Homework 2

Guilherme Albertini

November 4, 2022

Theory

Problem 1.1: Convolutional Neural Networks

1. Given an input image of dimension 21×12, what will be output dimension after applying a convolution with 4×5 kernel, stride of 4, and no padding?

   5×2

2. Given an input of dimension $C \times H \times W$ what will be the dimension of the output of a convolutional layer with kernel of size $K \times K$, padding P, stride S, dilation D, and F filters. Assume that H $\geq$ K, W $\geq$ K.

Define Padding along height on top P_{H1}

Define Padding along height on bottom P_{H2}

Define Padding along width on left P_{W1}

Define Padding along width on right P_{W2}

Define Kernel width K_H

Define Kernel height K_W

Define Stride horizontal S_W

Define Stride vertical S_H

Define Batch Count B

Note that for Dilated kernel:

$K' = K + (K-1)(D-1) = K + KD - K - D + 1 = D(K-1) + 1$

.

Effect of adding padding and applying kernel to dimensions:

$$\begin{aligned}
H_P &= P_{H1} + P_{H2} + H \\
W_P &= P_{W1} + P_{W2} + W \\
H_{PK} &= H_1 - [D_H(K_H - 1) + 1] \\
&= P_{H1} + P_{H2} + H - [D_H(K_H - 1) + 1] \\
W_{PK} &= W_1 - [D_W(K_W - 1) + 1] \\
&= P_{W1} + P_{W2} + W - [D_W(K_W - 1) + 1]
\end{aligned}$$

Considering stride to dimensions:

$$\begin{aligned}
H_{PKS} &= \left\lfloor \frac{H_P - [D_H(K_H - 1) + 1] + S_H}{S_H} \right\rfloor \\
&= \left\lfloor \frac{P_{H1} + P_{H2} + H - [D_H(K_H - 1) + 1]}{S_H} \right\rfloor + 1 \\
W_{PKS} &= \left\lfloor \frac{W_P - [D_W(K_W - 1) + 1] + S_W}{S_W} \right\rfloor \\
&= \left\lfloor \frac{P_{W1} + P_{W2} + W - [D_W(K_W - 1) + 1]}{S_W} \right\rfloor + 1
\end{aligned}$$

We can make simplifcations that I think are implied here:

$$\begin{aligned}
S &= S_W = S_H \\
D &= D_W = D_H \\
K &= K_W = K_H \\
B &= 1 \\
P &= P_{W1} + P_{W2} = P_{H1} + P_{H2}
\end{aligned}$$

Thus the output dimension is:

$$\begin{aligned}
F \times &\left(\left\lfloor \frac{2P + H - [D(K-1) + 1]}{S} \right\rfloor + 1 \right) \\
\times &\left(\left\lfloor \frac{2P + W - [D(K-1) + 1]}{S} \right\rfloor + 1 \right)
\end{aligned}$$

3. Let's consider an input $x[n] \in \mathbb{R}^5$, with $1 \leq n \leq 7$, e.g. it is a length 7 sequence with 5 channels. We consider the convolutional layer f_W with one filter, with kernel size 3, stride of 2, no dilation, and no padding. The only parameters of the convolutional layer is the weight $W, W \in \mathbb{R}^{1\times 5\times 3}$ and there is no bias and no non-linearity.

   (a) What is the dimension of the output $f_W(x)$? Provide an expression for the value of elements of the convolutional layer output $f_W(x)$. Example answer format here and in the following sub-problems: $f_W(x) \in \mathbb{R}^{42\times 42\times 42}$, $f_W(x)[i, j, k] = 42$.

The general recurrence equation (which is first-order, non-homogeneous, with variable coefficients): $r_{l-1} = s_l r_l - (s_l - k_l) = s_l(r_l - 1) + k_l$

$$f_W(x) \in \mathbb{R}^3$$
$$f_W(x)[r] = \sum_{c=1}^{5}\sum_{k=1}^{3} x[k + 2(r-1), c] W_{1,c,k}$$

For $r = \{i : i \in \mathbb{N}, i \in [1, \dim(f_W)]\}$

(b) What is the dimension of $\frac{\partial f_W(x)}{\partial W}$? What are its values?

