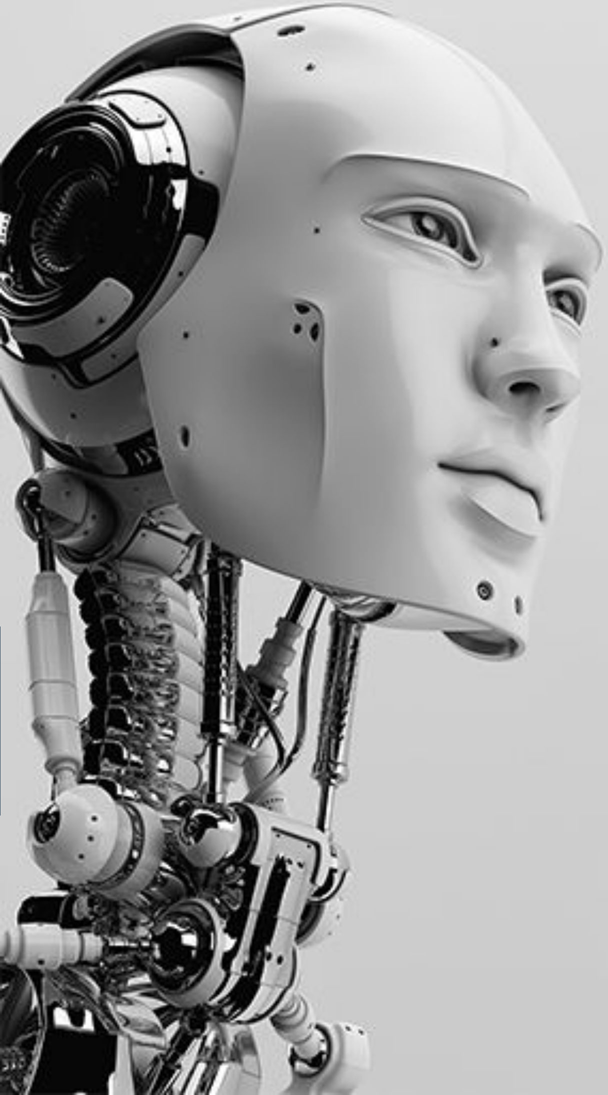




Abstract

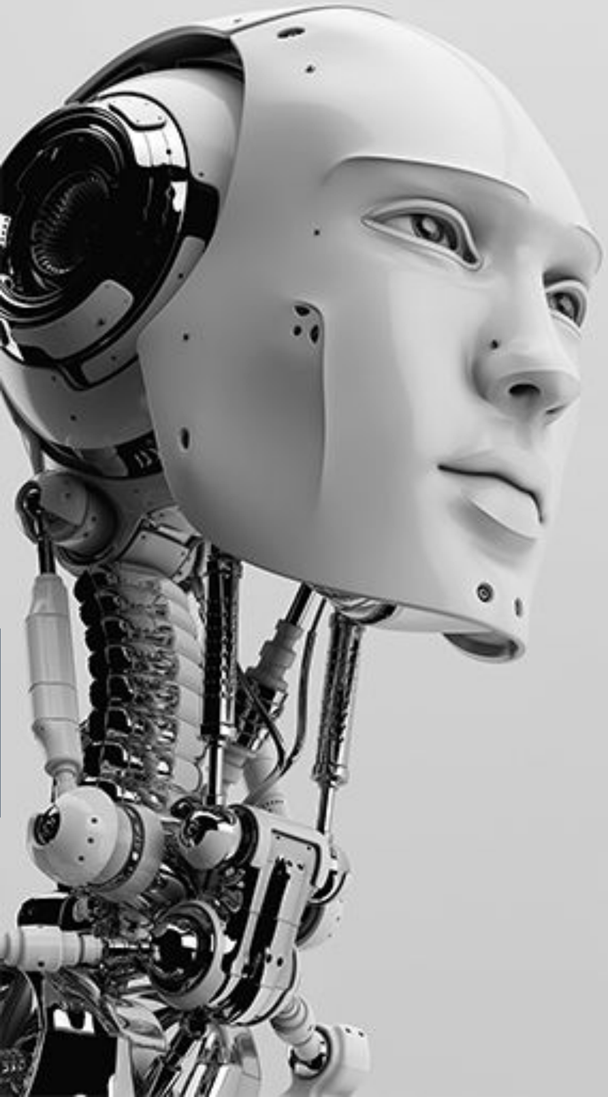
- The designer is always put to challenge to design a logo for a new organization that represents the organization's value/mission/nature of service they provide.
- If he/she designs a logo, can we automatically pull out logos that match the one that is designed or is unique.
- This reduces a lot of rework for the designers.





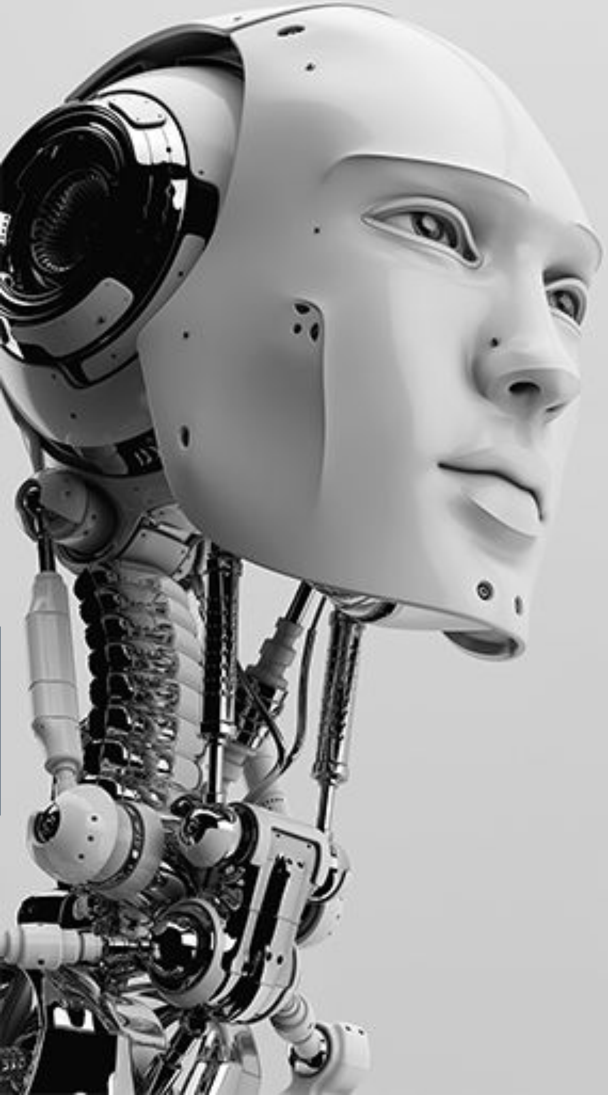
Applications

- Logo Detection - (Finding all the logos in a video/image (opencv))
- **Making sure new logos are not similar to existing ones**
- Feature Extraction to generate new unique logos
- Making commercial use of scraped logos (Manual selection required)
- State of art classification - 500 classes (80%+ Accuracy)



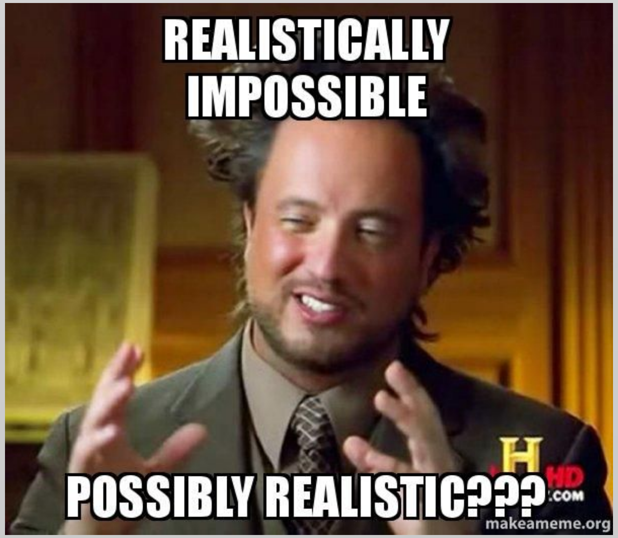
Methodology

- **Data Gathering:** Gather the images
- **Preprocessing:** Clean and resize the images, Remove noise
- **Modelling:** Extract the features, Classify the features
- **Evaluating:** Select the similar logo(s), Rank the brand/logos
- **Deployment:** API for similar logo recognition

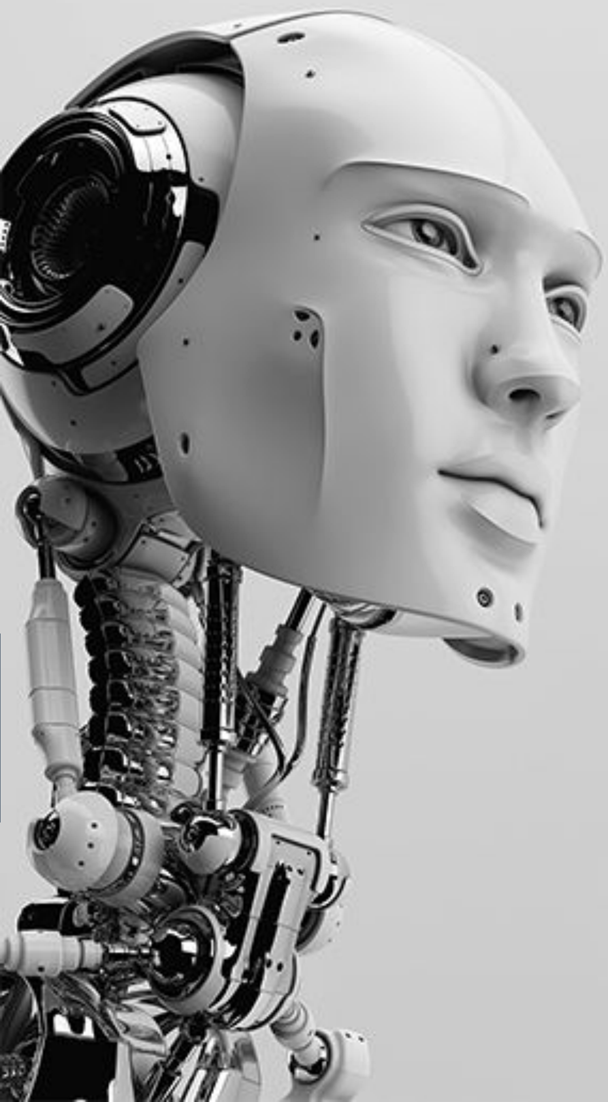


Ideal Data Required

- Get the logos of all the existing companies
- If not all, atleast for the top companies across the globe
- Few hundreds of logos for each company

