Homework 4 - Version 1.1

Deadline: Tues, Mar.24, at 11:59pm.

Submission: You must submit your solutions as a PDF file through MarkUs[1]. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

See the syllabus on the course website[2] for detailed policies. You may ask questions about the assignment on Piazza[3]. *Note that 10% of the homework mark (worth 1 pt) may be removed for a lack of neatness.*

The teaching assistants for this assignment are Sajad Norouzi and John Chen.
mailto:csc413-2020-01-tas@cs.toronto.edu

Clarifications

All the modifications to the original assignment will be marked with red in the main body of the assignment. A complete list of the modifications can be found below:

[1] https://markus.teach.cs.toronto.edu/csc413-2020-01
[2] https://csc413-2020.github.io/assets/misc/syllabus.pdf
[3] https://piazza.com/class/k58ktbdnt0h1wx?cid=1

1 Architectural Choice v.s. Vanishing / Exploding Gradients

For any successful deep learning system, choosing the right network architecture is as important as choosing a good learning algorithm. In this question, we will explore how various architectural choices can have a significant impact on learning. We will analyze the learning performance from the perspective of vanishing /exploding gradients as they are backpropagated from the final layer to the first.

1.1 Warmup: A Single Neuron RNN

Consider an n layered fully connected network that has scalar inputs and outputs. For now, assume that all the hidden layers have a single unit, and that the weight matrices are set to 1 (because each hidden layer has a single unit, the weight matrices have a dimensionality of $\mathbb{R}^{1\times 1}$).

1.1.1 Effect of Activation - Sigmoid [1pt]

Lets say we're using the sigmoid activation. Let x be the input to the network and let $f : \mathbb{R}^1 \to \mathbb{R}^1$ be the function the network is computing. Do the gradients necessarily have to vanish or explode as they are backpropagated? Answer this by showing that $0 \leq |\frac{\partial f(x)}{\partial x}| \leq (\frac{1}{4})^n$, where n is the number of layers in the network.

1.1.2 Effect of Activation - Tanh [1pt]

Instead of sigmoid, now lets say we're using the tanh activation (otherwise the same setup as in 1.1.1). Do the gradients necessarily have to vanish or explode this time? Answer this by deriving a similar bound as in Sec 1.1.1 for the magnitude of the gradient.

1.2 Matrices and RNN

We will now analyze the recurrent weight matrices under Singular Value Decomposition. SVD is one of the most important results in all of linear algebra. It says that any real matrix $M \in \mathbb{R}^{mxn}$ can be written as $M = U\Sigma V^T$ where $U \in \mathbb{R}^{mxm}$ and $V \in \mathbb{R}^{nxn}$ are square orthogonal matrices, and $\Sigma \in \mathbb{R}^{mxn}$ is a rectangular diagonal matrix with nonnegative entries on the diagonal (i.e. $\Sigma_{ii} \geq 0$ for $i \in \{1, \ldots, \min(m, n)\}$ and 0 otherwise). Geometrically, this means any linear transformation can be decomposed into a rotation/flip, followed by scaling along orthogonal directions, followed by another rotation/flip.

1.2.1 Gradient through RNN [1pt]

Let say we have a very simple RNN-like architecture that computes $x_{t+1} = tanh(Wx_t)$. You can view this architecture as a deep fully connected network that uses the same weight matrix at each layer. Suppose the largest singular value of the weight matrix is $\sigma_{max}(W) = \frac{1}{2}$. Show that the largest singular value of the input-output Jacobian has the following bound: $0 \leq \sigma_{max}(\frac{\partial x_n}{\partial x_1}) \leq (\frac{1}{2})^2$. (Hint: if $C = AB$, then $\sigma_{max}(C) \leq \sigma_{max}(A)\sigma_{max}(B)$. Also, the input-output Jacobian is the multiplication of layerwise Jacobians).

1.2.2 Gradient through Residual / GRU / LSTM Layers [0pt]

Suppose all the layers of a neural network you're training has the following operation: $z_{t+1} = z_t + f_t(z_t)$ where z_t and z_{t+1} are activations in consecutive layers. f_t is any continuous, differentiable function. This is true for ResNets, GRUs and LSTMs.

In this part, we are dealing with a ResNet like model that computes $z_{t+1} = z_t + f_t(z_t)$ at each layer. Let $\{\sigma_1, \ldots, \sigma_n\}$ be a list of all the singular values (in ascending order) of the Jacobian for the nonlinear part of the model f_t (i.e. $\frac{\partial f_t(z_t)}{\partial z_t}$. For simplicity, assume that the Jacobian matrix has the following properties: the first half of the singular values are bounded above by σ_{small}, which is much smaller than 1 (i.e. $\forall i \in \{1, \ldots, \frac{n}{2}\}, \sigma_i \leq \sigma_{small}(\frac{\partial f_t(z_t)}{\partial z_t}) \ll 1$. Suppose the latter half of the singular values are lower bounded by σ_{big}, which is much larger than 2 (i.e. $\forall i \in \{\frac{n}{2}, \ldots, n\}, \sigma_i(\frac{\partial f_t(z_t)}{\partial z_t}) \geq \sigma_{big} \gg 2$).

