Marking Scheme TEK5040/9040 H2022 Exam

1 Field of view (weight 12%)

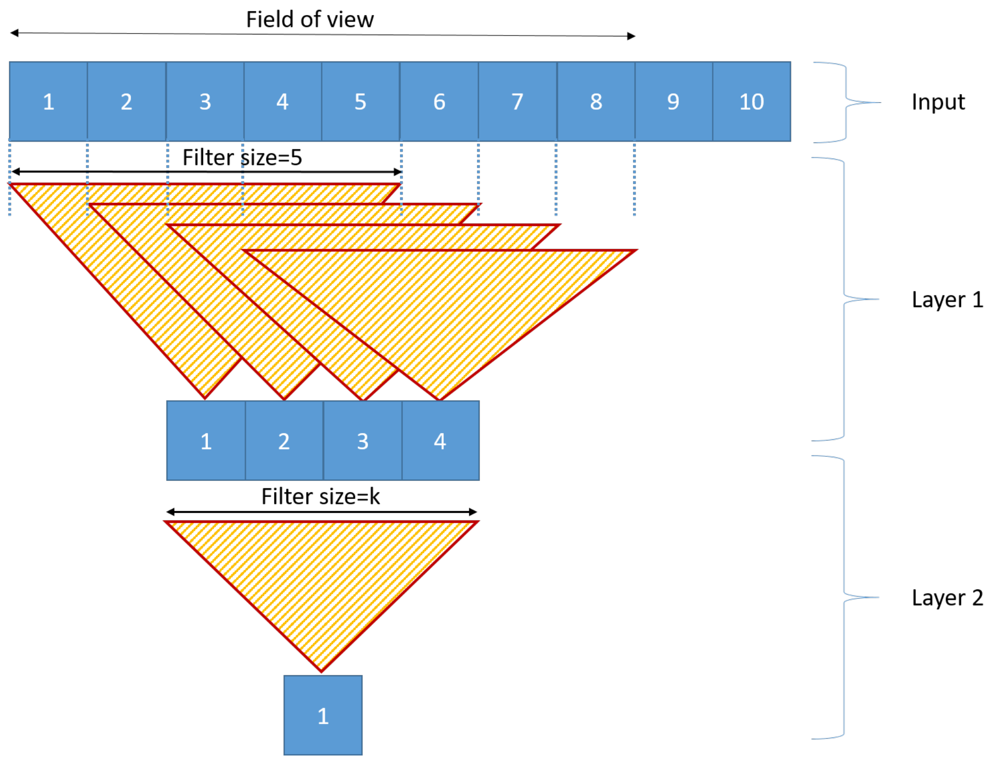

Figure 1: Field of view

Answer: k = 4

1.1 Marks

- **12 points** if the answer is correct
- If the answer is incorrect and there is illustration or description:
  - **6 points** if the first layer is correct (Reduce proportionally for missing/incorrect parts)
  - **6 points** if the second layer is correct (Reduce proportionally for missing/incorrect parts)

2 Self-attention (weight 12%)

The new output sequence is $\boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_1$

Explanation: Output order changes in the same way as the input order because of symmetry of the self-attention operation (permutation equi-variance).

Alternatively, one can write the equations assuming query $\boldsymbol{q}_i$ and key $\boldsymbol{k}_i$ are derived from $\boldsymbol{x}_i$ for $i = 1, 2, 3$. Then for the original order, output corresponding to $\boldsymbol{x}_1$ would be $\boldsymbol{z}_1 = \alpha_1(1) \cdot \boldsymbol{x}_1 + \alpha_1(2) \cdot \boldsymbol{x}_2 + \alpha_1(3) \cdot \boldsymbol{x}_3$,