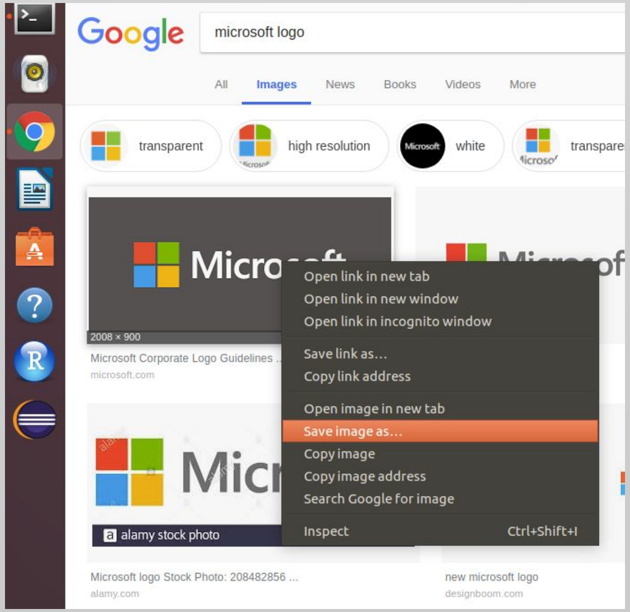


BUT WE WON'T GIVE UP THE GOOD FIGHT!
WE GOT THIS!! BECAUSE...IN THE END...WE'RE SUPERSTARS!
makeameme.org



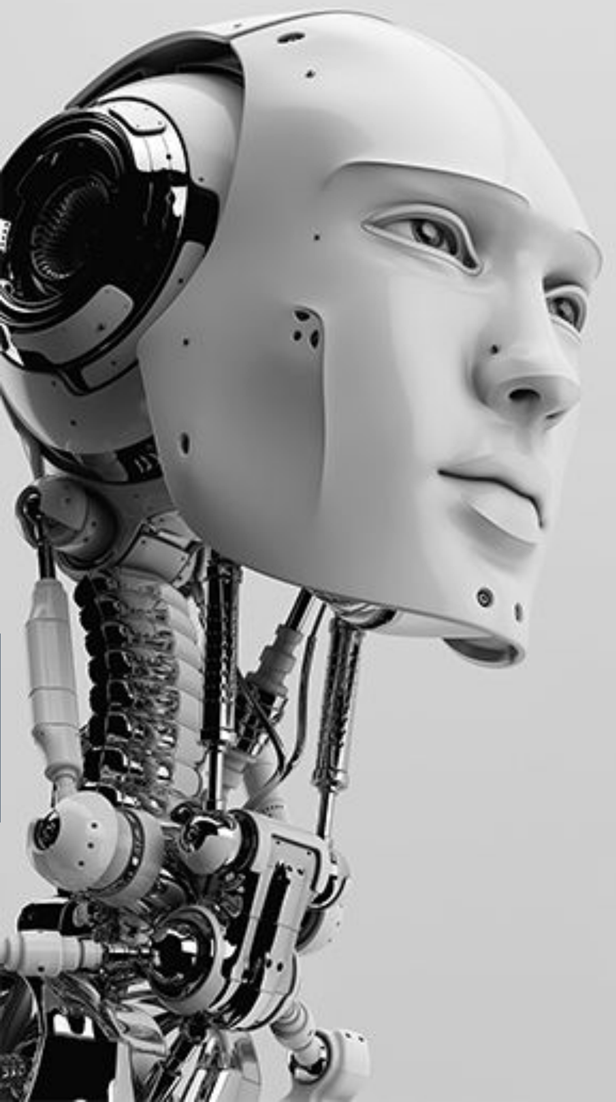
Data Gathering

- Manually gathered Fortune 500 and Forbes 300 listed companies
- Listed out top 500 companies from the above list
- Planned to download 100 logos for each company
- Should I do it manually like this?



microsoft logo

Data Gathered

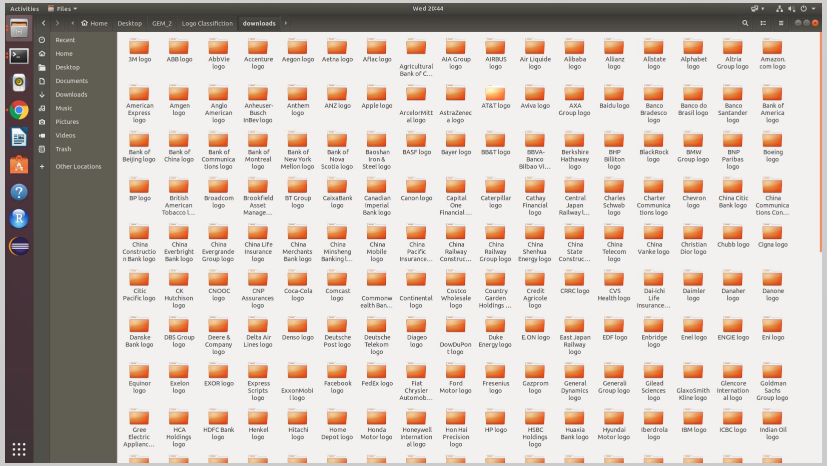


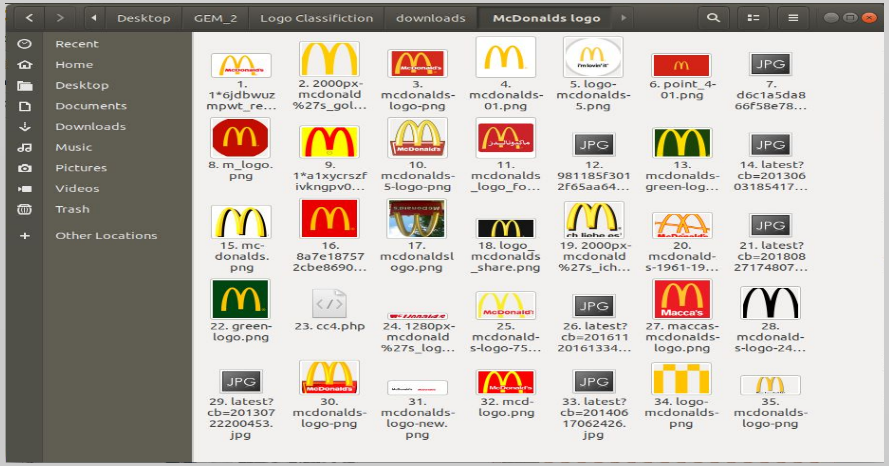
- Thanks to ready-to-run Python Script to download hundreds of images from 'Google Images'.
- scraped 100 logo images for each company i.e. 50000 images - has a lot of noise

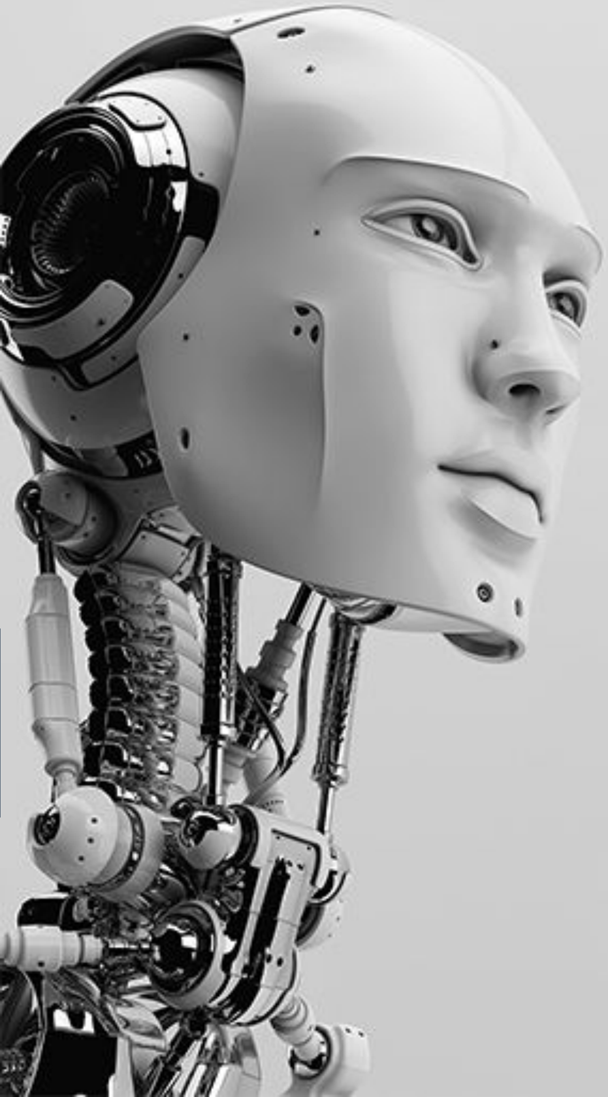
```
$ pip install google_images_download
```

```
$git clone https://github.com/hardikvasa/google-images-download.git
$ cd google-images-download && sudo python setup.py install
```

```
$python google-images-download.py --keywords "Microsoft logo, ANZ logo, ….." --limit 100 --format png
```

