University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SPRING, 22/23 TRIMESTER EXAMINATIONS

COMP47590

Advanced Machine Learning

Module Coordinator: Assoc Professor Brian Mac Namee

Student Number

Seat Number

Time Allowed: 120 minutes

Materials Permitted in the Exam Venue:
Non-programmable or scientific calculator
Programmable calculator

Materials to be Supplied to Students:
8 Page Answer Booklets
New Cambridge Statistical Tables

Instructions to Students:
Answer any three out of four questions. All questions carry equal marks. Total marks available 90. The value of each part of each question is shown in brackets next to it.

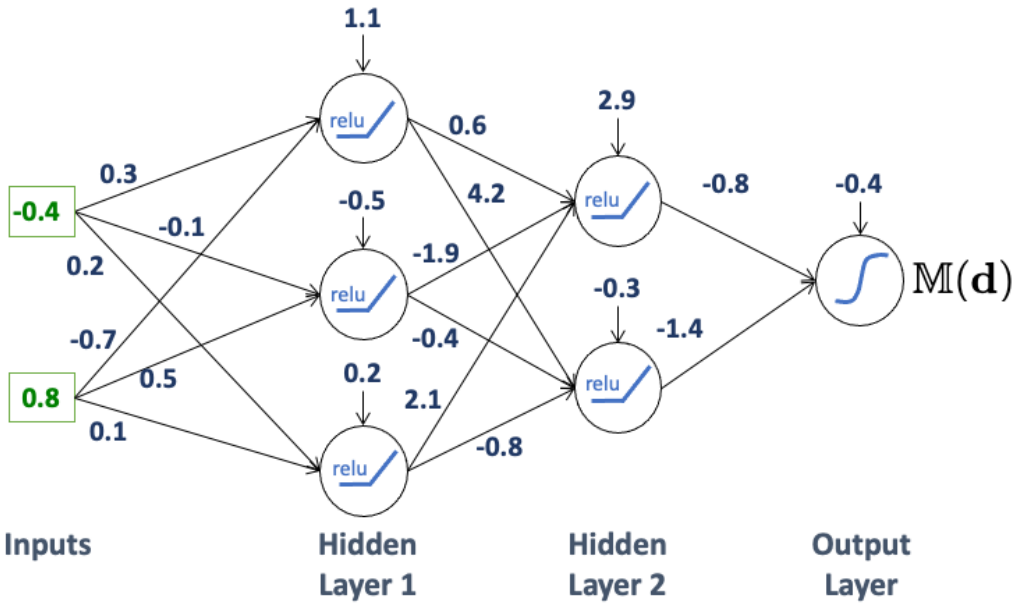
SOLUTIONS

SOLUTIONS

SOLUTIONS

SOLUTIONS

1. **(a)** The image below shows a *feed forward artificial network*. The computational units in the two hidden layers use *rectified linear* (*relu*) activation functions and the output layer unit uses a *sigmoid* activation function. The *weights* and *biases* are shown along the links in the network.



(i) Perform a **forward propagation** through the network using an input feature vector of [-0.4, 0.8]. Show your workings.

[12 marks]

2 Setup Input, Weight and Bias Matrices

The network inputs:

$$\mathbf{d} = \begin{bmatrix} -0.4 \\ 0.8 \end{bmatrix}$$

Weights and biases for Layer 1:

$$\mathbf{W}^{[1]} = \begin{bmatrix} 0.3 & -0.7 \\ -0.1 & 0.5 \\ 0.2 & 0.1 \end{bmatrix}$$

$$\mathbf{b}^{[1]} = \begin{bmatrix} 1.1 \\ -0.5 \\ 0.2 \end{bmatrix}$$

Weights and biases for Layer 2:

$$\mathbf{W}^{[2]} = \begin{bmatrix} 0.6 & -1.9 & 2.1 \\ 4.2 & -0.4 & -0.8 \end{bmatrix}$$

$$\mathbf{b}^{[2]} = \begin{bmatrix} 2.9 \\ -0.3 \end{bmatrix}$$

Weights and biases for Layer 3:

$$\mathbf{W}^{[3]} = \begin{bmatrix} -0.8 & -1.4 \end{bmatrix}$$
$$\mathbf{b}^{[3]} = \begin{bmatrix} -0.4 \end{bmatrix}$$

3 Forward Propagate

To perform a forward propagation for the first layer in the network, first calculate $ extbf{z}^{[1]}$:

$$\begin{aligned} \mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\,\mathbf{d} + \mathbf{b}^{[1]} \\ &= \begin{bmatrix} 0.3 & -0.7 \\ -0.1 & 0.5 \\ 0.2 & 0.1 \end{bmatrix} \begin{bmatrix} -0.4 \\ 0.8 \end{bmatrix} + \begin{bmatrix} 1.1 \\ -0.5 \\ 0.2 \end{bmatrix} \\ &= \begin{bmatrix} 0.42 \\ -0.06 \\ 0.2 \end{bmatrix} \end{aligned}$$

