tion 5 Consider a self-attention layer with the following input token embeddings:

x_1: 0.0, 1.0
x_2: 1.0, 0.0
x_3: 0.5, 0.5
x_4: 0.8, 0.2
x_5: 0.2, 0.8

Let the key matrix W^K and the value matrix W^V both be the identity matrix.

(a) Suppose the query matrix is $W_1^Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. What are the attention weights when calculating the output token embeddings $z_1, \ldots, z_5$ of the self-attention layer? What are the output token embeddings.

(b) Suppose the query matrix is $W_2^Q = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. What are the attention weights when calculating the output token embeddings $z_1, \ldots, z_5$ of the self-attention layer? What are the output token embeddings.

(c) In Part (a), what would be the attention weights if the input taken embeddings were:

x_1: 0.8, 0.2
x_2: 0.2, 0.8
x_3: 0.0, 1.0
x_4: 1.0, 0.0
x_5: 0.5, 0.5

Note how the attention matrix influences the output (a vs b), and how the attention weights changes with respect to input (a vs c).

ion: **(a)** Since W_Q, W_K, W_V are all identity matrices, we have

$$Q = K = V = \begin{bmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \\ 0.5 & 0.5 \\ 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}.$$

Then,

$$\frac{QK^\top}{\sqrt{d_k}} = \begin{bmatrix} 0.7071 & 0. & 0.3536 & 0.1414 & 0.5657 \\ 0. & 0.7071 & 0.3536 & 0.5657 & 0.1414 \\ 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.1414 & 0.5657 & 0.3536 & 0.4808 & 0.2263 \\ 0.5657 & 0.1414 & 0.3536 & 0.2263 & 0.4808 \end{bmatrix}$$

and thus the attention weights are

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) = \begin{bmatrix} 0.2754 & 0.1358 & 0.1934 & 0.1564 & 0.2391 \\ 0.1358 & 0.2754 & 0.1934 & 0.2391 & 0.1564 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1598 & 0.2443 & 0.1976 & 0.2244 & 0.174 \\ 0.2443 & 0.1598 & 0.1976 & 0.174 & 0.2244 \end{bmatrix}.$$

Finally, the output token embeddings are

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V = \begin{bmatrix} 0.4054 & 0.5946 \\ 0.5946 & 0.4054 \\ 0.5 & 0.5 \\ 0.5574 & 0.4426 \\ 0.4426 & 0.5574 \end{bmatrix}.$$

Note that $\mathbf{z_1} = \mathbf{0.28x_1 + 0.16x_2 + 0.19x_3 + 0.15x_4 + 0.24x_5}$, x_1 and x_5 rec attentions.

(b) First, we have

$$Q = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.5 & 0.5 \\ 0.2 & 0.8 \\ 0.8 & 0.2 \end{bmatrix}, \qquad K = V = \begin{bmatrix} 0.0 & 1.0 \\ 1.0 & 0.0 \\ 0.5 & 0.5 \\ 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}.$$

Then,

$$\frac{QK^\top}{\sqrt{d_k}} = \begin{bmatrix} 0. & 0.7071 & 0.3536 & 0.5657 & 0.1414 \\ 0.7071 & 0. & 0.3536 & 0.1414 & 0.5657 \\ 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.5657 & 0.1414 & 0.3536 & 0.2263 & 0.4808 \\ 0.1414 & 0.5657 & 0.3536 & 0.4808 & 0.2263 \end{bmatrix}$$

and thus the attention weights are

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) = \begin{bmatrix} 0.1358 & 0.2754 & 0.1934 & 0.2391 & 0.1564 \\ 0.2754 & 0.1358 & 0.1934 & 0.1564 & 0.2391 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2443 & 0.1598 & 0.1976 & 0.174 & 0.2244 \\ 0.1598 & 0.2443 & 0.1976 & 0.2244 & 0.174 \end{bmatrix}.$$