Note: There are a few ways one could interpret the transpose of the tensor (W) depending on which dimensions are to be transposed in numerator format. Using a chosen transpose with numerator layout format.

$$\frac{\partial f_W(x)}{\partial W} \in \mathbb{R}^{3\times(3\times5\times1)}$$
$$\frac{\partial f_W(x)}{\partial W}[r, c, k] = x[k + 2(r-1), c]$$

(c) What is the dimension of $\frac{\partial f_W(x)}{\partial x}$? What are its values?

See note above.

$$\frac{\partial f_W(x)}{\partial x} \in \mathbb{R}^{3\times(7\times5)}$$
$$\frac{\partial f_W(x)}{\partial x}[r, c, k] = \begin{cases} W_{1,c,k-2(r-1)} & \text{if } k - 2(r-1) \in [1, 3] \\ 0 & \text{otherwise} \end{cases}$$

(d) Now, suppose you are given the gradient of the loss ℓ with respect to the output of the convolutional layer $f_W(x)$, i.e. $\frac{\partial \ell}{\partial f_W(x)}$. What is the dimension of $\frac{\partial \ell}{\partial W}$? Provide its expression. Explain the similarities and differences of this and expression in (a).

$$\frac{\partial \ell}{\partial W} = \frac{\partial \ell}{\partial f_W}\frac{\partial f_W}{\partial W}$$

$$\frac{\partial \ell}{\partial W} \in \mathbb{R}^{3\times 5\times 1}$$

$$\left(\frac{\partial \ell}{\partial W}\right)[1, c, k] = \sum_{r=1}^{3} \left(\frac{\partial \ell}{\partial f_W(x)}\right)[r]x[k + 2(r-1), c]$$

Both the backward and forward pass of the convolutional layer apply a convolution but the stride dilates in the backward pass; we can consider dilation factor D as the gradient of the loss with respect to the output of the convolutional layer.

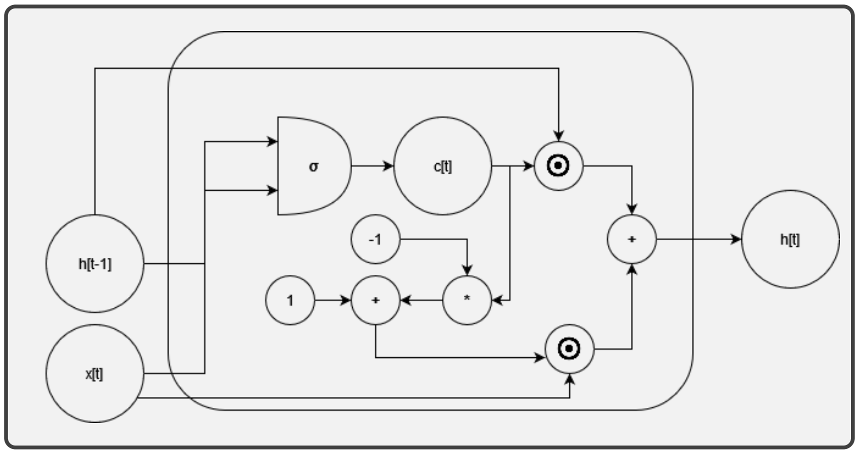
Problem 1.2: Recurrent Neural Networks

In this section consider simple recurrent neural network defined by:

$$c[t] = \sigma(W_c x[t] + W_h h[t-1]) \tag{1}$$
$$h[t] = c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t] \tag{2}$$

here σ is element-wise sigmoid, $x[t] \in \mathbb{R}^n, h[t] \in \mathbb{R}^m, W_c \in \mathbb{R}^{m\times n}, W_h \in \mathbb{R}^{m\times m}, W_x \in \mathbb{R}^{m\times n}$ and $\odot$ is a Hadamard product, $h[0] := 0$.

