I9
Dynamic Vision and Learning
Technical University of Munich

E1119
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Introduction to Deep Learning

Exam:	IN2346 / endterm	**Date:**	Monday 1st August, 2022
Examiner:	Prof. Leal-Taixé	**Time:**	08:15 – 09:45

	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
I									

Working instructions

- This exam consists of **24 pages** with a total of **9 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 90 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources:
  - one **non-programmable pocket calculator**
  - one **analog dictionary** English ↔ native language
- Subproblems marked by * can be solved without results of previous subproblems.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.

Left room from ______ to ______ / Early submission at ______

Problem 1 Multiple Choice (18 credits)

Mark correct answers with a cross ☒

To undo a cross, completely fill out the answer option ■

To re-mark an option, use a human-readable marking × ■

Please note:

- For all multiple choice questions any number of answers, i.e. either zero (!), one or multiple answers can be correct.
- For each question, you'll receive 2 points if all boxes are answered correctly (i.e. correct answers are checked, wrong answers are not checked) and 0 otherwise.

1.1 You're training a neural network with 3 fully-connected layers, ReLU as an activation function and a keeping dropout probability of 0.2. Your task is to classify a dataset that is made of 2 classes - what is a good choice of (activation layer, loss function) at the end of the architecture?

☐ (TanH, Binary Cross Entropy)

☐ (ReLU, Mean Squared Error (L_2))

☒ (Sigmoid, Binary Cross Entropy)

☒ (Softmax, (Multi-class) Cross Entropy)

1.2 What is true about fully-connected layers?

☐ They are non-linear functions.

☒ They can be represented as a convolutional layer.

☐ Given the input $X_{N\times D}$ and $W_{D\times M}$, the length of the learnable parameters β and γ in a following batch normalization layer is D.

☐ Once initialized, they can accept any input size.

1.3 If your input batch of images is 16×32×64×64 (N×C×H×W), how many parameters are there in a single 1x1 convolution filter operating on this input, including bias?

☐ 65

☐ 2,097,153

☒ 33

☐ 17

1.4 You're building a neural network consisting of convolutional layers and TanH as the activation function. What could mitigate / reduce the likelihood for the gradient to vanish during training?

☒ Use Xavier initialization for your convolutional layers.

☒ Organize the layers in residual blocks.

☒ Replace TanH with Leaky ReLU with α = 0.2.

☒ Reduce the number of layers.

1.5 What is the benefit of using Momentum in optimization?

□ It introduces just a single learnable parameter.

□ It effectively scales the learning rate to act the same amount across all dimensions.

□ It combines the benefits of multiple optimization methods.

☒ It is more likely to avoid local minima when used with stochastic gradient descent.

1.6 A sigmoid layer ...

□ ... could be used only on the logits of a classification neural network.

□ ... could boost performance when used in linear regression on normalized data with mean=0 and standard deviation=1.

□ ... maps all values into the interval $[-0.5, 0.5]$.

□ ... is a zero-centered non-linear activation function.

1.7 Given a Max-Pool layer with kernel size of 2×2 on a window with all unique positive values:

☒ In order to backpropagate through the layer, one needs to pass information about the positions of the max values from the forward pass.

□ 75% of the derivatives differ from zero.

□ The layer's weights are updated with a chosen optimizer method.

☒ It performs features selection.

1.8 You are given the following fully-connected network. What is true about the Xavier initialization? **Note**: Each fully-connected layer has X as the input, W as the weight matrix and a vector b as the bias.

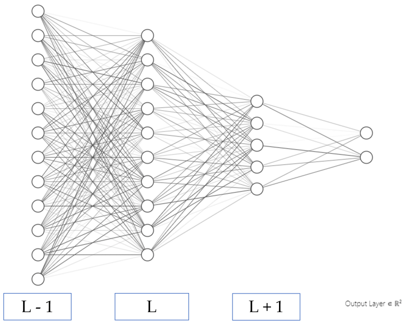

Figure 1.1: A fully-connected network.