Finally, the output token embeddings are

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V = \begin{bmatrix} 0.5946 & 0.4054 \\ 0.4054 & 0.5946 \\ 0.5 & 0.5 \\ 0.4426 & 0.5574 \\ 0.5574 & 0.4426 \end{bmatrix}.$$

Note that $\mathbf{z_1} = \mathbf{0.16x_1 + 0.28x_2 + 0.19x_3 + 0.24x_4 + 0.16x_5}$, x_2 and x_4 receive the most attentions. **(c)** Since W_Q, W_K, W_V are all identity matrices, we have

$$Q = K = V = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \\ 0.5 & 0.5 \end{bmatrix}. \tag{9}$$

Then,

$$\frac{QK^\top}{\sqrt{d_k}} = \begin{bmatrix} 0.4808 & 0.2263 & 0.1414 & 0.5657 & 0.3536 \\ 0.2263 & 0.4808 & 0.5657 & 0.1414 & 0.3536 \\ 0.1414 & 0.5657 & 0.7071 & 0. & 0.3536 \\ 0.5657 & 0.1414 & 0. & 0.7071 & 0.3536 \\ 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \end{bmatrix} \tag{10}$$

and thus the attention weights are

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) = \begin{bmatrix} 0.2244 & 0.174 & 0.1598 & 0.2443 & 0.1976 \\ 0.174 & 0.2244 & 0.2443 & 0.1598 & 0.1976 \\ 0.1564 & 0.2391 & 0.2754 & 0.1358 & 0.1934 \\ 0.2391 & 0.1564 & 0.1358 & 0.2754 & 0.1934 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}. \tag{11}$$

Note that $\mathbf{z_1} = \mathbf{0.22x_1 + 0.17x_2 + 0.16x_3 + 0.24x_4 + 0.20x_5}$. The attentions different from (a) even the query, key and value matrices are the same. The demonstrate the concept of dynamic weights.

stion 6 BERT input representation is the sum of token embedding, segment embedding, and positional embedding. Briefly explain why positional embedding is introduced in the architecture.

tion: In BERT, unlike AutoRegressive approach, input tokens are fed into the architecture at once. Without positional embedding, the input representation of a token holds no information about its position in a sentence, in other words, each token has only one input representation (assuming same segment embedding). By adding positional embedding to input representation, each token can have different representations according to its position, and holds positional information.

ion 2: In class, we discussed two loss functions for the generator G in GAN:

$$\begin{aligned} L_1 &= \frac{1}{m}\sum_{i=1}^{m}[-\log D(G(z^i))] \\ L_2 &= \frac{1}{m}\sum_{i=1}^{m}\log[1 - D(G(z^i))] \end{aligned}$$

(a) What is the common goal that both loss functions aim to achieve?

(b) What are the advantages and disadvantages of each loss function? Why? How are the disadvantages mitigated in practice?

(c) Assume the discriminator D is optimal. Which loss function is an approximation of the Jensen-Shannon divergence between the data distribution and the generator distribution?

er: (a) Both loss functions aim to maximize the probability of fake images being realistic, so as to fool the discriminator. (1 point)

(b) L_2 leads to slow training. The reason is that, at the start, D can easily distinguish between fake images from real ones. This means $D(G(z^i)) \approx 0$ and hence $\log[1 - D(G(z^i))] \approx 0$. Those remain true for small changes in G. Hence, L_2 gives small gradients for improving G. (2 points)
L_1 leads to unstable training. The reason is that, at the start, $-\log D(G(z^i))$ is large and its changes a lot for small changes in G. Hence, L_1 gives large gradients for improving G, which imply unstable training (1 point).
In practice, L_1 is used. To make the training more stable, D is not trained to optimal. (1 point)

(c) L_2.

ion 3: What are the inputs and output of the noise predictor in denoising diffusion probabilistic nodels (DDPM)? What is the loss function used to train it?

