Deep Learning (19238501)

Exam

Name, given name: ____________________

Matriculation number: ____________________

Institution: ____________________

I hereby declare that I have done the work this exam independently and without help from others, and that I have not communicated my solution to other students. I am aware that clearly plagiarized solutions will lead to exams graded with 0 points and if two or more submissions contain clear duplicates beyond a reasonable probability of coincidence, all exams containing duplicate solutions will be graded with 0 points.

Signature: ____________________

Make sure all notes are clearly assigned to the relevant **number of the examination task**!

Do not hesitate to ask any exam-related questions whatsoever per E-mail to jonas.koehler@fu-berlin.de or leon.klein@fu-berlin.de.

Question:	I	II	III	IV	V	Total
Points:	14	6	10	5	15	50
Score:						

Grade:

Examiner:

I. Multiple choice

The multiple-choice questions can have one or more correct answers. You are expected to explain your answers in one or two sentences.

(A) Your training data consists of strings with 100 characters each. The characters are lower case letters from the English alphabet, i.e. there are 26 different ones. You encode your data with one-hot encoding. For training you use a learning rate of 0.001 and a batch size of 32. How many entries of each training batch tensor are non-zero? (2 pts)

- ☐ 83.200
- ☐ 80.000
- ☐ 3.200
- ☐ 2.500

(B) Consider a dense neural network with an input layer with 25 nodes, 3 hidden layers with $30, 5, 50$ nodes respectively, and an output layer with 25 nodes. The loss function is given by the Mean Squared Error. Which of the following can describe this architecture? (2 pts)

- ☐ Residual neural network
- ☐ Multilayer perceptron
- ☐ Recurrent neural network
- ☐ Autoencoder

(C) You have set up a neural network for a supervised property prediction problem of a scalar property with a mean-squared error as loss function. Although you have 1000 data points you decide to use just a single batch with 32 samples for training. While performing gradient descent you notice that the training loss stagnates and does not converge to zero. Which of the following are likely true? (2 pts)

- ☐ L_2 regularization of the weights will help with convergence
- ☐ The neural network is not large enough to represent the predictor.
- ☐ When using the whole data set, the neural network is going to underfit.
- ☐ When using the whole data set, the neural network is going to overfit.

(D) Consider a dense neural network with 5 hidden layers, ReLU activations, and a scalar output. All weights are small positive values and all biases zero. Given input which consists of vectors with only negative entries what can be said about the output? (2 pts)

- ☐ The output of the network will be zero.
- ☐ The output of the network will be negative.
- ☐ The output of the network will be larger than zero.
- ☐ The output can be positive or negative.

(E) Which of the following is true about LSTMs? (2 pts)

- ☐ They consist of exactly three cells
- ☐ They can be used to classify sequences
- ☐ They share weights between unfolded cells
- ☐ They do not have any trainable weights

(F) Given color images with dimension 64×64 and 3 color channels, what is the output shape when using a 2D convolution with kernel size 5×5, valid padding, stride 1 and 32 filters followed by a 2×2 max pooling layer? (4 pts)

- ☐ $60 \times 60 \times 96$
- ☐ $32 \times 32 \times 32$
- ☐ $32 \times 32 \times 96$
- ☐ $30 \times 30 \times 32$

How many trainable parameters do we have (ignoring bias)?

- ☐ 9600
- ☐ 2400
- ☐ 800
- ☐ 75

II. Image Classification and Regularization

You want to classify 64×64 RGB images from a image dataset with 10^5 images into 8 classes. You chose the following architecture:

- Layer 1: 2D Convolution with valid padding, a 8×8 kernel, and 64 output channels
- Layer 2: ReLU
- Layer 3: 2D Convolution with valid padding, a 4×4 kernel, and 8 output channels
- Layer 4: ReLU and flattening
- Layer 5: Linear layer with 2048 output neurons
- Layer 6: ReLU
- Layer 7: Linear layer with 8 output neurons
- Layer 8: Softmax layer

You train using Adam optimization with a learning rate of 10^{-4}, a batch size of 128, and a dropout rate of 10%.

Answer the following questions. You are expected to briefly explain your answers in a few sentences.

(A) What is true for the model? (2 pts)

☐ The network output uses one-hot encoding.

☐ The network is a fully connected multilayer perceptron.

(B) During training, you notice that the validation loss stagnates quite early on, while the training loss keeps decreasing. Which of the following strategies would be promising to reduce overfitting? (4 pts)

☐ Perform data augmentation by random rotation, cropping, flipping.

☐ Perform data augmentation by blurring and adding random noise.

☐ Increase the dropout rate.

☐ Choose a smaller learning rate

☐ Rerun the optimization 100 times and take the parameters with the smallest training loss.

☐ Incorporate pooling layers.

☐ Remove convolutional layers to reduce the number of parameters.

☐ Decrease the batch size to avoid convergence to local minima and saddle points.