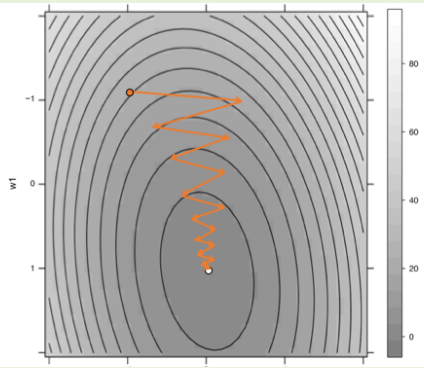
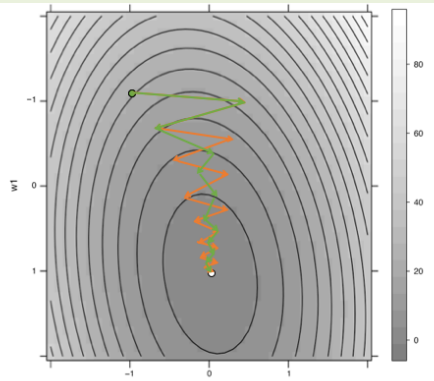
then apply the activation function, in this case a relu function, to calculate the activation of the nodes at Layer 1:

$$\begin{aligned} \mathbf{a}^{[1]} &= \mathrm{g}(\mathbf{z}^{[1]}) \\ &= \mathrm{g}\left(\begin{bmatrix} 0.42 \\ -0.06 \\ 0.2 \end{bmatrix} \right) \\ &= \begin{bmatrix} 0.42 \\ 0.0 \\ 0.2 \end{bmatrix} \end{aligned}$$

To perform a forward propagation for the second layer in the network, first calculate $ extbf{z}^{[2]}$:

$$\begin{aligned} \mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\,\mathbf{a}^{[1]} + \mathbf{b}^{[2]} \\ &= \begin{bmatrix} 0.6 & -1.9 & 2.1 \\ 4.2 & -0.4 & -0.8 \end{bmatrix} \begin{bmatrix} 0.42 \\ 0.0 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 2.9 \\ -0.3 \end{bmatrix} \\ &= \begin{bmatrix} 3.572 \\ 1.304 \end{bmatrix} \end{aligned}$$

			then apply the activation function, in this case a extbf{relu}, to calculate the activation of the output nodes of the network: $$\begin{aligned}\mathbf{a}^{[2]} &= g(\mathbf{z}^{[2]}) \\ &= g\left(\begin{bmatrix}3.572\\1.304\end{bmatrix}\right) \\ &= \begin{bmatrix}3.572\\1.304\end{bmatrix}\end{aligned}$$ To perform a forward propagation for the third layer in the network, first calculate $ extbf{z}^{[3]}$: $$\begin{aligned}\mathbf{z}^{[3]} &= \mathbf{W}^{[3]}\,\mathbf{a}^{[2]} + \mathbf{b}^{[3]} \\ &= \begin{bmatrix}-0.8 & -1.4\end{bmatrix}\begin{bmatrix}3.572\\1.304\end{bmatrix} + \begin{bmatrix}-0.4\end{bmatrix} \\ &= \begin{bmatrix}-5.083\end{bmatrix}\end{aligned}$$ then apply the activation function, in this case a extbf{softmax function}, to calculate the activation of the output nodes of the network: $$\begin{aligned}\mathbf{a}^{[3]} &= g(\mathbf{z}^{[3]}) \\ &= g\left(\begin{bmatrix}-5.083\end{bmatrix}\right) \\ &= \begin{bmatrix}0.006\end{bmatrix}\end{aligned}$$ D: Half an attempt D+: First layer matrices set up correctly C: First layer activation calculated B-: Matrices set up wrong, but calculations made B: Set up correctly but failed to apply relu B: Correct workings but did not work out final sigmoid or some other significant calculation error (e.g. put matrices in wrong order in calculation) B+: Small calculation error but correct workings A: No, or minimal, workings A+: Correct answer and well formatted workings.
		(ii)	If the target feature value for the current input vector is 1.0, calculate the **loss** associated with this training instance using **cross entropy loss**.
			[2 marks]
			Calculate cross entropy loss Loss = -(1*log(0.00616) + 0*log(0.99384)) = 5.0894 E: stated log loss equation D+: Incorrectly applied log loss C+: Correctly applied log loss but calculation error B+: log10 instead natural log