☒ $\text{Var}(X_L) = \text{Var}(X_{L+1})$

□ $\text{Var}(W_{L-1}) = \text{Var}(W_L)$

□ $\text{Var}(X_L) = \text{Var}(W_L)$

□ $\text{Var}(W_L) = \frac{1}{m}$, where m is the number of columns of W_L.

1.9 You're given the following fully-connected toy network with:

1. All weights are initialized with the value 1.
2. The optimizer is stochastic gradient descent (SGD).
3. The learning rate is $\alpha = 1$.
4. The same input $X = [1, 1, 1, 1]$ is fed time after time, for 3 iterations.

What is the sign of the weights of the first layer, L_1, by the end of the loop?

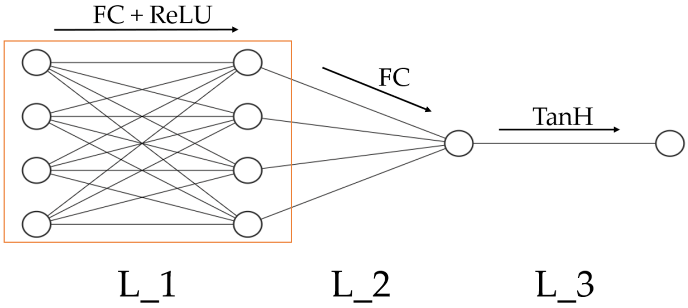

Figure 1.2: Toy network

□ All negative.

□ All zeros.

☒ All positive.

□ Both negative and positive.

Problem 2 Unsorted Questions (14 credits)

2.1 List one advantage and one disadvantage of having a small batch size for training.

□ 0 □ 1 □ 2

(1p): Advantages: Less memory consumption of a batch. Faster computation for one iteration/for updates/ for a batch. Escape local minima.
(1p): Disadvantage: Converges slower. Small batches can offer a regularizing effect that provides better generalization (Noise). More iterations needed.

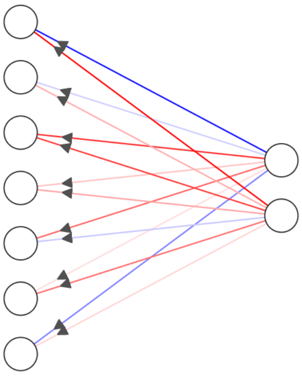

Figure 2.1: Backpropagation of a fully-connected layer

Given a fully-connected layer $Y = XW + b$, where $X_{N\times D}$, $W_{D\times M}$, an upstream gradient $dout = \frac{\partial L}{\partial Y}$ during the backpropagation process and a loss function $L : \mathbb{R}^M \rightarrow \mathbb{R}$.

2.2 What are the dimensions of b and $dout$?

□ 0 □ 1

(0.5p): $1 \times M$
(0.5p): $N \times M$

2.3 What is the derivative of $\frac{\partial L}{\partial X}$ and what are its dimensions?

□ 0 □ 1

(0.5p): $dout \cdot W^T$
(0.5p): $N \times D$

2.4 Consider the L_2 - regularization mechanism with a corresponding coefficient λ. Write down the update rule of gradient descent with weight decay, for the iteration $k+1$, a weight matrix θ and a learning rate coefficient α. Use $\nabla L(\theta_k)$ as the derivative of $\frac{\partial L}{\partial \theta}$.

□ 0 □ 1

The L_2 regularization term, that is added to the scores of a linear layer is $Y = f(x, \theta, b) + \frac{1}{2}\sum_i \sum_j |\theta_{i,j}|_2^2$.
So after taking the derivatives, the optimizer step is:
(1p): $\theta_{k+1} = (1 - \alpha\lambda)\theta_k - \alpha\nabla L(\theta_k) = \theta_k - \alpha(\nabla L(\theta_k) + \lambda\theta_k)$

0 1

2.5 What is the purpose of weight decay?

(1p): Regularization.

0 1

2.6 What can we expect to see in terms of the error curves in a graph, when the corresponding coefficient λ in weight decay is too high while training?

(1p): Underfitting. Both training and validation loss will plateau on a relatively high value.

