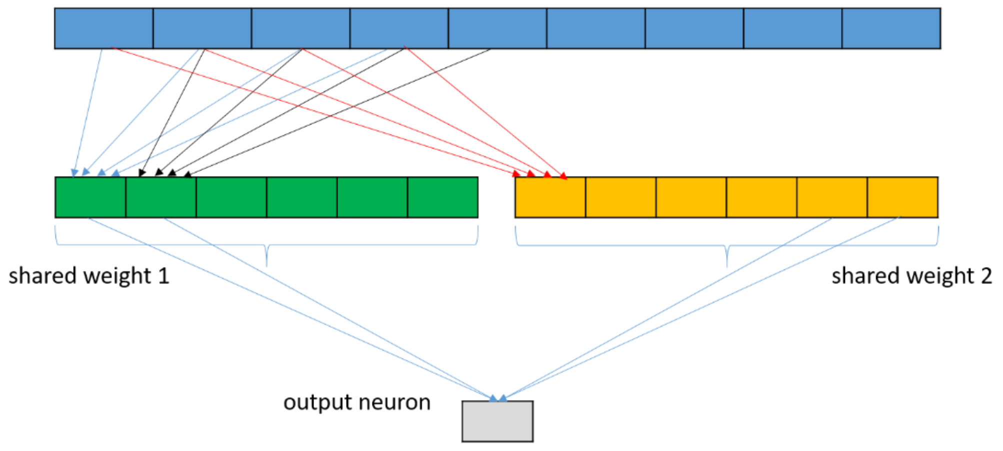
CMP_SC 8770 and ECE 8890: Spring 2019

Neural Networks

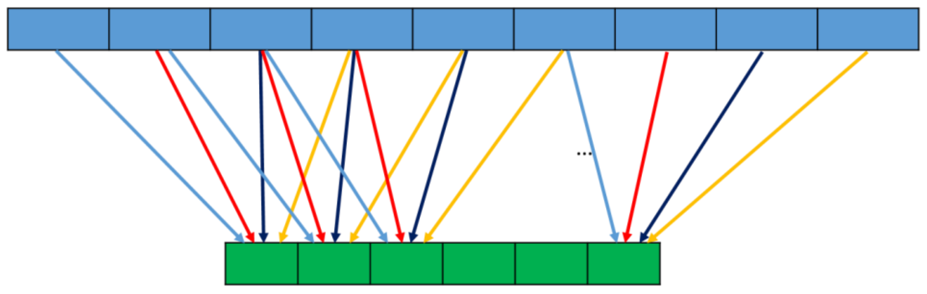
Project 1: Convolutional Neural Network (CNN)

[10%] Part 1: Validation of a simple CNN (due Feb the 19th, start of class, hand in a hard copy)

Implement the following



Here is a zoomed in view on just the first shared weight (shared weights are color coded)



Description:

1) Blue cells are inputs: 9 numbers.
2) Green cells are neurons: 6 in total, each has four weights (excluding the bias).
3) Orange cells are neurons: 6 in total, each has four weights (excluding the bias).
4) Gray is an output neuron.

Notes:

1) We only drew the first two connections (out of six) for “shared weight 1” and one for “shared weight 2” (out of six) to get the point across. It gets too hectic visually if we draw them all!
2) For “shared weight 1”, the blue and black arrows are what we are sharing. There are four more neurons with shared weights in “shared weight 1” that we did not draw (too hard to “see”).
3) We did not draw any bias terms.

Assume all neurons have i) sum aggregation and ii) Sigmoid activation with lambda = 1.

Use a learning rate (aka α) of value 0.1.

Set initial parameters of the system to

- Green shared weights

w1	w2	w3	w4	bias
0.1	-0.2	0.3	-0.4	1

- Orange shared weights

w1	w2	w3	w4	bias
1	0.4	-0.2	-0.1	1

- Weights to the output neuron

w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	bias
1	1	-1	1	-1	-1	-1	1	-1	1	-1	1	-1

Consider the following six training samples

x1	x2	x3	x4	x5	x6	x7	x8	x9	label
0	0	0	1	1.2	0.2	0.9	0.5	0	0
0.8	0.6	0.1	0.2	0	0	0	0.1	0.2	1
1	0	0	0.8	0.6	0.1	0.2	0	0	1
0	0.1	0	0	0	1	1.2	0.2	0.9	0
1	1.2	0.2	0.9	0	0	0.1	0	0.1	0
0	0.15	0.1	0	0	0.8	0.6	0.1	0.2	1