			B: Wrong sign A+: Correctly applied log loss function to result from previous section.
	(b)	**Gradient descent with momentum**, **RMSprop**, and **adam** are three common adaptations to the basic gradient descent algorithm used to train neural networks. Explain how these approaches improve upon basic gradient descent and how they differ from each other.	
			[8 marks]
		Sample Answer (Diagrams such as those included in this answer would be useful.) The gradient descent algorithm optimizes network weight values during a journey across an error (or loss) surface.  All of the adaptations listed improve this process by taking a more direct route across the error surface. This is achieved by adding different types of momentum terms. **Gradient descent with momentum** adds a momentum term to the gradient descent process to avoid sudden changes in exploration of the error surface. A diagram such as the following would help.  This is achieved by using an exponentially weighted moving average applied to the gradient terms over subsequent iterations.	

$$v_{d\mathbf{W}} = \beta v_{d\mathbf{W}} + (1-\beta)\, d\mathbf{W}$$
$$v_{d\mathbf{b}} = \beta v_{d\mathbf{b}} + (1-\beta)\, d\mathbf{b}$$

Weight and bias terms are then updated with these averaged gradients rather than raw gradient values.

$$\mathbf{W} = \mathbf{W} - \alpha v_{d\mathbf{W}}$$
$$\mathbf{b} = \mathbf{b} - \alpha v_{d\mathbf{b}}$$

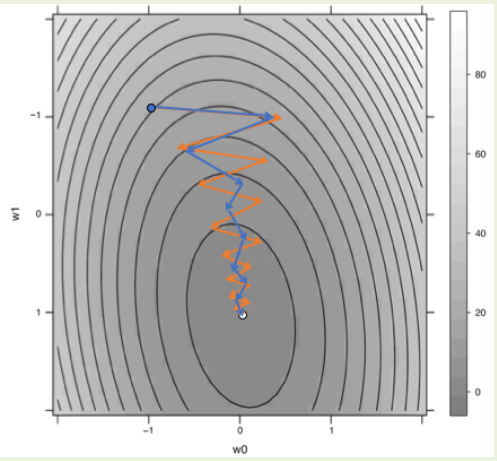
RMSprop builds on top of the same idea as gradient descent with momentum but uses the square of the gradients in the exponentially weighted moving average.

$$v_{d\mathbf{W}} = \beta\, v_{d\mathbf{W}} + (1-\beta)\, d\mathbf{W}^{*2}$$
$$v_{d\mathbf{b}} = \beta v_{d\mathbf{b}} + (1-\beta)\, d\mathbf{b}^{*2}$$

These values are then used in the weight and bias update rules.

$$\mathbf{W} = \mathbf{W} - \alpha \frac{d\mathbf{W}}{\sqrt{v_{d\mathbf{W}}} + \epsilon}$$
$$\mathbf{b} = \mathbf{b} - \alpha \frac{d\mathbf{b}}{\sqrt{v_{d\mathbf{b}}} + \epsilon}$$

This gives a more direct route across the error surface.



Adam mixes the gradient descent with momentum and RMSprop approaches in a weighted sum:

$$v_{d\mathbf{W}} = \beta_1 v_{d\mathbf{W}} + (1-\beta_1)\, d\mathbf{W}$$
$$v_{d\mathbf{b}} = \beta_1 v_{d\mathbf{b}} + (1-\beta_1)\, d\mathbf{b}$$

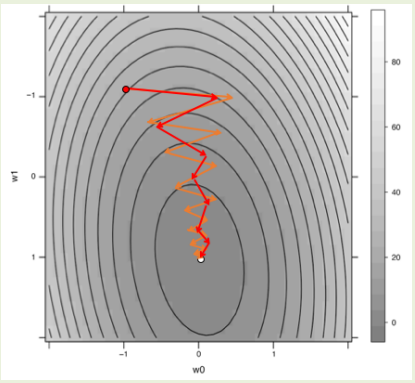
Gradient Descent with Momentum

$$s_{d\mathbf{W}} = \beta_2\, s_{d\mathbf{W}} + (1-\beta_2)\, d\mathbf{W}^{*2}$$
$$s_{d\mathbf{b}} = \beta_2 s_{d\mathbf{b}} + (1-\beta_2)\, d\mathbf{b}^{*2}$$

RMSprop

$$\mathbf{W} = \mathbf{W} - \alpha \frac{v_{d\mathbf{W}}}{\sqrt{s_{d\mathbf{W}}} + \epsilon}$$
$$\mathbf{b} = \mathbf{b} - \alpha \frac{v_{d\mathbf{b}}}{\sqrt{s_{d\mathbf{b}}} + \epsilon}$$

This gives a very efficient route across an error surface.