er:

s: A noisy image $\mathbf{x}_t$, and embedding of time step t

ut: An approximation $\epsilon_\theta(\mathbf{x}_t, t)$ of the compound noise $\bar{\epsilon}_t$ that was added to $\mathbf{x}_0$ to create $\mathbf{x}_t$:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t \qquad \bar{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Training Objective: $\min E_{\mathbf{x}_0, t, \bar{\epsilon}_t} ||\bar{\epsilon}_t - \epsilon_\theta(\mathbf{x}_t, t)||^2$

estion 5: Here is the parameter update rule for Deep Q-Networks:

$$\theta \leftarrow \theta - \alpha\nabla_\theta([r(s,a) + \gamma \max_{a'} Q(s', a'; \theta^-)] - Q(s, a; \theta))^2$$

What do s, a, $r(s,a)$ and s' stand for? What about θ^-? What is the objective that the update rule is intended to achieve?

ution: The tuple $(s,\ a,\ r(s,a),\ s')$ is an experience of the agent has with its environment. θ^- denotes the parameters of a target network, which is updated once in a while. The objective of the rule to change the parameters of the Q-network such that it better approximates the TD-target $r(s,a) + \gamma \max_{a'} Q(s', a'; \theta^-)$, which is an improved estimate of the value for the pair (s,a) given the experience tuple.

estion 6: Here is the update rule for the actor in the Actor-Critic algorithm:

$$\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(a|s)\hat{A}^\pi(s,a),$$

where $\hat{A}^\pi(s,a) \leftarrow r + \gamma\hat{V}_\phi^\pi(s') - \hat{V}_\phi^\pi(s)$.

What do s, a, $r(s,a)$ and s' stand for? What about $\hat{V}_\phi^\pi(s)$? Intuitively, what does the update rule try to achieve?

ution: The tuple $(s,\ a,\ r,\ s')$ is an experience of the agent has with its environment. $\hat{V}_\phi^\pi(s)$ is an estimate of the value of the state s given by the current value network. The update rules tries to increase the probabilities of the actions that lead to high rewards and decrease the probabilities of the actions that lead to low rewards.

tion 7: (a) In the context of deep image classification, what is adversarial attack?

(b) The CW attack finds an adversarial example $\mathbf{x}'$ for a benign example $\mathbf{x}$ by solving the following optimization problem:

$\min_{\mathbf{x}'} c||\mathbf{x} - \mathbf{x}'||_2^2 + l(\mathbf{x}')$
s.t. $\mathbf{x}' \in [0,1]^n$
where $l(\mathbf{x}') = \max\{\max_{i \neq t} Z_i(\mathbf{x}') - Z_t(\mathbf{x}'), -\kappa\}$.

What do the terms $Z_t(\mathbf{x}')$, $Z_i(\mathbf{x}')$ stand for? What are the key ideas behind this attack?

ion: (a) In the context of deep image classification, adversarial attack means to add small perturbations to an input image that are (almost) imperceptible to human vision system, and that cause the classifier to misclassify the input with high confidence.

(b) $Z_i(\mathbf{x}')$ is the logit of class i and t stands for the target class. The CW attack aims to maximize the logit $Z_t(\mathbf{x}')$ of the target class and to minize the logits of all other classes $\max_{i \neq t} Z_i(\mathbf{x}')$. When then difference $Z_t(\mathbf{x}') - \max_{i \neq t} Z_i(\mathbf{x}')$ becomes large enough $\geq \kappa$, it turns its attention to minimize the pertubations $||\mathbf{x} - \mathbf{x}'||_2^2$.

tion 8: In some pixel-level explanations, we need to compute $\frac{\partial z_c(\mathbf{x})}{\partial x_i}$. The backpropagation algorithm for the task is given in Lecture 16. Suppose the last layer of the model is a Softamx layer. How would you change to algorithm if we are to compute $\frac{\partial \log P(y=c|\mathbf{x})}{\partial x_i}$?