1. Draw a diagram for this RNN.



2. What is the dimension of $c[t]$?

$$c[t] \in \mathbb{R}^m$$

3. Suppose that we run the RNN to get a sequence of $h[t]$ for t from 1 to K. Assuming we know the derivative $\frac{\partial \ell}{\partial h[t]}$, provide the dimension of an expression for values of $\frac{\partial \ell}{\partial W_x}$. What are the similarities between backward and forward pass of this RNN?

$$\frac{\partial \ell}{\partial W_x} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial h[t]} \frac{\partial h[t]}{\partial W_x}$$

Note that the first term of $\frac{\partial h[t]}{\partial W_x}$ has dependence on the prior term recursively so need chain rule: $\frac{\partial h[t]}{\partial h[t-1]} \frac{h[t-1]}{W_x}$

$$= \sum_{t=1}^{K} \frac{\partial \ell}{\partial h[t]} \left(\frac{\partial([1 - c[t]] \odot W_x x[t])}{\partial W_x} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial h[i]}{\partial h[i-1]} \frac{\partial h[i-1]}{\partial W_x} \right) \right)$$

Note:

$$\begin{aligned} \frac{\partial([1 - c[t]] \odot W_x x[t])}{W_x} &= \frac{diag(1 - c[t]) \partial (W_x x[t]) + diag(W_x x[t]) \partial (1 - c[t])}{\partial W_x} \\ &= diag(1 - c[t]) \frac{\partial W_x x[t]}{\partial W_x} \\ &= (1 - c[t]) \odot X \end{aligned}$$

Thus,

$$\frac{\partial \ell}{\partial W_x} = \sum_{t=1}^{K} \frac{\partial \ell}{\partial h[t]} \left((1 - c[t]) \odot X + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial h[i]}{\partial h[i-1]} \right) [(1 - c[t]) \odot X] \right)$$

$$X_j = [0 \times (j-1), x[t], 0, \ldots], X \in \mathbb{R}^{m \times (m \times n)}$$

Both the forward and backward passes use recurrences.

Aside: Diagonal matrices can eliminate Hadamard products.

$$
\begin{aligned}
h[t] &= c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t] \\
dh[t] &= h[t-1] \odot dc[t] + c[t] \odot d(h[t-1]) \\
&\quad + (1 - c[t]) \odot d(W_x x[t]) + W_x x[t] \odot d(1 - c[t]) \\
&\quad = diag(h[t-1]) dc[t] + diag(c[t]) dh[t-1] - diag(W_x x[t]) dc[t] \\
&\quad + diag(1 - c[t])(dW_x x[t] + W_x[t] dx[t])
\end{aligned}
$$

Note $\frac{\partial W_x}{\partial h[t-1]} = \frac{\partial x[t]}{\partial h[t-1]} = 0$:

$$
\frac{\partial h[t]}{\partial h[t-1]} = diag(c[t]) + diag(h[t-1]) \frac{\partial c[t]}{\partial h[t-1]} - \\
diag(W_x x[t]) \frac{\partial c}{\partial h[t-1]}
$$

And note, as $\sigma'(x) = \sigma(x)(1 - \sigma(x))$:

$$
\begin{aligned}
c[t] &= \sigma(W_c x[t] + W_h h[t-1]) \\
dc[t] &= \sigma(W_c x[t] + W_h h[t-1]) \odot (1 - \sigma(W_c x[t] + W_h h[t-1])) \\
&\quad \odot d[W_c x[t] + W_h h[t-1]] \\
&\quad = diag[\sigma(W_c x[t] + W_h h[t-1]) \\
&\quad \odot (1 - \sigma(W_c x[t] + W_h h[t-1]))] d[W_c x[t] + W_h h[t-1]] \\
&\quad \implies \frac{\partial c[t]}{\partial h[t-1]} = diag[\sigma(W_c x[t] + W_h h[t-1]) \\
&\quad \odot (1 - \sigma(W_c x[t] + W_h h[t-1]))] W_h
\end{aligned}
$$

4. Can this network be subject to vanishing or exploding gradients?

The vector $h[t]$ is not being multiplied by matrices throughout timesteps so will not have exploding gradients. It can vanishing gradients as the element-wise multiplication of values of $h[t]$ and $c[t]$ are between 0 and 1.

