Homework 2: Convolutional Neural Networks and Recurrent Neural Networks

CSCI-GA 2572 Deep Learning

Fall 2025

The goal of homework 2 is to get you to work with convolutional neural networks and recurrent neural networks.

In the theoretical part, you will work on figuring out how backpropagation works in these networks. In part 2, we will implement and train them.

In part 1, you should submit all your answers in a pdf file. As before, we recom- mend using LaTeX.

For part 2, you will implement some neural networks by adding your code to the provided `ipynb` file.

As before, please use numerator layout.

The due date of homework 2 at 10/5. Submit the following files in a zip file `your_net_id.zip` through NYU Brightspace:

- `hw2_theory.pdf`
- `hw2_cnn.ipynb`
- `hw2_rnn.ipynb`
- `08-seq_classification.ipynb`

The following behaviors will result in penalty of your final score:

1. 10% penalty for submitting your file without using the correct naming for- mat (including naming the zip file, PDF file or python file wrong, adding extra files in the zip folder, like the testing scripts in your zip file).
2. 20% penalty for late submission within the first 24 hours after the deadline. We will not accept any late submission after the first 24 hours.
3. 20% penalty for code submission that cannot be executed following the steps we mentioned.

1 Theory (50pt)

1.1 Convolutional Neural Netoworks (15 pts)

(a) (1 pts) 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?

$$H_{out} = \left\lfloor \frac{21 - 4 + 0}{4} \right\rfloor + 1 = 5$$

$$W_{out} = \left\lfloor \frac{12 - 5 + 0}{4} \right\rfloor + 1 = 2$$

Therefore, the output dimension is 5×2.

(b) (2 pts) 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 $H \geq K$, $W \geq K$.

Considering the dilation, the effective kernel size is:

$$K_{eff} = D \cdot (K - 1) + 1$$

With this effective kernel size, we can compute

$$H_{out} = \left\lfloor \frac{H - K_{eff} + 2P}{S} \right\rfloor + 1$$

$$W_{out} = \left\lfloor \frac{W - K_{eff} + 2P}{S} \right\rfloor + 1$$

Therefore, the output dimension becomes $F \times H_{out} \times W_{out}$.

(c) (12 pts) In this section, we are going to work with 1-dimensional convolutions. Discrete convolution of 1-dimensional input $x[n]$ and kernel $k[n]$ is defined as follows:

$$s[n] = (x * k)[n] = \sum_m x[n - m]\, k[m]$$

However, in machine learning, convolution is usually implemented as cross-correlation, which is defined as follows:

$$s[n] = (x * k)[n] = \sum_m x[n + m]\, k[m]$$

Note the difference in signs, which will get the network to learn a "flipped" kernel. In general it doesn't change much, but it's important to keep it in mind. In convolutional neural networks, the kernel $k[n]$ is usually 0 everywhere, except a few values near 0: $\forall |n| > M,\ k[n] = 0$. Then, the formula becomes:

$$s[n] = (x * k)[n] = \sum_{m=-M}^{M} x[n + m]\, k[m]$$

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, $W \in \mathbb{R}^{1 \times 5 \times 3}$, there's no bias and no non-linearity.

(i) (1 pts) 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:** $f_W(x) \in \mathbb{R}^{42\times42\times42}$, $f_W(x)[i,j,k] = 42$.

$f_W(x) \in \mathbb{R}^{1\times3}$

$$f_W(x)[1,p] = \sum_{c=1}^{5}\sum_{m=1}^{3} x\big[c, s_p + m - 1\big] W[1,c,m], \text{where } s_p = 2p-1 \text{ and } p \in \{1,2,3\}.$$

(ii) (3 pts) What is the dimension of $\dfrac{\partial f_W(x)}{\partial W}$? Provide an expression for the values of the derivative $\dfrac{\partial f_W(x)}{\partial W}$.

As a Jacobian (matrix) it is $(1\times3)\times(1\times5\times3)$.
As a tensor, you can view it as $\mathbb{R}^{1\times3\times1\times5\times3}$.

$$\frac{\partial\, f_W(x)[1,p]}{\partial\, W[1,c,m]} \;=\; x\big[c,\; s_p + m - 1\big], \qquad p = 1,2,3,\; c = 1,\ldots,5,\; m = 1,2,3.$$

(iii) (3 pts) What is the dimension of $\dfrac{\partial f_W(x)}{\partial x}$? Provide an expression for the values of the derivative $\dfrac{\partial f_W(x)}{\partial x}$.