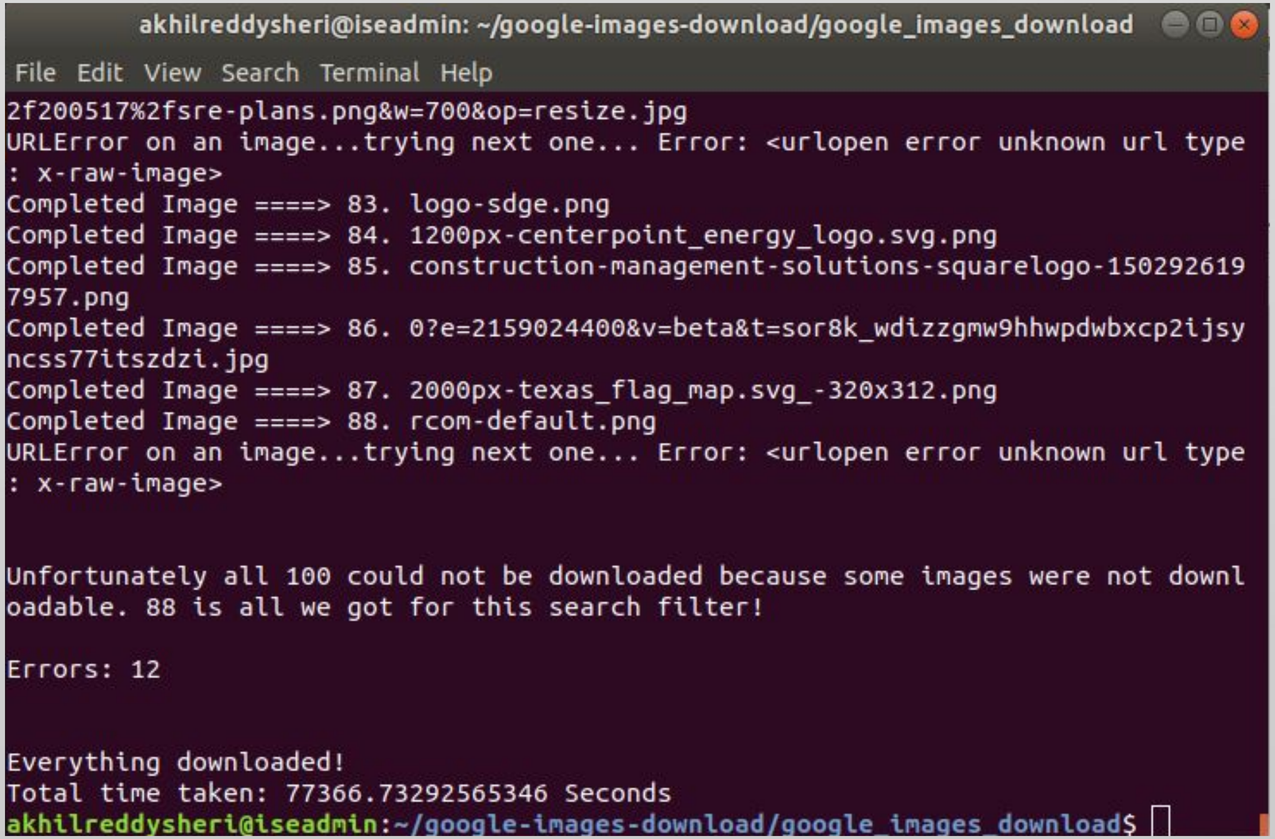






Data Gathered

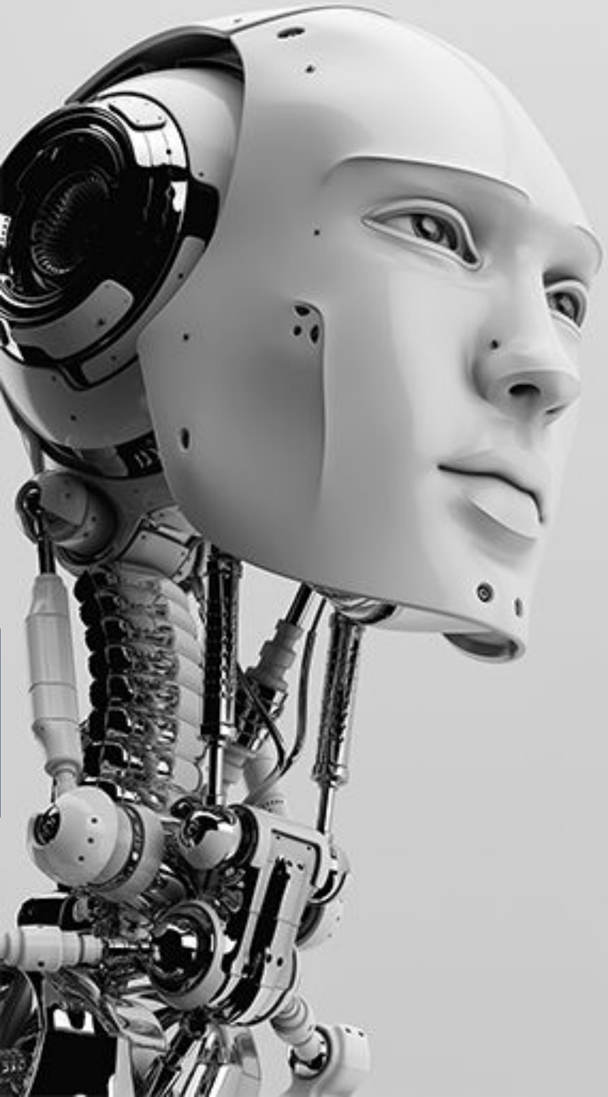
- Marathon!

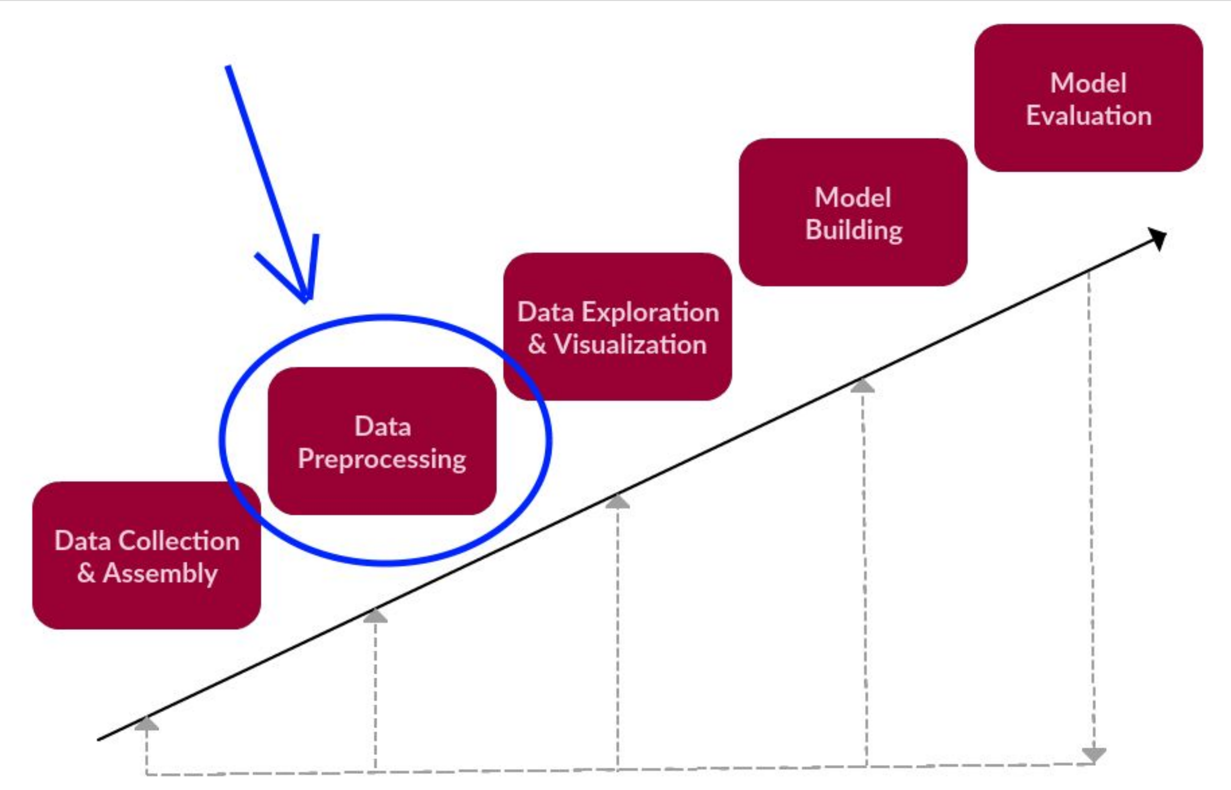


akhilreddysheri@iseadmin: ~/google-images-download/google_images_download
File Edit View Search Terminal Help

```
2f200517%2fsre-plans.png&w=700&op=resize.jpg
URLError on an image...trying next one... Error: <urlopen error unknown url type
: x-raw-image>
Completed Image ====> 83. logo-sdge.png
Completed Image ====> 84. 1200px-centerpoint_energy_logo.svg.png
Completed Image ====> 85. construction-management-solutions-squarelogo-150292619
7957.png
Completed Image ====> 86. 0?e=2159024400&v=beta&t=sor8k_wdizzgmw9hhwpdwbxcp2ijsy
ncss77itszdzi.jpg
Completed Image ====> 87. 2000px-texas_flag_map.svg_-320x312.png
Completed Image ====> 88. rcom-default.png
URLError on an image...trying next one... Error: <urlopen error unknown url type
: x-raw-image>


Unfortunately all 100 could not be downloaded because some images were not downl
oadable. 88 is all we got for this search filter!

Errors: 12


Everything downloaded!
Total time taken: 77366.73292565346 Seconds
akhilreddysheri@iseadmin:~/google-images-download/google_images_download$ 
```