E: Explaaintion given, but doesn't really match what these approaches do.

D+: very minimal explanation of three approaches

C+: solid exp[lanation of three approaches (no equations)

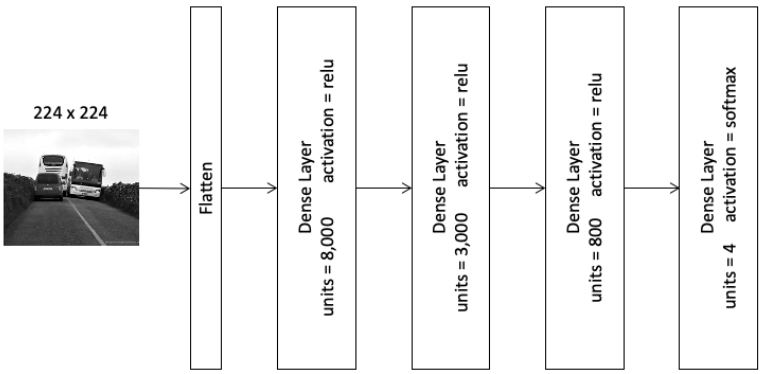
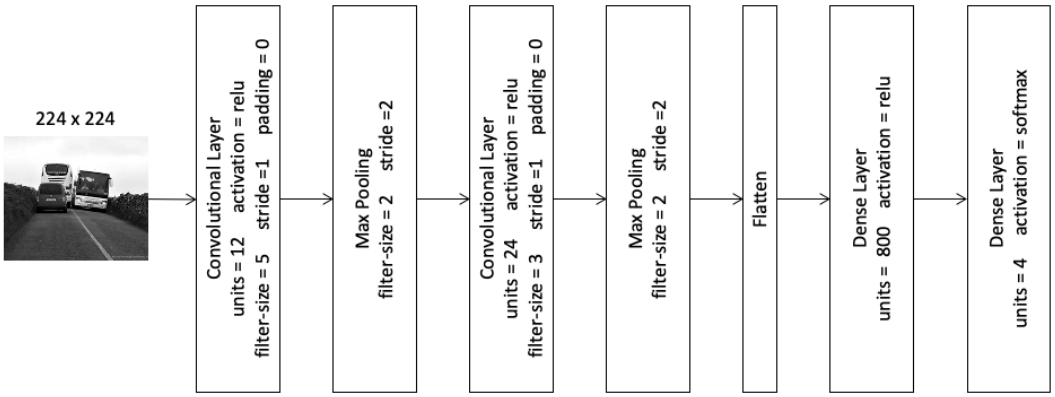
B: solid explantion with equations

A: Correct explanation of three approaches and relevant advantages/disadvantages/insight for each.

(c) We can describe what artificial neural networks do as learning representations (or **embeddings**) of input data that make downstream tasks, for example classification, straight forward. Good embeddings are often said to be:

- **Distributed**
- **Abstract**
- **Invariant**
- **Disentangled**

		Describe what each of these terms means in relation to embeddings generated by artificial neural networks.
		[8 marks]
		A short description of each term along the following lines is required. – Distributed: Expressed across multiple features. – Abstract: Can capture abstract concepts. – Invariant: Are invariant to small and local changes in input data. – Disentangled: Are not overly connected to a single task. **[2 marks each]** E: Explanation given but doesn't really match what these approaches do. D+: very minimal explanation of three approaches C+: solid explanation of four ideas B+: Really good explanation of three ideas and why they are important A+: Correct explanation of four ideas and relevant advantages/disadvantages/insight for each.

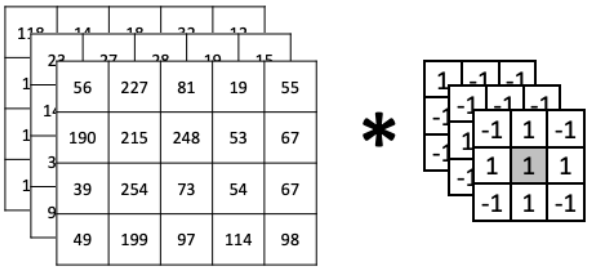
2.	**(a)**	You have been tasked with training a neural network to control a self-driving car from image input. The model should output one of four control signals - *left, right, brake, or accelerate* - from each input image frame. The only input to the model is a 224 pixel by 224 pixel greyscale image from the front of the car. Image (a) shows the architecture of a multi-layer perceptron neural network designed for this problem. Image (b) shows the architecture of a convolutional neural network designed for this problem. Both architectures are composed of four layers.  (a) Multi-layer perceptron network architecture  (b) Convolutional neural network architecture Calculate the number of parameters (weights and biases) that need to be learned for each network architecture.
		[12 marks]
		<u>Sample Answer</u> **Multi-layer perceptron:** This is pretty straight-forward as it involves multiplying through the sizes of the network layers. Input: 224 * 224 = 50,176 Layer 1: 50,176 * 8,000 + 8,000 = **401,416,000**

