Basic Vision

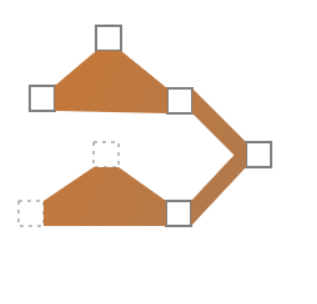
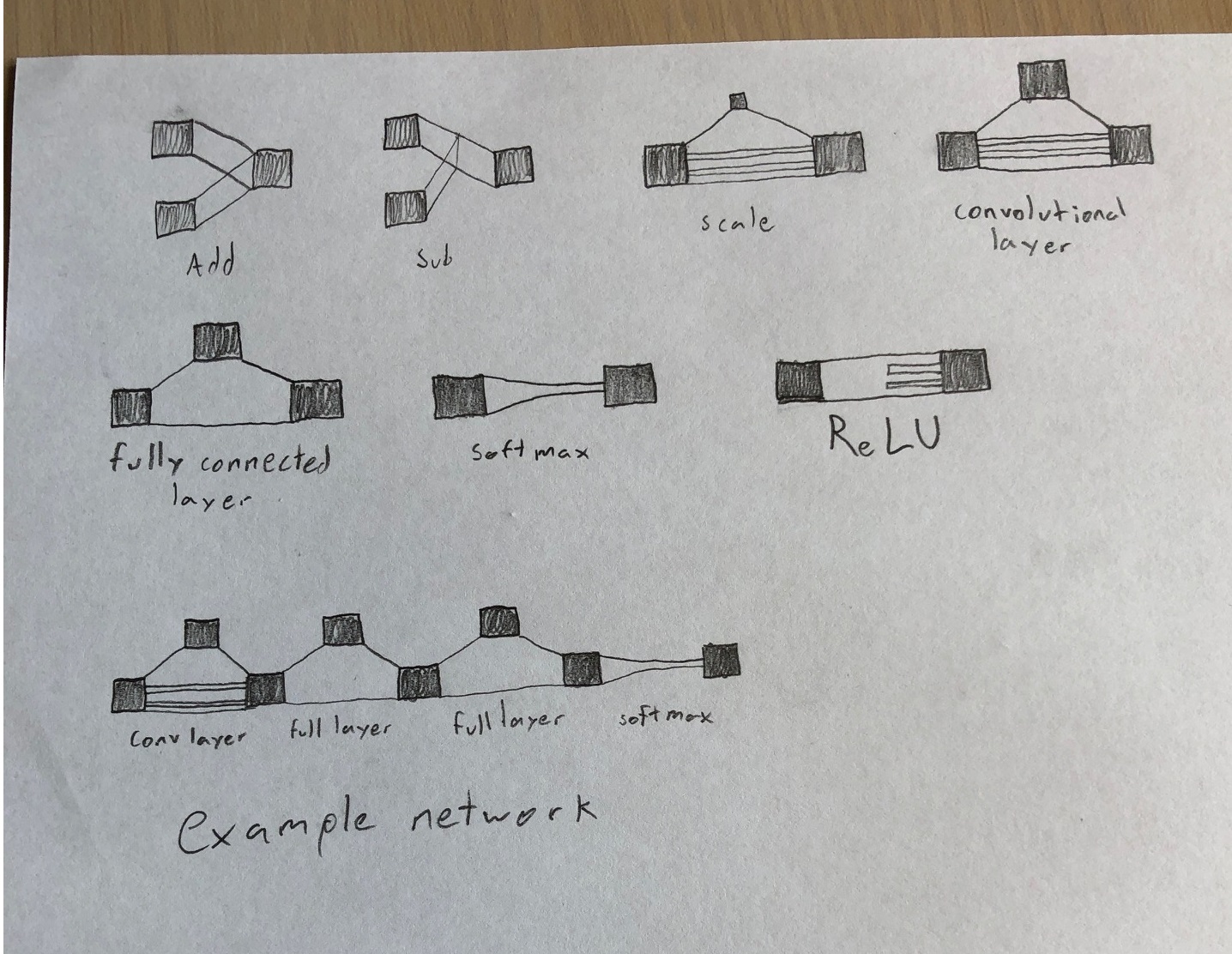
Name?:

Nous

Glia

Neurula - i like this one(nico)

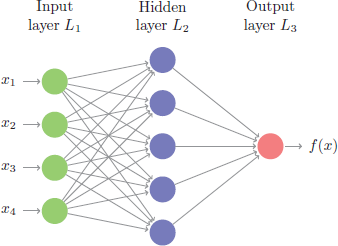
# Front end:



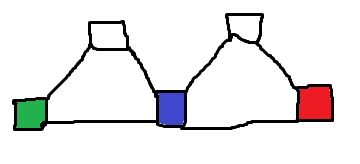
Users will drag these blocks from a side menu, onto a canvas (prob using actual html+js canvas) to design their network.

The squares are tensors, the orange connectors are the tensor operations. A tensor is basically an array of data, like a vector, or matrix, or higher dimensional equivalent.