Linear Regression is a well-known problem in Machine Learning, where we aim to find the best coefficient matrix W, given a function $\hat{Y} = f(X) \sim XW$, where $X = \{x_1, ..., x_n\}$ and such that

$$W^* = \underset{W}{\operatorname{argmin}} \frac{1}{2n} \sum_{i=1}^{n} (y_t - \hat{y}_i)^2$$

Consider that we have a classical problem setting with a low amount of data samples.

0 1

2.7 If our algorithm converges to a minimum, why is it guaranteed to be the global minimum?

(1p): Linear regression is a convex function.

0 1

2.8 What advantage does the linear regression method has over the iterative deep-learning methods, given a linearly distributed dataset?

(1p): It has a closed-form solution. Mathematically computing the solution $W = (X^T X)^{-1} X^T Y$ is superior.

0 1 2

2.9 Mention 2 drawbacks of ReLU. Explain how they affect the optimization problem.

(0.5p): Not zero centered/ All weights can change only with the same sign → (0.5p) The zigzag effect, slower training.
(1p): Dying ReLU → Vanishing gradient.
(1p): ReLU not differentiable at zero.

You're working on a group project, and define some deep-learning architecture. You've found a good set of hyperparameters and you are ready to start your training. Now, your partner suggests that you could merge the validation set into the training set, so it will be bigger.

0 1

2.10 What is the purpose of the validation set while training?

(1p): Sanity check. The validation set is unseen data, that helps to determine whether the model has overfitted or underfitted.

2.11 You take their advice, and see that your training loss is converging at a very low error rate for a long time. However, when testing, your error rate is really high. Name the problem, and how can we avoid training for such a long time **by using the validation set.**

□ 0
□
□ 1
□
□ 2

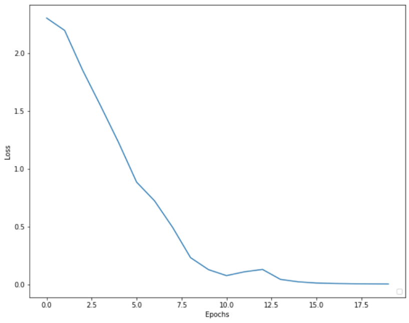

Figure 2.2: Low training error rate

(1p): Overfitting. (1p): Early stopping.

Problem 3 Data Augmentation (4 credits)

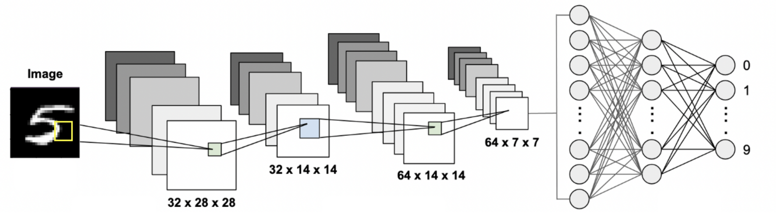

Figure 3.1: MNIST classification architecture

You have a classification task on the MNIST dataset. After training, your model achieves great results. In order to further improve, a friend has suggested you to apply data augmentation.

0 □ 1 □

3.1 Explain the concept of data augmentation. (When would one use it? What is crucial to keep?)

Data augmentation is a regularization technique in which we modify the training set inputs to introduce a larger variety of training data.
(0.5p): One uses data augmentation for: Regularize the training/minimize the generalization gap/reduce overfitting/one use it when the dataset is too small.
(0.5p): One should keep in mind that augmented data must come from the same distribution as original data.

0 □ 1 □

3.2 On which dataset out of the split training, validation, test would you apply data augmentation?

Training.

You remember from class that data augmentation improves performance. In order to optimize your model, you decide to apply "Horizontal Flip" and "Gaussian Blur". Surprisingly, you notice that training and testing errors have both gone higher.

0 □ 1 □

3.3 Why is the training error higher?

(1p): Training error is higher because: There are now instances that present cases not seen before in original training dataset / data augmentation increases the diversity of training data / more variance in train data.

0 □ 1 □

3.4 Why is the test error higher?

(1p): The augmented data we create (horizontal flipped one) do not belong to the MNIST dataset distribution, hence do not come from same distribution as test data.

