CMPE Bonus Work 1

Farazuddin Mohammad (016176836)

1 Problem Statement

To perform Training and Inference on the Fast Semantic Segmentation model built using the Pytorch framework. The model inference is performed using the Intel's OpenVINO Framework. Create a Web Server which has the Pytorch Framework's pretrained model i.e DenseNet121 for Image Classification and expose the model inference as a REST API's POST method.
At the End we will do a performance comparison in terms of inference time with the Intel OpenVINO and just the original Pytorch Models.

2 Data Set Info

For the Fast Sematic Segmentation we have selected the Cityscapes dataset, the dataset contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities, with high quality pixel-level annotations of 5000 frames in addition to a larger set of 20000 weakly annotated frames. The dataset is thus an order of magnitude larger than similar previous attempts. The Cityscapes Dataset is intended for assessing the performance of vision algorithms for major tasks of semantic urban scene understanding: pixel-level, instance-level, and panoptic semantic labeling; supporting research that aims to exploit large volumes of (weakly) annotated data, e.g. for training deep neural networks.
For the Image classification using DenseNet121 we have used the standard ImageNet Dataset with 1,000 classes and more than 10,000 training images.
Device CPU: Intel(R) Xeon(R) CPU @ 2.20GHz

3 Performing Inference on a Model

- Train and Save your Prediction Model.
- Convert PyTorch model to ONNX
- Convert ONNX Model to OpenVINO IR Format
- Load and Preprocess an Input Image
- Load the OpenVINO IR Network and Run Inference on the ONNX model
- Test inference on the ONNX Model in OpenVINO Runtime
- Test inference on the OpenVINO IR Model in OpenVINO Runtime
- Test inference on the Original Pytorch Model

Below is the Code to Convert any trained Pytorch model to ONNX format.

```
import torch

# Instantiate your model. This is just a regular PyTorch model that will be
    ↪ exported in the following steps.
model = SomeModel()
# Evaluate the model to switch some operations from training mode to inference.
model.eval()
# Create dummy input for the model. It will be used to run the model inside
    ↪ export function.
dummy_input = torch.randn(1, 3, 224, 224)
# Call the export function
torch.onnx.export(model, (dummy_input, ), 'model.onnx')
```