		Layer 2: 8,000 * 3,000 + 3,000 = **24,003,000** Layer 3: 3,000 * 800 + 800 = **2,400,800** Layer 4: 800 * 4 + 4 = **3,204** Total parameters: **427,823,004** **C:** Wrong input size. **B:** Missed a layer **B+-:** right workings but calculation mistake **A+:** Right answer **Convolutional neural network:** Students need determine the number of weights based on the size of each filter and the size of each layer. To calculate the number of activations at the flattening layer they also need to keep track of the number of activations flowing through the network. Layer 1 Dim: 224 x 224 x 1 Layer 1: 5 * 5 * 1 * 12 + 12 = **312** Layer 1 Output Dim: 220 x 220 x 12 Layer 1 Output Dim After Pooling: 110 x 110 x 12 Layer 2: 3 * 3 * 12 * 24 + 24 = **2,616** Layer 2 Output Dim: 108 x 108 x 24 Layer 2 Output Dim After Pooling: 54 x 54 x 24 Layer 2 Dimensionality after flattening: 54 x 54 x 24 = **69,984** Layer 3: 69,984 * 800 + 800 = **55,988,000** Layer 4: 800 * 4 + 4 = **3,204** Total parameters: **55,994,132**

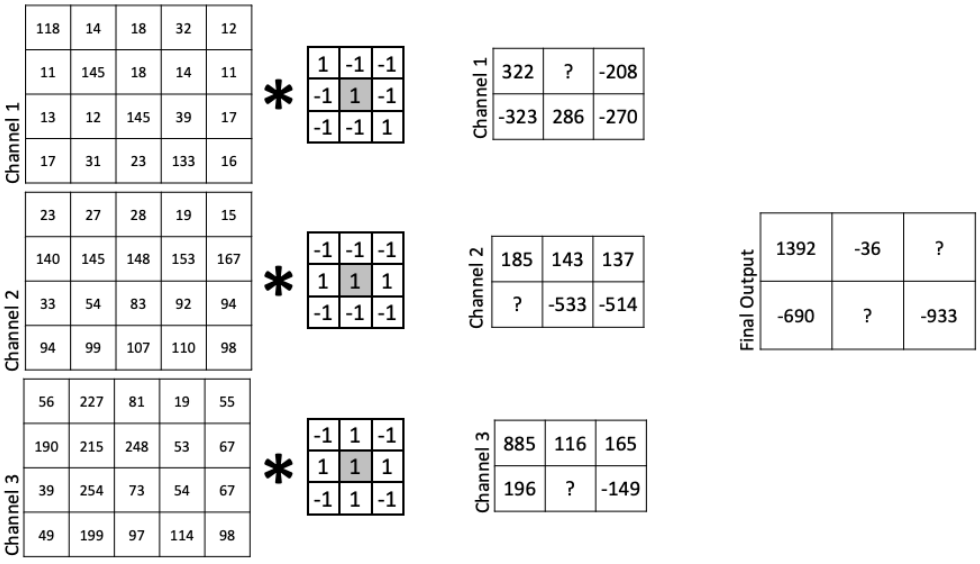
D: used image size not convolutions for size
D+: multiple issues
C+: Incorrect flattening
B: Missed a layer

B: FORGOT BIAS TERMS
B+: right workings but calculation mistake
A+: Right answer

(b) The image below shows a 3-channel input that is being convolved (cross correlated) with a 3-channel 3 x 3 kernel.