Problem 4 An application (16 credits)

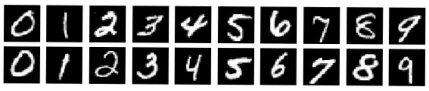

Figure 4.1: Example images of the MNIST dataset.

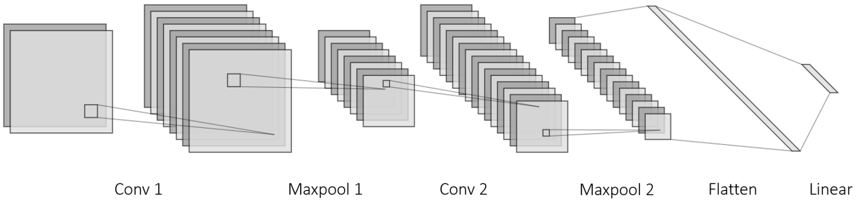

Figure 4.2: Sketch of the given architecture

We train a network which calculates **the sum of two digits** as a classification problem, represented by images from the MNIST dataset.
The MNIST dataset comprises 50,000 unique 1x28x28 grayscale images of handwritten digits, in the range of [0...9], equally distributed.

Notes:

- The input of our network are two images stacked in feature space, with channels=2, height=28 and width=28.
- The output of the network is a vector of length *n*, corresponding to the number of possible sums.
- Note that if the original MNIST contains a balanced dataset of 50,000 images and 10 labels, now it holds 50,000 · 50,000 = 2.5B different possible pairs.
- Convolutional layers are defined as Conv2d(<input channels>, <output channels>)
- Maxpool layers are defined as Maxpool2d(<input channels>)

We give the following network architecture that classifies into *n* possible output classes:

- Conv2d(2, 8) → MaxPool2d(2) → BatchNorm2d() → ReLU()
- Conv2d(8, 16) → MaxPool2d(2) → BatchNorm2d() → ReLU()
- Flatten()
- $Linear(k, n)$

where $k \in \mathbb{N}$ is a variable defining the input dimension of the Linear layer after the Flatten operation.

4.1 Give a triplet of (kernel size, stride, padding) for the convolutional layers, that will keep the spatial dimensions of their input (i.e., $H \times W \to H \times W$)

□ 0
□ 1
□ 2

(2p): Example: (k=3, s=1, p=1).

For the following consider a batch size N and a triplet (kernel size, stride, padding) that keeps the spatial dimension of the input.

4.2 What is the shape of the input of our network? What is the shape of the network output?

(0.5p): Shape of input: $N \times 2 \times 28 \times 28$.
(0.5p): Shape of output: $N \times 19$, since there are 19 different possible sums.

4.3 What is the value of k (the input dimension of the Linear Layer)? Show your calculation steps. You can provide the final answer using products.

(1p): The spatial size decreases: $28 \rightarrow 28 \rightarrow 14 \rightarrow 14 \rightarrow 7$. So $X = 16 \cdot 7 \cdot 7$.

4.4 What loss function should we use?

(1p): MCE - (Multiclass) Cross-Entropy.

4.5 In the exercises, we used a fully-connected network to classify the MNIST dataset.
How can we resemble a fully-connected layer, from an input of an image with size ($2 \times 28 \times 28$) to an output of 100 neurons, by using convolutions?

(2p): Use 100 filters, kernel-size=28, padding=0, stride=0, and flatten the output.

Receptive Field:

4.6 State the definition of the receptive field of a neuron in an intermediate layer of a neural network.

(1p): The size of the region in the input image that a single neuron in the intermediate layer is affected by.

4.7 What is the receptive field of a neuron after **Maxpool 1**?
We assume

1. Convolutions have a kernel size of 5 and a stride 1.
2. Maxpools have a kernel size of 2 and a stride 2.

For reference the optional formula $r_k = r_{k-1} + (\prod_{i=1}^{k-1} s_i) \cdot (f_k - 1)$ *for* $k \geq 2$
(1p): $r_1 = 3 \rightarrow r_2 = r_1 + s_1 \cdot s_2 \cdot (f_2 - 1) = 5 + 1 \cdot (2 - 1) = 6 \rightarrow (6 \times 6)$.

