CS 482/682 Machine Learning: Deep Learning
Mock exam questions

Note: Please note that the primary purpose of this mock exam is for you to review your understanding of important concepts taught in this class. Specifically, the mock exam **is not** representative of the true midterm exam with respect to length or composition of questions (there will be no true/false or multiple choice questions.

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	10	12	25	26	8	4	25	28	18	156
Score:										

1. **True/False Questions**
   (a) (2 points) In standard recurrent neural networks, the magnitude of the largest singular value of the weight matrix $\boldsymbol{W}$ is related to the exploding/vanishing gradient problem.
   ◯ True ◯ False
   (b) (2 points) When moving from fully connected multi-layer perceptrons to convolutional neural networks, a key assumption is translation invariance.
   ◯ True ◯ False
   (c) (2 points) There is no obvious relation between convolutional layers and fully-connected layers.
   ◯ True ◯ False
   (d) (2 points) Linear regression with L_2 loss is convex and we can always find the globally optimal solution. ◯ True ◯ False
   (e) (2 points) Perceptrons can model any Boolean function.
   ◯ True ◯ False
2. **Multiple-choice Questions** (More than one answer can be correct!)
   (a) (4 points) Convolutional neural network (ConvNet) architectures ...
   ◯ Most of the parameters of VGG-16 are in the early convolutional layers
   ◯ Blocks of 3 3×3 convolutions with stride 1, as is done in VGG, mimic the 11×11 convolutions in AlexNet but has fewer parameters and more non-linearities
   ◯ Residual connections, as in the ResNet, introduce "skip ahead connections" with "addition nodes" such that the gradient is distributed, enabling more effective optimization
   ◯ Convolutions with strides > 1 can be used instead of pooling layers
   (b) (4 points) Recurrent neural networks and LSTM ...
   ◯ Can model sequential data and are often used for natural language processing tasks
   ◯ Do not suffer from vanishing and exploding gradient problems

◯ Vanilla-RNN can learn to retain the most relevant parts of each frame

◯ LSTMs use the forget gate to extend the number of frames it can retain information over

(c) (4 points) You observe that your training loss initially decreased but does not converge to a stable solution. Which of the following may help you achieve convergence?

◯ Increasing the network depth

◯ Increasing the batch size

◯ Increasing the learning rate

◯ Increasing the dropout rate

3. **Convolutional Neural Networks: Concepts** In this problem you will try to reason why the sigmoid activation function in intermediate layers has largely been replaced by Rectified Linear Units (ReLU).

(a) (4 points) First, state the mathematical formula of the sigmoid $\sigma(x)$ and sketch (with annotations of y-axis crossing and limits) the curve in the interval $[-10, 10]$.

(b) (4 points) The derivative of the sigmoid function $\sigma(x)$ is given by $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$. Plot the derivative into the same diagram. What are the minimum and maximum values of the derivative?

(c) (4 points) Briefly explain the consequences of the previous observations for parameter updates during training of *very deep (or recurrent)* neural networks via backpropagation when using sigmoid as activation function?

(d) (5 points) A common alternative to the sigmoid activation function, that has contributed substantially to the rise of deep learning, is the rectified linear unit (ReLU) activation function. Give its

formula, plot its curve, and briefly explain how ReLU solves the aforementioned problem of the sigmoid.

(e) (8 points) However, the ReLU is not ideal. First, give two examples of problems that remain or are introduced when using the ReLU. Second, introduce a small modification to ReLU that solves these problem, give it's formula, and plot it in the diagram of (d).

4. **Convolutional Neural Networks: Regularization** In this problem you will consider regularization approaches to avoid overfitting when training convolutional neural networks (ConvNets).

(a) (8 points) First, briefly explain the *bias – variance trade-off* and provide explanations on what we mean by bias and variance. Second, describe how this relates to optimizing the parameters of ConvNets.

(b) (9 points) In class, we discussed batch normalization as a way of introducing regularization. We now consider the training phase. Using bullet points, please outline the procedure of applying batch normalization. Assume that your batch contains $i = 1, \ldots, N$ instances $\boldsymbol{x}_i^{(j)}$ with $j = 1, \ldots, M$ feature maps. Please provide 1) a formula describing the computations, and 2) a brief rationale on why this computation is performed.

(c) (2 points) How is batch normalization applied in the testing phase?

(d) (7 points) Dropout is another way of regularizing ConvNets. First, please briefly describe how dropout is implemented and why it regularizes ConvNets during training. Second, please briefly describe the change(s) that must be made for testing? Finally, can you find a reason to apply dropout during testing?

5. **Convolutional Neural Networks: Optimization** In this problem, you will analyze different optimization strategies that can be applied to find optimal parameters when training neural networks using gradient descent-type minimization, e. g. backpropagation. We seek to find $\hat{\boldsymbol{x}}$ such that $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x})$, where $L(\boldsymbol{x}) : \mathbb{R}^2 \mapsto \mathbb{R}$ is an arbitrary multi-dimensional scalar-valued function (function accepts vector and yields scalar), e. g. a loss function.