where $\alpha_1(i) = \frac{\boldsymbol{q}_1 \cdot \boldsymbol{k}_i}{\sum_{j=1}^{3} \boldsymbol{q}_1 \cdot \boldsymbol{k}_j}$.
When the order is changed the output corresponding to $\boldsymbol{x}_1$ would be $\boldsymbol{z}'_1 = \alpha'_1(2) \cdot \boldsymbol{x}_2 + \alpha'_1(3) \cdot \boldsymbol{x}_3 + \alpha'_1(1) \cdot \boldsymbol{x}_1$, where $\alpha'_1(i) = \frac{\boldsymbol{q}_1 \cdot \boldsymbol{k}_i}{\sum_{j=1}^{3} \boldsymbol{q}_1 \cdot \boldsymbol{k}_j}$. Then we see that $\alpha'_1(i) = \alpha_1(i)$ for $i = 1, 2, 3$ and hence $\boldsymbol{z}'_1 = \boldsymbol{z}_1$.
Same applies for outputs corresponding to $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$. Therefore, outputs change in the same order as the inputs.

Remedy for the computational complexity:

- Consider only a neighbourhood of each input element rather than the whole sequence in calculating the self-attention

Disadvantage of the remedy:

- Remedy will reduce the field of view. Therefore we need to stack several self-attention layers in order to increase the field-of-view (for example to cover the whole sequence). Hence the maximum path length (i.e. the maximum number of operations between an output element and an input element) would be dependent on the length of the input sequence.

2.1 Marks

- **4 points** to the correct output sequence
- **2 points** to the explanation. Mentioning at least one of symmetry, permutation equi-variance or equivalent is enough.
- **3 points** to the remedy. (Look for other advanced techniques)
- **3 points** to mentioning at least one of reduced field of view, higher maximum path length or equivalent

3 Policy Gradients (weight 14%)

Sub-question 1
Back-propagating $R(\tau)$ would be unsuccessful, because

- The student cannot back-propagate through sampling operations, because there is no smooth relationship between the probability distribution parameters and the samples.
- He/she might not be able to back-propagate through the environment if the environment model is unknown.
- He/she might not be back-propagate through the reward function is it is not differentiable.

Sub-question 2
Gradients of the expected reward with respect to the policy network parameters can be calculated by back-propagating

$$\log(\pi_\theta(a_t|s_t)R(\tau)$$

at each time step t. In this case a_t is the sampled action and $\pi_\theta(a_t|s_t)$ is the probability of the sample action with respect to the current policy distribution $\pi_\theta(\cdot|s_t)$. Gradients are accumulated over the time steps and empirical average is calculated over different trajectories.

Alternatively one can back propagate $\log(\pi_\theta(a_t|s_t)$ and multiply the gradients with $R(\tau)$.

Sub-question 3
$\log(\pi_\theta(a_t|s_t)$ is equivalent to cross-entropy loss in maximum likelihood supervised training. Therefore $\nabla_\theta \log(\pi_\theta(a_t|s_t)$ is equivalent to the gradients based on cross entropy loss. The difference in policy gradients is the scaling factor $R(\tau)$. When $R(\tau)$ is positive, we scale the gradients in the positive direction, meaning that trajectories with positive rewards are made more likely. Conversely, when $R(\tau)$ is negative, we scale the gradients in the negative direction, meaning that trajectories with negative rewards are made less likely.

3.1 Marks

- **4 points** for sub-question 1, for at least one of the possible reasons or equivalent
- **4 points** for sub-question 2, for mentioning either $\log(\pi_\theta(a_t|s_t)R(\tau)$ or $\log(\pi_\theta(a_t|s_t)$.
- **6 points** for sub-question 3. Split: 2 points for identifying the scale factor $R(\tau)$. 2 point each for positive and negative scaling

4 Generative Adversarial Networks (weight 12%)

- We can apply a conditional GAN to solve this problem, because "paired" data available. In this case a *data sample* would be an image and the *condition* would be the corresponding description word. Figure 2 shows a suitable architecture. In training, we train the discriminator to maximize the conditioned real data likelihood $\log(D(x_{real}, c)$ and to minimize the conditioned fake data likelihood $\log(D(x_{fake}, c)) = \log(D(G(z, c), c))$ (or equivalently maximize $\log(1 - D(x_{fake}, c)) = \log(1 - D(G(z, c), c)))$. The generator is trained maximize the conditional fake data likelihood $\log(D(x_{fake}, c)) = \log(D(G(z, c), c))$
- Once we have trained the system, we can use the generator, to generate an image by inputting a noise vector z and the given descriptor word c. We can generate a diverse set of images by inputting different noise vectors drawn from the same distribution (eg: unit Gaussian distribution) together with the given description word.
- GANs are trained using a min-max type of objective, whereas diffusion models are trained through plain minimization of an objective. As a result, training progress is easier to interpret and control in diffusion models. Therefore, diffusion models are less susceptible to mode collapse and the issues of generation diversity.