Problem 1.3: AttentionRNN(2)

Now define AttentionRNN(2) as:

$$q_0[t], q_1[t], q_2[t] = Q_0 x[t], Q_1 h[t-1], Q_2 h[t-2] \quad (3)$$

$$k_0[t], k_1[t], k_2[t] = K_0 x[t], K_1 h[t-1], K_2 h[t-2] \quad (4)$$

$$v_0[t], v_1[t], v_2[t] = V_0 x[t], V_1 h[t-1], V_2 h[t-2] \quad (5)$$

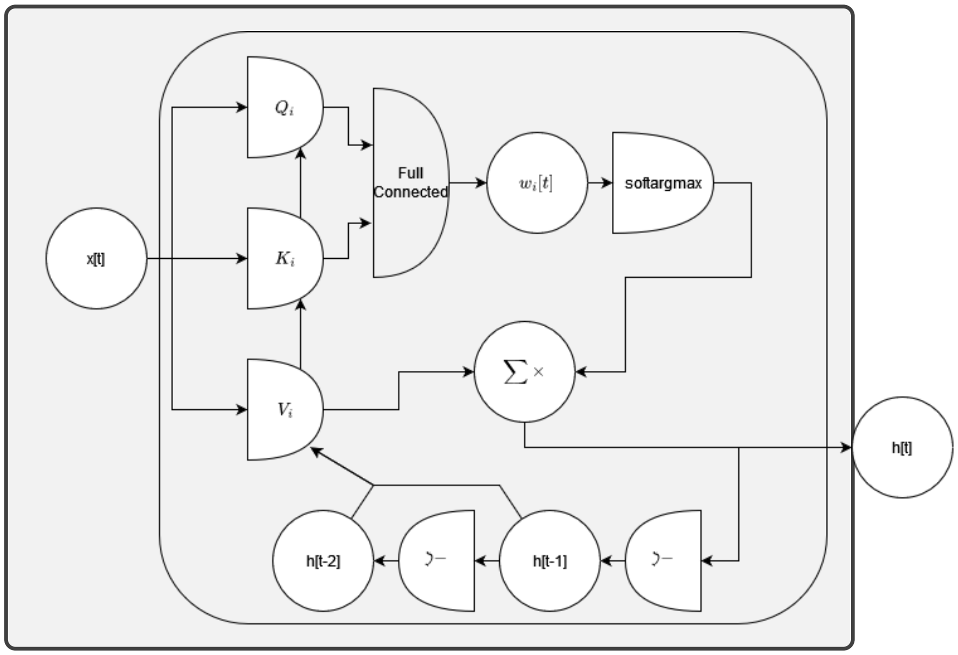
$$w_i[t] = q_i[t]^T k_i[t] \quad (6)$$

$$a[t] = \text{softargmax}([w_0[t], w_1[t], w_2[t]]) \quad (7)$$

$$h[t] = \sum_{i=0}^{2} a_i[t] v_i[t] \quad (8)$$

where $x[t], h[t] \in \mathbb{R}^n$ and $Q_i, K_i, V_i \in \mathbb{R}^{n \times n}$. We define $h[t] = 0$ for $t < 1$. You may safely ignore base cases in the following.

1. Draw a diagram for this RNN.



2. What is the dimension of $a[t]$?

$$a[t] \in \mathbb{R}^n$$

3. Extend this to AttentionRNN(k), a network that uses the last k state vectors h. Write out a system of equations that defines it.