(a) (8 points) Consider the following notation. At iteration t, where $t \gg 0$, we have:

$$\begin{aligned} \text{Gradient:} \quad & \boldsymbol{g}_t = \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_t) \\ \text{?:} \quad & \\ \text{Element-wise update:} \quad & (\mathrm{d}\boldsymbol{x}_t)_i = \\ \text{Update step:} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \mathrm{d}\boldsymbol{x}_t \end{aligned}$$

where $(\boldsymbol{x}_t)_i$ corresponds to the i-th element of $\boldsymbol{x}$. Please provide the missing computations for SGD with momentum, and RMSProp.

6. **Convolutional Neural Networks: Architecture**

One important consideration when designing convolutional neural network architectures is the *receptive field*. Large receptive fields can be achieved via large kernels, but there is another solution. Consider a convolutional kernel of size 7×7 that obviously has a receptive field of 7×7 pixels. Please briefly describe an alternative way of achieving the same receptive field (Hint: Think about the VGG architecture). Name and explain one advantage of the alternative method you just described.

7. **Multi-layer perceptron** In this question, you will study the capacity of multi-layer perceptrons (MLPs). Consider the dataset shown in Fig. 1 that was sampled from a 2D distribution. Note how the decision boundary that optimally separates these 2D points can be modeled by two simple polygons. In the following exercise, you will design an MLP architecture that exploits this

(a) (4 points)

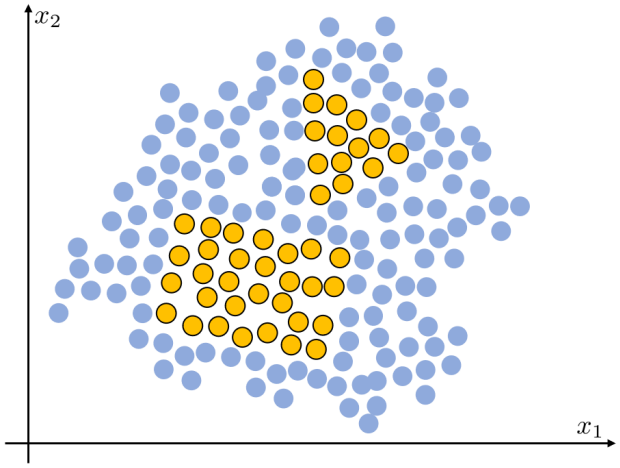

Figure 1: Visualization of an example 2D dataset with true/false annotation. For mathematical modeling, *true* corresponds to an output of 1 while *false* corresponds to an output of 0. Every instance $\boldsymbol{x}_i = (x_1^{(i)}, x_2^{(i)})$ is visualized as circle and colorized depending on the respective annotation: Instances are colored in blue if *false* and colored in orange with black outline if *true*.

observation to classify data sampled from this distribution. Here, we consider perceptrons of model $f(\boldsymbol{u}, \boldsymbol{W}, b) = \begin{cases} 1, & \text{if } \boldsymbol{W}\boldsymbol{u} > b \\ 0, & \text{else} \end{cases}$ (i. e. threshold activation).

(a) (7 points) As we learned in class, every perceptron over the inputs $\boldsymbol{x}$ defines a linear decision boundary. Start by drawing all required linear decision boundaries on Fig. 1 and label them (e.ġ. $L_1, \ldots, L_n$). Please label the positive direction of the boundary (i.e. $f(x) > 0$).

(b) (8 points) Using the above result, draw the computational graph of an MLP with two hidden layers that classifies the dataset. Make use of your annotations in Fig. 1 (e.ġ. $L_1, \ldots, L_n$) to link your computational graph with your drawings.

(c) (10 points) By design, the previous MLP has two hidden layers. However, we have seen in class that the same (and any arbitrary decision boundary) can be approximated using an *MLP with a single hidden layer*. Please provide a brief walk-through on how this can be achieved (6 bullet points maximum!). You can use small schematic drawings to support your reasoning.

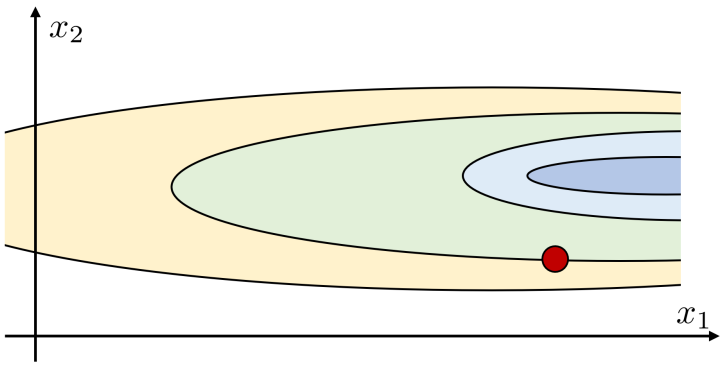

Figure 2: Contour lines of an arbitrary cost function $L(\boldsymbol{x}) : \mathbb{R}^2 \mapsto \mathbb{R}$. The function is convex in the displayed area with larger ellipses representing contour lines of greater values of $L(\boldsymbol{x})$. The red circle represents the current estimate $\boldsymbol{x}_t$ and we attempt to find the $\hat{\boldsymbol{x}}$ that minimizes L. *Please add your plots to this figure.*