ion: Let $Z = \sum_c e^{z_c(\mathbf{x})}$. The first step of backprop should be changed as follows:

$$\begin{aligned} \frac{\partial \log P(y=c|\mathbf{x})}{\partial u_j} &= \frac{\partial \log \frac{e^{z_c(\mathbf{x})}}{Z}}{\partial u_j} \\ &= \frac{\partial z_c(\mathbf{x})}{\partial u_j} - \frac{1}{Z}\frac{\partial \sum_{c'} e^{z_{c'}(\mathbf{x})}}{\partial u_j} \\ &= w_{cj} - \sum_{c'} P(y = c'|\mathbf{x}) w_{c'j}. \end{aligned}$$

stion 1 Consider the following dataset:

Instance	y	x_1	x_2
1	1	0	0
2	1	0	0
3	1	0	1
4	1	0	1
5	0	1	0
6	0	1	0
7	1	1	1
8	0	1	1

(a) Give the Naïve Bayes model for the data. There is no need to use Laplace smoothing, and there is no need to show the process of calculation.

(b) Calculate the posterior probabilities of the Instances 1 and 7 belonging to the two classes according to the model of the previous sub-question. Show the process of calculation.

tion: (a) $p(y=0)=3/8$, $p(x_1=0|y=0)=0$, $p(x_2=0|y=0)=2/3$, $p(x_1=0|y=1)=4/5$, $p(x_2=0|y=1)=2/5$

(b)

$$\begin{aligned}
p(\mathbf{x}_1|y=0) &= p(x_1=0|y=0)p(x_2=0|y=0)=0\\
p(\mathbf{x}_1|y=1) &= p(x_1=0|y=1)p(x_2=0|y=1)=8/25\\
p(y=0|\mathbf{x}_1) &= \frac{p(y=0)p(\mathbf{x}_1|y=0)}{p(y=0)p(\mathbf{x}_1|y=0)+p(y=1)p(\mathbf{x}_1|y=1)}=0\\
p(y=1|\mathbf{x}_1) &= 1-p(y=0|\mathbf{x}_1)=1.
\end{aligned}$$

$$\begin{aligned}
p(\mathbf{x}_7|y=0) &= p(x_1=1|y=0)p(x_2=1|y=0)=1/3\\
p(\mathbf{x}_7|y=1) &= p(x_1=1|y=1)p(x_2=1|y=1)=3/25\\
p(y=0|\mathbf{x}_7) &= \frac{p(y=0)p(\mathbf{x}_7|y=0)}{p(y=0)p(\mathbf{x}_7|y=0)+p(y=1)p(\mathbf{x}_7|y=1)}=\frac{\frac{3}{8}\frac{1}{3}}{\frac{3}{8}\frac{1}{3}+\frac{5}{8}\frac{3}{25}}=\frac{5}{8}\\
p(y=1|\mathbf{x}_7) &= 1-p(y=0|\mathbf{x}_7)=\frac{3}{8}.
\end{aligned}$$

estion 4: Why is the sigmoid activation function not recommended for hidden units, but it is fine for an output unit.

ution: The sigmoid activation function $\sigma(z)=\frac{1}{1+exp(-z)}$ is not recommended for hidden units because it saturates across most of its domain. In other words, its derivative is usually small, preventing errors to back-propagate to units at previous layers.

