CMPE 255 - Bonus Work
Satyadivya Maddipudi
Student No. 016011775

Options Selected:

1. Inference using OpenVINO
2. Serving with REST APIs

Datasets used:
For OpenVINO:
https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
For Serving with REST APIs:
https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

Colab link:
https://colab.research.google.com/drive/1mIt1tyurHxBYxHoZezljbQsHAYQwpkqa#scrollTo=xFQPBYE0UfPs

Github link:
https://github.com/DivyaMaddipudi/CMPE-255

1. **Inference using OpenVINO**

   Once the tensorflow model is trained we have to convert that model with OpenVINO model optimizer.

   Command For model optimize:

   ```
   mo_command = f"""mo
                    --saved_model_dir "{model_fname}"
                    --input_shape "[1,180,180,3]"
                    --data_type "{ir_data_type}"
                    --output_dir "{model_fname}"
                    --model_name "{ir_model_name}"
                    """
   ```

   Now run the model optimizer:

   ```
   # Run the Model Optimizer (overwrites the older model)
   print("Exporting TensorFlow model to IR... This may take a few minutes.")
   mo_result = %sx $mo_command
   print("\n".join(mo_result))
   ```

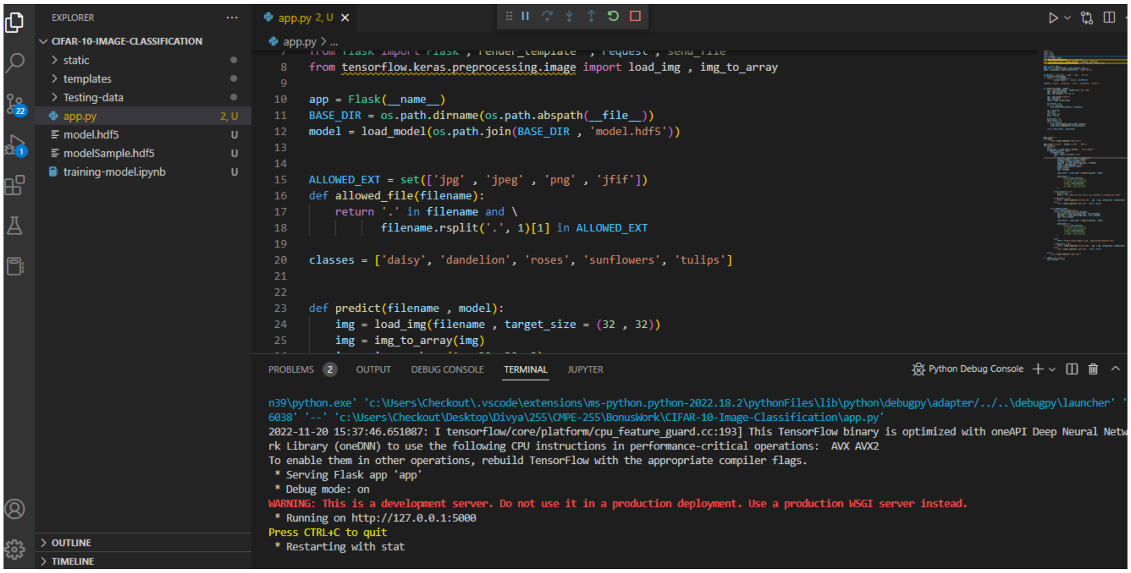
This will create two files ending with .bin and .xml extension.

Once those files are generated, start the OpenVINO inference setup and run it.

2. Serving with REST APIs

For this I use the CIFAR10 dataset, here I trained a model using tensorflow which classifies the given image and wrote endpoints using flask. React is used in the front-end.

Once the model is saved and downloaded as a hdf5 file, use flask to serve the backend and link to the model file. Then run the app.py file.



```python
from flask import Flask , render_template , request , send_file
from tensorflow.keras.preprocessing.image import load_img , img_to_array

app = Flask(__name__)
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
model = load_model(os.path.join(BASE_DIR , 'model.hdf5'))


ALLOWED_EXT = set(['jpg' , 'jpeg' , 'png' , 'jfif'])
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXT

classes = ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']


def predict(filename , model):
    img = load_img(filename , target_size = (32 , 32))
    img = img_to_array(img)
```

```
n39\python.exe' 'c:\Users\Checkout\.vscode\extensions\ms-python.python-2022.18.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '
6038' '--' 'c:\Users\Checkout\Desktop\Divya\255\CMPE-255\BonusWork\CIFAR-10-Image-Classification\app.py'
2022-11-20 15:37:46.651087: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Netw
rk Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
```

