Introduction to Deep Learning (I2DL)
Mock Exam - *Solutions*

IN2346 - SoSe 2020

Technical University of Munich

Problem		Full Points	Your Score
1	Multiple Choice	10	
2	Short Questions	12	
3	Backpropagation	9	
Total		**31**	

Total Time: **31 Minutes**

Allowed Ressources: **None**

The purpose of this mock exam is to give you an idea of the type of problems and the structure of the final exam. The mock exam is not graded. The final exam will most probably be composed of 90 graded points with a total time of 90 minutes.

Multiple Choice Questions:

- For all multiple choice questions any number of answers, i.e. either zero (!) or 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.

How to Check a Box:

- Please **cross** the respective box: ☒ (interpreted as **checked**)
- If you change your mind, please **fill** the box: ■ (interpreted as **not checked**)
- If you change your mind again, please **circle** the box: ◉ (interpreted as **checked**)

Part I: Multiple Choice (10 points)

1. (2 points) To avoid overfitting, you can...

   ☐ increase the size of the network.

   √ **use data augmentation.**

   ☐ use Xavier initialization.

   √ **stop training earlier.**

2. (2 points) What is true about Dropout?

   ☐ The training process is faster and more stable to initialization when using Dropout.

   ☐ You should not use Leaky ReLu as non-linearity when using Dropout.

   √ **Dropout acts as regularization.**

   √ **Dropout is applied differently during training and testing.**

3. (2 points) What is true about Batch Normalization?

   √ **Batch Normalization uses two trainable parameters that allow the network to undo the normalization effect of this layer if needed.**

   √ **Batch Normalization makes the gradients more stable so that we can train deeper networks.**

   √ **At test time, Batch Normalization uses a mean and variance computed on training samples to normalize the data.**

   √ **Batch Normalization has learnable parameters.**

4. (2 points) Which of the following optimization methods use first order momentum?

   ☐ Stochastic Gradient Descent

   √ **Adam**

   ☐ RMSProp

   ☐ Gauss-Newton

5. (2 points) Making your network deeper by adding more parametrized layers will always...

   √ **slow down training and inference speed.**

   ☐ reduce the training loss.

   ☐ improve the performance on unseen data.

   √ **(Optional: make your model sound cooler when bragging about it at parties.)**

Part II: Short Questions (12 points)

1. (2 points) You're training a neural network and notice that the validation error is significantly lower than the training error. Name two possible reasons for this to happen.

Solution:

The model performs better on unseen data than on training data - this should not happen under normal circumstances. Possible explanations:

- Training and Validation data sets are not from the same distribution
- Error in the implementation
- ...

3. (2 points) You're training a neural network for image classification with a very large dataset. Your friend who studies mathematics suggests: "If you would use Newton-Method for optimization, your neural network would converge much faster than with gradient descent!". Explain whether this statement is true (1p) and discuss potential downsides of following his suggestion (1p).

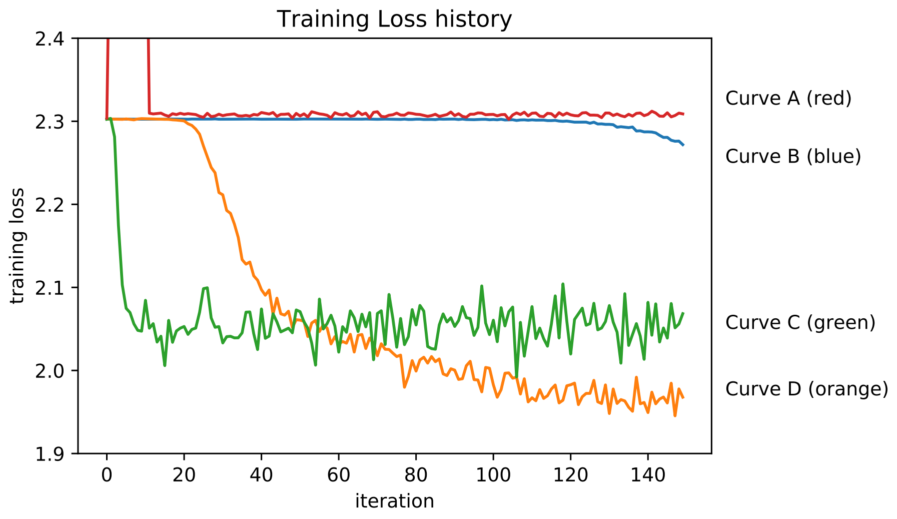
Solution:

Faster convergence in terms of number of iterations ("mathematical view"). (1 pt.)

However: Approximating the inverse Hessian is highly computationally costly, not feasible for high-dimensional datasets. (1 pt.)

4. (2 points) Your colleague trained a neural network using standard stochastic gradient descent and L2 weight regularization with four different learning rates (shown below) and plotted the corresponding loss curves (also shown shown below). Unfortunately he forgot which curve belongs to which learning rate. Please assign each of the learning rate values below to the curve (A/B/C/D) it probably belongs to and explain your thoughts.

```
learning_rates = [3e-4, 4e-1, 2e-5, 8e-3]
```



Solution:

Curve A: 4e-1 = 0.4 (Learning Rate is way too high)

Curve B: 2e-5 = 0.00002 (Learning Rate is too low)

Curve C: 8e-3 = 0.008 (Learning Rate is too high)

Curve D: 3e-4 = 0.0003 (Good Learning Rate)

5. (1 point) Explain why we need activation functions.