It is fine for an output unit because of the use of negative log-likelihood (i.e., cross entropy) as the loss function. As such, we back-propagate the gradient of the **logarithm of a sigmoid function**, i.e., $-\log\sigma((2y-1)z)$, instead of the gradient of a sigmoid function itself. The function $-\log\sigma((2y-1)z)$ saturates only when the training example is classified correctly by the model.

estion 5: What is dropout used for in deep learning? Why does it work? Answer briefly.

ution: Dropout is regularization technique used in deep learning to avoid overfitting. It associates a binary mask variable with some of the units. During training, values for the mask variables are randomly sampled for each minibatch of data, and only parameters for the units with mask variable taking value 1 are oupdated on the minibatch. It reduces overfitting by preventing complex co-adaptation of parameters.

estion 6: What are the key ideas behind the Adam algorithm for training deep neural networks? Answer briefly.

ution: There are three key ideas: The use of momentum to accelerate learning; Adaption of learning rate to slow down changes on parameters that have changed a lot before; The correction of bias in moment estimates.

tion 1: Consider the image X and filter F given below. Let X be convolved with F using no padding and a stride of 1 to produce an output Y. Assume the bias is 2 and the activation function is ReLU. What are values of the cells a, b, e and f in the output Y?

$X =$

1	0	-2	3	4	1
2	9	5	6	0	-1
0	-3	1	3	4	4
6	5	2	0	6	8
-5	4	-3	1	3	-2
4	1	2	8	9	7

$F =$

-1	-1	-1
-1	8	-1
-1	-1	-1

$Y =$

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

As given $X=\begin{bmatrix}1&0&-2&3&4&1\\2&9&5&6&0&-1\\0&-3&1&3&4&4\\6&5&2&0&6&8\\-5&4&-3&1&3&-2\\4&1&2&8&9&7\end{bmatrix}$, and $F=[\begin{matrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{matrix}]$

Element a (Top-left corner of Y), the convolution:

$$\begin{aligned}&=(-1\cdot1)+(-1\cdot0)+(-1\cdot-2)+(-1\cdot2)+(8\cdot9)+(-1\cdot5)+(-1\cdot0)+(-1\cdot-3)\\&\quad+(-1\cdot1)=-1+0+2-2+72-5+0+3-1=68\end{aligned}$$

Adding the bias: $a=ReLU(68+2)=ReLU(70)=70$

Element b (Top-middle of Y), the convolution:

$$\begin{aligned}&=(-1\cdot0)+(-1\cdot-2)+(-1\cdot3)+(-1\cdot9)+(8\cdot5)+(-1\cdot6)+(-1\cdot-3)+(-1\cdot1)\\&\quad+(-1\cdot4)=0+2-3-9+40-6+3-1-4=22\end{aligned}$$

Adding the bias: $b=ReLU(22+2)=ReLU(24)=24$

Element e (Middle-left of Y), the convolution:

$$\begin{aligned}&=(-1\cdot2)+(-1\cdot9)+(-1\cdot5)+(-1\cdot0)+(8\cdot-3)+(-1\cdot1)+(-1\cdot6)+(-1\cdot5)\\&\quad+(-1\cdot2)=-2-9-5+0-24-1-6-5-2=-53\end{aligned}$$

Adding the bias: $e=ReLU(-53+2)=ReLU(-51)=0$

Element f (Middle-middle of Y), the convolution:

$$\begin{aligned}&=(-1\cdot9)+(-1\cdot5)+(-1\cdot6)+(-1\cdot-3)+(8\cdot1)+(-1\cdot3)+(-1\cdot5)+(-1\cdot2)\\&\quad+(-1\cdot0)=-9-5-6+3+8-3-5-2+0=-19\end{aligned}$$

Adding the bias: $f=ReLU(-19+2)=ReLU(-17)=0$

Thus, $a=70, b=24, e=f=0$.

stion 2 [Optional]
Suppose there are K i.i.d training sets $S_k=\{\mathbf{x}_{ki},y_{ki}\}_{i=1}^m$ $(k=1,\dots,K)$ for a regression problem with a hypothesis class $\mathcal{H}$. For each k, let

$$h_k=\arg\min_{h\in\mathcal{H}}\hat{\epsilon}(h),\ \text{where}\ \hat{\epsilon}(h)=\frac{1}{m}\sum_{i=1}^m(y_{ki}-h(\mathbf{x}_{ki}))^2$$

The variance component of the expected error of h_k is:

$$\text{Var}(h_k)=E_{\mathbf{x}}E_{S_k}[E_{S_k}(h_k(\mathbf{x}))-h_k(\mathbf{x})]^2.$$

Because the training sets are i.i.d, $\text{Var}(h_k)$ is the same for different k. Let $\text{Var}(h_k)=\sigma^2$.

