IMPERIAL COLLEGE LONDON

TIMED REMOTE ASSESSMENTS 2021-2022

MEng Honours Degree in Electronic and Information Engineering Part IV
MEng Honours Degree in Mathematics and Computer Science Part IV
MEng Honours Degrees in Computing Part IV
MSc Advanced Computing
MSc Artificial Intelligence
MSc in Computing (Specialism)
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant assessments for the Associateship of the City and Guilds of London Institute

PAPER COMP70010=COMP97111=COMP97112

DEEP LEARNING

Wednesday 23 March 2022, 14:00
Writing time: 90 minutes
Upload time: 25 minutes

Answer ALL TWO questions
Open book assessment

This time-limited remote assessment has been designed to be open book. You may use resources which have been identified by the examiner to complete the assessment and are included in the instructions for the examination. You must not use any additional resources when completing this assessment.

The use of the work of another student, past or present, constitutes plagiarism. Giving your work to another student to use constitutes an offence. Collusion is a form of plagiarism and will be treated in a similar manner. This is an individual assessment and thus should be completed solely by you. The College will investigate all instances where an examination or assessment offence is reported or suspected, using plagiarism software, vivas and other tools, and apply appropriate penalties to students. In all examinations we will analyse exam performance against previous performance and against data from previous years and use an evidence-based approach to maintain a fair and robust examination. As with all exams, the best strategy is to read the question carefully and answer as fully as possible, taking account of the time and number of marks available.

Paper contains 2 questions

1 Convolutions and Networks

a Please answer these multiple choice questions. Each question has at least one correct option unless explicitly mentioned. No explanation is required.

i) Which of the following are true about Batch Normalization?

☐ **Batch Norm layers are skipped at test time because a single test example cannot be normalized.**

☐ **Its learnable parameters can only be learned using gradient descent or mini-batch gradient descent, but not other optimization algorithms.**

☐ **It helps speed up learning in the network, *e.g.*, with respect to instance normalisation.**

☐ **It introduces noise to a hidden layer's activation, because the mean and the standard deviation are estimated with a mini-batch of data.**

ii) If your input image is 64x64x16, how many parameters are there in a single 1x1 convolution filter, including bias?

☐ **2**

☐ **1**

☐ **4097**

☐ **17**

iii) The shape of your input image is (n_h, n_w, n_c); the convolution layer uses a 1-by-1 filter with stride = 1 and padding = 0. Which of the following statements are correct?

☐ **You can reduce n_c by using 1×1 convolution. However, you cannot change n_h, n_w.**

☐ **You can use a standard max pooling to reduce n_h, n_w, but not n_c.**

☐ **You can use a 1×1 convolution to reduce n_h, n_w, n_c.**

☐ **You can use maxpooling to reduce n_h, n_w, n_c.**

iv) Which of the following would you consider to be valid activation functions (element-wise non-linearities) to train a neural net in practice?

(1) $f(x) = 0.8x + 1$

(2) $f(x) = \begin{cases} min(x, 0.1x), & x >= 0 \\ min(x, 0.1x) & x < 0 \end{cases}$

(3) $f(x) = \begin{cases} max(x, 0.1x), & x >= 0 \\ min(x, 0.1x) & x < 0 \end{cases}$

(4) $f(x) = -min(2, x)$

v) You are benchmarking runtimes for layers commonly encountered in CNNs. Which of the following would you expect to be the fastest (in terms of floating point operations)?

- ☐ **Max pooling**
- ☐ **Conv layer (convolution operation + bias addition)**
- ☐ **Average pooling**
- ☐ **Batch Normalization**

vi) Your model for classifying different virus variants is getting a high training set error. Which of the following are promising things to try to improve your classifier?

- ☐ **Increase the regularization parameter lambda**
- ☐ **use a bigger neural network**
- ☐ **use a lower dropout probability (assume the classifier has dropout layers)**
- ☐ **Get more training data**

vii) You are designing a deep learning system to diagnose chest cancer through X-ray images. What do you think might be the two most appropriate evaluation metrics?