4.8 What is the receptive field of a neuron after the **Linear Layer** using the same assumptions?

(1p): A linear layer corresponds to the entire input, i.e. the receptive field is (28×28) (the whole input image).

4.9 We decided to sample the input images randomly, and not set a fixed split to the dataset. Each time we test our network we randomly sample two digit images. State one problem that can arise by doing so.

□ 0
□
□ 1
□
□ 2

(2p): Mixes train and test set. Therefore our network trains on test samples. Thus our test would be biased and we can't check if our model performs well on unseen data.
OR
(2p): Impossible to compare two different runs/architectures (not deterministic).

4.10 After training, you observe 95% test accuracy for the class 12 but 35% test accuracy for class 18. What is the problem?

□ 0
□
□ 1
□
□ 2

(2p): An imbalanced dataset. There are many more pairs that could sum to 12, than pairs summing to 18.

4.11 A friend suggests approaching the task in a different way, namely as a regression task. Name two required changes to the network.

□ 0
□
□ 1
□
□ 2

(1p): 1 output neuron instead of 19, because regression is a continuous function.
(1p): Change the Loss function: MSE or L1.

Problem 5 Autoencoder (11 credits)

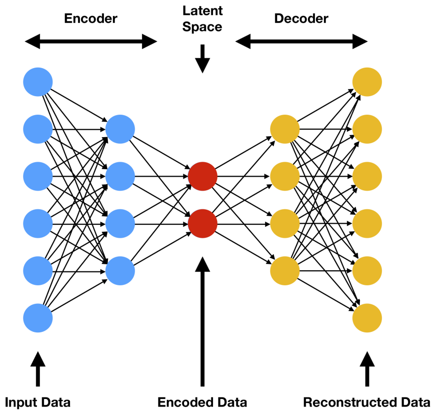

Figure 5.1: A Fully-Connected Autoencoder

0 □ 1 □

5.1 The size of the bottleneck (latent space) of an autoencoder is an important hyperparameter. Explain the consequences, if the latent space is too small.

(1p): Too much information is lost which means the signal cannot be reconstructed accurately.

0 □ 1 □

5.2 And what if the latent space is too big?

(1p): Model will learn the identity mapping / no compression / no meaningful features are learned.

0 □ 1 □

5.3 Explain the main difference regarding data of training an autoencoder in comparison to training a classification network.

(1P): No ground-truth labels necessary / unsupervised vs. supervised.

0 □ 1 □ 2 □

5.4 You train a classification network on a big dataset, but only a small proportion of your dataset is labeled. Explain the steps of using an autoencoder to deal with this issue.

(1P): Train autoencoder on the whole (unlabelled) dataset for reconstruction.
(1P): Take the encoder model, add classification layers and train/finetune it on the labeled data.

0 □ 1 □

5.5 How is this technique called? As the classification dataset is quite small, suggest a reasonable approach to handle the weights of the network while training to avoid using the pretrained advantage of using an autoencoder as described above.

(1.5p): (0.5p) Transfer learning. (1p) Freeze the weights / part of the weights of the pretrained encoder.

5.6 You now successfully trained an autoencoder on MNIST. You randomly sample a latent vector and decode it using the trained decoder, but the resulting image does not look like a number at all. Explain why this is to be expected.

□ 0 □ 1

(1p): During training we only map the input distribution to the latent space using an encoder. As this function does not need to be surjective, we can't expect randomly sampled vector's to be decoded into MNIST numbers.

5.7 Name a solution how we can train an autoencoder such that we can sample a latent vector and expect our decoder to generate MNIST images for randomly sampled latent vectors.

□ 0 □ 1

(1p): Variational Autoencoder / enforce a loss on the latent space so that it follows a known distribution, like a Gaussian.

We saw in the lecture that the Autoencoder architecture could be also used for the task of semantic segmentation.