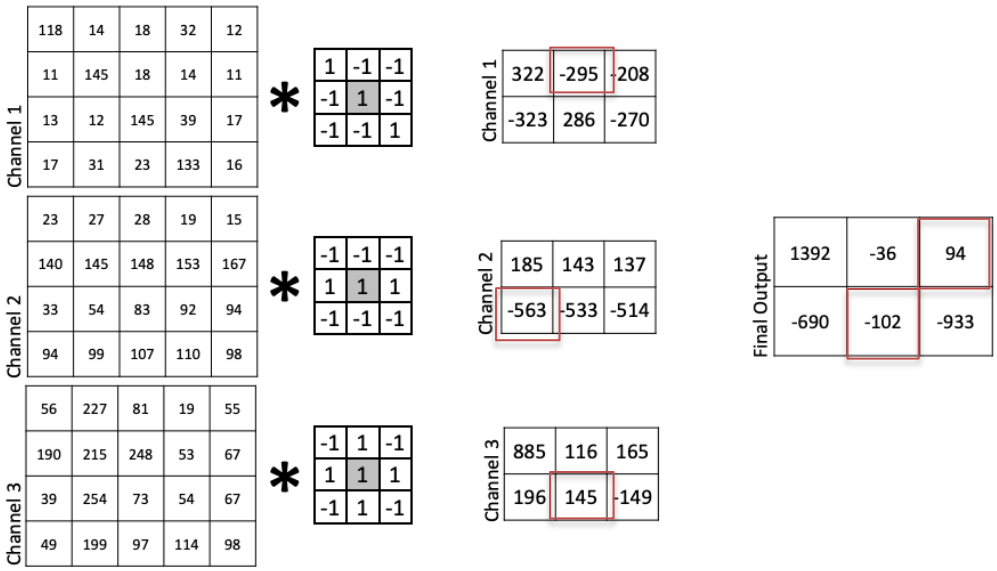


The image below expands the three-channel input and three-channel kernel so that all values can be seen and shows the intermediate convolution result for each channel as well as the final output. Calculate the values marked with a **?** in the intermediate convolution results and the final output.



[8 marks]

Simple convolutions are calculated for each channel and then summed for the final output.



E: POORLY APPLIED CONVOLUTION.

C+: Missing final computation.

B+: Simple calculation error.

A+: All numbers correct.

	(c)	The 2017 paper "*Attention is all you need*" by Vaswani et al is now one of the most cited papers in machine learning research. Explain what **attention** is and how the **transformer** architecture utilises it.
		[10 marks]

Sample Answer

Attention is a mechanism used in deep learning that allows a model to selectively focus on certain parts of the input during processing. It helps the model to identify the most important features and context of the input data while ignoring irrelevant information.

The Transformer architecture is a type of neural network architecture that is specifically designed to handle sequence-to-sequence tasks such as language translation, text summarization, and speech recognition. It replaces the recurrent and convolutional layers commonly used in these tasks with a self-attention mechanism.

		The self-attention mechanism in the Transformer architecture allows the model to weigh the importance of different parts of the input sequence when making predictions. The self-attention layer computes a weighted sum of the input sequence, where the weights are based on the similarity between each input element and every other element in the sequence. The Transformer architecture utilizes multi-head attention, where the self-attention mechanism is applied multiple times in parallel, each with its own set of weights. This allows the model to capture different relationships between different parts of the input sequence and is particularly effective for capturing long-range dependencies. **D+:** Basic idea of context-sentsitive reproresntaiton only **C:** REASONABLE EXPLANATION OF ATTENTION MEDCHANISM WITH KEY IDEAS. **B+:** Excellent explanation of what attention is but no motivation for why. **A+:** Excellent explanation with details and motivation.

3.	**(a)**	Describe the concept of **discounted return** that is frequently used in reinforcement learning.
		[4 marks]
		Sample Answer Calculating the expected return from a sequence of actions is key in developing reinforcement learning systems. In a basic formulation of expected return that simply sums rewards, e.g. $$G = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_e$$ expected future rewards are considered to be as valuable as the immediate reward that the agent will receive from taking the next immediate action. Just like we might be more excited about receiving a gift of $100 today than a promise to receive a gift of $100 in a year's time, it is reasonable when calculating expected return to pay more attention to the immediate reward we expect to receive from taking the next action, than to the rewards that we expect to receive in 10 or even 100 action's time. This is known as discounted return. We can define discounted return as: $$G_\gamma = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e$$ where γ is a discount factor that can take a value between 0 and 1. **F:** Explanation of something else. **D:** Incorrect explanation, but something right. **C:** Good explanation but no equation. **B:** Good explanation plus equation **A:** Excellent explanation plus equation with some insight
	(b)	An intelligent agent trained to play a video game completes an episode and receives the following sequence of rewards over six timesteps: $\{r_0 = 84, r_1 = 98, r_2 = -57, r_3 = -104, r_4 = -96\}$ Compare the **discounted returns** calculated at time $t = 0$ based on this reward sequence when discounting factors of 0.9 and 0.1 are used.
		[6 marks]
		Sample Answer Students should begin by calculating the discounted returns using: $$G_\gamma = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e$$

The following tables shows the calculations for discounting factor of 0.9 and 0.1:

Time		0	1	2	3	4	
Reward		84	98	-57	-104	-96	
		1	0.9	0.81	0.729	0.6561	
Discounted Return	0.9	84	88.2	-46.17	-75.816	-62.9856	**-12.7716**
		1	0.1	0.01	0.001	0.0001	
Discounted Return	0.1	84	9.8	-0.57	-0.104	-0.0096	**93.1164**

Students should then discuss that with the lower discount factor much less attention is paid to later rewards and so the overall return is much lower.