Details

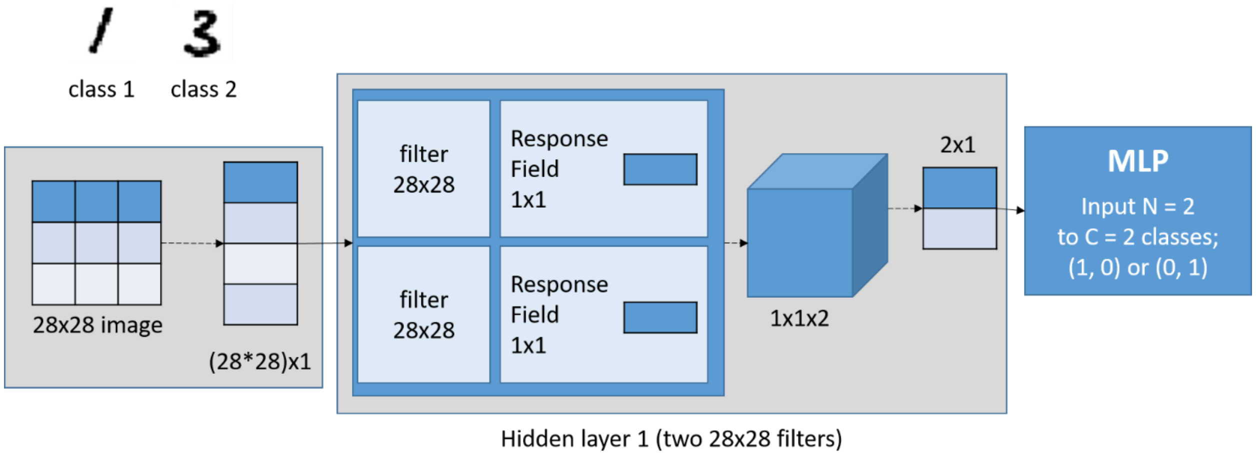
- Do not sort the data
- Do one epoch
- Do serial/online processing (i.e., no batch or mini-batch)

Report your weights at the end of the first epoch

YOU CANNOT USE ANY neural network libraries (e.g., PyTorch, TensorFlow, etc.)

YOU CAN USE a linear algebra library (e.g., NumPy)

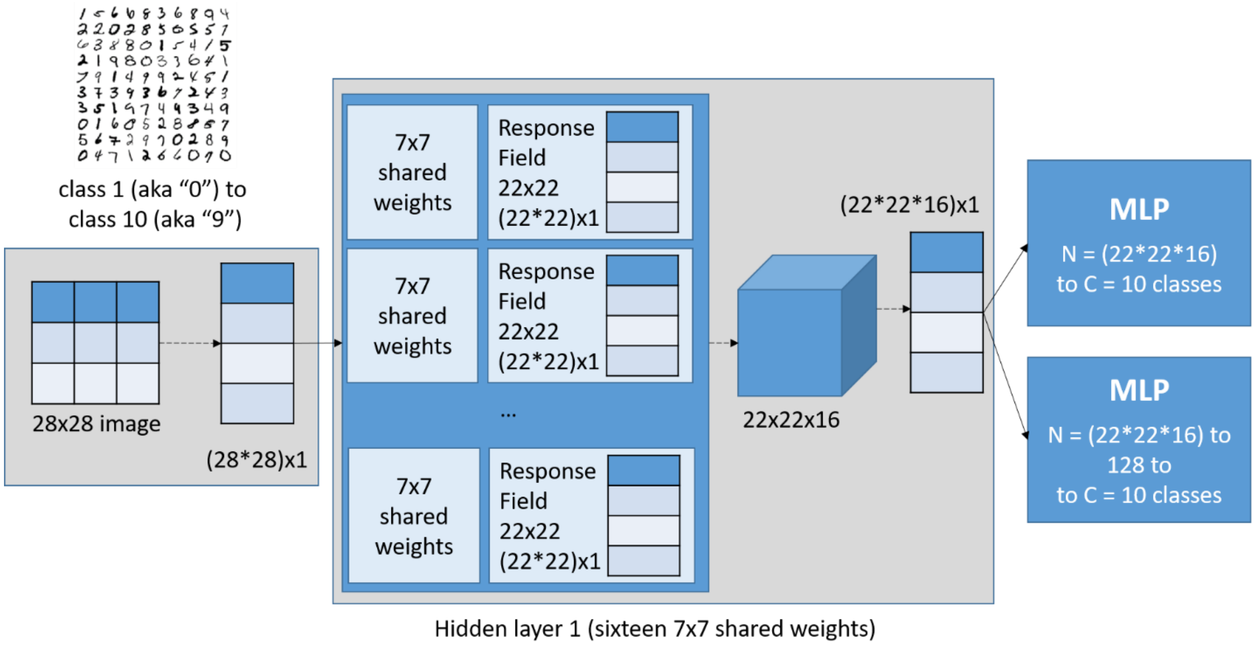
[70%] Part 2: Simple CNN (Due March 12th, start of class, hand in hard copy Project 1 Report)