Now define AttentionRNN(k) as:

$$\begin{aligned}
q_0[t], q_1[t], \dots, q_k[t] &= Q_0 x[t], Q_1 h[t-1], \dots, Q_k h[t-k] \\
k_0[t], k_1[t], \dots, k_k[t] &= K_0 x[t], K_1 h[t-1], \dots, K_k h[t-k] \\
v_0[t], v_1[t], \dots, v_k[t] &= V_0 x[t], V_1 h[t-1], \dots, V_k h[t-k] \\
w_i[t] &= q_i[t]^T k_i[t] \\
a[t] &= \text{softargmax}([w_0[t], w_1[t], \dots, w_k[t]]) \\
h[t] &= \sum_{i=0}^{k} a_i[t] v_i[t]
\end{aligned}$$

4. Modify the above network to produce AttentionRNN(∞), a network that uses every past state vector. Write out a system of equations that defines it. We can do this by tying together some set of parameters, e.g. weight sharing.

Now define AttentionRNN(∞) as the following for $i \in [1, T]$:

$$\begin{aligned}
q_0[t], \dots, q_i[t] &= Q_0 x[t], Q_i h[t-i], \dots, Q_i h[1] \\
k_0[t], \dots, k_i[t] &= K_0 x[t], K_i h[t-i] \dots, K_i h[1] \\
v_0[t], \dots, v_i[t] &= V_0 x[t], V_i h[t-i] \dots, V_i h[1] \\
w_i[t] &= q_i[t]^T k_i[t] \\
a[t] &= \text{softargmax}([w_0[t], \dots, w_{t-1}[t]]) \\
h[t] &= \sum_{i=0}^{t-1} a_i[t] v_i[t]
\end{aligned}$$

5. Suppose the loss ℓ is computed, and we know the derivative $\frac{\partial \ell}{\partial h[i]}$ for all $i \geq t$. Write down expression for $\frac{\partial h[t]}{\partial h[t-1]}$ for AttentionRNN(2).

$\frac{\partial h[t]}{\partial h[t-1]}$ with $h[t] = \sum_{i=1}^{2} a_i[t]v_i[t]$ has $h[t] = a_0[t]v_0[t] + a_1[t]v_1[t] + a_2[t]v_2[t]$. Accordingly, only q_1, v_1, k_1 depend on $h[t-1]$.

$$
\begin{aligned}
a_0[t] &= \frac{\exp(q_0[t]^T k_0[t])}{\sum_{i=0}^{2} \exp(q_i[t]^T k_i[t])} \\
a_1[t] &= \frac{\exp(q_1[t]^T k_1[t])}{\sum_{i=0}^{2} \exp(q_i[t]^T k_i[t])} \\
a_2[t] &= \frac{\exp(q_2[t]^T k_2[t])}{\sum_{i=0}^{2} \exp(q_i[t]^T k_i[t])} \\
v_0[t] &= V_0 x[t] \\
v_1[t] &= V_1 h[t-1] \\
v_2[t] &= V_2 h[t-2] \\
\frac{\partial h[t]}{\partial h[t-1]} &= \left(\frac{\partial a_0[t]v_0[t]}{\partial h[t-1]}, \frac{\partial a_1[t]v_1[t]}{\partial h[t-1]}, \frac{\partial a_2[t]v_2[t]}{\partial h[t-1]} \right) \\
\frac{\partial a_0[t]v_0[t]}{\partial h[t-1]} &= v_0[t]\frac{-\partial Z}{Z^2} = \frac{-v_0[t]}{Z^2}\partial(\exp(q_1[t]^T k_1 t)) \\
&= \frac{-v_0[t]}{Z^2}(k_1 Q_q + {q_1}^T k_1[t]) \\
&= -v_0[t](k_1 Q_q + q_1^T K_1)\frac{\exp(q_1[t]^T k_1[t])}{\sum_{i=1}^{2} \exp(q_i[t]^T k_i[t])^2} \\
\frac{\partial a_2[t]v_2[t]}{\partial h[t-1]} &= -v_2[t](k_1 Q_q + q_1^T K_1)\frac{\exp(q_1[t]^T k_1[t])}{\sum_{i=1}^{2} \exp(q_i[t]^T k_i[t])^2} \\
\frac{\partial a_1[t]v_1[t]}{\partial h[t-1]} &= v_1[t]\frac{\partial a_1[t]}{\partial h[t-1]} + a_1[t]\frac{\partial v_1[t]}{\partial h[t-1]} \\
&= \frac{\partial \exp(q_1[t]^T k_1[t])}{Z} - \exp(q_1[t]^T k_1[t])\frac{\partial Z}{Z^2} \\
&= \frac{(k_1 Q_1 + q_1^T K_q)\exp(q_1[t]^T k_1[t])}{Z} \\
&\quad - \exp(q_1[t]^T k_1[t])(k_1 Q_q + q_1^T K_1)\frac{\exp(q_1[t]^T k_1[t])}{\sum_i = 0}^2 \exp(q_i[t]^T k_i[t])
\end{aligned}
$$