III. Transpose Autoencoder

Consider a linear encoder $E_{\boldsymbol{a}} : \boldsymbol{x} \mapsto z$, $z = \boldsymbol{a}^T \boldsymbol{x}$, with a trainable $\boldsymbol{a} \in \mathbb{R}^n$. $E_{\boldsymbol{a}}$ performs dimension reduction by mapping n-dimensional data points from a mean-free dataset to a single latent variable z. Instead of defining an independent decoder network, we use the transpose decoder $D_{\boldsymbol{a}} : z \mapsto \hat{\boldsymbol{x}} = \boldsymbol{a} z$.

The autoencoder is trained to minimize the reconstruction loss

$$L(\boldsymbol{a}) = \frac{1}{N} \sum_{i=1}^{N} \| D_{\boldsymbol{a}}(E_{\boldsymbol{a}}(\boldsymbol{x}_i)) - \boldsymbol{x}_i \|_2^2,$$

where $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ is the dataset.

Answer the following questions. You do not have to prove anything as long as you are able to argue clearly.

(A) Which optimizer would you chose for training and why? How would you choose the batch size? (2 pts)

(B) What can you say about the magnitude $|\boldsymbol{a}^*|$ of the optimal parameter $\boldsymbol{a}^*$? (2 pts)

(C) How does this transpose autoencoder relate to PCA? (2 pts)

(D) Now you want to enforce $\boldsymbol{a}$ to be a unit vector and optimize the reconstruction loss with SGD. How can you make sure that the network trains successfully and $|\boldsymbol{a}| = 1$ is satisfied at all times during training? (2 pts)

- ☐ Optimize only for first $n-1$ entries $\boldsymbol{a}_{<n} \in \mathbb{R}^{n-1}$ and set $a_n = \sqrt{1 - \sum_{i=1}^{n-1} a_i^2}$.
- ☐ Optimize for $\tilde{\boldsymbol{a}} \in \mathbb{R}^n$ and set $\boldsymbol{a} = \tilde{\boldsymbol{a}} / |\tilde{\boldsymbol{a}}|$.
- ☐ Optimize for $\tilde{\boldsymbol{a}} \in \mathbb{R}^n$ and set $\boldsymbol{a} = \sqrt{\text{Softmax}(\tilde{\boldsymbol{a}})}$, where the square root is evaluated element-wise.
- ☐ Optimize for $\tilde{\boldsymbol{a}} \in \mathbb{R}^n$ and set $\boldsymbol{a} = \sqrt{\log(\text{Softmax}(\tilde{\boldsymbol{a}}))}$, where the square root is evaluated element-wise.

(E) For the version with $|\boldsymbol{a}| = 1$, formulate an alternative loss function that yields the same optimum and only invokes the encoder but not the decoder. This alternative loss function should have the form (2 pts)

$$L_2(\boldsymbol{a}_1, \dots, \boldsymbol{a}_B) = \frac{1}{B-1} \sum_{i=1}^{B} g\left(E_{\boldsymbol{a}}(x_i), \frac{1}{B} \sum_{i=1}^{B} E_{\boldsymbol{a}}(x_i) \right),$$

where B denotes the batch size. Specify $g(z, \overline{z})$ and explain the equivalence (no need to prove anything if you can argue clearly). How does this loss function behave on the version of the autoencoder with unconstrained $|\boldsymbol{a}|$?

IV. Universal Representation Theorem

Consider a dense neural network $f(x) : \mathbb{R} \to \mathbb{R}$ with sigmoid activations and one hidden layer with N neurons.

(A) Which of the following expressions are true? (5 pts)

- ☐ For all functions $g \in C^2([0,1])$ and all $\varepsilon > 0$ there exist a number N and a choice of network parameters such that $\sup_{x\in[0,1]} |f(x) - g(x)| < \varepsilon$.
- ☐ Let $g_2 : (0,1) \to [-1,1],\ g_2(x) = \sin(1/x)$. For all $\varepsilon > 0$ there exist a number N and a choice of network parameters such that $\sup_{x\in(0,1)} |f(x) - g_2(x)| < \varepsilon$.
- ☐ Let $g_3 : \mathbb{R} \to [-1,1],\ g_3(x) = \sin(x)$. For all $\varepsilon > 0$ there exist a number N and a choice of network parameters such that $\sup_{x\in\mathbb{R}} |f(x) - g_3(x)| < \varepsilon$.
- ☐ Let $g_4 : \mathbb{R} \to \mathbb{R}$ be a polynomial of arbitrary but fixed degree. For all $\varepsilon > 0$ there exist a number N and a choice of network parameters such that $\|e^{-x^2}(f - g_4)\|_2 < \varepsilon$, where $\|\cdot\|_2$ denotes the L^2-norm.
- ☐ For all $\varepsilon > 0$ there exists a number N such that for all functions $g : [0,1] \to \mathbb{R}$ there is a choice of network parameters with $\|f - g\|_2 < \varepsilon$.