As a Jacobian it is $(1\times3)\times(5\times7)$.
As a tensor: $\mathbb{R}^{1\times3\times5\times7}$.

$$\frac{\partial\, f_W(x)[1,p]}{\partial\, x[c,n]} = \begin{cases} W\big[1,\; c,\; n - s_p + 1\big], & \text{if } n \in \{s_p,\; s_p+1,\; s_p+2\}, \\ 0, & \text{otherwise,} \end{cases}$$

where $p = 1,2,3,\; c = 1,\ldots,5,\; n = 1,\ldots,7$.

(iv) (5 pts) Now, suppose you are given the gradient of the loss ℓ w.r.t. the output of the convolutional layer $f_W(x)$, i.e. $\dfrac{\partial \ell}{\partial f_W(x)}$. What is the dimension of $\dfrac{\partial \ell}{\partial W}$? Provide an expression for $\dfrac{\partial \ell}{\partial W}$. Explain similarities and differences of this expression and expression in (i).

Let $g[1,p] = \frac{\partial \ell}{\partial f_W(x)}[1,p]$ with $p \in \{1,2,3\}$ and $s_p = 2p-1$.

$$\frac{\partial \ell}{\partial W} \in \mathbb{R}^{1\times5\times3}$$

$$\frac{\partial \ell}{\partial W[1,c,m]} = \sum_{p=1}^{3} g[1,p]\; x\big[c,\; s_p + m - 1\big], \qquad c = 1,\ldots,5,\; m = 1,2,3.$$

Relation to (i): same local products $x[c,\cdot]\, W[1,c,m]$; forward computes $f_W(x)[1,p] = \sum_{c,m} x[c, s_p + m - 1]\, W[1,c,m]$ (sum over c,m at fixed p), while the gradient sums over p at fixed (c,m) and is weighted by $g[1,p]$.

1.2 Recurrent Neural Networks (30 pts)

Part 1

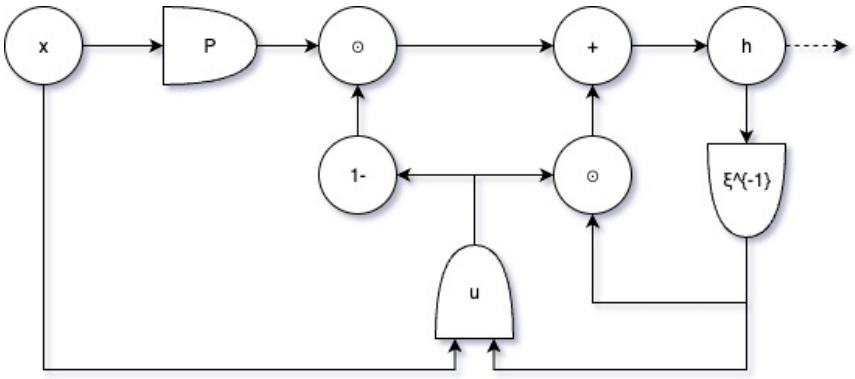
In this section we consider a simple recurrent neural network defined as follows:

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

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

(a) (4 pts) Draw a diagram for this recurrent neural network, similar to the diagram of RNN we had in class. We suggest using diagrams.net.

Let $c[t] = u(x[t], h[t-1]) = \sigma(W_c x[t] + W_h h[t-1])$,
and $W_x x[t] = P(x)$



(b) (1pts) What is the dimension of $c[t]$?

Since $c[t] = \sigma(W_c x[t] + W_h h[t-1])$ with $W_c \in \mathbb{R}^{m \times n}$ and $W_h \in \mathbb{R}^{m \times m}$, $c[t] \in \mathbb{R}^m$.

(c) (5 pts) Suppose that we run the RNN to get a sequence of $h[t]$ for t from 1 to K. Assuming we know the derivative $\dfrac{\partial \ell}{\partial h[t]}$, provide dimension of and an expression for values of $\dfrac{\partial \ell}{\partial W_x}$. What are the similarities of backward pass and forward pass in this RNN?