Part 2 builds on Part 1 (concepts and codes). Do the following:

- Make two shared weight filters in hidden layer 1 that are 28x28 each. Connect their outputs to two perceptron's that produce an (1,0) for class 1 and (0,1) for class 2.
- Use a small subset of the MNIST data, see http://derektanderson.com/eecs_spring_2019_nn/ and Part2.zip. Class 1 is a set of "1 digits". Class 2 is a set of "3 digits". The data is organized into (a) training and (b) testing. At each epoch, (1) use the training data to update the network weights then (2) evaluate the test data and generate an error value.
- *Options*: (1) grouping (serial, batch, mini-batch), (2) order (fixed or shuffle each epoch), (3) network selection (activation function), (4) initialization method, and (5) learning rate. You need to pick and vary at least two of these five *options*.
- Report (1) convergence/error plots and (2) the final 28x28 filters (render as images in [0,255]).

[15%] Part 3: Multi-Class CNN (Due March 12th, start of class, hand in hard copy Project 1 Report)


class 1 (aka "0") to
class 10 (aka "9")
28x28 image
(28*28)x1
7x7 shared weights
Response Field 22x22 (22*22)x1
...
22x22x16
(22*22*16)x1
MLP
N = (22*22*16) to C = 10 classes
MLP
N = (22*22*16) to 128 to to C = 10 classes
Hidden layer 1 (sixteen 7x7 shared weights)

In Part 3, you need to:

- Make sixteen shared weight filters in hidden layer 1 that are 7x7 each. In case (1), connect their output to ten neurons (one for each class). In case (2), map their output to 128 hidden layer neurons then to 10 neurons (one for each class).
- Follow all the details specified in Part 2, except use the Part3.zip data located at http://derektanderson.com/eecs_spring_2019_nn/
- Note: we show the image unrolled into a single column vector (like parts 1 and 2). You can use that format, but then you need to determine how the connections are made to the 7x7 filters that are shared across the image (discussed in class week of 2/4/2019). Alternately, you can leave the image as a 28x28 square and implement the sliding window directly. This really depends on your "mental map" for these required calculations. The heavy-duty linear algebra gang may prefer the former. The more traditional image processing gang may feel more comfortable with the latter.

[5%] Part 4: Deconvolution (Due March 12th, start of class, hand in hard copy Project 1 Report)

Take the first example image from each digit and push it through your trained network. For each image, generate the sixteen possible response fields. For each response field, apply “deconvolution” (convolution transpose). Show your results and discuss them.

Report (20 to 30 pages)

(1) Title Page: 1 page: class, project, date, name.

(2) Technical Description: up to 5 pages: prove to us that you understand this topic (description, math, pictures, etc.)

(3) Code Description: 2 pages: how to run your code and give a picture that connects all the code parts

(4) Experiments and Results: can be 10 or more pages: documentation, tables, plots, discussion.

(5) Lessons Learned: around 2 pages: discuss unexpected things, summarize major findings, etc.

Note: we provided an example student paper from a Pattern Recognition class that Derek taught a few years back. We will discuss your report content and organization, and our expectations in class. Note, the provided student paper is not exactly the same format as what I asked you for above. However, it is a great reference as to what can be accomplished in a course project. Again, we will discuss in class what we believe makes a good report!