Reminders/Definitions:

- $C^2([0,1])$ *is the set of twice continuously differentiable functions* $f : [0,1] \to \mathbb{R}$. *On compact intervals like* $[0,1]$, *differentiability is defined as being differentiable on some open set that contains the compact interval.*
- *The* L^2*-norm for the function given in the fourth answer can be computed as*

$$\|f\|_2 = \left(\int f(x)^2 dx\right)^{\frac{1}{2}}.$$

V. Euclidean Message Passing Networks

Point-cloud data sets (e.g. molecules, LIDAR scans) require different neural network architectures than classic CNNs to handle arbitrary indexing and global orientation of the data. Such data is usually structured as follows:

- Each particle j has a 3D position $\mathbf{r}_j \in \mathbb{R}^3$
- Each particle j has some feature $\mathbf{f}_j \in \mathbb{R}^F$ attached to it (e.g. charge, atom type).
- A single data item is given by a system of N particles. Thus, a data item $\mathbf{x} = (\mathbf{r}, \mathbf{f})$ is given by a position tensor $\mathbf{r} \in \mathbb{R}^{N\times 3}$ and a feature tensor $\mathbf{f} \in \mathbb{R}^{N\times F}$. E.g. a molecule would consist of a bunch of particles each having a 3D position and an atom type. The particle with index j would have the position $\mathbf{r}_j$ and the attached feature $\mathbf{f}_j$. If we consider a full batch with batch size B we will have a $\mathbb{R}^{B\times N\times 3}$ tensor for the positions and a $\mathbb{R}^{B\times N\times F}$ tensor for the features.

A recently proposed neural network layer for such data works as follows: the input features $\mathbf{f} \in \mathbb{R}^{N\times F}$ are transformed into output features $\mathbf{f}' \in \mathbb{R}^{N\times F'}$ in two steps. First, the particles interact in a *message passing step*. Second, the features are transformed in a *point-wise convolution*.

In the *message passing step* we first compute all pair-wise distances $d_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|_2$ between particles, which gives the distance matrix $\mathbf{d} \in \mathbb{R}^{N\times N}$. Then we use a multilayer perceptron $\phi\colon \mathbb{R} \to \mathbb{R}^F$ which transforms each entry of $\mathbf{d}$ into $\mathbf{d}' = \phi(\mathbf{d}) \in \mathbb{R}^{N\times N\times F}$. *Note:* we share ϕ across all entries of $\mathbf{d}$. Finally, we compute the output of the message passing layer as

$$\bar{\mathbf{f}}_{j,k} = \rho\left(\sum_i \mathbf{d}'_{j,i,k}\mathbf{f}_{i,k}\right) \tag{1}$$

where ρ is some point-wise nonlinearity (e.g. a ReLU or Softplus).

In the *point-wise convolution* we take another multilayer perceptron $\psi\colon \mathbb{R}^F \to \mathbb{R}^{F'}$ and transform each particle's feature $\bar{\mathbf{f}}_j$ individually into

$$\mathbf{f'}_j = \psi\left(\bar{\mathbf{f}}_j\right). \tag{2}$$

Note: we share ψ across all particles!

Important: all programming solutions should be valid `Python 3` code and make proper use of the `numpy` and `PyTorch` API. We will subtract points for wrong code.

(A) Implement the *point-wise convolution* layer. You may use the following code template for your solution. For the multilayer perceptron ψ chose one hidden layer with 128 hidden units and a softplus nonlinearity. (4 pts)

```
import torch
class PointwiseConvolutionLayer(torch.nn.Module):
    def __init__(self, N, F, F_prime):
        super().__init__()
        # ... your code here ...

    def forward(self, f_bar_batch):
        # ... your code here ...
```

(B) Implement the *message-passing step*. You may use the following code template for your solution. For the multilayer perceptron ϕ chose one hidden layer with 128 hidden units and a softplus nonlinearity. (6 pts)

Note: we have already inserted the code for computing the all-distance matrix **d**.

```
class MessagePassingLayer(torch.nn.Module):
    def __init__(self, N, F):
        super().__init__()
        # ... your code here ...

    def forward(self, r_batch, f_batch):
        # the following line will compute pairwise distances
        # of shape (B, N, N) when given input tensors of shape (B, N, 3)
        d = torch.cdist(r_batch, r_batch)
        # ... your code here ...
```

(C) Write a property predictor module that predicts a scalar property for the *whole* system of N particles (e.g. a molecule). The module should use the message passing and pointwise convolution layers from above. Furthermore, the property should neither depend on global rotations or translations of the system nor on the order of two particles that have the exact same features. (5 pts)

Argue in two or three sentences, why the predicted property is invariant under these modifications.

```
class PropertyPredictor(torch.nn.Module):
    def __init__(self, N, F, F_prime=1):
        # ... your code here ...

    def forward(self, r_batch, f_batch):
        # ... your code here ...
```