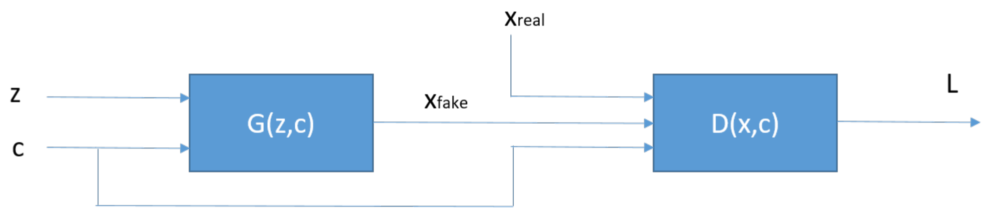

Figure 2: Conditional GAN

4.1 Marks

- Conditional GAN: Total **8 points**
  - **1 point** for proposing a conditional GAN.
  - **2 point** for mentioning condition=word description, data_sample= image.

  - **1 points** for stating the training objective.
  - **4 points** for the correct figure
- Diversity: Total **2 points** for mentioning diverse noise vectors
- Advantage of diffusion model: Total **2 points** for mentioning at least one of easier training, less mode collapse or better diversity.

5 Learning Concepts (weight 12%)

a. (10%) Meta-Learning
k-class n-shot learning means that a learning system learns to classify a given input to one of the k-classes using only n labelled data examples from each of the k-classes.

In a 5-class one shot learning problem, we have 5 classes and each class have only one example. In meta-training we should have a data-set $D_{meta-train} = \{(D^{(n)}_{support}, D^{(n)}_{query}) |\ n = 1, 2, \cdots, N\}$ where the each support set

$$D^{(n)}_{support} = \{(\boldsymbol{x}^{(n)}_1, c^{(n)}_1), (\boldsymbol{x}^{(n)}_2, c^{(n)}_2), (\boldsymbol{x}^{(n)}_3, c^{(n)}_3), (\boldsymbol{x}^{(n)}_4, c^{(n)}_4), (\boldsymbol{x}^{(n)}_5, c^{(n)}_5)\}$$

and the query set

$$D^{(n)}_{query} = \{(\boldsymbol{x}^{(n)}, c^{(n)})\}$$

where $c^{(n)} \in C^{(n)} = \{c^{(n)}_1, c^{(n)}_2, c^{(n)}_3, c^{(n)}_4, c^{(n)}_5\}$

An example $D^{(1)}_{support}$ could have $C^{(1)} = \{\text{volvo, toyota, audi, kia, bmw}\}$ and $\boldsymbol{x}^{(n)}_i,\ i = 1, 2, 3, 4, 5$ are images of cars of corresponding makes. The query set input $\boldsymbol{x}^{(n)}$ should be an image of a car of one of those makes. Another support set $D^{(2)}_{support}$ can contain completely different set of classes, for example $C^{(2)} = \{\text{cat, dog, ox, lion, elephant}\}$

The meta-testing should have at least one support set $\hat{D}_{support}$ and a query image $\hat{\boldsymbol{x}}$. In this case too $\hat{D}_{support}$ takes a similar form:

$$\hat{D}_{support} = \{(\hat{\boldsymbol{x}}_1, \hat{c}_1), (\hat{\boldsymbol{x}}_2, \hat{c}_2), (\hat{\boldsymbol{x}}_3, \hat{c}_3), (\hat{\boldsymbol{x}}_4, \hat{c}_4), (\hat{\boldsymbol{x}}_5, \hat{c}_5)\}$$

But the class set $\hat{C} = \{\hat{c}_1, \hat{c}_2, \hat{c}_3, \hat{c}_4, \hat{c}_5\}$ can be completely different, for example it could be $\{\text{circle, triangle, square, rectangle, hexagon}\}$ and the query image $\hat{\boldsymbol{x}}$ is classified to one of these classes.

b. (2%) Active Learning
Samples that are most difficult to classify are considered the most valuable samples. These samples can be predicted by calculating quantities such as confidence, classification margin, entropy and variance of the outputs.

