Klausur Winter 2019/2020, Fragen und Antworten

Introduction to Deep Learning (Technische Universität München)

Introduction to Deep Learning (I2DL)

Mock Exam - *Solutions*

IN2346 - WiSe 2019/2020

Technical University of Munich

Problem		Full Points	Your Score
1	Multiple Choice	18	
2	Short Questions	24	
3	Logistic Regression	12	
4	Backpropagation	12	
5	Layers	12	
6	CNN Architectures	12	
Total		**90**	

Total Time: **90 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.

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 (18 points)

1. (2 points) Which of the followings models can be used in unsupervised learning?
   - √ **Autoencoder**
   - √ **PCA**
   - √ **K-means**
   - ☐ Linear Regression

2. (2 points) To avoid overfitting, you can...
   - ☐ increase the size of the network.
   - √ **use data augmentation.**
   - ☐ use Xavier initialization.
   - √ **stop training earlier.**

3. (2 points) You want to train a neural network to predict housing prices in Munich. Which of the following output functions is best suited for this regression problem?
   - ☐ Softmax Function
   - ☐ Sigmoid Function
   - √ **Identity Function**
   - ☐ TanH Function

4. (2 points) Compared to the L1 loss, the L2 loss...
   - ☐ is robust to outliers.
   - ☐ is costly to compute.
   - √ **has a different optimum.**
   - ☐ will lead to sparser solutions.

5. (2 points) What is true about Dropout?
   - ☐ When using dropout, our network can be seen as an ensemble of multiple independent smaller networks.
   - √ **Dropout reduces the co-adaptation between neurons.**
   - √ **Dropout acts as regularization.**
   - ☐ Dropout can also be applied at test time, but the output has to be scaled by the number of training samples.

6. (2 points) What is true about Batch Normalization?

   ☐ Batch Normalization is applied after activation functions to normalize the mean and variance of the output.

   √ **Batch Normalization makes the gradients more stable, so 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.**

7. (2 points) Which of the following optimization methods use first order momentum?

   ☐ Stochastic Gradient Descent

   √ **Adam**

   ☐ RMSProp

   ☐ Gauss-Newton

8. (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.

   √ **make your model sound cooler when bragging about it at parties.**

9. (2 points) What can 1×1 convolutions be used for?

   ☐ To perform feature selection.

   √ **To reduce the number of channels before a costly operation.**

   √ **To make a network more complex.**

   √ **To replace fully-connected layers in order to make a network fully-convolutional.**

Part II: Short Questions (24 points)

1. (2 points) 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.)

$15 \times 15 \times 8$ (1 pt.)

2. (2 points) Your colleague wrote the following Python code to compute the score for each sample in a linear regression setting. What problem do you see with this implementation? Write a better implementation that fixes this problem.

```
# the following variables have been defined outside this snippet:
# X: numpy matrix of shape N*D containing N samples with D features
# w: numpy vector of length D containing the weights
# b: scalar representing the bias parameter

def score_for_sample(x, w, b):
    score = b
    for i in range(len(x)):
        score += x[i] * w[i]
    return score

scores = np.zeros(len(X))
for i in range(len(X)):
    scores[i] = score_for_sample(X[i], w, b)
```

Solution:

Using FOR-loops will lead to bad performance. Use vectorization instead. (1 pt.)

```
scores = X.dot(w) + b
```

(1 pt.)

3. (2 points) Show that $softmax(s_i)$ is equal to $softmax(s_i - const)$ where s_i is the score for a specific class i and $const$ is a constant that we subtract from all scores of a sample. Explain why this property of the Softmax function is useful when training neural networks.

Solution:

$softmax(s_i - const) = \frac{e^{(s_i - const)}}{\sum_k^C e^{(s_k - const)}} = \frac{e^{s_i} \cdot e^{-const}}{\sum_k^C e^{s_k} \cdot e^{-const}} = \frac{e^{s_i} \cdot e^{-const}}{e^{-const} \cdot \sum_k^C e^{s_k}} = \frac{e^{s_i}}{\sum_k^C e^{s_k}} = softmax(s_i)$ (1 pt.)

We use this property to improve numeric stability by subtracting a sample's maximum score from all scores of that sample. (1 pt.)

4. (2 points) You're training a neural network and notice that the validation error is significantly lower than the training error. What could be a reason for that?

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

(1 pt. if a reasonable explanation was given)

5. (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.