Image Preprocessing





Data Collection & Assembly

Data Preprocessing

Data Exploration & Visualization

Model Building

Model Evaluation

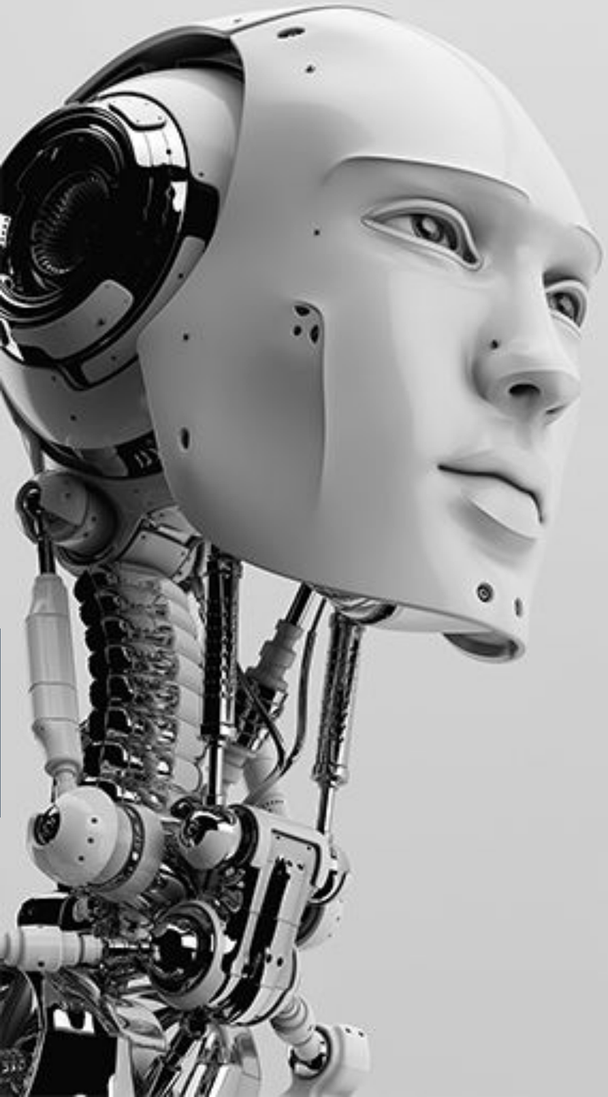
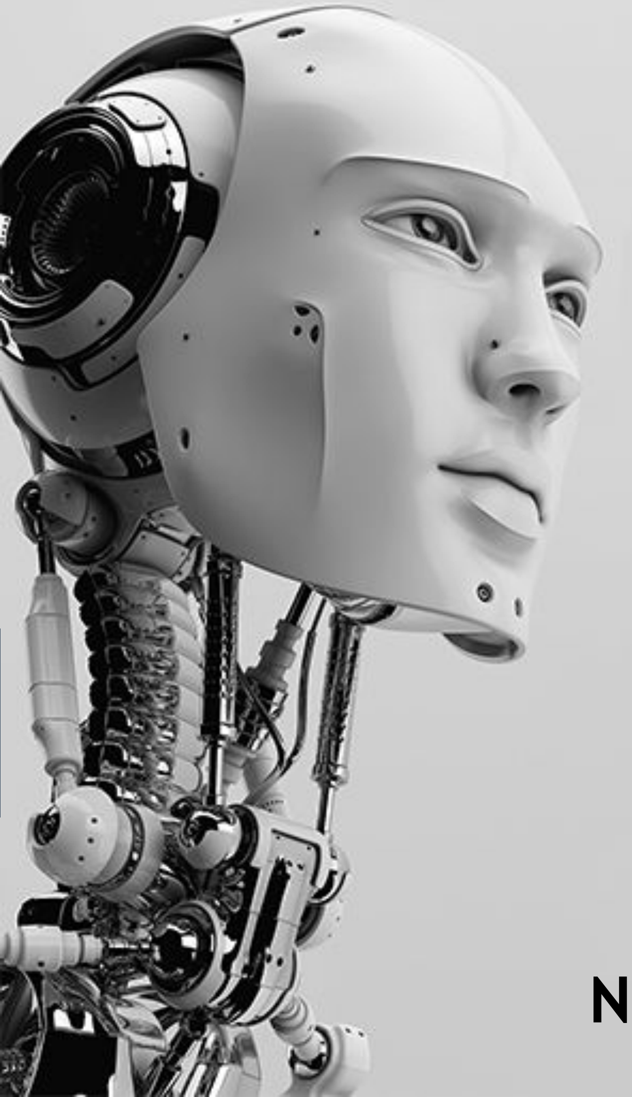
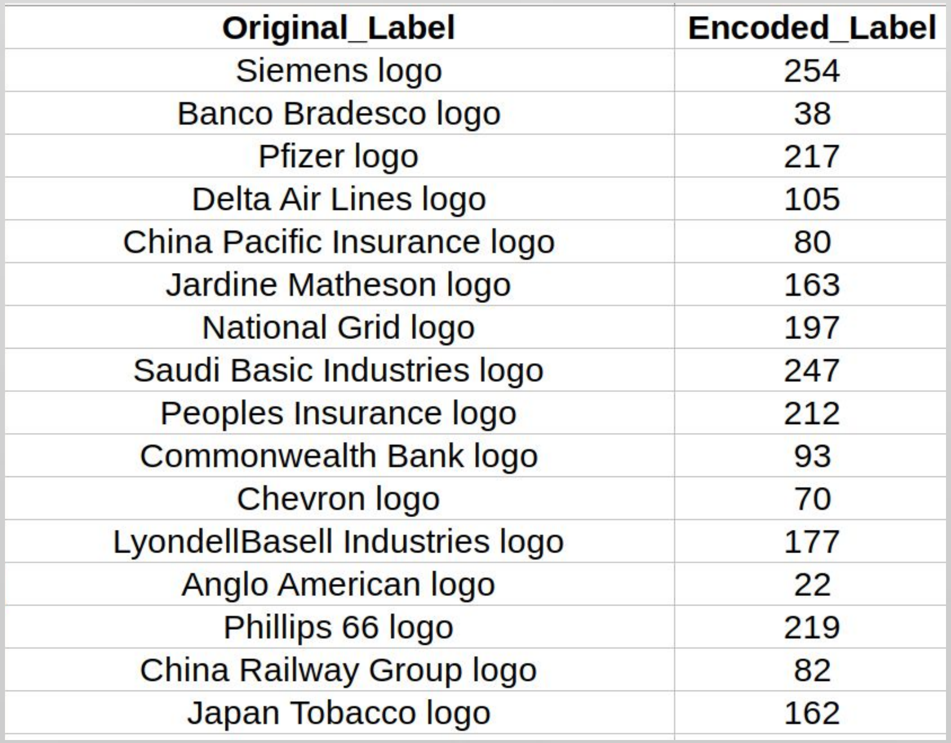


Image Preprocessing

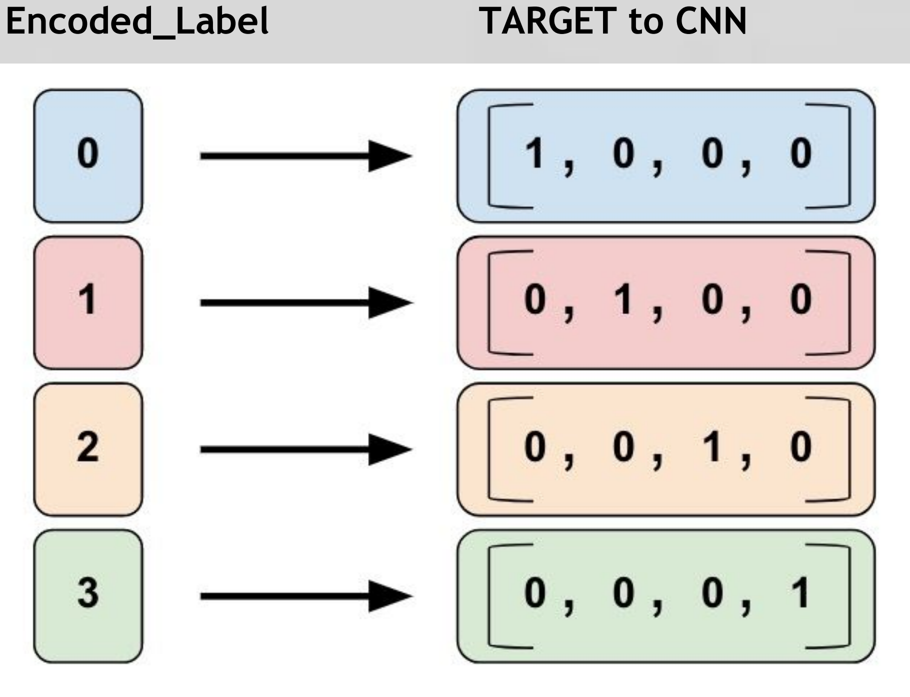
- As the images are scraped, there are corrupted images, so, Subsetted 12 images for each brand
- Resized all images to size (50,50,3)
- Created a numpy array and combined all the train images
- Numpy array is of size (no_of_train_images, 50, 50, 3)
- Similarly created target variable which is of size (no_of_train_images,)

Label Encoding and Categorical Encoding - Target Label





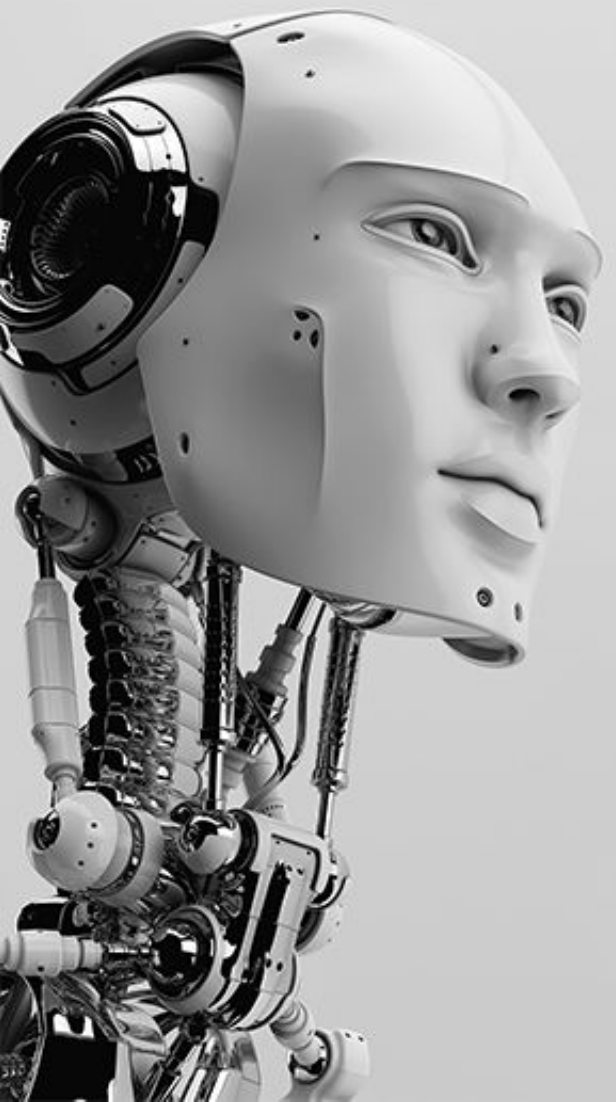
Original_Label	Encoded_Label
Siemens logo	254
Banco Bradesco logo	38
Pfizer logo	217
Delta Air Lines logo	105
China Pacific Insurance logo	80
Jardine Matheson logo	163
National Grid logo	197
Saudi Basic Industries logo	247
Peoples Insurance logo	212
Commonwealth Bank logo	93
Chevron logo	70
LyondellBasell Industries logo	177
Anglo American logo	22
Phillips 66 logo	219
China Railway Group logo	82
Japan Tobacco logo	162