5.1 Marks

- Meta-learning: **Total 10 points**
  - **2 points** for correct k-class n-shot description
  - **4 points** for acceptable support set in meta-training
  - **2 points** for acceptable query set for meta-training
  - **2 points** for acceptable support set and query set (or test input) for meta-testing
- Active learning: **2 points** either for mentioning difficult-to-classify samples or one of the techniques (i.e. confidence, classification margin, entropy or variance)

6 Bayesian Deep Learning (weight 12%)

a. (6%) Bayesian Deep Learning

Output variance can be used as a confidence measure in safety-critical applications such as autonomous vehicles and medical diagnostics. When the variance is low, confidence of the output is considered high and vice-versa.

In most cases of practical deep learning, we cannot evaluate the integral analytically because $p(\boldsymbol{w}|\boldsymbol{D})$ does not have a convenient functional form. Therefore we can try use sampling based numerical techniques. However, in this case too we need to draw samples from $p(\boldsymbol{w}|\boldsymbol{D})$. We can then use Bayes Formula to compute this posterior distribution:

$$p(\boldsymbol{w}|\boldsymbol{D}) = \frac{p(\boldsymbol{D}|\boldsymbol{w})p(\boldsymbol{w})}{\int p(\boldsymbol{D}|\boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}}$$

or for the discrete case

$$P(\boldsymbol{w}|\boldsymbol{D}) = \frac{p(\boldsymbol{D}|\boldsymbol{w})P(\boldsymbol{w})}{\sum_{\boldsymbol{w}} p(\boldsymbol{D}|\boldsymbol{w})P(\boldsymbol{w})}$$

Denominator on the right hand side of the formula is difficult to evaluate when a deep network is used to represent $p(\boldsymbol{D}|\boldsymbol{w})$. The reasons for this is that

- Analytical evaluation of the integral/sum is often not possible, therefore numerical evaluation is necessary
- Numerical evaluation becomes intractable even for a moderate number of parameters N, because the number of different weight vectors increases exponentially with N.

b. (6%) Variational Inference
The basic steps are as follows:

- Pick a variational distribution $q(\boldsymbol{w}, \lambda)$ which has a convenient analytical form (eg: Gaussian distribution). Let the parameter set of this distribution be λ
- Adjust the parameter vector λ such that the KL-divergence between $q(\boldsymbol{w}, \lambda)$ and $p(\boldsymbol{w}|\boldsymbol{D})$ is minimized
  - Since we cannot minimize KL-divergence directly, maximize equivalently the Evidence Lower Bound (ELBO) $\mathcal{L}(\lambda)$ with respect to λ. Here we can use a technique such as Black-box variational inference and Bayes by Backprop. Let the optimized parameter set be $\lambda^\star$
- Use $q(\boldsymbol{w}, \lambda^\star)$ in place of $p(\boldsymbol{w}|\boldsymbol{D})$ in the given integral I
- Now draw samples from $q(\boldsymbol{w}, \lambda^\star)$ and evaluate the integral as an empirical average

$$I \approx \frac{1}{L}\sum_{i=1}^{L} F(\boldsymbol{w}_i), \quad \text{where } \boldsymbol{w}_i \sim q(\boldsymbol{w}, \lambda^\star)$$

6.1 Marks

- Bayesian deep learning : Total **6 points**
  - **2 points** for mentioning low variance as high confidence
  - **2 points** for the correct Bayes formula (either discrete or continuous)
  - **1 point** for pointing the denominator for the difficulty
  - **1 point** for mentioning exponential increase of different weight vectors (or equivalent)
- Variational Inference: Total **6 points**