8. **Convolutional Neural Networks: Optimization** In this problem, you will analyze different optimization strategies that can be applied to find optimal parameters when training neural networks using gradient descent-type minimization, e. g. backpropagation. We seek to find $\hat{\boldsymbol{x}}$ such that $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x})$, where $L(\boldsymbol{x}) : \mathbb{R}^2 \mapsto \mathbb{R}$ is an arbitrary multi-dimensional scalar-valued function (function accepts vector and yields scalar), e. g. a loss function, shown in Fig. 2. Note, how values of the function change much more drastically in one direction compared to the other. The value at contour lines (lines on which $L(x)$ has the same value) is proportional to the size of the ellipse (i. e. *smallest* ellipse has *smallest* value!). **After multiple optimization steps**, the optimization algorithm has arrived at $\boldsymbol{x}_t$, that is highlighted as red circle in Fig. 2.

   (a) (8 points) Name and describe 2 problems of optimization with vanilla SGD and name and describe 2 approaches to mitigate them.

(b) (8 points) Consider the following notation. At iteration t, where $t \gg 0$, we have:

$$\begin{aligned} \text{Gradient:} \quad & \boldsymbol{g}_t = \nabla_{\boldsymbol{x}} L(\boldsymbol{x}_t) \\ \text{?:} \quad & \\ \text{Element-wise update:} \quad & (\mathrm{d}\boldsymbol{x}_t)_i = \\ \text{Update step:} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \mathrm{d}\boldsymbol{x}_t \end{aligned}$$

where $(\boldsymbol{x}_t)_i$ corresponds to the i-th element of $\boldsymbol{x}$. Please provide the missing computations for SGD with momentum, and AdaGrad.

(c) (12 points) Now consider the situation in Fig. 2. In that figure, please plot the next update directions for vanilla SGD, SGD with momentum, and AdaGrad (do not consider the learning rate, i. e. arrows should be legible). For every optimizer, you should only draw **a single arrow**. Please also provide a brief statement in the space provided below explaining why you have decided to draw the update in this specific way.

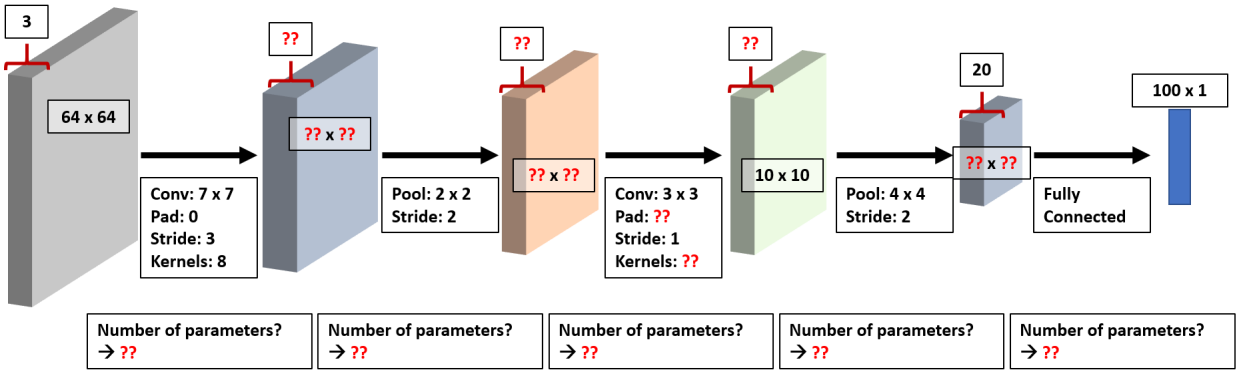

Figure 3: A shallow convolutional neural network for image classification. Your task is to derive the missing quantities highlighted as ??. Note: You will calculate the number of parameters in part (b).

9. **Convolutional Neural Networks: Architecture** Consider the network architecture provided in Fig. 3. The final output is a vector of size 100×1, e. g. for image classification in a problem with 100 different classes. Assume *ReLU activation* for all convolutional and fully connected layers.

   (a) (8 points) Several quantities (shown with question marks as ??) are missing from the schematic in Fig. 3. Please derive them from the provided values. If you write them directly into the figure make sure it is obvious (e. g. using arrows) which ?? they belong to.

   (b) (4 points) Now, please show the calculation for the number of parameters per layer and the total number of parameters. It is enough to write the formulae for the number of parameters; it is not necessary to perform the calculations to get a final number.

   (c) (6 points) In class, we discussed the counter-intuitive observation that very deep networks (e. g. with 50 or more layers) that follow the simple structure shown in Fig. 3 do not necessarily outperform shallower networks. Please provide the reason for this phenomenon, name the observations that led to this conclusion, and *briefly* describe a way of overcoming this problem. To help your

argumentation, sketch the key changes to the architectural setup.