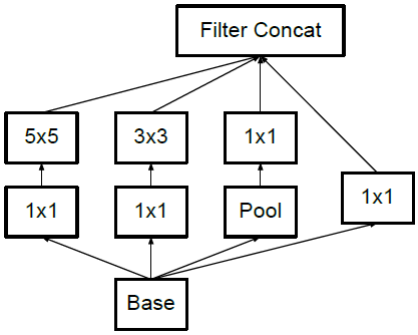
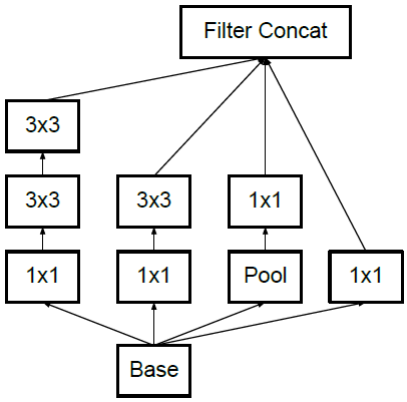
1 Modern CNN

You have been introduced to the original inception module.



1.1 (5 Points)

Show that the following module is more parameter efficient than the original one.

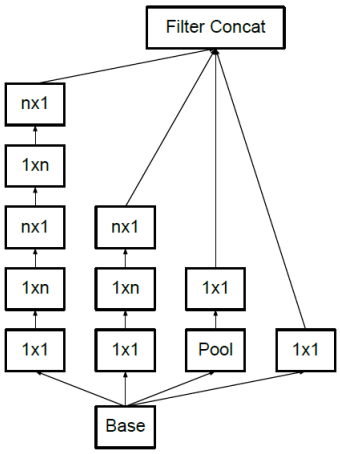


1.2 (5 Points)

Show that you can save more computation by factorizing 3×3 convolutions into two 2×2 convolutions. (One branch is enough.)

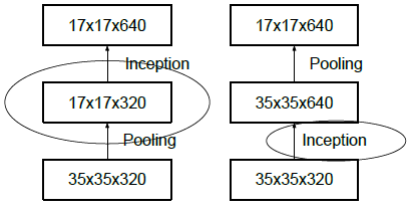
1.3 (10 Points)

Show that using asymmetric convolutions as follows saves computation. (One branch is enough.) How do you compare this enhancement with the previous one?



1.4 (10 Points)

Consider the following methods for grid size reduction. Which one do you prefer in terms of computational cost? Which one creates a representational bottleneck?



2 Autoencoders

1000 gene expression profile samples are given to us. These samples contain normal and cancer cases. The objective is to design a model which can distinguish cancer samples from normal ones. To do so, we have split the data into 800 train and 200 test samples. Each gene expression profile is a 20,000 dimension vector of numbers between 0 and 15. We have designed a 7-layer Autoencoder. The first and last layers have 2000 neurons, the second and 6th layers have 200 neurons and the fourth layer has only 20 neurons. Consider that all of the layers are fully-connected layers and the activation function for all of the neurons is sigmoid. The loss function has been chosen to be MSE.

2.1 (5 Points)

We have trained the network for 100 epochs but the loss function value does not decrease. What is the main reason?

2.2 (10 Points)

After solving the problem, we again train the network for 100 epochs. We observe that MSE is less than 0.1 for test samples. Can we thereby conclude that the network is performing well? If not, what should we check?

2.3 (10 Points)

After the training process, we notice that with every small change in input values, a lot of neurons from the latent layer will change. What should we do in order to solve this issue? Provide a formula if possible.

2.4 (10 Points)

Finally we succeed to train the network in a way that with low test error we can reconstruct input samples from the 20 latent neurons. What property of our input data has helped us in this success?

3 Generative adversial networks (Optional)

In this question, we will workout backpropagation with Generative Adversarial Networks (GANs). Recall that a GAN consists of a Generator and a Discriminator playing a game. The Generator takes as input a random sample from some noise distribution (e.g., Gaussian), and its goal is to produce something from a target distribution (which we observe via samples from this distribution). The Discriminator takes as input a batch consisting of a mix of samples from the true dataset and the Generator's output, and its goal is to correctly classify whether its input comes from the true dataset or the Generator.
Definitions:

- $X^1, ..., X^n$ is a minibatch of n samples from the target data generating distribution For this question, we suppose that each X^i is a k dimensional vector. For example, we, we might be interested in generating a synthetic dataset of customer feature vectors in a credit scoring application
- $Z^1, ..., Z^n$ is a minibatch of n samples from some predetermined noise distribution Note that in general these minibatch sizes may be different.

