Subtask 3

LeNet

To implement LeNet we have created a function called "convolute_multiple" which takes following inputs

1. float** input_layers: Representing input in a way such that input[i] gives the input matrix in row-major form from the ith channel(0 based indexing).
2. int input_width: The dimension of input_matrices in the input channels
3. int input_channels: Number of input channels
4. float***filter_layers: Represents filters in the following way. filter_layers[i] gives the ith filter(0 based indexing). filter_layers[i][j] gives the jth filter matrix in row-major order from the ith filter.
5. float* biases- gives array of biases
6. int filter_width: The dimension(number of rows/columns) of filter_matrices in the filters
7. int output_channels: Number of output channels
8. int pad_size: padding_size
9. bool toPad: boolean which determines if padding is to be done
10. bool relu: determines if relu activation has to be done on the output or not

This gives output as float** represented in the same form as float** input_layers

We have used convolute_openblas used in convolute_multiple.cpp since it was observed that it ran much faster than convolute_pthread for inputs of the given size and almost identically with convolute_mkl.cpp.

`execute.sh`

performs following tasks:

1. Takes two command line arguments. $1 = name of the image file you want to predict $2 = name of a file in which you need to temporarily store the processed data (taken from user since if we choose a default name, it may overwrite an existing file with same name)
2. Image is processed
3. "exe" file is run and the output is printed on terminal

Therefore to run LeNet implementation

```
chmod +x execute.sh
./execute.sh {digit_image} {temp_data_file_name}
```