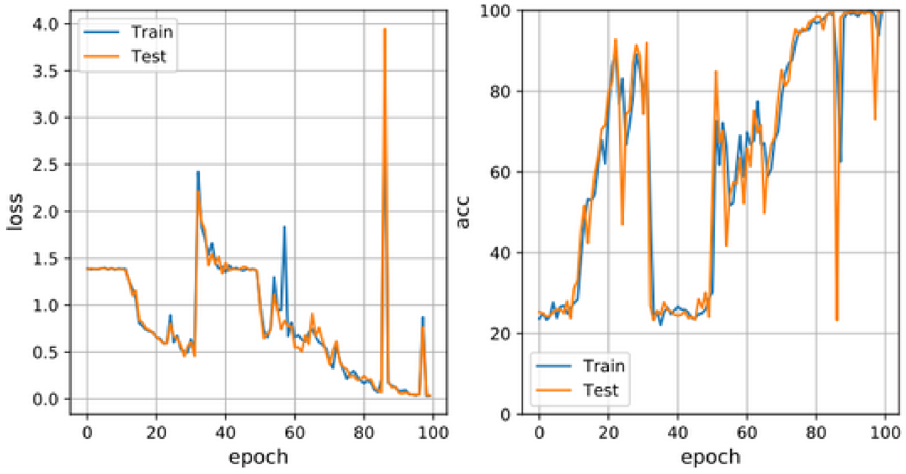
And so,

$$
\frac{\partial h[t]}{\partial h[t-1]} = \frac{\partial a_0[t]v_0[t]}{\partial h[t-1]} + \frac{\partial a_2[t]v_2[t]}{\partial h[t-1]} + v_1[t]\frac{\partial a_1[t]}{\partial h[t-1]} + a_1[t]\frac{\partial v_1[t]}{\partial h[t-1]}
$$

6. Suppose we know $\frac{\partial h[t]}{\partial h[T]}$ and $\frac{\partial \ell}{\partial h[t]} \forall t > T$. Write down expression for $\frac{\partial \ell}{\partial h[T]}$ for AttentionRNN(k).

$$\frac{\partial \ell}{\partial h[T]} = \sum_{i=1}^{k} \frac{\partial \ell}{\partial h[T+i]} \frac{\partial h[T+i]}{\partial h[T]}$$

Problem 1.4: Debugging Loss Curves



1. What causes the spikes on the left?

   Underparametrization of the model at that point in training.

2. How can they be higher than the initial value of the loss?

   At that point in training, the selected parametrization produced a model that was less performant than this initialization (i.e. was worse than randomly selecting one of the classes).

3. What are some ways to fix them?

   We can select a more conservative learning rate or clip the gradient.

4. Explain why the loss and accuracy are at these set values before training starts. You may need to check the task definition in the notebook.