Solution:

Without non-linearities, our network can only learn linear functions, because the composition of linear functions is again linear.

6. (3 points) When implementing a neural network layer from scratch, we usually implement a 'forward' and a 'backward' function for each layer. Explain what these functions do, potential variables that they need to save, which arguments they take, and what they return.

Solution:

Forward Function:

- takes output from previous layer, performs operation, returns result (1 pt.)
- caches values needed for gradient computation during backprop (1 pt.)

Backward Function:

- takes upstream gradient, returns all partial derivatives (1 pt.)

7. (0 points) Optional: Given a Convolution Layer with 8 filters, a filter size of 6, a stride of 2, and a padding of 1. For an input feature map of $32 \times 32 \times 32$, what is the output dimensionality after applying the Convolution Layer to the input?

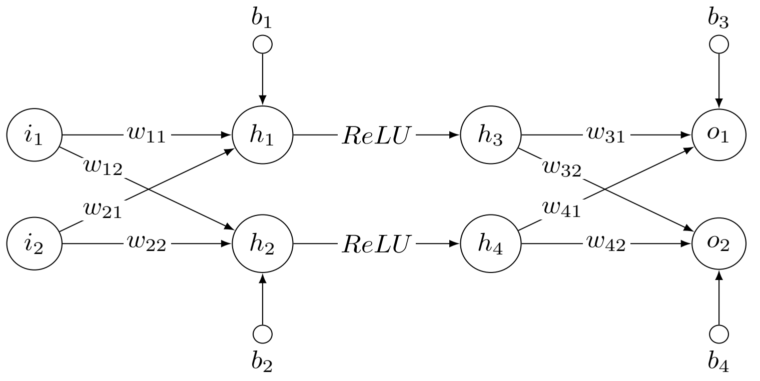
Solution:

$\frac{32-6+2\cdot 1}{2} + 1 = 14 + 1 = 15$ (1 pt.)

8x15x15

Part III: Backpropagation (9 points)

1. (9 points) Given the following neural network with fully connection layer and ReLU activations, including two input units (i_1, i_2), four hidden units (h_1, h_2) and (h_3, h_4). The output units are indicated as (o_1, o_2) and their targets are indicated as (t_1, t_2). The weights and bias of fully connected layer are called w and b with specific sub-descriptors.



The values of variables are given in the following table:

Variable	i_1	i_2	w_{11}	w_{12}	w_{21}	w_{22}	w_{31}	w_{32}	w_{41}	w_{42}	b_1	b_2	b_3	b_4	t_1	t_2
Value	2.0	-1.0	1.0	-0.5	0.5	-1.0	0.5	-1.0	-0.5	1.0	0.5	-0.5	-1.0	0.5	1.0	0.5

(a) (3 points) Compute the output (o_1, o_2) with the input (i_1, i_2) and network paramters as specified above. Write down all calculations, including intermediate layer results.

Solution:

Forward pass:

$$h_1 = i_1 \times w_{11} + i_2 \times w_{21} + b_1 = 2.0 \times 1.0 - 1.0 \times 0.5 + 0.5 = 2.0$$
$$h_2 = i_1 \times w_{12} + i_2 \times w_{22} + b_2 = 2.0 \times -0.5 + -1.0 \times -1.0 - 0.5 = -0.5$$
$$h_3 = max(0, h_1) = h_1 = 2$$
$$h_4 = max(0, h_2) = 0$$
$$o_1 = h_3 \times w_{31} + h_4 \times w_{41} + b_3 = 2 \times 0.5 + 0 \times -0.5 - 1.0 = 0$$
$$o_2 = h_3 \times w_{32} + h_4 \times w_{42} + b_4 = 2 \times -1.0 + 0 \times 1.0 + 0.5 = -1.5$$

(b) (1 point) Compute the mean squared error of the output (o_1, o_2) calculated above and the target (t_1, t_2).

Solution:

$$MSE = \frac{1}{2} \times (t_1 - o_1)^2 + \frac{1}{2} \times (t_2 - o_2)^2 = 0.5 \times 1.0 + 0.5 \times 4.0 = 2.5$$

(c) (5 points) Update the weight w_{21} using gradient descent with learning rate 0.1 as well as the loss computed previously. (Please write down all your computations.)

Solution:

Backward pass (Applying chain rule):

$$\frac{\partial MSE}{\partial w_{21}} = \frac{\partial \frac{1}{2}(t_1 - o_1)^2}{\partial o_1} \times \frac{\partial o_1}{\partial h_3} \times \frac{\partial h_3}{\partial h_1} \times \frac{\partial h_1}{\partial w_{21}} + \frac{\partial \frac{1}{2}(t_2 - o_2)^2}{\partial o_2} \times \frac{\partial o_2}{\partial h_3} \times \frac{\partial h_3}{\partial h_1} \times \frac{\partial h_1}{\partial w_{21}}$$

$$= (o_1 - t_1) \times w_{31} \times 1.0 \times i_2 + (o_2 - t_2) \times w_{32} \times 1.0 \times i_2$$

$$= (0 - 1.0) \times 0.5 \times -1.0 + (-1.5 - 0.5) \times -1.0 \times -1.0$$

$$= 0.5 + -2.0 = -1.5$$

Update using gradient descent:

$$w_{21}^{+} = w_{21} - lr * \frac{\partial MSE}{\partial w_{21}} = 0.5 - 0.1 * -1.5 = 0.65$$

Additional Space for solutions. Clearly mark the problem your answers are related to and strike out invalid solutions.