  - **3 points** for mentioning $q(\boldsymbol{w}, \lambda)$ to approximate $p(\boldsymbol{w}|\boldsymbol{D})$
  - **2 points** for mentioning Evidence Lower Bound (ELBO)
  - **1 point** for mentioning sample mean estimation of the integral

7 Deep Learning for Control (weight 12%)

a. (4%) Inverse Reinforcement learning

Objective function of Inverse Reinforcement learning is

$$L(\psi) = \sum_{\tau \in D} \log(p(\tau))$$

where

$$p(\tau) = \frac{1}{Z} \exp(R_\psi(\tau))$$

and

$$Z = \sum_{\tau \in D_{all}} \exp(R_\psi(\tau))$$

b. (8%) Behaviour cloning

Weaknesses of behaviour cloning:

- They are prone to compounding errors
- They do not consider long term goals and hence lead to reactive policies

Arguments for reinforcement learning (RL):

- RL involves more exploration and hence less chance for compounding errors
- RL considers long term planning and therefore leads to proactive policies

Arguments against reinforcement learning:

- RL is a trial and error based method and this can lead to poor sample efficiency (slow training)
- RL needs a pre-designed reward function, which is difficult in complex control problems.
- Training RL systems in real world environments involves high risks.

7.1 Marks

- Inverse Reinforcement Leraning : Total **4 points**
  - **4 points** for the complete objective function (reduce marks proportionally for missing components)
- Behaviour Cloning: Total **8 points**
  - **2 points** for two correct/acceptable weaknesses of behaviour cloning (1 point each)
  - **2 points** for two acceptable advantages of RL (1 point each)
  - **4 point** for two acceptable weaknesses of RL (2 points each)

8 3D Processing and tracking(weight 14%)

a. (3%) 3D Processing

Node feature dimension = 4, because it contains (x,y,z,r)

b. (4%) 3D Processing

Disadvantage of $\boldsymbol{h}_\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{h}_\Theta(\boldsymbol{x}_i)$:
This is corresponding to the Pointnet. This edge function would not capture locality information (i.e patterns in the neighbourhoods)

Pros/cons of of $\boldsymbol{h}_\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{h}_\Theta(\boldsymbol{x}_j)$:
Advantage: This would capture locality information
Disadvantage: Global structure is only weakly represented

c. (4%) Multi-object tracking

Initial node features= Appearance features (i.e. features extracted from the image patches around detections)
Initial edge features= Geometric features (i.e. geometric relationships between different detections such difference in bounding box co-ordinates)

$\mathcal{G}_s$ is not dependent on explicit edge features because in this case the focus is on segmentation of point clouds which is equivalent to node classification.

$\mathcal{G}_t$ would benefit from explicit edge features because the focus is on finding likely associations which is equivalent to edge classification.

d. (3%) Single Object tracking

Fully convolutional Siamese network (SiamFC) is faster than MDNet mainly because MDNet requires online re-training, whereas SiamFC does not require re-training when deployed. Additionally, in SiamFC, the target image and the search image are compared at all possible locations using a single forward pass, while in MDNet, possible patches are sent through the network one-by-one.

8.1 Marks

- 3D Processing : Total **3 points** for the correct answer 4
- 3D Processing: Total **4 points**
  - **2 points** for disadvantage of $\boldsymbol{h}_\Theta(\boldsymbol{x}_i)$
  - **1 points** for advantage of of $\boldsymbol{h}_\Theta(\boldsymbol{x}_j)$
  - **1 point** for disadvantage of $\boldsymbol{h}_\Theta(\boldsymbol{x}_j)$
- Multi-object tracking: Total **4 points**
  - **1 points** for appearance features or equivalent
  - **1 points** for geometric features or equivalent
  - **1 point** for mentioning $\mathcal{G}_s$ performs node classification or equivalent
  - **1 point** for mentioning $\mathcal{G}_t$ performs edge classification or equivalent
- Single object tracking: Total **3 points** for mentioning at least one of ”no re-training” or fast matching in SiamFC.