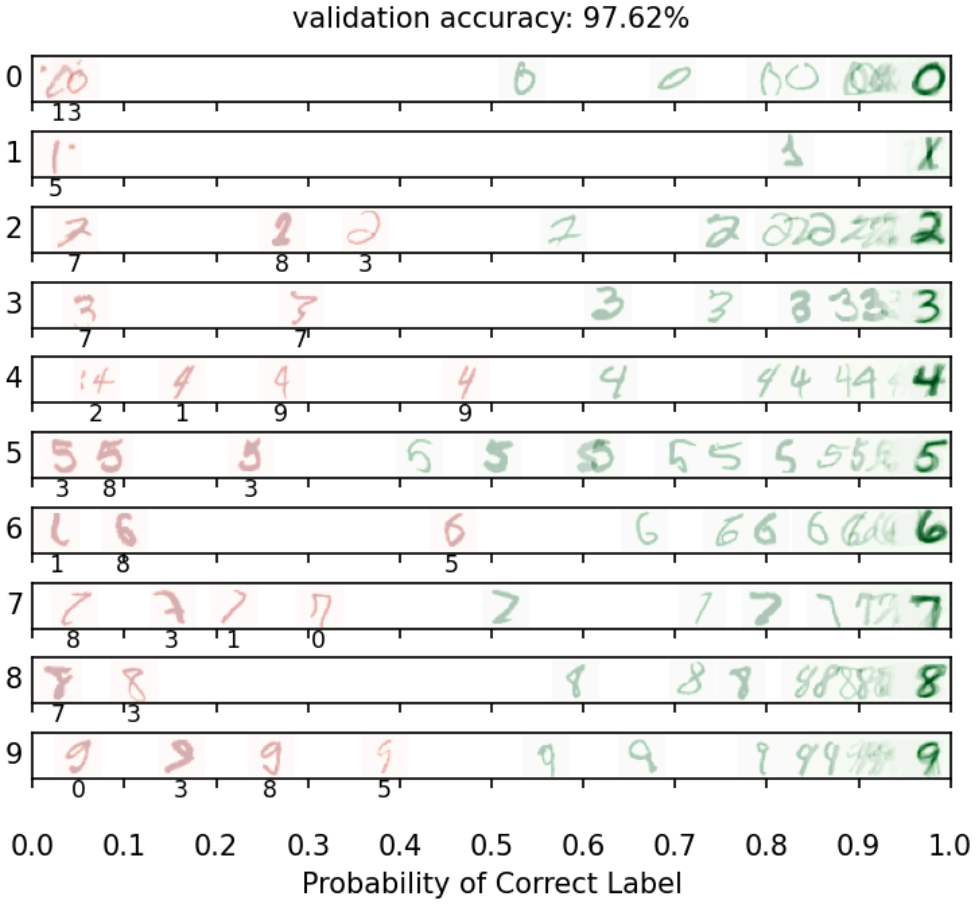
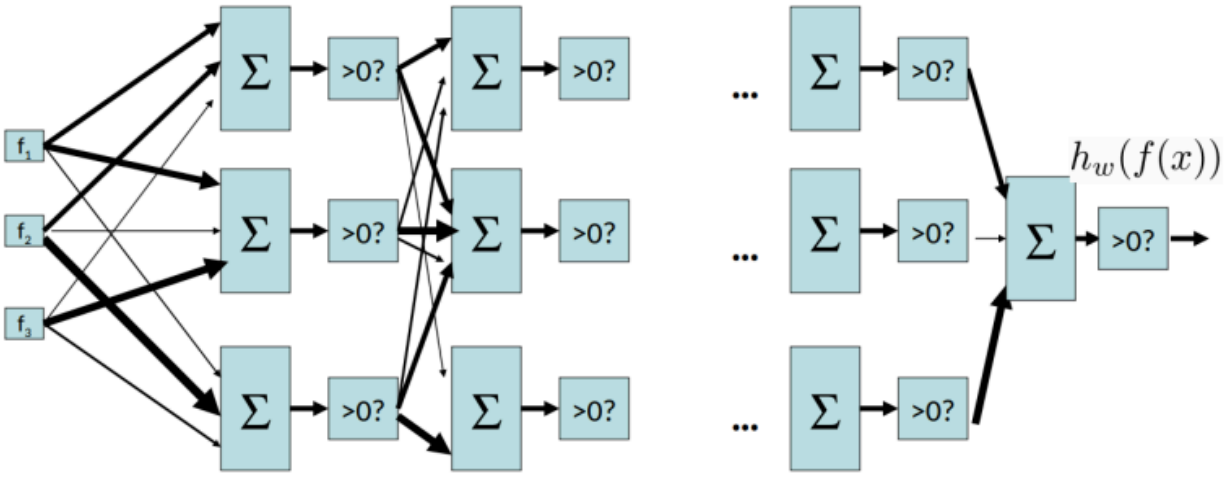
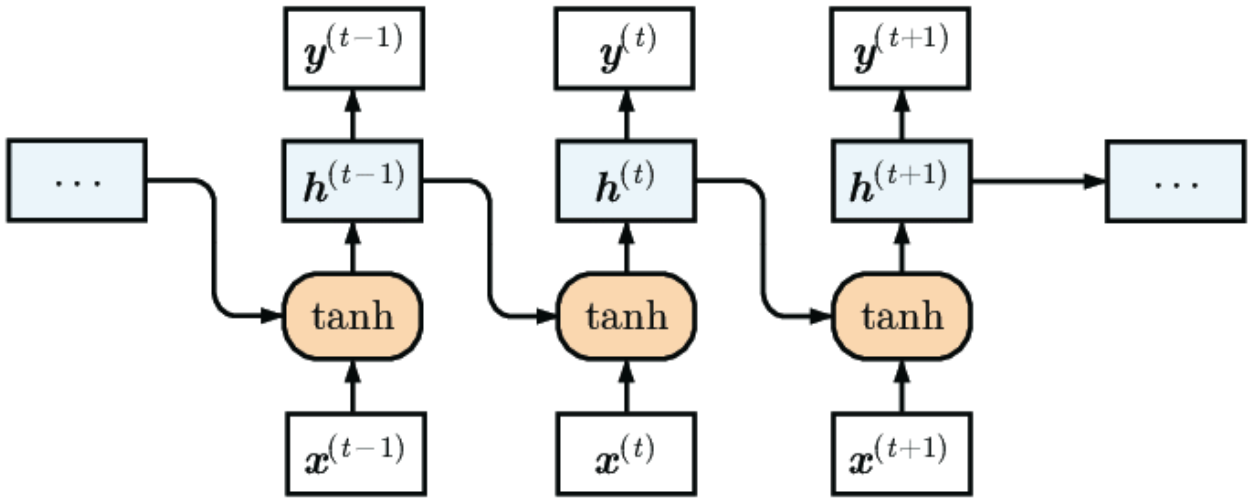
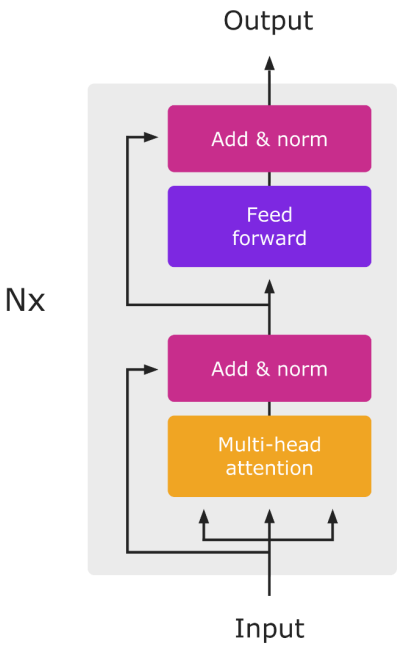