D:

D: small mistake

C+: Correct answers but little or no insight

A: Correct answers plus good insight

(c) To try to better understand the slightly baffling behaviour of her new baby girl, Maria - a scientifically minded new mother - monitored her baby over a period of time, recording her activity at 20 minute intervals. The activity stream looked like this (with time flowing down through the columns):

0	SLEEPING	12	SLEEPING	24	SLEEPING	36	CRYING
1	CRYING	13	SLEEPING	25	HAPPY	37	HAPPY
2	SLEEPING	14	SLEEPING	26	CRYING	38	HAPPY
3	SLEEPING	15	CRYING	27	SLEEPING	39	HAPPY
4	SLEEPING	16	CRYING	28	SLEEPING	40	HAPPY
5	HAPPY	17	SLEEPING	29	HAPPY	41	HAPPY
6	HAPPY	18	SLEEPING	30	HAPPY	42	SLEEPING
7	HAPPY	19	HAPPY	31	HAPPY	43	SLEEPING
8	SLEEPING	20	SLEEPING	32	HAPPY	44	SLEEPING
9	SLEEPING	21	HAPPY	33	HAPPY	45	SLEEPING
10	SLEEPING	22	HAPPY	34	HAPPY	46	SLEEPING
11	SLEEPING	23	CRYING	35	CRYING	47	SLEEPING

Maria noticed that her baby could occupy one of three states - HAPPY, CRYING, or SLEEPING - and moved quite freely between them.

(i) Based on the sequence of states given above calculate a transition matrix that gives the probability of moving between each of the three states.

[10 marks]

Sample Answer

The first step to building the transition matrix is to count the frequency of each possible state transition. Working down through the list of states we can count the number of times we move from one state to the next. This gives a transition frequency table:

	SLEEPING	CRYING	HAPPY
SLEEPING	15	2	5
CRYING	4	2	1
HAPPY	3	3	12

By normalising each row in the table (dividing each value by the sum of values in the row) we can calculate the final transition matrix:

	SLEEPING	CRYING	HAPPY
SLEEPING	0.682	0.091	0.227
CRYING	0.571	0.286	0.143
HAPPY	0.167	0.167	0.667

D:

C:

B: Correct counts but not normalised (or incorrect normalisation)

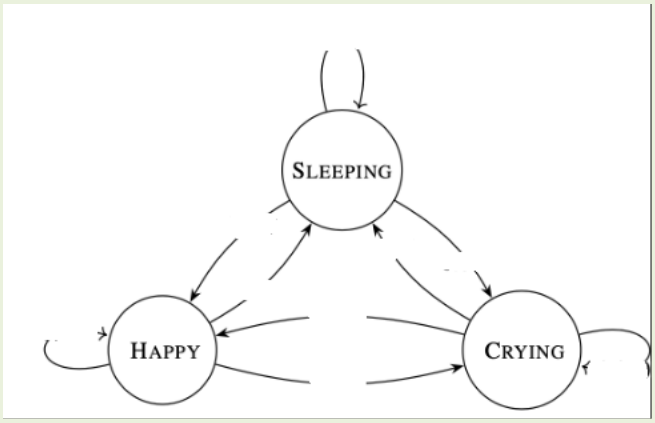
A+: Perfect

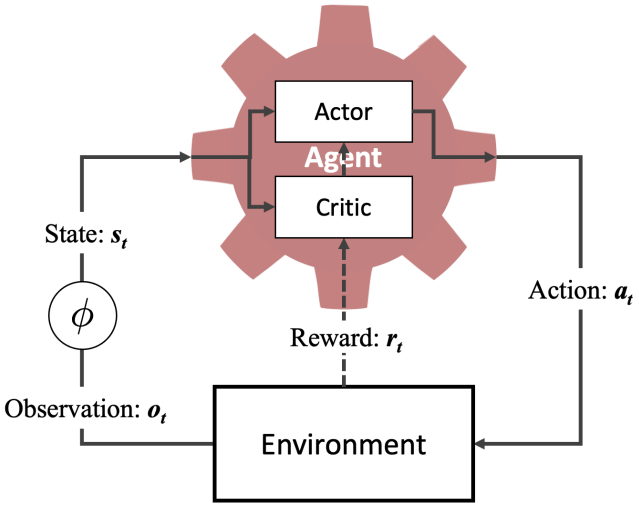
(ii) Draw a Markov process diagram to capture the behaviour of a small baby as described above.

[5 marks]

Sample Answer

A diagram like this one is appropriate for this answer.