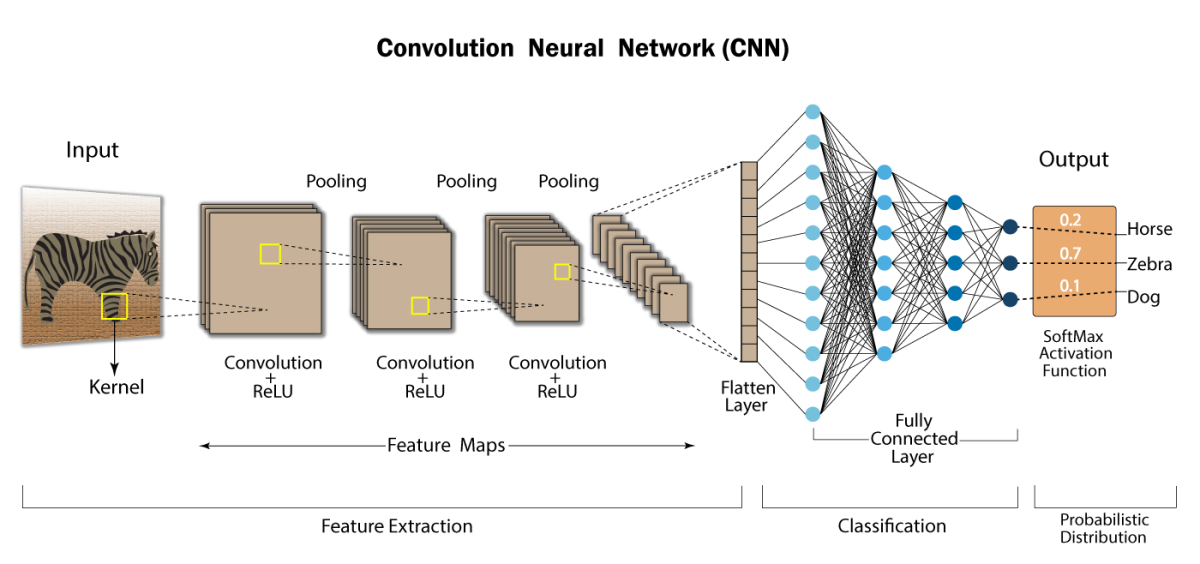
For example the following network



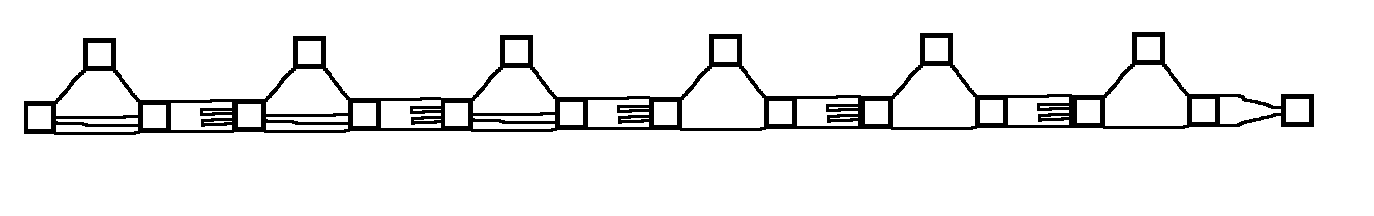
would look like this:



And a complicated network such as

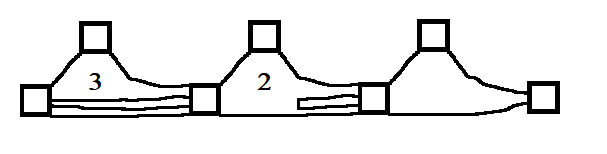


would look like



We’ll probably color the different operators to make them easier to differentiate

And eventually we’ll make abstractions so this will end up looking like



## 

## 

## 

## 

## 

## 

## 

## Backend:

Once the user presses a button that says something like “generate code”,

a representation of the network will be passed to the back end through an API

The back end will be good ol’ python.

The network will be represented as a class

Pseudo code:

class Network{

Operator[] operators; // list of operators found in the network

Tensor[] tensors; // list of operators found in the network

int[] inputs; // list of indexes of the tensors array, which tensors are inputs

int[] outputs; // list of indexes of the tensors array, which tensors are outputs

}

class Operator{

int operator\_type; // int representing the operator type

int[] inputs; // list of indexes of the tensors array, which tensors are inputs

int[] outputs; // list of indexes of the tensors array, which tensors are outputs

string s; // Extra info

}

class Tensor{

int[] shape; // dimensions of the tensor, like tensorflow shape

}

The operator types will just be an agreement on what operator each integer represents, such as

0: fully connected layer (dense)

1: Convolutional layer

2: ReLU

3: Softmax

Python functions will convert an object of this class into code, such as:

model = Sequential()

model.add(Conv2D(16, kernel\_size=5, activation='relu', input\_shape=(28,28,1)))

model.add(Conv2D(8, kernel\_size=5, activation='relu'))

model.add(Flatten())

model.add(Dense(10, activation='softmax'))