validation accuracy: 97.62%
0
1
2
3
4
5
6
7
8
9
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Probability of Correct Label


f1
f2
f3
Σ
>0?
...
$h_w(f(x))$




```python
class DigitConvolutionalModel(Module):
    """
    A model for handwritten digit classification using the MNIST dataset.

    This class is a convolutational model which has already been trained on MNIST.
    if Convolve() has been correctly implemented, this model should be able to achieve a high accuracy
    on the mnist dataset given the pretrained weights.

    Note that this class looks different from a standard pytorch model since we don't need to train it
    as it will be run on preset weights.
    """

    def __init__(self):
        # Initialize your model parameters here
        super().__init__()
        output_size = 10

        self.convolution_weights = Parameter(ones((3, 3)))
        """ YOUR CODE HERE """
        self.layer1 = Linear(676,32)
        self.layer2 = Linear(32,10)
        self.optimizer = optim.Adam(self.parameters(), lr = 0.005)

    def run(self, x):
        return self(x)

    def forward(self, x):
        """
        The convolutional layer is already applied, and the output is flattened for you. You should treat x as
        a regular 1-dimentional datapoint now, similar to the previous questions.
        """
        x = x.reshape(len(x), 28, 28)
        x = stack(list(map(lambda sample: Convolve(sample, self.convolution_weights), x)))
        x = x.flatten(start_dim=1)
        """ YOUR CODE HERE """
        return self.layer2(relu(self.layer1(x)))


    def get_loss(self, x, y):
        """
```


Output
Add & norm
Feed forward
Nx
Add & norm
Multi-head attention
Input