4 Pytorch Segmentation Model Inference

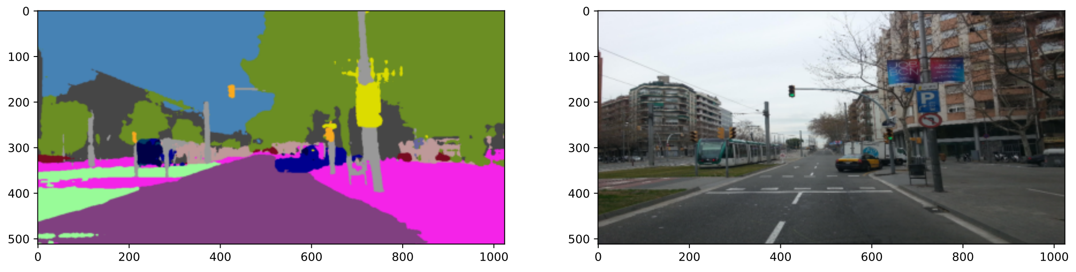


Figure 1: Original Pytorch Model Prediction

5 Inference ONNX Model in OpenVINO Runtime

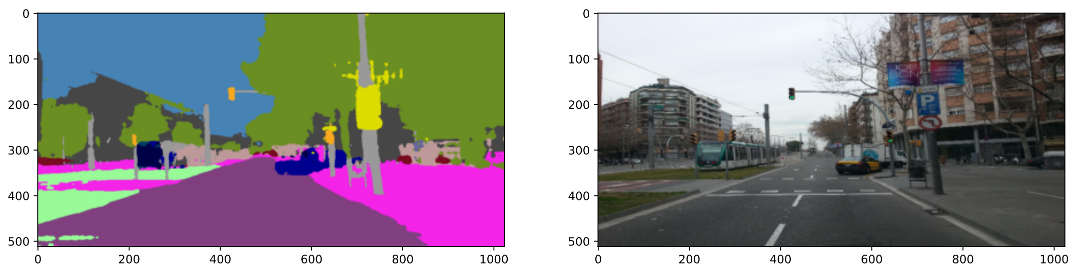


Figure 2: ONNX Model in OpenVINO Runtime Inference

6 Inference OpenVINO IR Model in OpenVINO Runtime

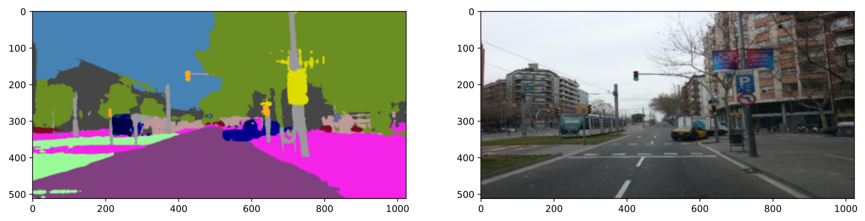


Figure 3: OpenVINO IR Model in OpenVINO Runtime Inference

7 Inference Performance Comparison

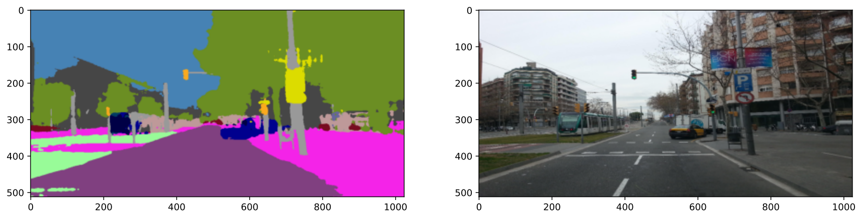


Figure 4: OpenVINO IR Model in OpenVINO Runtime Inference

8 Inference Performance Comparison

https://drive.google.com/file/d/1b5twCa8T44pWy6cTvclkcAT1L9ozdrrc/view?usp=sharing

9 Flask Application

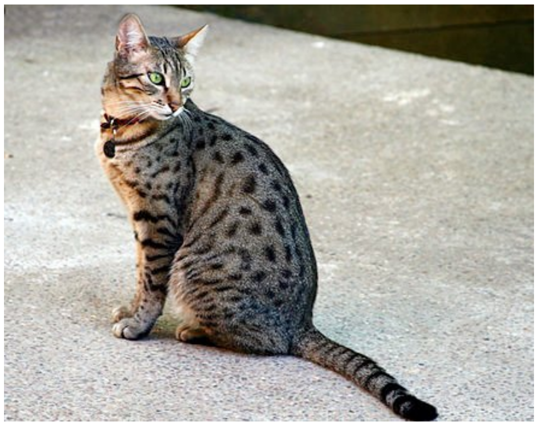
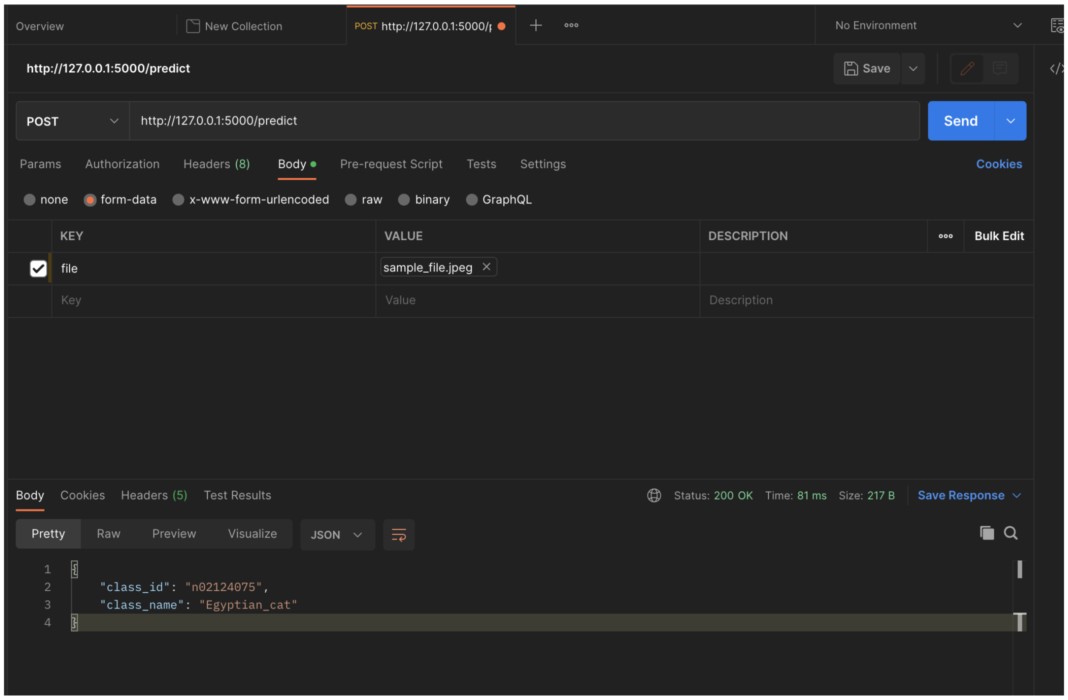


Figure 5: Input Image



POST http://127.0.0.1:5000/predict

Body: ● form-data

KEY	VALUE	DESCRIPTION
✓ file	sample_file.jpeg	
Key	Value	Description

Status: 200 OK Time: 81 ms Size: 217 B

JSON

```
{
    "class_id": "n02124075",
    "class_name": "Egyptian_cat"
}
```

Figure 6: Flask Applications POST Request