(a) Let $\bar{h}=\frac{1}{K}\sum_{k=1}^K h_k$. Show that the variance component of the expected error of $\bar{h}$ is:

$$\text{Var}(\bar{h})=\frac{1}{K}\sigma^2.$$

(b) Based on part (a), a variance reduction technique called **bagging** is proposed. Find out how bagging works, and explain why it reduces variance.

tion: (a) Let $e_k(\mathbf{x})=E_{S_k}[h_k(\mathbf{x})]-h_k(\mathbf{x})$. Then, different e_k's are independent of each other, and $E_{S_k}[e_k(\mathbf{x})]=0$. Hence, we have:

$$\begin{aligned}
\text{Var}(\bar{h}) &= E_{\mathbf{x}}E_{S_1,\dots,S_k}[E_{S_1,\dots S_k}(\bar{h}(\mathbf{x}))-\bar{h}(\mathbf{x})]^2\\
&= E_{\mathbf{x}}E_{S_1,\dots,S_k}[E_{S_1,\dots S_k}([\frac{1}{K}\sum_{k=1}^K h_k(\mathbf{x}))-\frac{1}{K}\sum_{k=1}^K h_k(\mathbf{x})]^2\\
&= E_{\mathbf{x}}E_{S_1,\dots,S_k}[\frac{1}{K}\sum_{k=1}^K e_k]^2\\
&= \frac{1}{K^2}E_{\mathbf{x}}[\sum_{k=1}^K E_{s_k}[e_k]^2+\sum_{j\neq k}E_{s_j}[e_j]E_{s_k}[e_k]] \qquad \text{(independence)}\\
&= \frac{1}{K^2}\sum_{k=1}^K E_{\mathbf{x}}E_{s_k}[e_k]^2 \qquad (E_{S_k}[e_k(\mathbf{x})]=0)\\
&= \frac{1}{K^2}\sum_{k=1}^K\sigma^2\\
&= \frac{1}{K}\sigma^2.
\end{aligned}$$

(b) In bagging, we start with one training set $S=\{\mathbf{x}_i,y_i\}_{i=1}^m$. K training sets S_1, …, S_K of the same sample size are created by randomly sampling from S with replacement. Those training sets are called *bootstrap samples*. A regression function h_k is learned from each bootstrap sample, and their average $\bar{h}=\frac{1}{K}\sum_{k=1}^K h_k$ is the final output regressor.
For simplicity, assume that each bootstrap sample consists of m data points. Let h be the regressor learned from S. Then, $\text{Var}(h)\approx\text{Var}(h_k)=\sigma^2$. However, $\text{Var}(\bar{h})\approx\frac{1}{K}\sigma^2$. Hence the variance is reduced by a factor of K.
Note that the conditions of Part (a) are not strictly satisfied, and $\text{Var}(h)>\text{Var}(h_k)$. Hence, the actual reduction is less than the factor of K.

stion 3: What is batch normalization? What is layer normalization? What are their pros and cons?

tion: Batch normalization is a technique to stabilize the learning process of deep neural network. To be specific, each batch of data could possess different distribution which causes "internal covariate shift". Batch normalization normalizes each batch of data in order to avoid the problem by using scaling factor and shifting factor, and in turn, results in stabilized learning process as well as faster convergence. However, there are some cons in this technique. First is the dependency on the size of batch. Since batch normalization is manipulating data of each batch, the data in batch decides the value of scaling and shifting factor. Therefore, careful choice of the batch size is crucial, and of course, when batch size is 1, batch normalization cannot be applied. In addition, at training phase, the scaling and shifting factor have to be identified and kept, since the values will be used in testing phase as well.