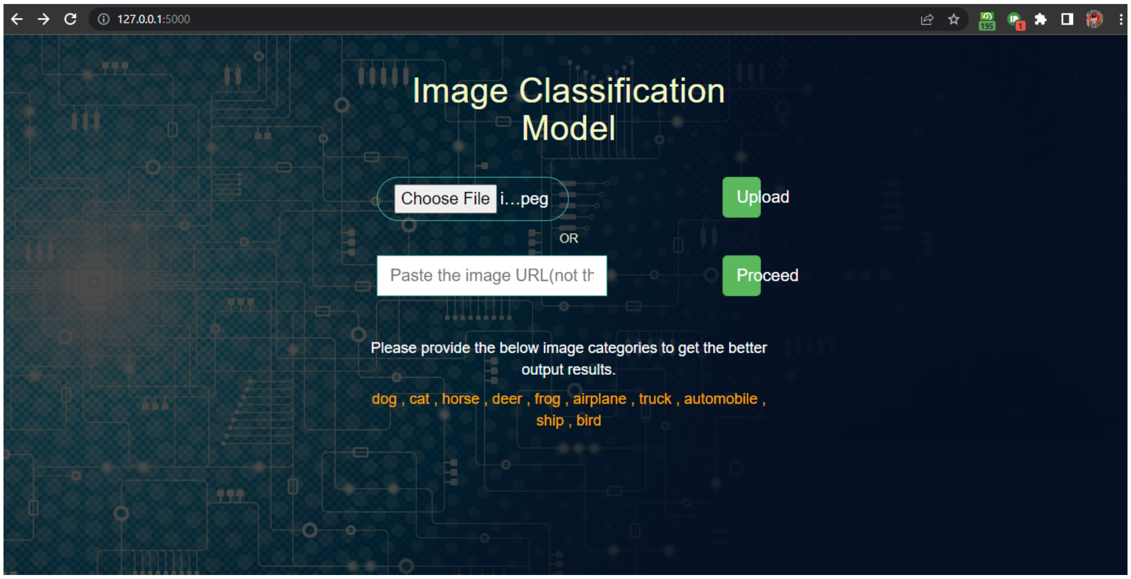
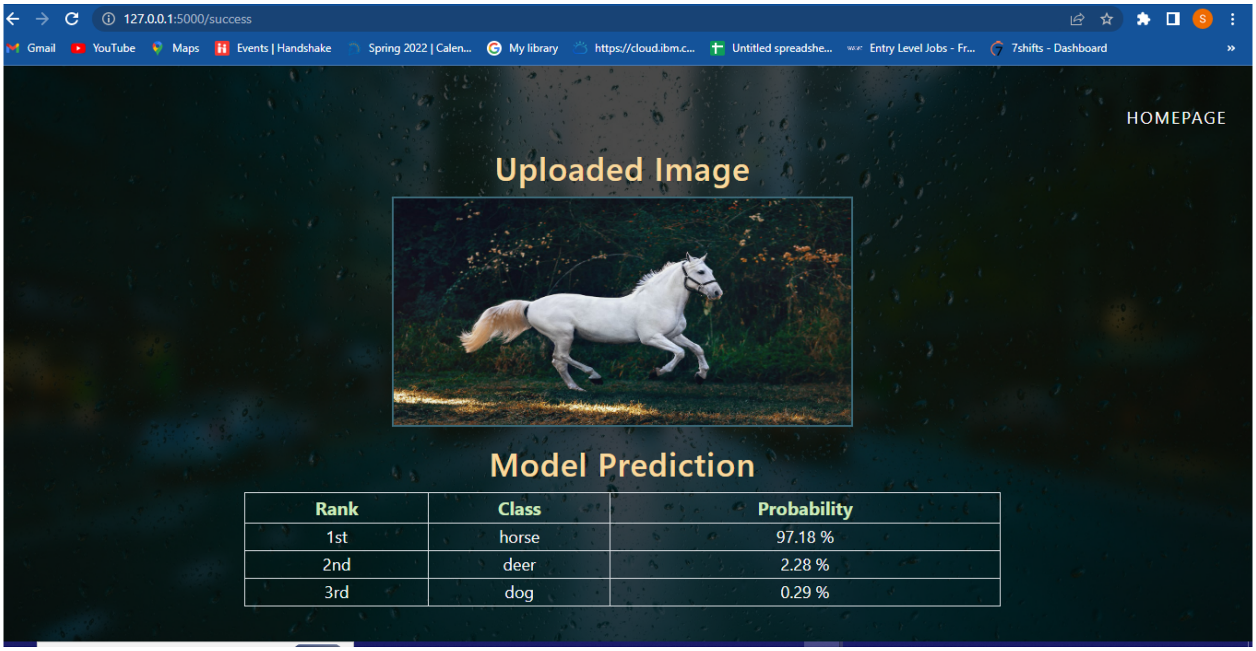


Image Classification Model

Please provide the below image categories to get the better output results.

dog , cat , horse , deer , frog , airplane , truck , automobile , ship , bird

Output:



Uploaded Image

Model Prediction

Rank	Class	Probability
1st	horse	97.18 %
2nd	deer	2.28 %
3rd	dog	0.29 %