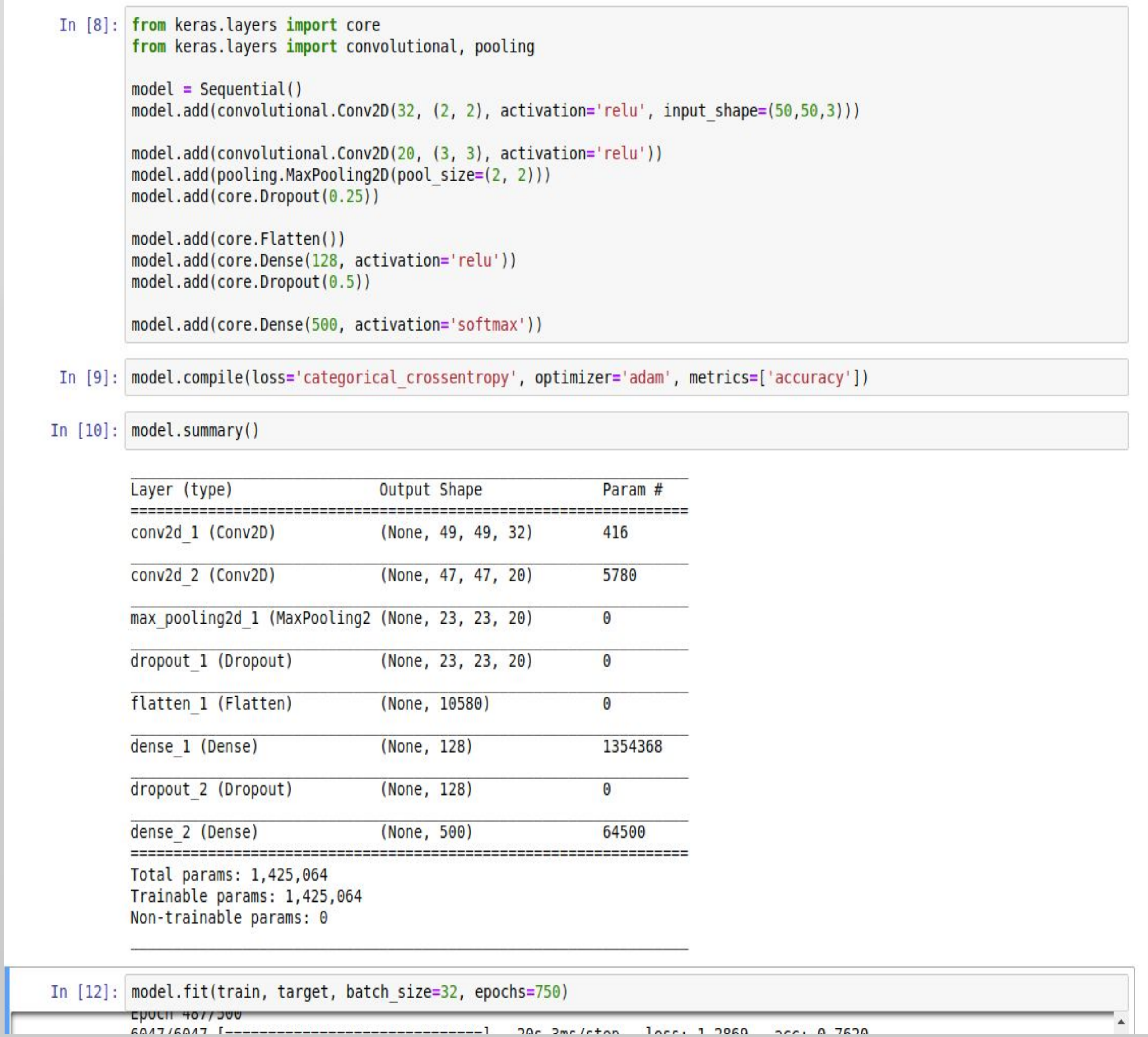


Encoded_Label	TARGET to CNN
0	[1, 0, 0, 0]
1	[0, 1, 0, 0]
2	[0, 0, 1, 0]
3	[0, 0, 0, 1]

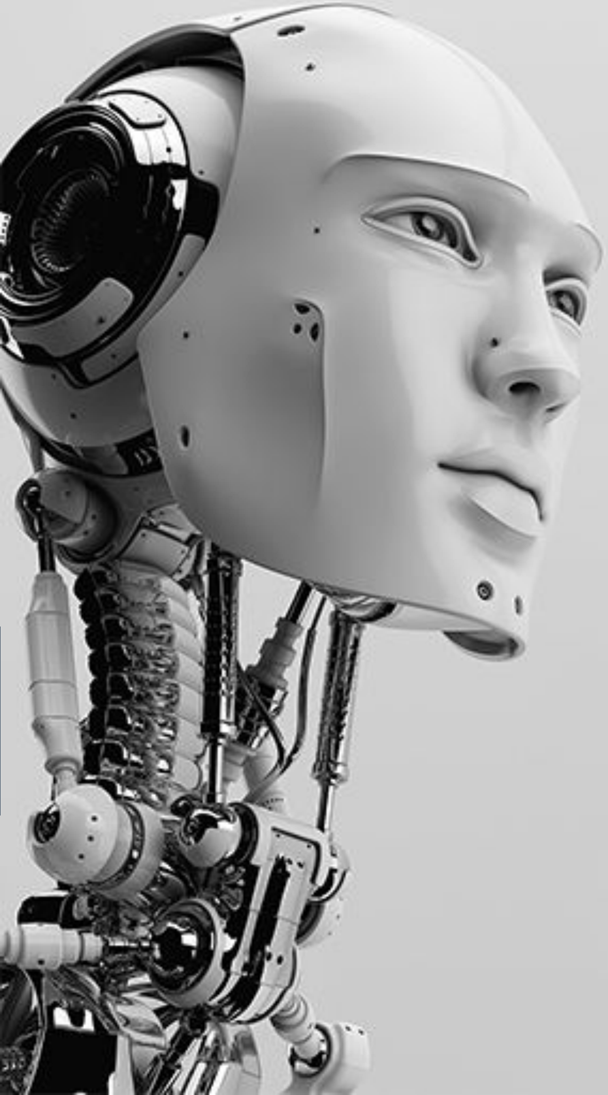
Now target variable of size (no_of_train_images, no_of_classes)

CNN Model



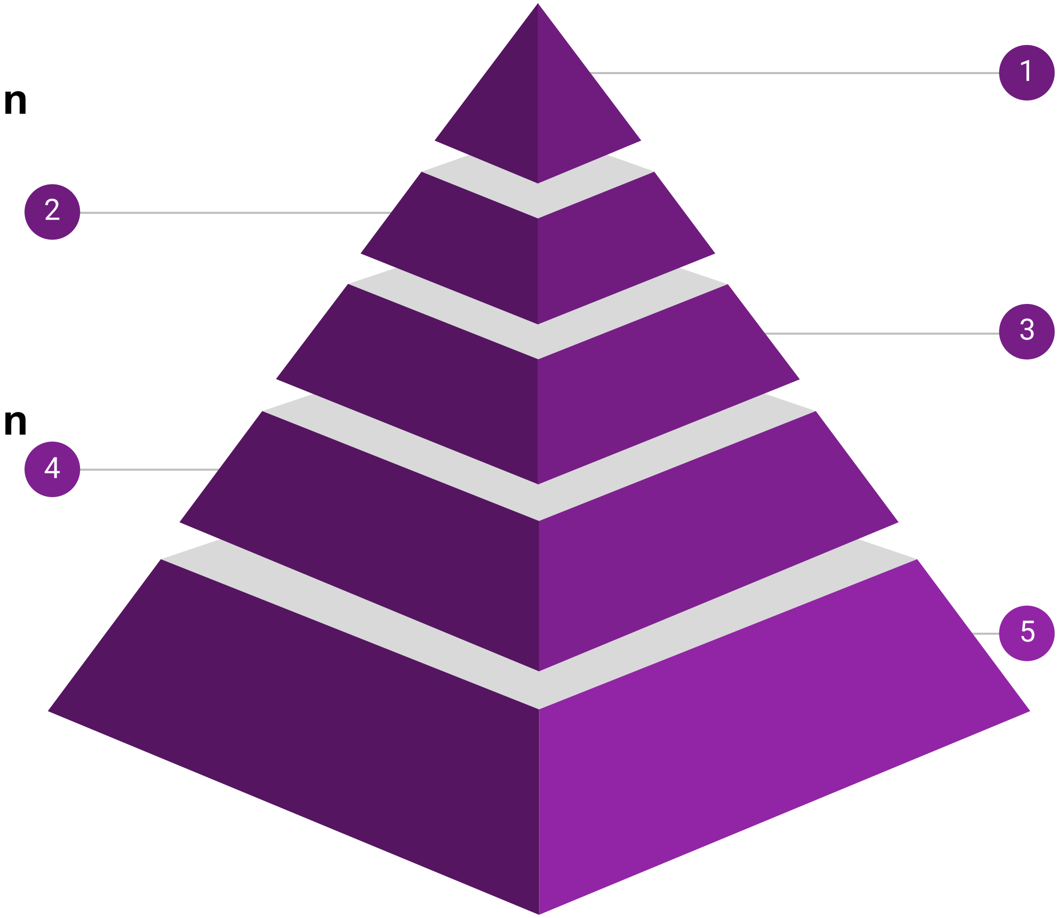


```
In [8]: from keras.layers import core
        from keras.layers import convolutional, pooling

        model = Sequential()
        model.add(convolutional.Conv2D(32, (2, 2), activation='relu', input_shape=(50,50,3)))

        model.add(convolutional.Conv2D(20, (3, 3), activation='relu'))
        model.add(pooling.MaxPooling2D(pool_size=(2, 2)))
        model.add(core.Dropout(0.25))

        model.add(core.Flatten())
        model.add(core.Dense(128, activation='relu'))
        model.add(core.Dropout(0.5))

        model.add(core.Dense(500, activation='softmax'))

In [9]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

In [10]: model.summary()

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 49, 49, 32)        416
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 47, 47, 20)        5780
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 20)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 23, 23, 20)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 10580)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1354368
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               64500
=================================================================
Total params: 1,425,064
Trainable params: 1,425,064
Non-trainable params: 0
_________________________________________________________________

In [12]: model.fit(train, target, batch_size=32, epochs=750)

Epoch 487/500
6047/6047 [==============================] - 20s 3ms/step - loss: 1.2869 - acc: 0.7620
```