- ☐ **Accuracy**
- ☐ **Precision**
- ☐ **Recall**
- ☐ **F1 score**

b Please provide short answers.

i) Why is scaling (γ) and shifting (β) often applied after the standard normalization in the batch normalization layer?

ii) You train a classification model for a user until it achieves $> 95\%$ accuracy on a held-back development set (for same user). However, upon deployment you get complaints the model fails to correctly work about half the time (50% misclassification rate). List one factor you think could have

contributed to the mismatch in misclassification rates between the dev set and deployment, and how you would try to fix this issue.

iii) What is the purpose of using 1×1 convolution?

iv) Why is the sigmoid activation function susceptible to the vanishing gradient problem?

c Consider a convolutional neural network block whose input size is $64 \times 64 \times 8$. The block consists of the following layers:

- A convolutional layer 32 filters, all stride 1, with height and width 3 and 0 padding which has both a weight and a bias (*i.e.*, CONV3-32)
- A 2×2 max-pooling layer with stride 2 and 0 padding (*i.e.*, POOL-2)
- A batch normalization layer (*i.e.*, BATCHNORM)

Compute the output activation volume dimensions and number of parameters of the layers. You can write the activation shapes in the format (H, W, C) where H, W, C are the height, width, and channel dimensions, respectively.

i) What are the output activation volume dimensions and number of parameters for CONV3-32?

ii) What is the output activation volume dimensions and number of parameters for POOL2?

iii) What is the output activation volume dimensions and number of parameters for BATCHNORM?

d You want to build a start-up's minimally viable product (MVP). The MVP model can distinguish RBG images of hot dogs (label: 1) from RBG images of hamburgers (label: 0) using a deep neural network with the following network architecture

$$
\begin{aligned}
z_1 &= W_1 x^{(i)} + b_1 \\
\alpha_1 &= RELU(z_1) \\
z_2 &= W_2 \alpha_1 + b_2 \\
\hat{y}^{(i)} &= \sigma(z_2) \\
\mathcal{L}^{(i)} &= \alpha \cdot y^{(i)} \cdot log(\hat{y}^{(i)}) + \beta \cdot (1 - y^{(i)}) \cdot log((1 - \hat{y}^{(i)})) \\
J &= -\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}^{(i)}
\end{aligned}
\tag{1}
$$

The dimensions are as follows:

$\hat{y}^{(i)} \in \mathbb{R}$

$y^{(i)} \in \mathbb{R}$

$x^{(i)} \in \mathbb{R}^{D_x \times 1}$

$W_1 \in \mathbb{R}^{D_{\alpha_1} \times D_x}$

$W_2 \in \mathbb{R}^{1 \times D_{\alpha_1}}$

Note that N is the size of the dataset and that the RGB images are flattened into vectors of length D_x before being fed into the network.

i) What are the dimensions of b_1 and b_2?

ii) Why are α and β useful?

iii) Unfortunately, your data set is imbalanced. The class counts are:

10000 images with a hot dog

1000 examples with a hamburger

What is a reasonable pair of values for (α, β)? Provide specific values for these weightings.

iv) What is $\frac{\partial J}{\partial \hat{y}}$?

v) With the architecture in d you approximate a function space $f(x)$. You decide to experiment with a different model architecture and choose to integrate residual neural network (ResNet) blocks. Which function (space) does a ResNet block approximate?

The four parts carry, respectively, 35%, 25%, 15%, and 25% of the marks.

2 Generative models, RNNs and Attention

a Consider a VAE with the graphical model visualised below:

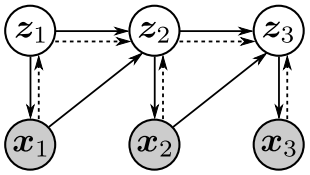

Fig. 1: Graphical model visualisation of p (solid arrows) and q (dashed arrows).

i) Write down the factorisation structure of $p(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3)$ and $q(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3 | \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ (in terms of product of conditional distributions). Assume the neural networks for the conditional q distributions are flexible enough. Would the q distribution be able to approximate the exact posterior $p(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3 | \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ well and why? Explain in 1-2 sentences.

ii) Consider a data distribution $p_{\text{data}}(\boldsymbol{x})$ and a VAE with the generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ and encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. Assume the neural networks are flexible enough so that the global optimum $q^*(\boldsymbol{z}|\boldsymbol{x}), p^*(\boldsymbol{z}|\boldsymbol{x})$ of the VAE objective can be achieved. What is the result for $\int q^*(\boldsymbol{z}|\boldsymbol{x}) p_{\text{data}}(\boldsymbol{x}) d\boldsymbol{x}$?

iii) Which of the following statements are TRUE for VAEs?

