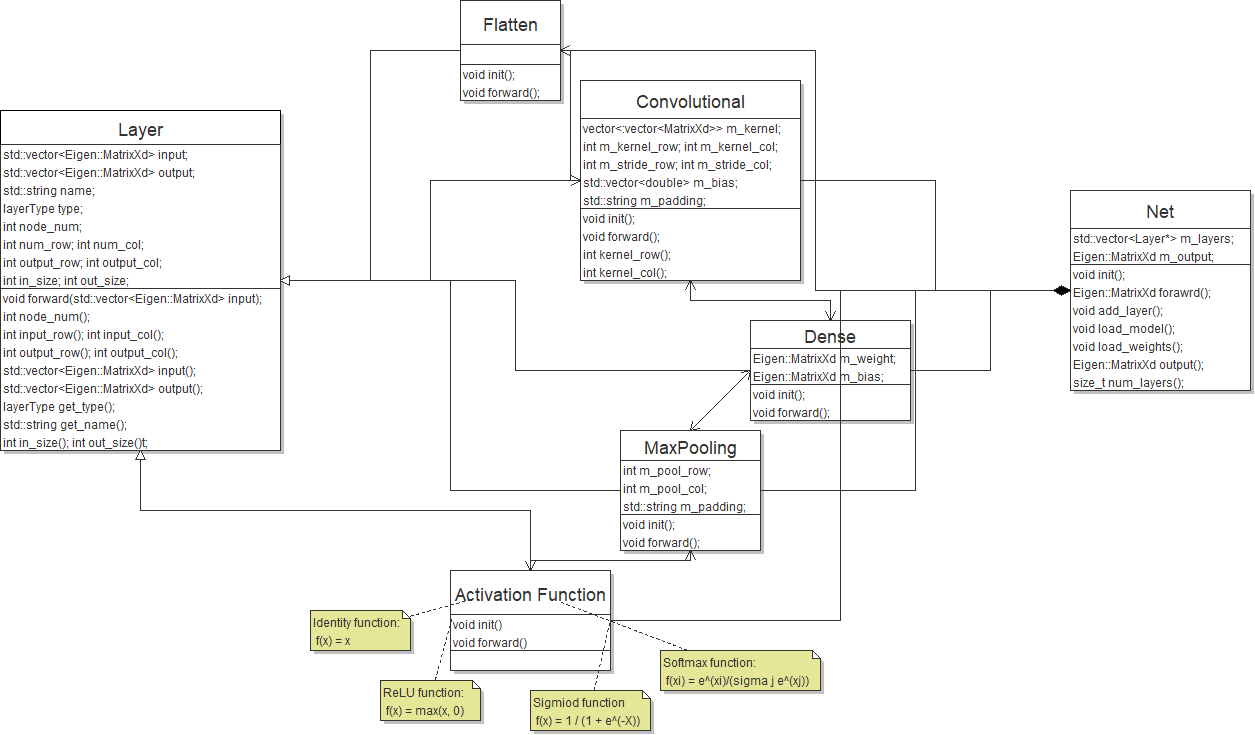
Report on CS133 Project: Neural Network Implementation

Abstract: The neural network consists of several layers, which pass the data forward to generate the result. The layers doing computation such as convolution, pooling, as well as the activation function. The network finally out put a series of possibilities, indicating how likely the number would be.

We use matrix to represent the data, which is included in the library of Eigen. One of the benefits is that there are lot of methods of matrix operation.

The framework of the neural network:



The class diagram shows the relationship between classes. A complete network contains a permutation of different layers, and all of those layers are inherited from the base layer class.

The layer class is designed to be the base class for all types of layer, containing 8 types in total, which are enumerated in the base class. There are two virtual function in the base class, init and forward. The former is used to initialize certain layer with given parameters, while the latter pass the data forward to next layer. The base layer contains several member variables to describe the size and shape of the input and output matrix. As those variables are protected, we also provide corresponding public getter function to reach the value from outside the class. Layers also have a name, represented by string and a type, which has been enumerated in the class definition.

The flatten layer is used to process the input data, which decrease the dimension of the input. It would transform the input data into a single column vector. The computation of the convolution is implemented in the forward function in the convolutional layer, achieved by a series of for loops. The layer of max pooling decrease the size of data, by representing four adjacent value by the largest one among them. The dense layer is fully connected, and the forward function compute the multiplication of input data and the weights, add by bias and pass it as output.

For convenience and consistence, we treat the activation function as independent layers. They just take the input and compute the value of the function and generate the output. We implemented four activation functions, identity, softmax, sigmoid and relu. The identity function returns original input as output, and softmax function calculate the proportion of the exponential value among the sum of exponential values. The sigmoid function mapped the value into an interval between 0 and 1, and the ReLU function returns the max value of input and 0.