Notes

- To check the compatibilty of tensorflow with python you can do it here:

https://www.tensorflow.org/install/pip

- If your python version is compatible to tensorflow simply create a virtual enviroment with the command:

```
virtualenv whatimage-back
```

- To check the directory of python write in the terminal:

```
which python
```

Applying your own CNN

- This section requires for you to have your own CNN or have the knowledge on how CNN work

- The example presented in this section will be based on the cifar 10 dataset:

https://www.cs.toronto.edu/~kriz/cifar.html

1) Save the model

First we need to save the model that we will load in django.

If you have your own CNN you can simply save the model with the code below:
`model.save('my_CNN_model.h5')`

2) Code for models.py

Now we need to load the model and use it in django. We also need to get the labels for the cifar 10 dataset.

```
from django.db import models
import pandas as pd
import numpy as np
import keras
import tensorflow as tf
from keras.preprocessing.image import img_to_array
from django.conf import settings
from keras.preprocessing import image
from tensorflow.keras.models import load_model
import os
from tensorflow.python import ops
```

```
# Create your models here.


class Image(models.Model):
    picture = models.ImageField()
    classified = models.CharField(max_length=10, blank=True)
    uploaded = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Image classified as {self.uploaded.strftime('%Y-
%m-%d %H:%M')}"

    def save(self, *args, **kwargs):
        LABELS = ['airplane', 'automobile', 'bird', 'cat',
                  'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
        img = image.load_img(self.picture, target_size=(32, 32))
        img_array = image.img_to_array(img)
        to_pred = np.expand_dims(img_array, axis=0)
        try:
            file_model = os.path.join(settings.BASE_DIR, 'img-
class-model.h5')
            graph = ops.get_default_graph()

            with graph.as_default():
                model = load_model(file_model)
                pred = LABELS[model.predict_classes(to_pred)[0]]
                self.classified = str(pred)
                print(f'classified as {pred}')
        except:
            print('failed to classify')
            self.classified = 'failed to classify'
        super().save(*args, **kwargs)
```