- The generator $g(.;\theta_g) : Z \to X$ is a neural network
- The discriminator $d(.;\theta_d) : X \to (0, 1)$ is a neural network

The log likelihood of the output produced by the discriminator is:

$$L(\theta_d, \theta_g) = \sum_{i=1}^{n} \log d(X^i; \theta_d) + \log(1 - D(G(Z^i))) \tag{1}$$

The training of such a GAN system proceeds as follows; given the generator's parameters, the discriminator is optimized to maximize the above likelihood. Then, given the discriminator's parameters, the generator is optimized to minimize the above likelihood. This process is iteratively repeated. Once training completes, we only require the generator to generate samples from our distribution of interest; we sample a point from our noise distribution and map it to a sample using our generator.
The Discriminator (The generator architecture is defined analogously)

- Consider the discriminator to be a network with layers indexed by $1, 2, .., L_d$ for a total of L_d layers.
- Let the discriminator's weight matrix for layer l be W_d^l; lets assume there are no biases for simplicity
- Let the activations produced by a layer l be given by A_d^l, and the pre activation values by Z_d^l
- Let $g_d^l(.)$ be the activation function at layer l

The goal of the discriminator is to maximize the above likelihood function.

3.1 (10 Points)

Write down $\bigtriangledown_{\theta_d} L(X; \theta_d, \theta_g)$ in terms of $\bigtriangledown_{\theta_d} D(.)$

3.2 (15 Points)

Write down

$$\partial L(\theta_d, \theta_g)/\partial z_d^{L_d} \tag{2}$$

taking help from your answer in the previous subpart. Remember that the activation function in the last layer of the discriminator is a sigmoid function as the output of the discriminator is a probability.

3.3 (10 Points)

What is mode collapse in GANs? How can it be avoided?

4 Sequence-to-Sequence Models Comprehension

Answer the following questions

4.1 (15 Points)

How can vanishing and exploding gradients be controlled in RNN?

4.2 (5 Points)

What are the advantages of bi-directional LSTM over simple LSTM?

4.3 (10 Points)

Why is encoder-decoder architecture used in Machine Translation? Explain problems of this architecture.

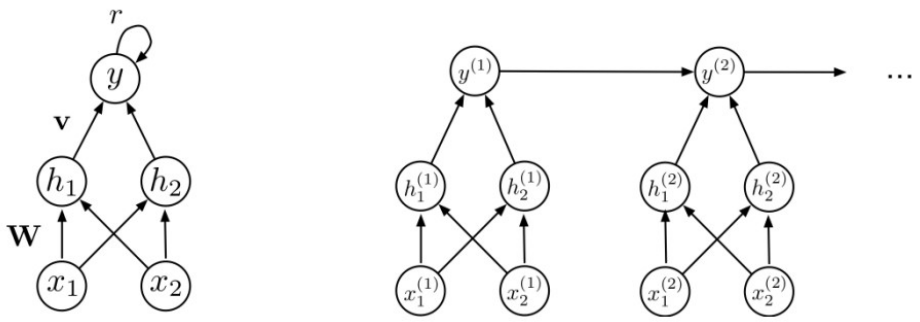
4.4 (10 Points)

What is beam search algorithm? Explain this algorithm briefly.

5 RNN Calculation

5.1 (20 points)

We want to process two binary input sequences with 0-1 entries and determine if they are equal. For notation, let $x_1 = x_1^{(1)}, x_1^{(2)}, , x_1^{(T)}$ be the first input sequence and $x_2 = x_2^{(1)}, x_2^{(2)}, , x_2^{(T)}$ be the second. We use the RNN architecture shown in the Figure.



We have the following equations in which W is a 2×2 matrix, b is a two dimensional bias vector, v is a two dimensional weight vector and r, C, C_0 are scalars.

$$h^{(t)} = g(Wx^{(t)} + b)$$

$$y^{(t)} = \begin{cases} g(v^T h^{(t)} + r y^{(t-1)} + C) & \text{if t} > 1 \\ g(v^T h^{(t)} + C_0)) & \text{if t} = 1 \end{cases}$$

$$g(z) = \begin{cases} 1 & \text{if z} > 1 \\ 0 & \text{if z} \leq 1 \end{cases}$$

Find W, b, v, r, C, C_0 such that at each step $y^{(t)}$ indicates whether all inputs have matched up to the current or not. time.