Code Walkthrough

Performance Evaluations



Model 1: CNN with 100 images in each class (Total 500 classes)

- 3 Convolution Layers, No Dropout
- Train - 0.22%
- Test - 0.02%

Model 2: CNN with 50 images in each class (Total 300 classes)

- 3 Convolution Layers
- Train - 51%
- Test - 30%

Model 3: CNN with 50 images in each class (Total 500 classes)

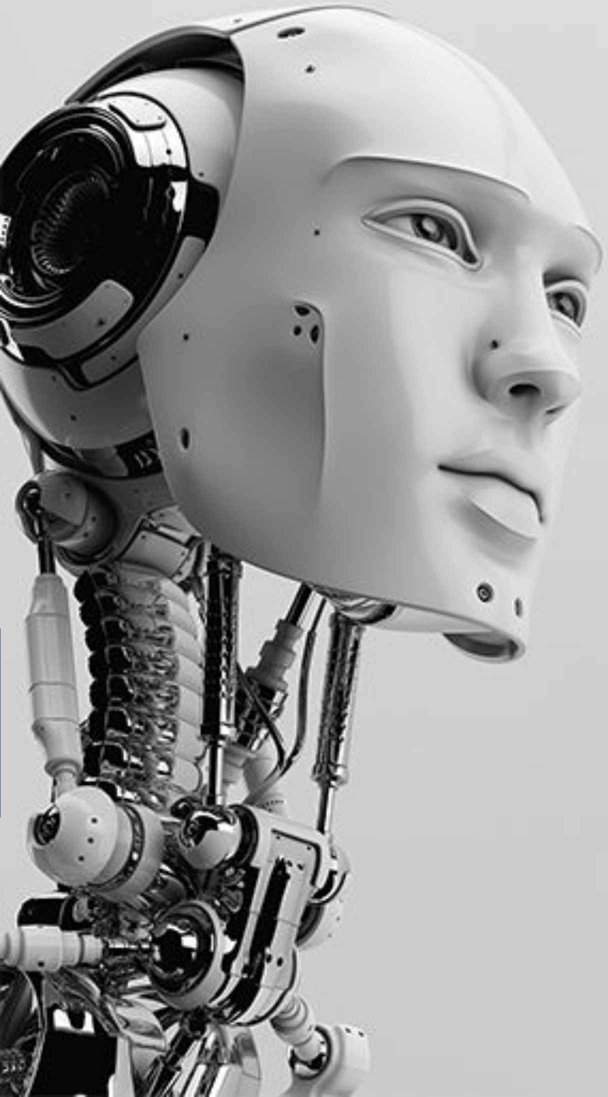
- 3 Convolution Layers
- Train - 55%
- Test - 35%

Model 4: CNN with 12 images in each class (Total 500 classes)

- 2 Convolution Layers, 200 epochs
- Train - 61%
- Test - 55%

Model 5: CNN with 12 images in each class (Total 500 classes)

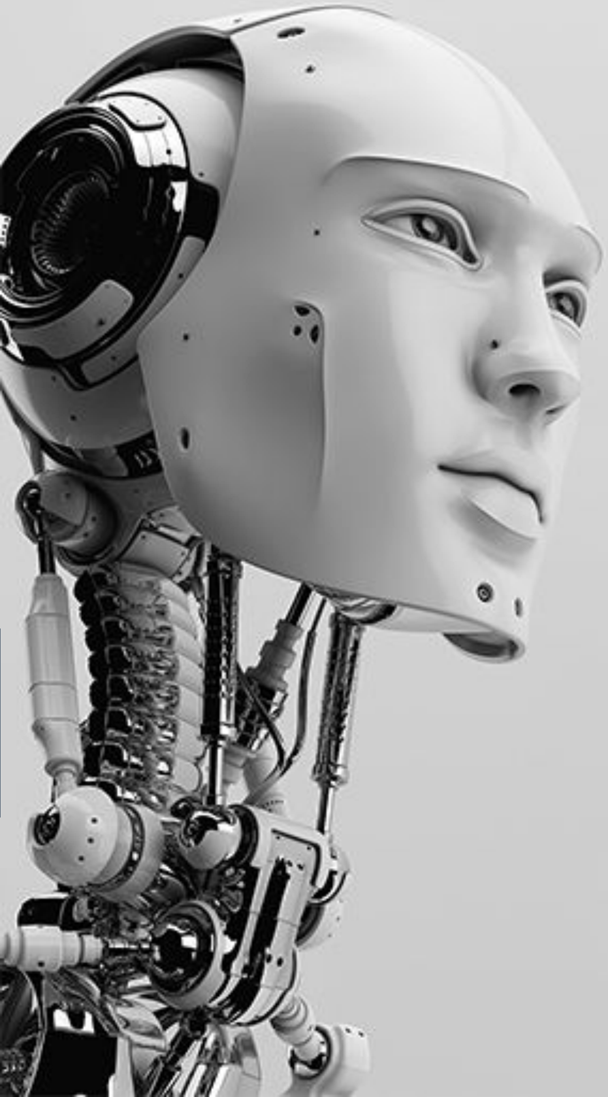
- 2 Convolution Layers, 750 epochs
- Train - 84%
- Test - 75%

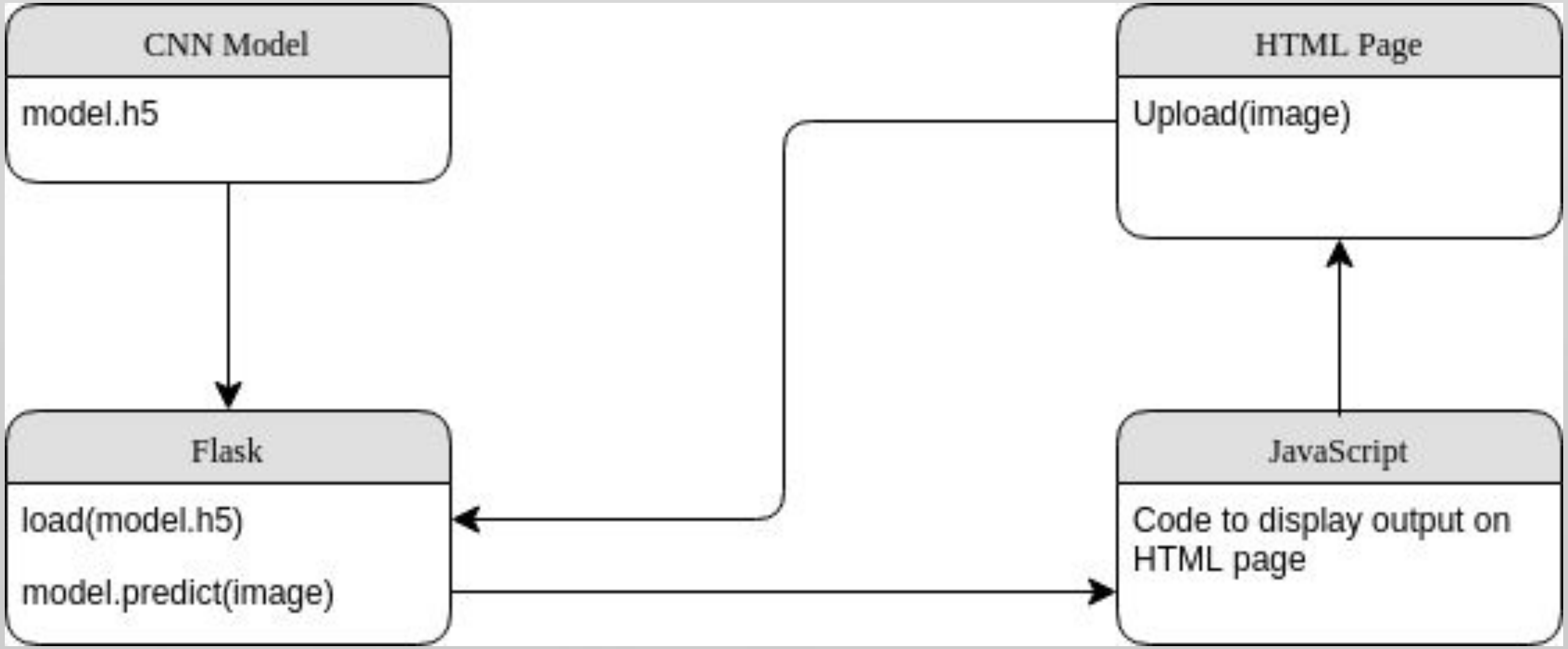


The Deployment Phase

- Flask (Web Framework)
- HTML (creating web pages)
- Java Script (For interactive web pages)
- Saved Model weights (For predictions/reusing for another application)

Deployment flow





CNN Model

model.h5

HTML Page

Upload(image)

Flask

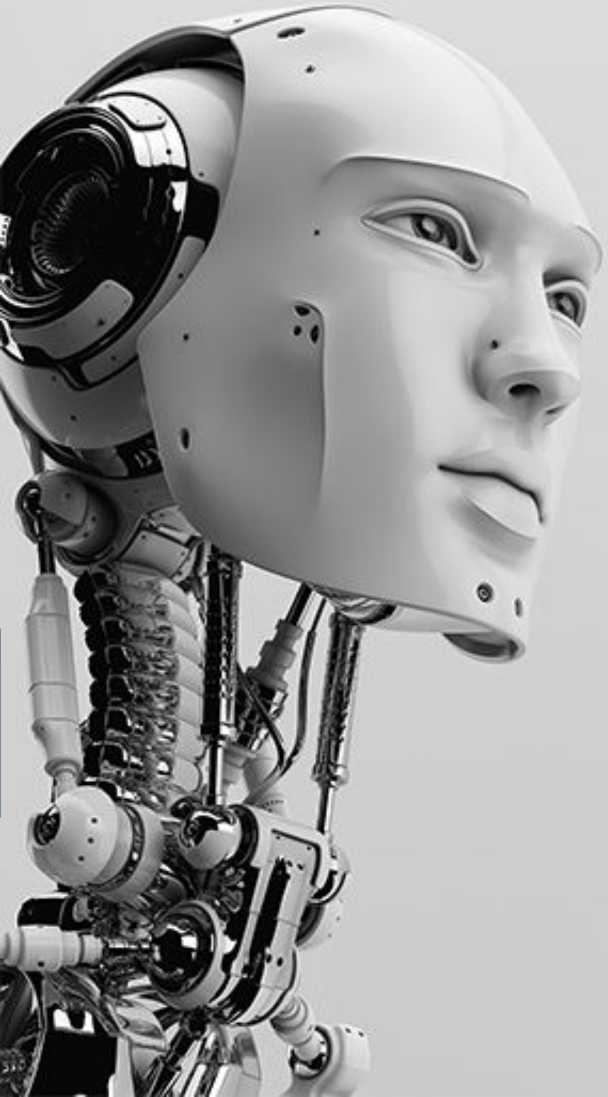
load(model.h5)