5.8 What is the semantic segmentation of an image?

□ 0 □ 1

(1p): Semantic segmentation is the task of classification for each individual pixel.

5.9 Specify the output dimensions for an input image of size C×H×W.

□ 0 ½

(0.5p): $\hat{C}\times H\times W$, where $\hat{C}$ is the number of classes in the dataset (in the setting of semantic segmentation).
(0.5p): Also accepted: C×H×W, where C is the number of input channels (in the setting of an auotencoder architecture).

5.10 Why do convolutions fit better to the task of semantic segmentation than fully-connected layers?

□ 0 □ 1

(1p): Fully-connected layers process global information from the entire input image, while convolutions extract local features from the neighborhood of each pixel.

Problem 6 Backpropagation (6 credits)

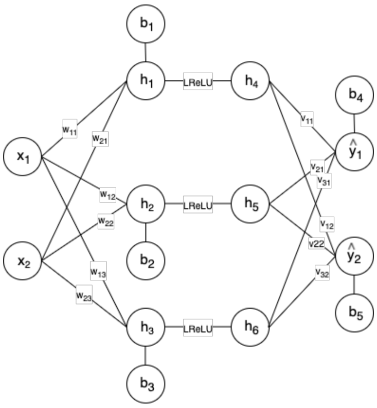

Figure 6.1: Simple network

Variable	x_1	x_2	w_{11}	w_{12}	w_{13}	w_{21}	w_{22}	w_{23}	b_1	b_2	b_3	v_{11}	v_{12}	v_{21}	v_{22}	v_{31}	v_{32}	b_4	b_5	y_1	y_2
Value	1.0	-2.0	-0.5	1.0	2.0	-1.0	0.5	1.5	0.5	0	-1.0	1.0	-0.5	-0.5	2.0	-1.0	1.0	-1.0	2.0	1.0	1.0

Table 6.1: Values of the variables

In the diagram, a simple network is given with weights, biases and Leaky ReLU activation with $\alpha = 0.5$.

0 □
1 □
2 □

6.1 Compute the output ($\hat{y_1}$ and $\hat{y_2}$) of this network. Therefore, you will need to calculate the following variables: $h_1, h_2, h_3, h_4, h_5, h_6, \hat{y}_1, \hat{y}_2$.

$h_1 = w_{11} * x_1 + w_{21} * x_2 + b_1 = -0.5 * 1 + (-1) * (-2) + 0.5 = 2$
$h_2 = w_{12} * x_1 + w_{22} * x_2 + b_2 = 1 * 1 + 0.5 * (-2) + 0 = 0$
$h_3 = w_{13} * x_1 + w_{23} * x_2 + b_3 = 2 * 1 + 1.5 * (-2) - 1 = -2$
$h_4 = max(0.5 * h_1, h_1) = max(0.5 * 2, 2) = 2$
$h_5 = max(0.5 * h_2, h_2) = max(0.5 * 0, 0) = 0$
$h_6 = max(0.5 * h_3, h_3) = max(0.5 * (-2), -2) = -1$
(1p): $\hat{y}_1 = v_{11} * h_4 + v_{21} * h_5 + v_{31} * h_6 + b_4 = 1 * 2 + (-0.5) * 0 + (-1) * (-1) - 1 = 2$
(1p): $\hat{y}_2 = v_{12} * h_4 + v_{22} * h_5 + v_{32} * h_6 + b_5 = (-0.5) * 2 + 2 * 0 + 1 * (-1) + 2 = 0$

0 □
1 □

6.2 Calculate the Mean Squared Error Loss using your results in the previous question and the target values (y_1 and y_2).

(1p): $MSE = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} (\hat{y_{i,j}} - y_{i,j})^2 = \frac{1}{2}(2-1)^2 + \frac{1}{2}(0-1)^2 = 1$
Note: k is the number of neurons in the output layer.