Let $g_t := \frac{\partial \ell}{\partial h[t]} \in \mathbb{R}^m$ be given. Only the second term of $h[t] = c[t] \odot h[t-1] + (1 - c[t]) \odot W_x x[t]$ depends on W_x. Thus,

$$\frac{\partial \ell}{\partial W_x} = \sum_{t=1}^{K} \left(g_t \odot (1 - c[t])\right) x[t]^\top \in \mathbb{R}^{m \times n}.$$

Similarity to the forward pass: the backward pass follows the same temporal structure as the forward computation but in reverse order. It reuses the same gating terms ($c[t]$ and $1 - c[t]$) to scale the gradients and accumulates contributions from each time step through outer products with $x[t]$.

(d) (2pts) Can this network be subject to vanishing or exploding gradients? Why?

Yes. During backpropagation, the gradients passed to $h[t-1]$ are repeatedly scaled by the gate values and their derivatives (for example, $c[t] \in (0,1)$ and $\sigma'(\cdot) \leq 1/4$), which can gradually shrink over time and cause *vanishing* gradients. Conversely, if the effective recurrent weights (e.g., from W_h in the gating path) have large magnitudes, these repeated multiplications can amplify the gradients, leading to *exploding* gradients. While the gating mechanism helps regulate these effects, it does not completely prevent them.

Part 2

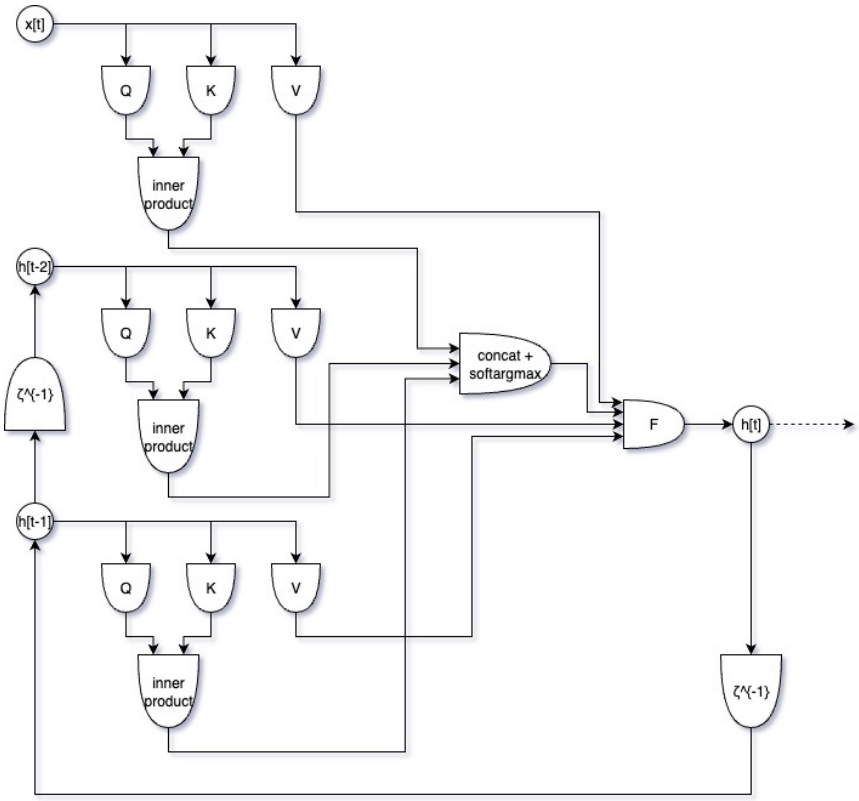
We define an AttentionRNN(2) as

$$q_0[t], q_1[t], q_2[t] = Q_0 x[t], Q_1 h[t-1], Q_2 h[t-2] \tag{3}$$
$$k_0[t], k_1[t], k_2[t] = K_0 x[t], K_1 h[t-1], K_2 h[t-2] \tag{4}$$
$$v_0[t], v_1[t], v_2[t] = V_0 x[t], V_1 h[t-1], V_2 h[t-2] \tag{5}$$
$$w_i[t] = q_i[t]^\top k_i[t] \tag{6}$$
$$a[t] = \text{softargmax}([w_0[t], w_1[t], w_2[t]]) \tag{7}$$
$$h[t] = \sum_{i=0}^{2} a_i[t]\, v_i[t] \tag{8}$$

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 these bases cases in the following questions.

(a) (4 pts) Draw a diagram for this recurrent neural network.

Let $h[t] = F(a[t], v_0[t], v_1[t], v_2[t]) = \sum_{i=0}^{2} a_i[t]\, v_i[t]$.