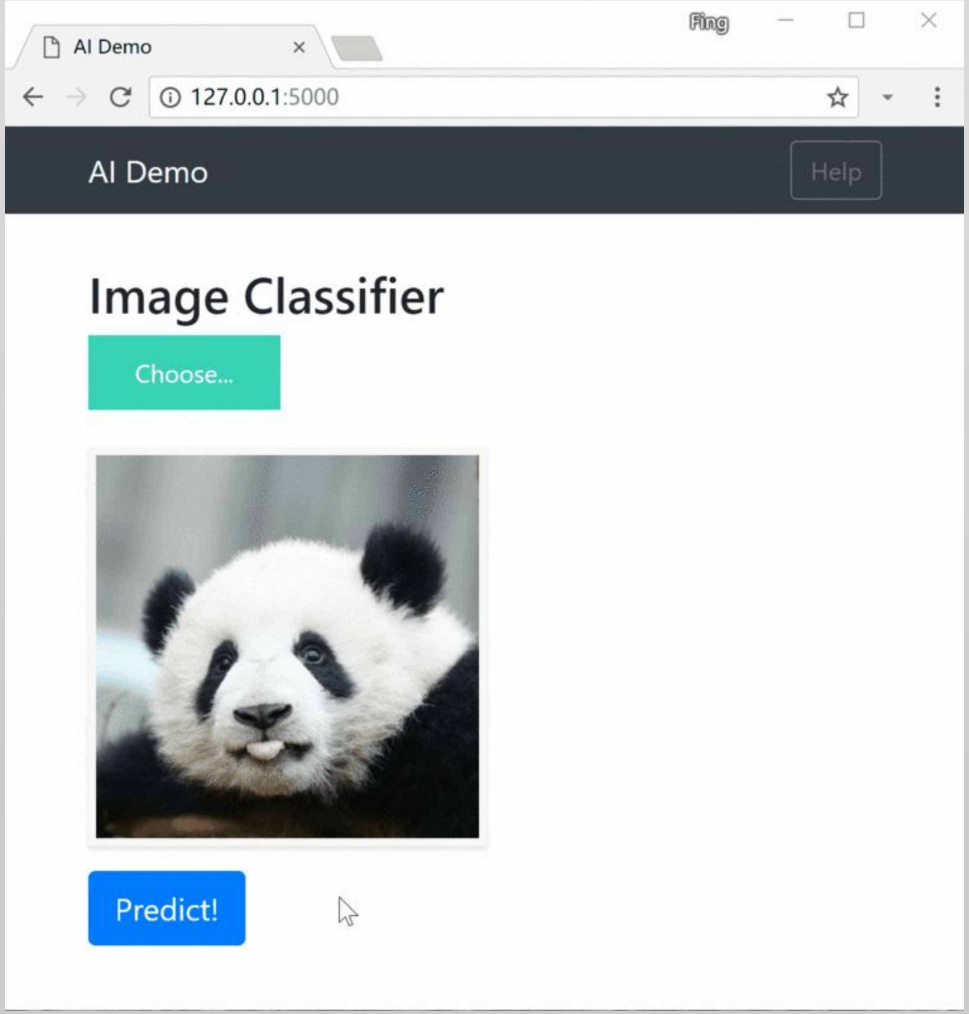
model.predict(image)