6.3 Do one backward pass on this network to update w_{11} with Stochastic Gradient Descent using learning rate of 0.1.

□ 0
□
□ 1
□
□ 2
□
□ 3

(1.5p): $\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial \hat{y}_1} \frac{\partial \hat{y}_1}{\partial h_4} \frac{\partial h_4}{\partial h_1} \frac{\partial h_1}{\partial w_{11}} + \frac{\partial L}{\partial \hat{y}_2} \frac{\partial \hat{y}_2}{\partial h_4} \frac{\partial h_4}{\partial h_1} \frac{\partial h_1}{\partial w_{11}} = ((\hat{y}_1 - y_1) * v_{11} + (\hat{y}_2 - y_2) * v_{12}) * 1 * x_1 = (((2-1) * 1) + (0-1) * (-0.5)) * 1 * 1 = 1.5$

(1.5p): $w_{11}^* = w_{11} - \alpha \frac{\partial L}{\partial w_{11}}$ $w_{11}^* = -0.5 - 0.1 * 1.5 = -0.65$

Problem 7 Batch Normalization (6 credits)

Given a batch of inputs X, the batch normalization layers normalize the features according to the formula:

$$\hat{X} = \frac{X - E[X]}{\sqrt{\mathrm{Var}[X]}}$$

$$y = \gamma \cdot \hat{X} + \beta$$

0 □ 1 □ □

7.1 In general, why is it useful to apply a batch normalization layer after linear (fully connected / convolutional) layers?

(1p): Gradients are less varying in their magnitudes (stable gradients) and a smoothing of the gradients which leads to efficient backprop. For training that means they are less sensitivity to initialisation, faster convergence and larger possible learning rates.

0 □ 1 □ 2 □ □ □

7.2 What are γ and β? What is their purpose?

(1p): γ and β are learnable parameters.
(1p): It enables to undo the scaling and shifting in training.

0 □ 1 □ □

7.3 Consider a BatchNorm2d() layer for convolutions that operates on in input $X_{8\times3\times32\times64}$ (*BatchSize* × *Channels* × *Height* × *Width*). How many parameters do the running-mean and running-variance variables hold?

(1p): $2 \cdot 3 = 6$.

0 □ 1 □ 2 □ □ □

7.4 What is the consequence on a batch normalization layer when choosing a small batch size?

(1p): The running variables would not accurately represent the correct mean and variance of the distribution of the dataset (noisy or non-representative estimate).
(1p): Hence damaging the performance during inference (testing) time.

Problem 8 Optimization (10 credits)

Remember that both Gradient descent (GD) and stochastic-gradient descent (SGD) could be performed by using batches of inputs.

8.1 What is the main difference in definition between GD and SGD?

□ 0 □ 1

(1p): GD performs the optimizer update every epoch (one pass over the entire training-set), while SGD performs that every iteration (batch).

8.2 Name two advantages of SGD over GD.

□ 0 □ 1 □ 2

(1p): SGD introduces noise (regularization) to the optimization step, hence could escape saddle points.
(1p): Requires more optimizer steps, but converges faster time-wise.

Revisit the formula of RMSProp:

$$s^{k+1} = \beta s^k + (1 - \beta)[\nabla_\theta L \circ \nabla_\theta L]$$

$$\theta^{k+1} = \theta^k - \alpha \frac{\nabla_\theta L}{\sqrt{s^{k+1}} + \epsilon}$$

Where θ^k are the learnable weights at time step k, α is the learning rate, β is the exponential coefficient and $\nabla_\theta L$ is the gradient of the loss w.r.t to θ ($\frac{\partial L}{\partial \theta}$)

8.3 Which problem of SGD is addressed by RMSProp?

□ 0 □ 1

(1p): SGD converges slowly or SGD has a noisy optimization trajectory.

8.4 How does RMSProp solve this problem?

□ 0 □ 1

(1p): Dampening the oscillations for high-variance directions which enables a higher learning rate.