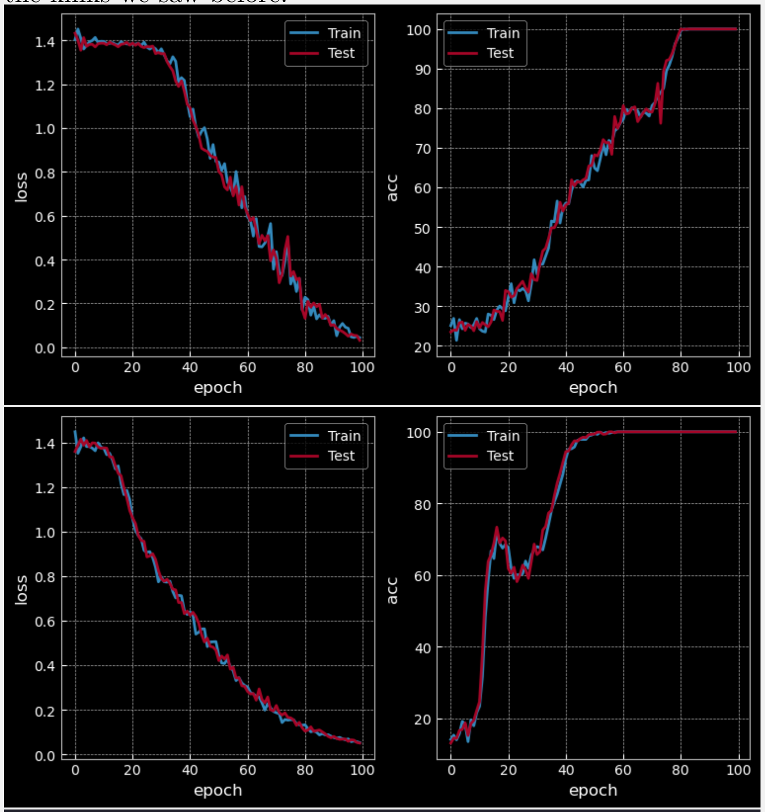
Models are initialized with near-zero random weights that essentially collapse output to near zero, thus energy of model $F(x, Y) \approx 0$. When we take a look at:

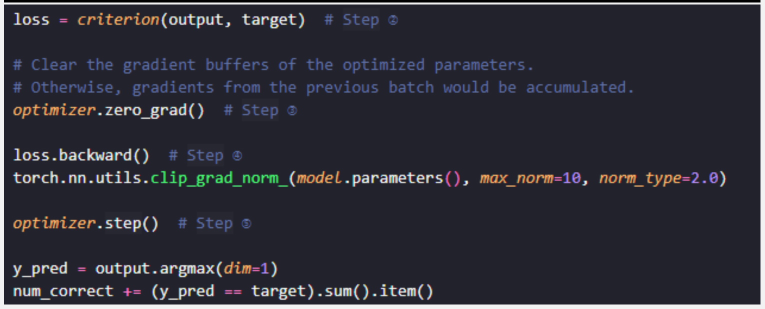
$$\begin{aligned}
L[F(x, Y), y] &= F(x, y) + \frac{1}{\beta} \log \sum_{y' \in Y} \exp(-\beta F(x, y')) \\
&= F(x, y) - softmin_{\beta}[F(x, Y)] \\
-\nabla_{F(x,y)} L[F(x, Y), y] &= \tilde{y} - y
\end{aligned}$$

Where $\tilde{y}$ is the softargmin at that point; the average of all classes produces the uniform distribution and y is the one-hot encoding for the correct class that will "push" its energy down by magnitude 1 while pulling up all other energies for each update step. When we shoot the untrained model's zero vector into softargmin we get each entry as $1/K = 1/4 = 0.25$ and compute cross entropy loss of $-\log_e(1/K) = \log(4) \approx 1.39$ which is what we see at the initial (untrained) loss region. We also see the 0.25 random choice for accuracy at this region above.

Extra Credit: Debugging Loss Curves

Extra Credit Images Shown Below. Note how gradient clipping and a more conservative learning rate of 0.005 was applied to smoothen out the kinks we saw before.





```
loss = criterion(output, target)  # Step ②

# Clear the gradient buffers of the optimized parameters.
# Otherwise, gradients from the previous batch would be accumulated.
optimizer.zero_grad()  # Step ③

loss.backward()  # Step ④
torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=10, norm_type=2.0)

optimizer.step()  # Step ⑤

y_pred = output.argmax(dim=1)
num_correct += (y_pred == target).sum().item()
```