JavaScript

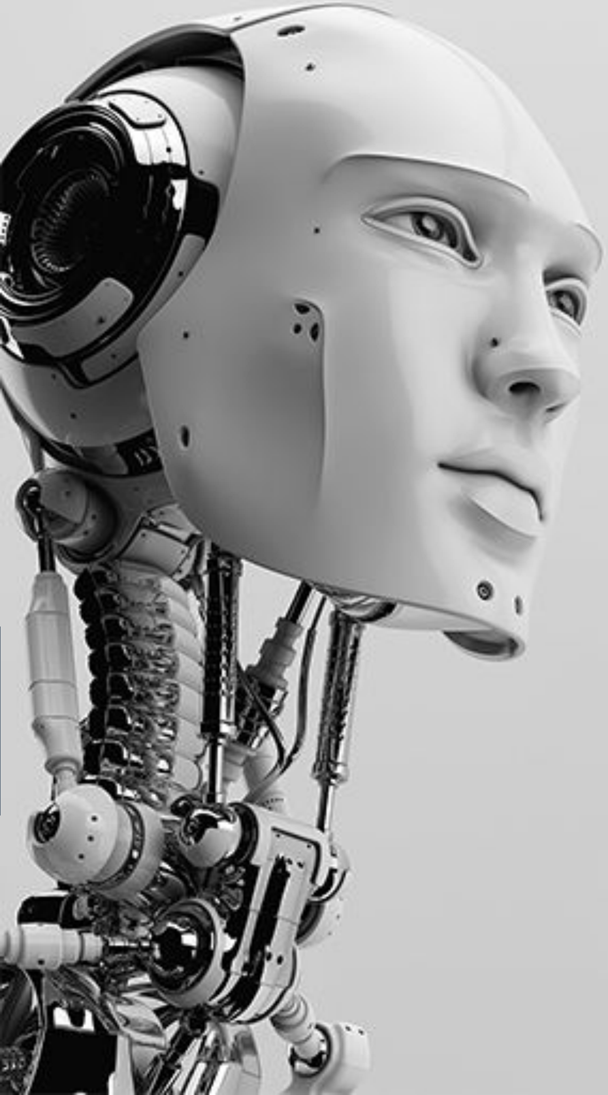
Code to display output on HTML page

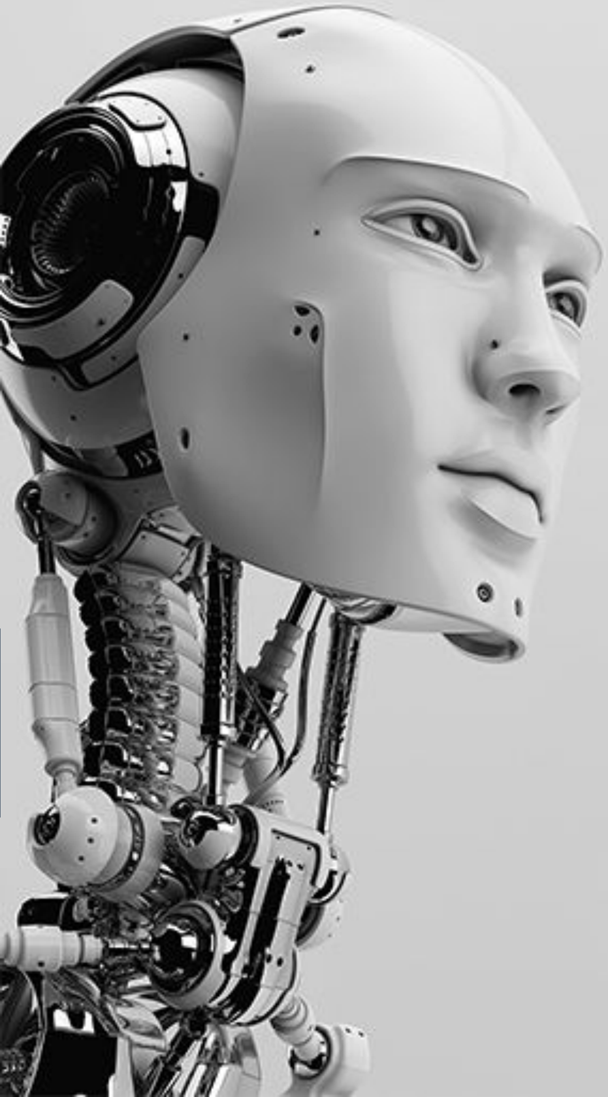
Deployment Example





Deployment Code Walkthrough

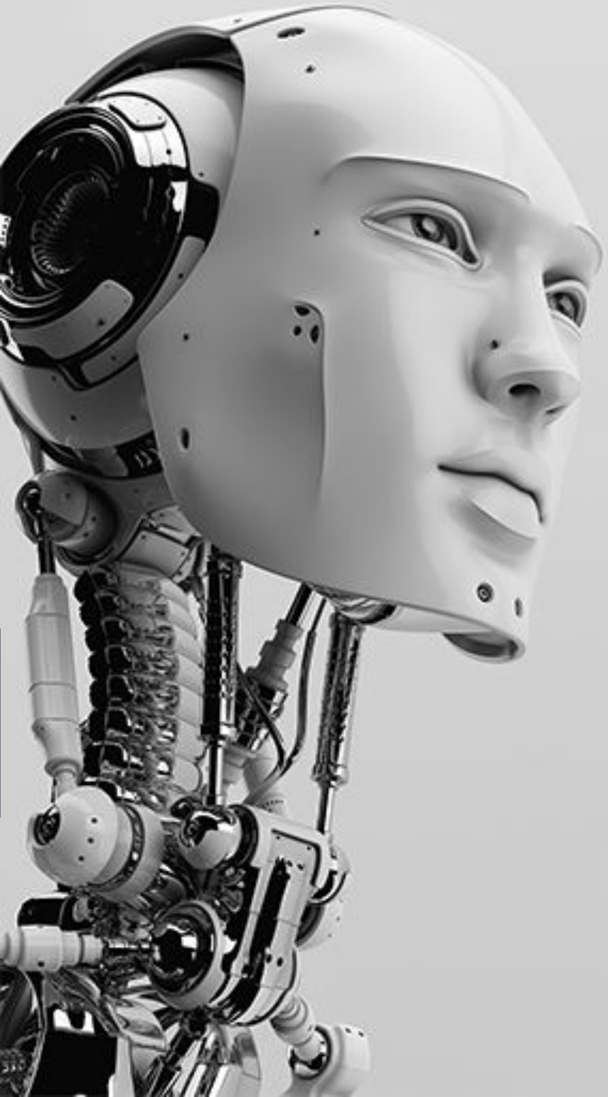


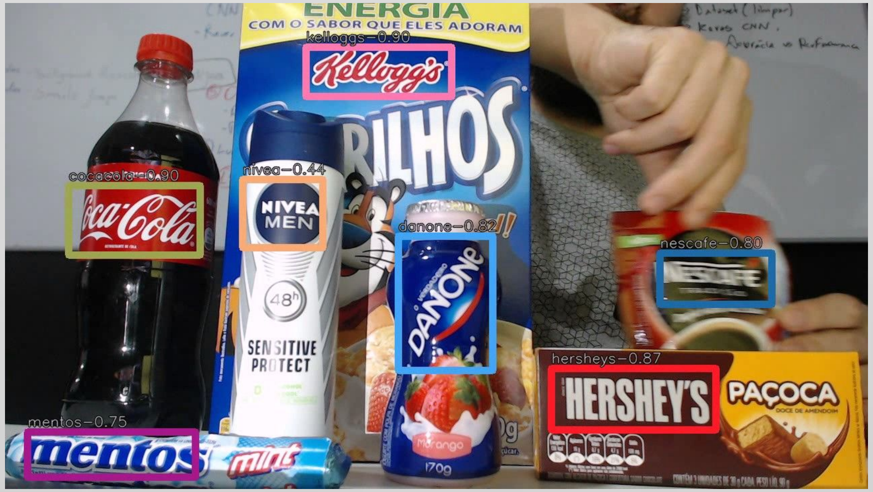


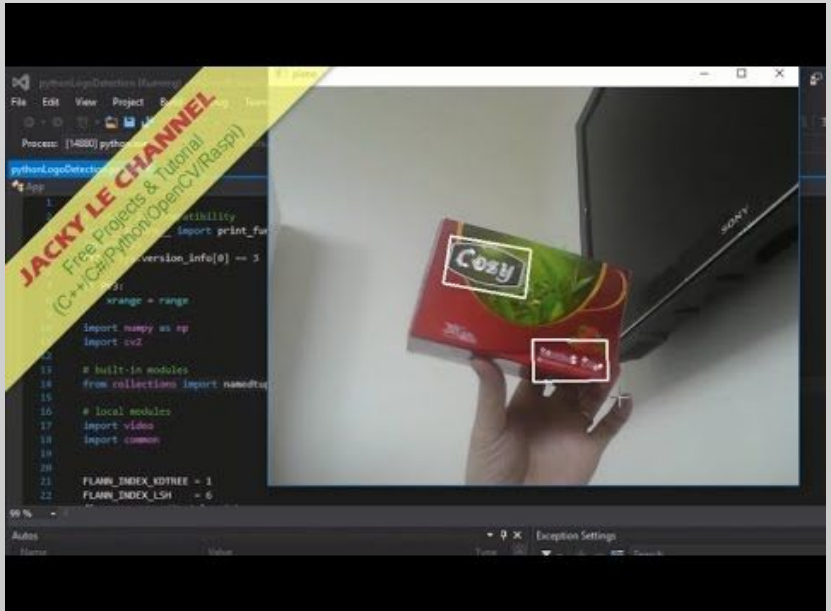
Future Scope

- Gather more data (more images for each class)
- Improving the prediction accuracy by adding more layers
- **Diverse Applications:**
  - **Virtual:** API to detect all the logos in a live video/image (opencv)
  - **Physical:**
    - Can attach a camera to the application such that if we draw a logo in a Sheet & click the capture button - it processes the image and implements the same work-flow
    - On the other hand, we can attach a ‘Digital Graphic Drawing Pad/Tablet’ to the system and draw the logo [which most professionals do], they can send that digital logo to the application and get the results

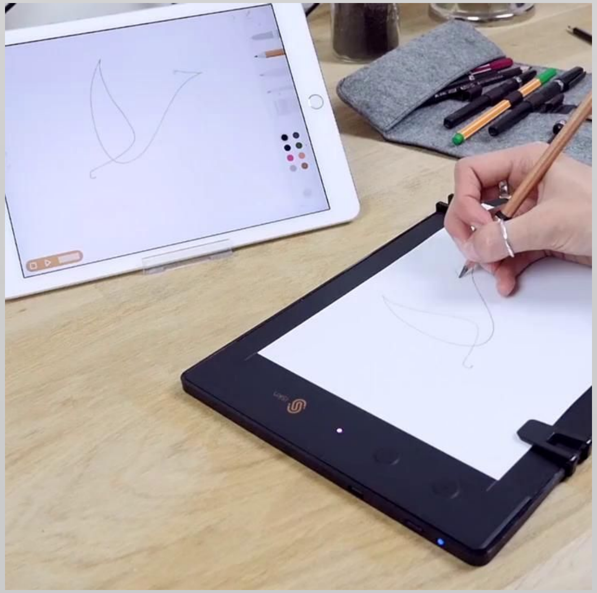
Future Scope

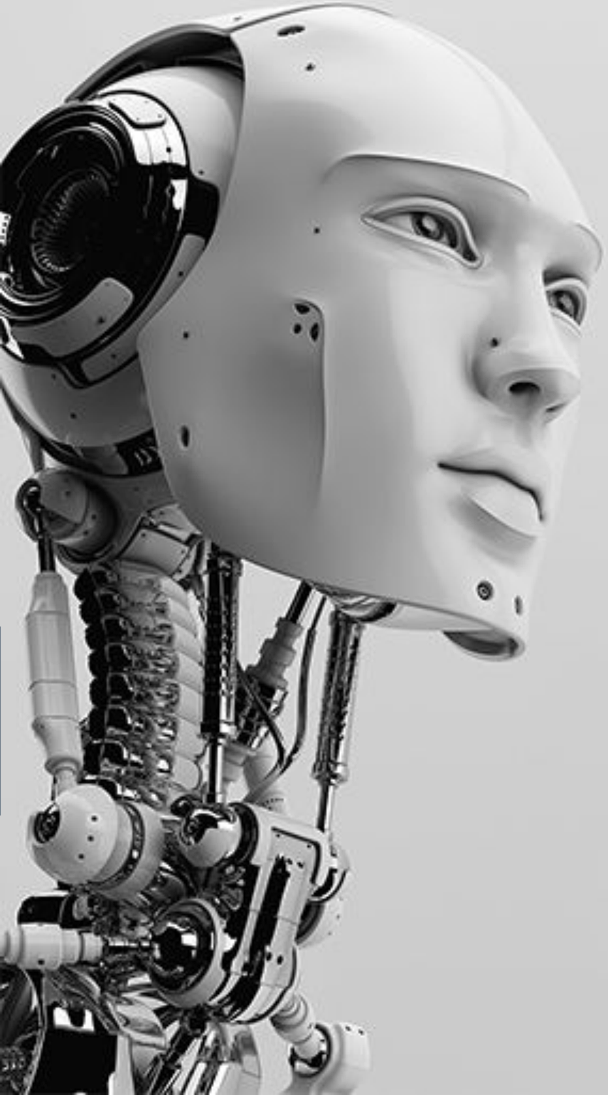






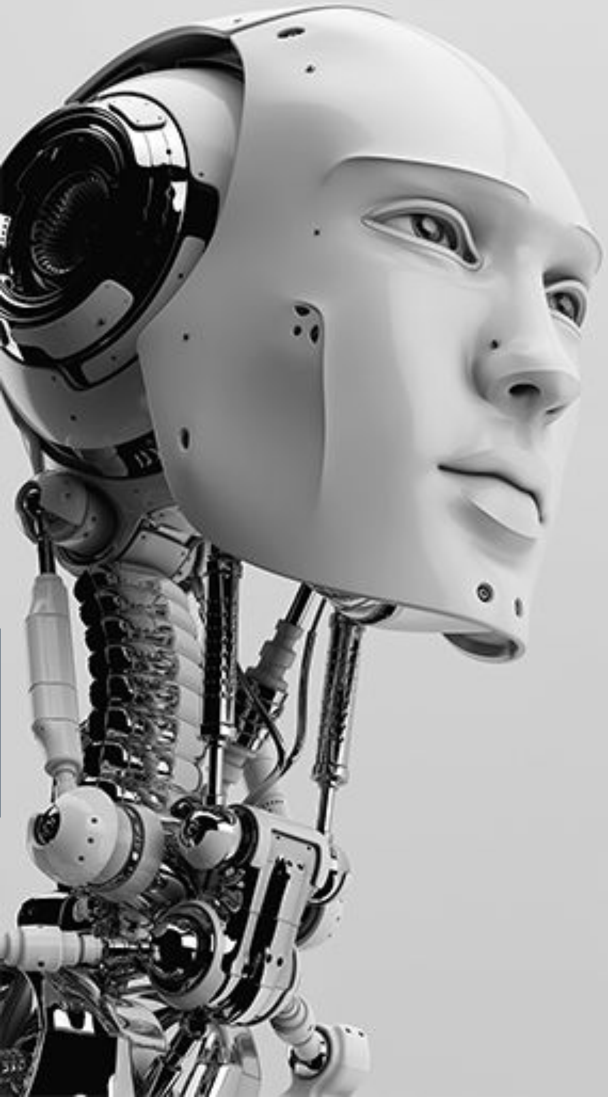
```
atibility
_ import print_fu

    version_info[0] == 3

3:
    xrange = range

import numpy as np
import cv2

# built-in modules
from collections import namedtu

# local modules
import video
import common


FLANN_INDEX_KDTREE = 1
FLANN_INDEX_LSH    = 6
```





Conclusion

- Achieved ~80% Accuracy on 500 classes with 12 images in each class, there is scope of further improvement.
- Scope of including more images for each class and increase the number of classes.
- Manually remove noise and train on ideal set of images for each class (requires laborious effort)



References

https://github.com/satojkovic/DeepLogo

https://www.upcounsel.com/trademark-infringement-penalties

https://keras.io/

https://github.com/mtobeiyf/keras-flask-deploy-webapp

https://pypi.org/project/google-images-download/1.0.1/

stackoverflow

google