(b) (1 pt) What is the dimension of $a[t]$?

$a[t]$ is the softmax over three scores $[w_0[t], w_1[t], w_2[t]]$, hence $a[t] \in \mathbb{R}^3$.

(c) (3 pts) Extend this to, AttentionRNN(k), a network that uses the last k state vectors h. Write out the system of equations that defines it. You may use set notation or ellipses (...) in your definition.

AttentionRNN(k). Let the sources be

$$s_0[t] = x[t], \qquad s_i[t] = h[t-i] \ (i = 1, \ldots, k).$$

With parameter sets $Q_i, K_i, V_i \in \mathbb{R}^{n \times n}$ (one per source),

$$\begin{aligned} q_i[t] &= Q_i \, s_i[t], \qquad k_i[t] = K_i \, s_i[t], \qquad v_i[t] = V_i \, s_i[t], \qquad i = 0, \ldots, k, \\ w_i[t] &= q_i[t]^\top k_i[t], \\ a[t] &= \mathrm{softargmax}\big([w_0[t], \ldots, w_k[t]]\big) \in \mathbb{R}^{k+1}, \\ h[t] &= \sum_{i=0}^{k} a_i[t] \; v_i[t]. \end{aligned}$$

(d) (3 pts) Modify the above network to produce AttentionRNN(∞), a network that uses every past state vector. Write out the system of equations that defines it. You may use set notation or

ellipses (...) in your definition. **HINT:** We can do this by tying together some set of parameters, e.g. weight sharing.

AttentionRNN(∞) (weight tying). Use all past states and share one set of weights across lags $i \geq 1$:

$$s_0[t] = x[t], \quad s_i[t] = h[t-i] \ (i = 1, \ldots, t-1), \quad Q_x, K_x, V_x, \ Q, K, V \in \mathbb{R}^{n \times n},$$

$$\begin{aligned}
q_0[t] &= Q_x s_0[t], \quad k_0[t] = K_x s_0[t], \quad v_0[t] = V_x s_0[t], \\
q_i[t] &= Q\, s_i[t], \quad k_i[t] = K\, s_i[t], \quad v_i[t] = V\, s_i[t] \quad (i \geq 1), \\
w_i[t] &= q_i[t]^\top k_i[t] \quad (i = 0, \ldots, t-1), \\
a[t] &= \text{softargmax}\big([w_0[t], w_1[t], \ldots, w_{t-1}[t]]\big), \\
h[t] &= \sum_{i=0}^{t-1} a_i[t]\ v_i[t].
\end{aligned}$$

(e) (5 pts) Suppose the loss ℓ is computed. Please write down the expression for $\dfrac{\partial h[t]}{\partial h[t-1]}$ for AttentionRNN(2).

Write $a = \text{softmax}(w) \in \mathbb{R}^3$ and $J = \text{diag}(a) - aa^\top \in \mathbb{R}^{3\times 3}$. Stack values $V := [v_0, v_1, v_2] \in \mathbb{R}^{n\times 3}$ and let $e_1 = (0,1,0)^\top$. Define the gradient of the score $w_1 = (Q_1 h[t-1])^\top (K_1 h[t-1])$ as

$$g_1(t) := \nabla_{h[t-1]} w_1[t] = \big(Q_1^\top K_1 + K_1^\top Q_1\big)\, h[t-1] \ \in \ \mathbb{R}^n.$$

Only v_1 depends on $h[t-1]$ linearly ($v_1 = V_1 h[t-1]$), and only w_1 depends on $h[t-1]$. Hence, Jacobian $\dfrac{\partial h[t]}{\partial h[t-1]}$ for AttentionRNN(2) is:

$$\frac{\partial h[t]}{\partial h[t-1]} = a_1[t]\ V_1 \ + \ \big(V\, J\, e_1\big)\, g_1(t)^\top \ \in \ \mathbb{R}^{n\times n}.$$

(f) (2 pts) Suppose we know the derivative $\dfrac{\partial h[t]}{\partial h[T]}$, and $\dfrac{\partial \ell}{\partial h[t]}$ for all $t > T$. Please write down the expression for $\dfrac{\partial \ell}{\partial h[T]}$ for AttentionRNN(k).