Show that $\sigma_{min}(\frac{\partial z_{t+1}}{\partial z_t}) \geq 1 - \sigma_{small}$ and $\sigma_{max}(\frac{\partial z_{t+1}}{\partial z_t}) \geq \sigma_{big} - 1$.
(Hint: $\sigma_i(A + B) \geq |\sigma_i(A) - \sigma_{max}(B)|$)

1.2.3 Benefits of Residual Connections [1pt]

From the results of 1.2.2, do residual connections/GRU/LSTM layers solve the vanishing gradients problem? Do they also solve the exploding gradients problem?

1.3 Batch Normalization

Batch normalization is one of the most commonly used techniques to improve training of deep neural networks. The method is also quite simple: at every layer, normalize the input by subtracting the mean and dividing by the standard deviation *of the mini-batch*. The motivation is that data normalization usually helps machine learning models, so it may help at every hidden layer.

1.3.1 Gradients of Batch Norm [0pt]

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 1: The forward pass computation of batch normalization. x_i is the ith input vector in a mini-batch of size m. y_i is the output vector for x_i.

Recall the forward computation of the batch normalization layer in Fig 1. Derive the Jacobian of the batch normalization layer for the ith training example, $\frac{\partial y_i}{\partial x_i}$. You should simplify your answer using $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}$.

1.3.2 Batch Normalization and ResNet [1pt]

We now apply batch normalization to neural networks with residual connections. Consider the following two ResNet architectures for the placement of the batch norm and the residual connection in Fig. 2. "Weight" layer can be either a matrix multiplication or convolution.

Which architecture is easier to learn in terms of exploding / vanishing gradient? Provide a brief justification for your answer.

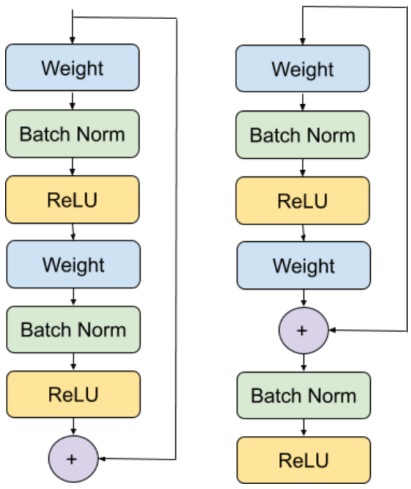


Figure 2: Two different form of ResNet Blocks

2 Auto-regressive Models

In sequence modeling, we aim to learn a distribution of a large amount of data in order to generate new samples, or evaluate the likelihood of any given data point. One approach to achieve this goal is by using auto-regressive models. Let's say we're working with image data. In an auto-regressive model, the joint distribution of pixels over a $H \times W$ image x is defined as the following product of conditional distributions, where x_i is a single pixel.

$$p(x) = \prod_{i=1}^{H \times W} p(x_i \mid x_1, ..., x_{i-1}). \tag{2.1}$$

For simplicity, we can assume each conditional distribution is a Gaussian with a fixed variance, so the model predicts the mean of the next pixel given all the previous pixels. When designing an autoregressive model, we also need to define an ordering for the pixels, which is usually a raster scan order for images, and the natural sequential ordering in texts or audios.

In this question, we will look at several autoregressive architectures and analyze their computational complexity.

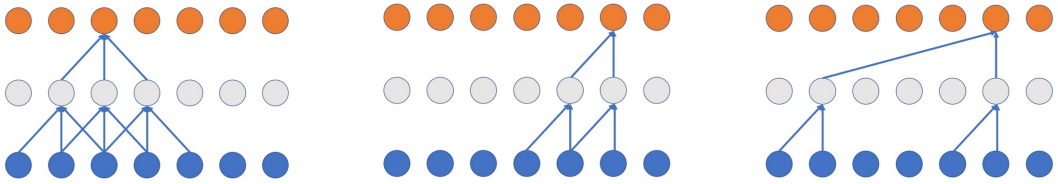
2.1 WaveNet

WaveNet, Oord et al. [2016], is an architecture proposed to model the distribution of raw audio waveforms. WaveNet consists of several 1-D convolutional layers where the last layer of the network

has the same dimensionality as the input corresponding to the mean of each conditional distribution in Eq. 2.1.

WaveNet has two key innovations over classical convolutional models. First, to make sure that at each time step the model just see the values from previous time steps, a causal convolution is used. Causal convolution can be implemented similar to conventional convolution, but with zeros at the position after the middle of filter (look at Fig 3.b).

In the standard causal convolution, the receptive field (context window) of neurons increases linearly with the number of layers. This is problematic when working with waveforms because the model needs to be very deep to capture any long-term context from the input sequence. To make receptive fields grow exponentially, dilated convolution is proposed by introducing zeros in between kernel values. For example in Fig. 3, the second layer has dilation factor of three. The WaveNet architecture starts with dilation factor of 1 and increase it by 1 for each subsequent layer. (For dilated convolution, see Lecture 9.)