☐ **Gaussian distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is the best choice for the encoder**

☐ **We can parameterise the variance of $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ as $\sigma = \textbf{Softplus}(\textbf{NN}_\phi(\boldsymbol{x}))$**

☐ **A VAE becomes a deterministic auto-encoder if trained without the KL regulariser**

☐ **A VAE is less likely to suffer from mode collapse due to the auto-encoder architecture design**

☐ **Test log-likelihood can be low even when reconstruction error is small**

b Consider designing a CycleGAN-style model as visualised in Fig. 2. It contains two mapping functions between domains X and Y, namely $G : X \to Y$ and $F : Y \to X$. Apart from the adversarial losses associated with discriminators D_X and D_Y, the approach also requires a "cycle-consistency loss" to ensure that $F(G(\boldsymbol{x})) \approx \boldsymbol{x}$ and $G(F(\boldsymbol{y})) \approx \boldsymbol{y}$. We assume the data distributions on the X, Y domains are $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{data}}(\boldsymbol{y})$ respectively, and we use θ to denote the parameters of the neural networks G and F.

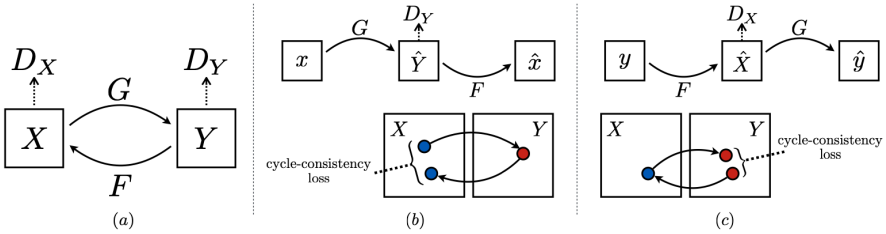

Fig. 2: Illustration of CycleGAN, taken from Zhu et al.

i) Provide your design for the cycle-consistency losses based on reconstruction error.

ii) Provide an alternative adversarial loss for the cycle-consistencies. You might need to introduce extra discriminators.
(Hint: can you relate this question to conditional GANs?)

c Consider the following RNN to process the input sequence $\boldsymbol{x}_{1:T}$:

$$\boldsymbol{h}_t = \sigma(W_h \boldsymbol{h}_{t-1} + W_x \boldsymbol{x}_t + \boldsymbol{b}_h), \quad t = 1, ..., T$$

with the convention that $\boldsymbol{h}_0 = \boldsymbol{0}$, and $\sigma(\cdot)$ is the activation function. Assume the final output is $\boldsymbol{y} = \boldsymbol{h}_T$ and there is a loss function $\mathcal{L}(\boldsymbol{y})$ applied to it.

i) What is the fully expanded form of $\frac{d\mathcal{L}(\boldsymbol{y})}{dW_x}$?

ii) Assume $\sigma(\cdot)$ is identity mapping, $\boldsymbol{h}_t \in \mathbb{R}^{3\times 1}$ and W_h is

$$W_h = \begin{pmatrix} -1.12 & 0.73 & 0.98 \\ 0.02 & -0.56 & 0.24 \\ 0.87 & -2.34 & 1.56 \end{pmatrix}.$$

State whether gradient vanishing/explosion would happen when $T \to +\infty$ and explain briefly why.

iii) Which of the following solutions can help address RNN's gradient vanishing/explosion problem in practice?

- ☐ **Use leaky ReLU activations**
- ☐ **Normalise the column vectors in W_h to have ℓ_2 norm 1**
- ☐ **Truncated back-propagation through time**
- ☐ **Use matrix W_h with orthogonal row vectors**
- ☐ **Use dropout during training**

d Attention mechanisms & Transformers:

i) Consider scaled dot-product hard attention: compute its output using the following inputs:

$$Q = \begin{pmatrix} 3 & 5 \\ -1 & 4 \\ 2 & -3 \end{pmatrix}, \quad K = \begin{pmatrix} 0 & 2 \\ 4 & -6 \end{pmatrix}, \quad V = \begin{pmatrix} 3 & 1 & -4 & 2 \\ 6 & -2 & 4 & 5 \end{pmatrix}.$$

ii) Let us compare the computational efficiency of a convolutional layer in CNN vs an attention layer in Vision Transformer. Assume an input grayscale image (thus only one channel) has size $H \times W$.

The convolution layer applies a filter of spatial size $h \times w$ and c channels directly to this image (with stride=1 and no padding).

On the other hand, the single-head attention layer first splits the image into non-overlapping patches of size $h \times w$ (assuming $H \bmod h = 0$ and $W \bmod w = 0$), and then flattens these patches as query/key vectors to perform self-attention. The value vectors of the attention layer is obtained by a linear projection of the queries with an $hw \times v$ matrix.

What are the computational complexities for the above two transformations? Comparing different settings of the hyper-parameters c and v, when would the convolution layer be more computationally efficient than the attention layer?

The four parts carry, respectively, 30%, 20%, 25%, and 25% of the marks.