Batch normalization is not suitable for RNN because it destroys sequential dependencies, which are important for NLP. It also requires a lot of memory as statistics for each time step have to be computed and stored. Furthermore, if given a longer sentence in testing than in training data, it is not possible to apply batch normalization considering the absent of statistics.

Layer normalization is related to batch normalization, but different in that it normalizes the inputs across features. Since, layer normalization normalizes inputs across features, any number of batch size could be used. In addition, past experiments show promising results on Recurrent Neural Network models with layer normalization.

ion 4 In an LSMT cell, $\mathbf{h}^{(t)}$ is computed from $\mathbf{h}^{(t-1)}$ and $\mathbf{x}^{(t)}$ using the following formulae:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}^{(t)}+\mathbf{U}_f\mathbf{h}^{(t-1)}+\mathbf{b}_f)\\
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}^{(t)}+\mathbf{U}_i\mathbf{h}^{(t-1)}+\mathbf{b}_i)\\
\mathbf{o}_t &= \sigma(\mathbf{W}_q\mathbf{x}^{(t)}+\mathbf{U}_q\mathbf{h}^{(t-1)}+\mathbf{b}_q)\\
\mathbf{c}_t &= \mathbf{f}_t\otimes\mathbf{c}_{t-1}+\mathbf{i}_t\otimes\tanh(\mathbf{U}\mathbf{x}^{(t)}+\mathbf{W}\mathbf{h}^{(t-1)}+\mathbf{b})\\
\mathbf{h}^{(t)} &= \mathbf{o}_t\otimes\tanh(\mathbf{c}_t)
\end{aligned}$$

(a) Intuitively, what are the functions of the forget gate $\mathbf{f}_t$ and the input gate $\mathbf{i}_t$ do? Answer briefly.

(b) Why do we use the sigmoid function for $\mathbf{f}_t$ and $\mathbf{i}_t$, but tanh for the memory cell $\mathbf{c}_t$ and the output $\mathbf{h}^{(t)}$? Answer briefly.

on: (a) The forget gate $\mathbf{f}_t$ determines which components of the previous state $\mathbf{c}_{t-1}$ and how much of them to remember/forget. The input gate $\mathbf{i}_t$ determines which components of the input from $\mathbf{h}^{(t-1)}$ and $\mathbf{x}^{(t)}$ and how much of them should go into the current state.

(b) The sigmoid function is used for $\mathbf{f}_t$ and $\mathbf{i}_t$ so that their values are likely to be close to 0 or 1, and hence mimicking the close and open of gates. The tanh function is used for $\mathbf{c}_t$ and $\mathbf{h}^{(t)}$ so that strong gradient signals can be backpropagated from $\mathbf{h}^{(t)}$ to $\mathbf{c}_t$, and then to $\mathbf{h}^{(t-1)}$.

stion 1: Here is the objective function of VAE:

$$\mathbf{L}(\mathbf{x}^{(i)},\theta,\phi)=E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]-KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})]$$

Explain why the first term is called the reconstruction error.

wer: In VAE, we

1. Start with an input training example $\mathbf{x}^{(i)}$ in the data space.
2. Map it to a distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ over a lower-dimensional latent space.
3. Sample a vector $\mathbf{z}$ from the distribution.
4. Map $\mathbf{z}$ back to a distribution $p_\theta(\mathbf{x}|\mathbf{z})$ over the data space.

How well does the resulting distribution match the training example $\mathbf{x}^{(i)}$? (In other words, how well is the input reconstructed from $\mathbf{z}$?) The answer is $\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$. How well can the input be reconstructed from the distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$? The answer is the first term of $\mathbf{L}$.

Note that it might more appropriate the call the first term the reconstruction loglikelihood. The "reconstruction error" or "reconstruction loss" should be $E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[-\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$.