Adam is the state-of-the-art optimizer for deep learning optimization problems, and is being used widely. It is defined by the formula:

$$\theta^{k+1} = \theta^k - \alpha \cdot \frac{\hat{m}^{k+1}}{\sqrt{\hat{v}^{k+1}} + \epsilon}$$

Including the bias-correction steps:

$$m^{k+1} = \beta_1 \cdot m^k + (1 - \beta_1)\nabla_\theta L(\theta^k)$$

$$v^{k+1} = \beta_2 \cdot v^k + (1 - \beta_2)[\nabla_\theta L(\theta^k) \circ \nabla_\theta L(\theta^k)]$$

$$\hat{m}^{k+1} = \frac{m^{k+1}}{1 - \beta_1^{k+1}}$$

$$\hat{v}^{k+1} = \frac{v^{k+1}}{1 - \beta_2^{k+1}}$$

0 □ 1 □

8.5 Which optimizers' concept does Adam combine?

(0.5p) RMSProp
(0.5p) Momentum

0 □ 1 □ 2 □

8.6 Why do we apply the bias correction?

(1p): The first iterations are very baised towards the random initialization of the of m_0 and v_0.
(1p): The bias correction uses the entire magnitude of the gradient (before the learning rate) and allows much bigger steps towards the optimum.

0 □ 1 □

8.7 Write the Netwon's method update step for w_{k+1} for a function $f : \mathbb{R}^2 \to \mathbb{R}, f(x, w) = wx$, in terms of $f(x, w)$.

(1p): $w_{k+1} = w_k - \frac{f'(x,w)}{f''(x,w)}$

0 □ 1 □

8.8 Name one advantage of Newton's Method.

(1p): Converges much faster or removes the need of a learning rate.

Problem 9 Advanced Topics (5 credits)

Consider the Recurrent Neural Network architecture (RNN):

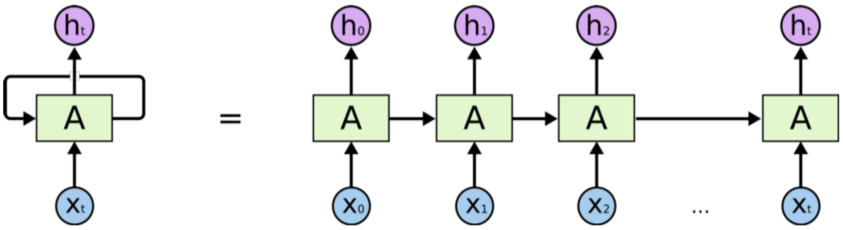

Figure 9.1: A Recurrent Neural Network

While training the model, we noticed that training loss does not drop.

9.1 What is the problem with RNNs that could cause this issue?

□ 0
□ 1

(1p): Vanishing / Exploding gradient (1p for either)

9.2 Explain why it occurs.

□ 0
□ 1

(1p): The shared weights are multiplied continuously by themselves. If their eigenvalues are lower than 1, the gradient would vanish. If > 1, they will explode.

9.3 Name one solution to the problem

□ 0
□ 1

(1p): If vanishing: Switch to LSTM/enforce eigenvalues are 1 during training (1p for either)
If exploding: gradient clipping

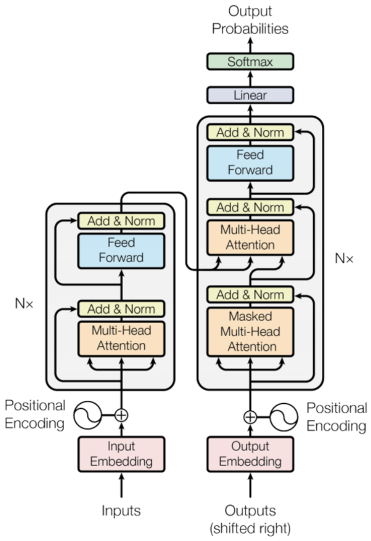

Figure 9.2: Transfomer Architecture

0 □ □ □ 1

9.4 Consider the transformer architecture above which is used for machine translation by training to predict the next word in a sentence. The network gets provided with a full input and output sentence. Why do the outputs need to be masked when fed into the decoder?

(1p): Because the full output sentence is presented we have to mask out the part of the output sentence after our predicted word.

0 □ □ □ 1

9.5 Can the transformer take an input sentence of arbitrary length? Explain why or why not.

(1p): No, because the attention layer (with fixed sized matrices K, Q, V) takes a fixed input.

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.