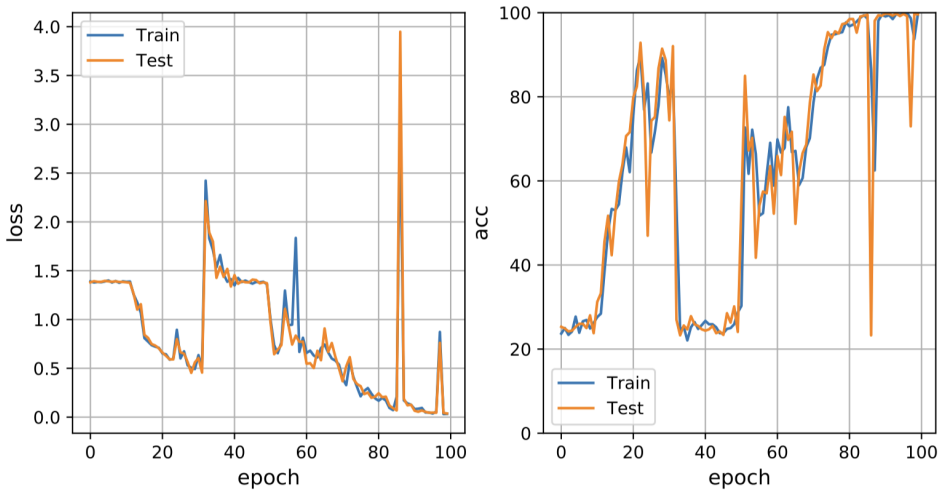
Gradient aggregation for AttentionRNN(k). If $\dfrac{\partial h[t]}{\partial h[T]}$ is known and $\dfrac{\partial \ell}{\partial h[t]}$ is known for all $t > T$, then

$$\frac{\partial \ell}{\partial h[T]} = \sum_{t=T+1}^{T+k} \frac{\partial \ell}{\partial h[t]}\ \frac{\partial h[t]}{\partial h[T]} \ \in \ \mathbb{R}^{1\times n},$$

since in AttentionRNN(k) only the next k states can depend directly on $h[T]$ (sum over all $t > T$ for the ∞ case).

1.3 Debugging loss curves (5pts)

In the class on Thursday September 30, when working with notebook `08-seq_classification`, we saw RNN training curves. In Section 8 "Visualize LSTM", we observed some "kinks" in the loss curve.



1. (1pts) What caused the spikes on the left?

   The spikes on the left are caused by *exploding gradients* early in training. When the recurrent weights and hidden states are still poorly initialized, the gradients can grow uncontrollably during backpropagation through time, leading to unstable parameter updates and sudden jumps in the loss before things settle.

2. (1pts) How can they be higher than the initial value of the loss?

   The loss can go higher than its initial value because the early gradient updates sometimes push the parameters in the wrong direction. In this stage the model's predictions can temporarily get worse before it starts learning meaningful patterns, which makes the loss briefly exceed its starting point.

3. (1pts) What are some ways to fix them?

   A few common fixes include applying *gradient clipping* to limit extreme updates, lowering the learning rate, or using more stable recurrent units such as LSTMs or GRUs. Careful weight initialization and normalization can also make training smoother.

4. (2pts) Explain why the loss and accuracy are at these values before training starts. You may need to check the task definition in the notebook.

   Before training begins, the model is effectively guessing at random across four classes. This gives about $1/4 = 25\%$ accuracy and a cross-entropy loss near $-\log(1/4) \approx 1.39$. These values simply reflect random predictions from an untrained network before any learning has taken place.

2 Implementation (50pts + 5pts extra credit)

There are three notebooks in the practical part:

- (25pts) Convolutional Neural Networks notebook: `hw2_cnn.ipynb`
- (20pts) Recurrent Neural Networks notebook: `hw2_rnn.ipynb`
- (5pts + 5pts extra credit) : This builds on Section 1.3 of the theoretical part.
  - (5pts) Change the model training procedure of Section 8 in `08-seq_classification` to make the training curves have no spikes. You should only change the training of the model, and not the model itself or the random seed.
  - (5pts extra credit) Visualize the gradients and weights throughout training before and after you fix the training procedure.

Plase use your NYU Google Drive account to access the notebooks. First two notebooks contain parts marked as TODO, where you should put your code. These notebooks are Google Colab notebooks, you should copy them to your drive, add your solutions, and then download and submit them to NYU Brightspace. The notebook from the class, if needed, can be uploaded to Colab as well.