(a) Standard 1-D Conv (b) WaveNet Causal Conv (c) Causal Dilated Conv

Figure 3: Different variants of convolutional layer showed in 1D

For the following questions, assume 1 pixel zero padding and stride of 1. We consider a convolution filter size of 3, i.e. there are two incoming weights for the causal convolution neurons in the case of a single channel. Denote the number of input channels and the number of filters at each layer to be k. The input Waveform has a length of t. For simplicity, we assume there are no bias parameters.

2.1.1 Connections [0pt]

Consider a d layer WaveNet model, what is the total number of connections for a WaveNet without dilated convolution? What is the total number of connection when using dilated convolution? Give your answers in terms of d, k, t. You only need to give big-O, not an exact count.

2.1.2 Parallelism [0pt]

Suppose that in each step we compute as many matrix-vector multiplications in parallel as we can, what is the minimum number of the sequential operations to compute the output of a WaveNet with/without dilation? Give your answers in terms of d, k, t. You only need to give big-O, not an exact count.

2.1.3 Discussion [0pt]

What is the benefit of using WaveNet over RNNs and Transformers? Discuss the pros and cons of these three models in terms of their computational, memory complexity, parallelization potential and the size of their context windows.

2.2 PixelCNN

When it comes to 2-D data, PixelCNN, Van den Oord et al. [2016], is a seminal work for modeling the distribution of images using convolutional architecture in an autoregressive manner. Similar to WaveNet, PixelCNN masks each convolutional filter to see just pixels appeared before the current pixel in raster scan order. A visualization of the 2-D causal convolutional filters is shown in Fig. 4. They stack several convolutional layers with masked filters and assume that the last layer gives the mean of conditional distribution for each pixel.

For the following questions, assume zero padding and stride of 1. We consider a 2-D convolution filter size of 3 as in Fig. 4. Denote the number of input channels and the number of filters at each layer to be k. The input image size is $H \times W$. For simplicity, we assume there are no bias parameters.

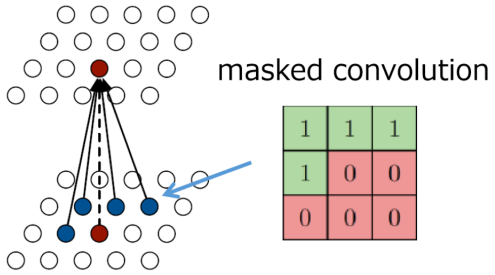


Figure 4: 2-D masked (causal) convolutional filters

2.2.1 Connections [1pt]

Consider a d layer PixelCNN model, what is the total number of connections? Give your answers in terms of d, k, H, W. You only need to give big-O, not an exact count.

2.2.2 Parallelism [1pt]

Suppose that in each step we compute as many matrix-vector multiplications in parallel as we can, what is the minimum number of the sequential operations to compute the output of a Pixel CNN in terms of d, k, H, W. You only need to give big-O, not an exact count.

2.3 Multidimensional RNN

One of the predecessors to the PixelCNN architecture was the multidimensional RNN (MDRNN). This is like the RNNs we discussed in lecture, except that instead of a 1-D sequence, we have a 2-D grid structure. Analogously to how ordinary RNNs have an input vector and a hidden vector for every time step, MDRNNs have an input vector and hidden vector for every grid square. Each hidden unit receives bottom-up connections from the corresponding input square, as well as recurrent connections from its north and west neighbors as follows:

The activations are computed as follows:

$$\mathbf{h}^{(i,j)} = \boldsymbol{\phi}\Big(\mathbf{W}_{\text{in}}^{\top}\mathbf{x}^{(i,j)} + \mathbf{W}_{\text{W}}^{\top}\mathbf{h}^{(i-1,j)} + \mathbf{W}_{\text{N}}^{\top}\mathbf{h}^{(i,j-1)}\Big).$$

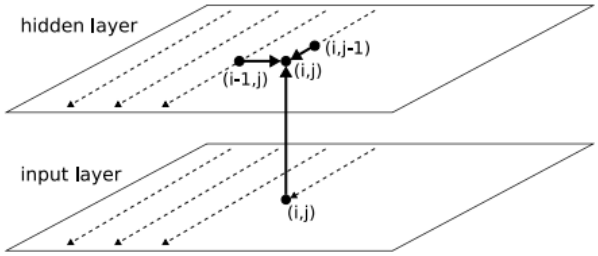


Figure 5: a single layer MDRNN

Denote the number of input channels and the number of recurrent neurons at each layer to be k. The input image size is $H \times W$. For simplicity, we assume there are no bias parameters.

2.3.1 Connections [1pt]

Consider a d layer MDRNN model, what is the total number of connections? Give your answers in terms of d, k, H, W. You only need to give big-O, not an exact count.

2.3.2 Parallelism [0pt]

Suppose that in each step we compute as many matrix-vector multiplications in parallel as we can, what is the minimum number of the sequential operations we need to compute the output in terms of d, k, H, W. You only need to give big-O, not an exact count.

2.3.3 Discussion [1pt]

What is the benefit of using PixelCNN over MDRNN. Discuss the pros and cons of the two models in terms of their computational, memory complexity, parallelization potential and the size of their context windows.

References

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.