# Converting python model and running in C++ OpenCL

One important aspect of deploying machine learning application in microscope is too able to run in native code, C++. Base framework is implemented to get started with training in python and porting to native code.

While training two separate folders one with ground truth, another with original images is fed. After training a script provided would be used to convert model to graph file and model weights. Below is the training flow diagram.

Train in python

Original images

Ground truth

Export the model

Model weights .npy files

Model graph .txt file

Weights of the models are stored in .npy format, in each model the number of files depending on the number of layers used. Some layers do not have any weights they directly apply some operation on to the incoming data, such as sigmoid, relu, max pooling layers.

So far in C++ below layers are implemented.

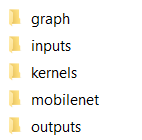
1. Batch normalization
2. Convolution layer (Stride 1X1 or 2X2)
3. Max pooling layer (to reduce the input size by half)
4. Depthwise convolution layer
   1. With bias
   2. Without bias
5. Pointwise convolution layer
6. Relu layer
7. Sigmoid layer
8. Argmax layer (for output size of 2 channels)
9. Padding layer
10. Resize, Input and preprocess layers

Above list of layers kernels written in OpenCL. There is one level of optimization performed when say Convolution layer followed by a Batch normalization layer. In this case we could merge mean and variance of batch normalization layer in convolution layer itself.

C++ native code supports running the exported model and generating the output. The command to execute the native code as follows

**SemanticSegmentation.exe <rootpath> <modelname>**

Root path should contain the following folders. For example we have to run the model named **mobilenet,** here is how the folder structure would look like.



Grpah folder contains **mobilenet.txt**

Input folder for loading input image,

as of now input image is in .npy format

OpenCL kernel of different layers

Exported model weights in .npy format

Output folder for saving the output image

The block diagram of running in native is given below.

Native C++

Model weights .npy files

Model graph .txt file

Input image .npy format

Output image .png format