[Face detection]

[Activity Report No.3]

Radu Beche

[In this report I will present you how I have successfully trained my model for detecting faces. Also, I will speak about how can a model be optimized so it can be deployed into the cloud]

[Face Detection]

[Activity Report]

What's new since last time?

Since last time I have partially trained the model on the WIDER face dataset. **The model is not yet completely trained thus it is a big dataset and requires a lot of computing time**. Waiting for the model to be trained, I have researched for tools and ways to be able to deploy my model to the cloud.

Dataset description

The Wider Face dataset is a face detection benchmark dataset, of which images are selected from the publicly available data. It contains **32,203** images and label **393,703** faces with a high degree of variability in scale, pose and occlusion as depicted in the sample images. The faced are categorized into many postures and poses.
Sample images:

Deploying a deep learning application

Models for deep learning have a very large memory footprint (over >50 mb usually) and requires a lot of computing power(TOPS).

[Project Roadmap]

- Research
- Training on dummy dataset
- Training on a face dataset
- Deploy application

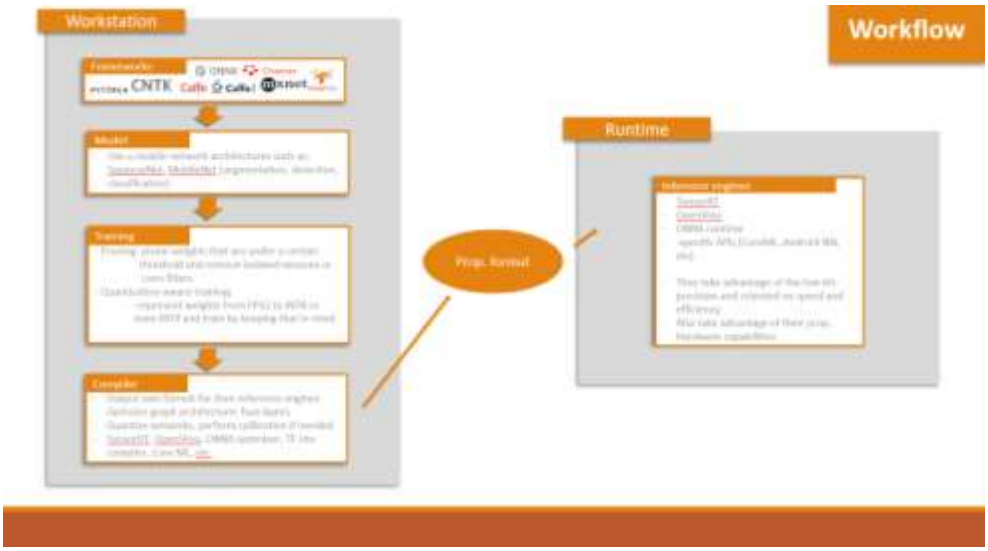

Figure 1. Workflow for developing NN applications

For optimizing the inference time, I will using an NVIDA tool named TensorRT.

NVIDIA TensorRT is a platform for high-performance deep learning inference. It includes a deep learning inference optimizer and runtime that delivers low latency and high-throughput for deep learning inference applications. With TensorRT, one can optimize neural network models trained in all major frameworks, calibrate for lower precision with high accuracy, and finally deploy to hyperscale.

TensorRT is built on CUDA, NVIDIA's parallel programming model, and enables to optimize inference for all deep learning frameworks leveraging libraries.

TensorRT provides INT8 and FP16 optimizations for production deployments of deep learning inference applications. Reduced precision inference significantly reduces application latency, which is a requirement for many applications.

How am I going to use this?

After training the model, I will try to use this NVIDIA tool for optimizing the developed model.

PS. I have attached a presentation of what a workflow should look like for Deep learning application.