			D: Diagram but no numbers **C:** **B:** **A:** Perfect
	(d)	The image below shows an illustration of reinforcement learning using **actor-critic** method.  Describe the role of the actor and critic models in this approach.	
			[5 marks]
		Attempt to merge best of policy gradient methods with value function methods.Composed of two models – **Critic** is an action value function model $Q_{\mathbb{M}_{w_C}}(s_t, a_t)$ – **Actor** is a policy model $\pi_{\mathbb{M}_{w_A}}(s_t)$ During learning when the actor model is being updated it uses outputs from the critic action value function to estimate expected return	

4.	**(a)**	Some benchmark experiments have found **ensemble models based on gradient boosting** can be prone to overfitting to incorrectly labelled instances in the training dataset (for example a positive instance mislabelled as a negative instance). Explain why this is the case and describe how a learning rate can be introduced to the gradient boosting algorithm to mitigate this.
		[10 marks]

		Sample Answer Gradient boosting builds an ensemble by iteratively training models and building models that correct the predictions of their predecessors. In this way the next model trained will focus on the parts of the training set that the previous model struggled with. The reason that boosting is sensitive to noise in the target features than bagging is that it is prone to over fitting to these noisy instances as models will typically struggle to correctly predict them which means subsequent models will focus too much on them. A learning rate in gradient boosting means that in the aggregation process later models are assigned less weight than earlier models. *A+ amazing explantions of both.* *B+ decent explanation of both problem – model focuses on the error cases. - and how learning rate helps – foscuses less on models trained later..* *C+ good problem explanation – model focuses on the error cases. No or bad explanatiomn of how learning rate helps – foscuses less on models trained later. Or vice versa* *D+ hovering around right ideas.* *E* *NG*
	(b)	When developing a machine learning model that will be deployed to perform a task for a user, we can describe three different goals of evaluation: 1. to determine which model is the most suitable for a task 2. to estimate how the model will perform after deployment 3. to convince users that the model will meet their needs Describe the differences between these goals, and how the evaluation methods used to achieve each of them can be different.
		[10 marks]
		Sample Answer *Students should outline that when performing an evaluation to prepare a model for deployment the first goal is to determine which approach might work best. Decision s to be made based on this evaluation include which modelling approach will be used, which data pre-processing techniques might be used, and the optimal algorithm hyper-parameters to be used. This evaluation is very much driven by the machine learning practitioner. Typically, k-fold cross validation can be used as the main evaluation method for this kind of evaluation.*

		Once all of the decisions about what modelling approach to take have been made the next goal attempts to evaluate the likely performance of a model after deployment. This is largely concerned with estimating generalisation error of the model. The only sensible way to evaluate this is to use a hold out test set that has been involved at all in training the model. This is the only reasonable way to evaluate generalisation error. *Once practitioners are convinced that a modelling approach will achieve acceptable performance after deployment the last step is to convince the people for whom the model is being built that it will meet their needs. This can be done with the same kinds of experiments used to evaluate likely generalisation error, but often different performance measures need to be used so as to speak to a different audience (not machine learning practitioners). It is also important to consider a deployment experiment at this stage too.* *A+* *B+* *C+* *C: Basic explanation of three approaches.* *D+* *D Little more than restating the question!* *E* *NG*
	(c)	The following is a definition of machine learning: *The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.* - Tom Mitchell Do you believe that this definition accurately defines machine learning? In your answer discuss the appropriateness of the definition, the scope of the definition, and any recommendations for improvements you would suggest.
		[10 marks]
		<u>Sample Answer</u>

		The definition given by Tom Mitchell accurately defines machine learning as a field of study that deals with the development of computer programs that can improve their performance over time with experience. The definition is appropriate as it highlights the central aspect of machine learning, which is learning from data. However, the definition's scope is limited to the concept of "improving with experience" and doesn't account for other aspects of machine learning such as pattern recognition, prediction, and decision-making. These aspects are also critical to machine learning and should be included in the definition. In terms of improvements to the definition, it could be expanded to include the concept of generalization. In machine learning, the ultimate goal is to develop models that can make accurate predictions on new, unseen data. Therefore, a better definition could be: "The field of machine learning is concerned with the development of computer programs that automatically learn patterns from data, generalize these patterns to make predictions on new, unseen data, and improve their performance over time through experience." Overall, the original definition accurately defines machine learning but could benefit from an expansion to include other critical aspects of the field. *A+* Exceptional discussion of appropriateness, scope , and improvements. Probably bringing in other definitions,. *B+* decent discussion of appropriateness, scope , and improvements. *C* *+Missing one* of appropriateness, scope , and improvements. *D+* *E* *NG*