6 Word Embedding

6.1 (20 points)

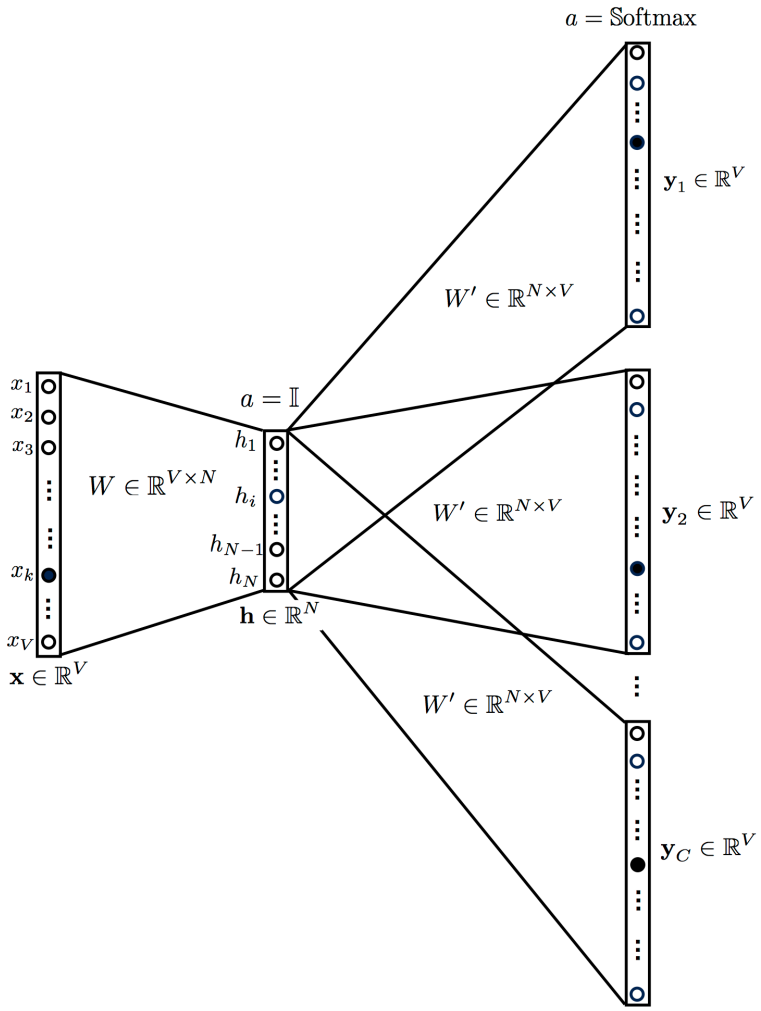
By using the following figure(next page), explain skip-gram. Your explanation must include the probability the model wants to estimate, the concept of the context window, the output of the model, and the loss function used in the training.

6.2 (20 points)

Suppose that the context window (C) is equal to 1. (For example for each center word, we just consider its left neighbor) We also denote the loss function as

$$L(x, \hat{y}, W, W')$$

In which $\hat{y}$ is the one-hot encoded vector of the word in the context window and x is the one-hot encoded vector of the center word.
Using the previous part, calculate $\frac{\partial L}{\partial w_k}$ in which w_k is the kth row of matrix W. For simplification purposes, you can assume that $\frac{\partial Softmax(y)}{\partial y}$ is given as $S'(y)$. However, we highly recommend a complete calculation.


$a = \mathbb{S}\text{oftmax}$
$\mathbf{y}_1 \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N \times V}$
x_1
x_2
x_3
x_k
x_V
$\mathbf{x} \in \mathbb{R}^V$
$a = \mathbb{I}$
h_1
h_i
h_{N-1}
h_N
$\mathbf{h} \in \mathbb{R}^N$
$W \in \mathbb{R}^{V \times N}$
$W' \in \mathbb{R}^{N \times V}$
$\mathbf{y}_2 \in \mathbb{R}^V$
$W' \in \mathbb{R}^{N \times V}$
$\mathbf{y}_C \in \mathbb{R}^V$