NN and Deep Learning

Math Exercise 3

Math 1

a) Since the convolution kernel can be characterized as

$$(x * w)_k = \sum_{i=1}^{W} w_i x_k \qquad \text{for } x \in \mathbb{R}^D.$$

Since there is no padding we see that there are $D - W + 1$ valid placements for the kernel.

And since there are M convolutions, there are total of $\underline{M \times (D - W + 1)}$ outputs.

Because convolutions haves weights, and there are M convolutions, there are

$\underline{M \times W \text{ weights}}$

6) In a fully connected layer there needs to be

$$\underline{M \times (W - D + 1) \text{ neurons}}$$

to replicate the convolution behaviour.

Neurons would connect to all D inputs.

Thus, there are

$$\underline{M \times (W - D + 1) \times D \text{ weights}},$$

most of which should be set to zero.

c) The fully connected layer can be characterized by

$$Z \in \mathbb{R}^{M(W-D+1) \times D}$$

matrix that maps $x \mapsto Zx =: z$.

In each neuron we want to have

$$z_k = \sum_{i=1}^{D} Z_{k,i} x_i$$
$$\approx \sum_{i=1}^{W} Z_{k,k-\lfloor W/2 \rfloor + i} \quad x_i \quad ,$$

meaning that entries outside the convolution window should be penalized and shrinked to zero.

We can achieve this by defining shrinkage matrix P s.t.

$$P_{i,k} = \begin{cases} 1, & i \notin \{k - \lfloor \frac{W}{2} \rfloor + i \,|\, i \in \mathbb{N}, \, i \leq W\} \\ 0, & i \in \quad = \| = . \end{cases}$$

Then regularization

$$R(x) := \| Px \|_1$$

penalizes those entries that are
not within the convolution
kernels window and the
ℓ^1 norm enforces sparsity.

$\boxed{B}$

Math 2

a) The blocks can be drawn as

1:
$$\begin{tikzcd} x \arrow[r, mapsto, "f"] \arrow[rrr, mapsto, bend right] & f_1(x) \arrow[r, mapsto, "\mathrm{ReLU}"] & \mathrm{ReLU}(f_1(x)) \arrow[r, mapsto, "\oplus"] & x + \mathrm{ReLU}(f_1(x)) \end{tikzcd}$$

2:
$$\begin{tikzcd} x \arrow[r, mapsto, "f_1"] \arrow[d, mapsto, "f_1"] & f_1(x) \arrow[r, mapsto, "\mathrm{ReLU}"] & \mathrm{ReLU}(f_1(x)) \arrow[r, mapsto, "f_2"] & f_2(\mathrm{ReLU}(f_1(x))) \arrow[r] & \oplus \; f_1(x) + f_2\ldots \\ f_1(x) \arrow[urrrr, mapsto] & & & & \end{tikzcd}$$

3
$$\begin{tikzcd} x \arrow[r, mapsto, "\mathrm{ReLU}"] \arrow[rrrrd, mapsto] & \mathrm{ReLU}(x) \arrow[r, mapsto, "f_1"] & f_1(\mathrm{ReLU}(x)) \arrow[r, mapsto, "\mathrm{ReLU}"] & \mathrm{ReLU}(f_1(\mathrm{Relu}(x))) \arrow[r, mapsto, "f_2"] & f_2(\mathrm{Relu}(f_1(\mathrm{Relu}(x)))) \arrow[d, mapsto] \\ & & & & \oplus \arrow[r, mapsto] & y. \end{tikzcd}$$

b)

1: $y = \begin{cases} x & , f_1(x) = w_1 x + b_1 \le 0 \\ (1+w_1)x + b_1 & , f_1(x) > 0 \end{cases}$

Thus, y is piecewise linear.

2: $y = \begin{cases} w_1 x + b_1 + b_2 & , f_1(x) \le 0 \\ f_1(x) + w_2 f_1(x) + b_2 \\ \quad = (1+w_2)(w_1 x + b_1) + b_2 & , f_1(x) > 0 \end{cases}$

piecewise linear with knot at $f_1(x) = 0$.

3:

$y = \begin{cases} x + f_2(\mathrm{ReLU}(b_1)) = \begin{cases} f(x) + b_2 & , x \le 0, b_1 \le 0 \\ f(x) + w_2 b_1 + b_2 & , x \le 0, b_1 > 0 \end{cases} \\ x + f_2(\mathrm{Relu}(f_1(x))) = \begin{cases} x + f_2(b) & , f_1(x) \le 0, x > 0 \\ x + f_2(f_1(x)) & , f_1(x) > 0, x > 0 \end{cases} \end{cases}$

C) Note the derivative is not defined at knots. Where the function is differentiable we get:

1:

$$\frac{dy}{dx} = \begin{cases} 1 & , \; f_1(x) \leq 0 \\ 1+w_1 & , \; f_1(x) > 0 \end{cases}$$

2:

$$\frac{dy}{dx} = \begin{cases} w_1 & , \; f_1(x) \leq 0 \\ (1+w_2)\,w_1 & , \; f_1(x) \geq 0 \end{cases}$$

3:

$$\frac{dy}{dx} = \begin{cases} w_1 & , \; x \leq 0 \\ 1 & , \; x > 0, \; f_1(x) \leq 0 \\ 1 + w_1 w_2 x & , \; x > 0, \; f_1(x) > 0 . \end{cases}$$

would probably prefer block 1, because it can be used to produce all the same functions as the more complicated blocks.

Block 1 also has the "most" stable gradient with block this.