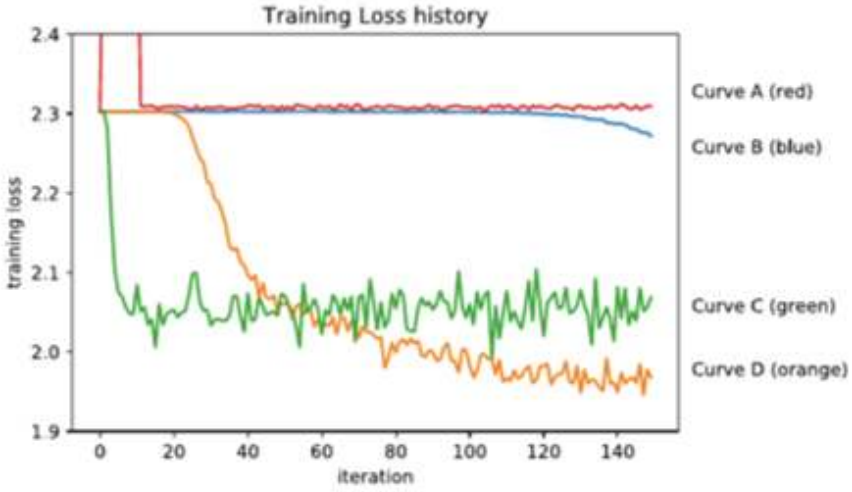
Solution:

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

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

6. (3 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.

$$\texttt{learning\_rates} = [3e{-}4, \quad 4e{-}1, \quad 2e{-}5, \quad 8e{-}3]$$



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)

7. (2 points) Explain how and why LSTM networks often outperform traditional RNNs.

Solution:

It is difficult for traditional RNNs to learn long-term dependencies due to vanishing gradients. (1 pt. for long-term dependencies)

The cell state in LSTMs improves the gradient flow and thereby allows the network to learn longer dependencies. (1 pt. for cell state)

8. (2 points) You're working for a cool tech startup that receives thousands of job applications every day, so you train a neural network to automate the entire hiring process. Your model automatically classifies resumes of candidates, and rejects or sends job offers to all candidates accordingly. Which of the following measures is more important for your model? Explain.

   $\text{Recall} = \frac{True\ Positives}{Total\ Positive\ Samples}$

   $\text{Precision} = \frac{True\ Positives}{Total\ Predicted\ Positive\ Samples}$

   Solution:

   Precision: High precision means low rate of false positives.

   False Negatives are okay, since we get "thousands of applications" it's not too bad if we miss a few candidates even when they'd be a good fit. However, we don't want False Positives, i.e. offer a job to people who are not well suited.

9. (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.

10. (2 points) How will weights be initialized by Xavier initialization? Which mean and variance will the weights have? Which mean and variance will the output data have?

    Solution:

    With Xavier initialization we initialize the weights to be Gaussians with zero mean and variance $Var(w) = 1/n$, where n is the amount of neurons in the input. (1 pt.)

    As a result, the output will have zero mean, and similar variance as the input. (1 pt.)

11. (2 points) Explain the purpose of separate validation and test datasets and why both are necessary.

Solution:

Both datasets are used to estimate the performance of our model on unseen data. During the development process, we use the validation set to tune hyperparameters and to make further design decisions. As a result, the validation dataset is not completely unseen anymore, for which reason we need the additional test set to accurately measure the performance of our final model on unseen data.

12. (2 points) Explain how K-fold Cross-Validation works, and how it is used to calculate validation scores for a given model.

Solution: We split the data into K parts. We train the model on K-1 parts and validate on the held out part. We repeat this process K times, so that each part was held out once, then we average results to obtain our final validation score.

(1 pt for data split into K-1 train and 1 validation; 1 pt for result averaging)

Part III: Logistic Regression (12 points)

1. (12 points) We want to build a binary classifier based on a logistic regression model that detects dogs, i.e. returns a probability p that a given sample is a dog, while the probability for not being a dog is $1 - p$.

   (a) (1 point) To predict probabilities p, we use the Sigmoid Function $\sigma(z)$. Please specify the formula for $\sigma(z)$.

   Solution:

   $\sigma(z) = \frac{1}{1+e^{-z}}$

   (b) (3 points) Show that the derivative of the Sigmoid Function $\sigma(z)$ w.r.t. the input z is equal to $\sigma(z)(1 - \sigma(z))$.

   Solution:

   $\frac{\partial \sigma(z)}{\partial z} = \frac{\partial}{\partial z}\frac{1}{1+e^{-z}} = \frac{-1\cdot(-e^{-z})}{(1+e^{-z})^2} = \frac{1\cdot e^{-z}}{(1+e^{-z})^2}$
   (1 pt. for differentiation of $\sigma(z)$)

   $= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \sigma(z) \cdot (1 - \sigma(z))$
   (1 pt. for splitting square into two factors, showing $\sigma(z)$)

   $1 - \sigma(z) = 1 - \frac{1}{1+e^{-z}} = \frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}} = \frac{e^{-z}}{1+e^{-z}}$
   (1 pt. for showing $1 - \sigma(z) = \frac{e^{-z}}{1+e^{-z}}$)

(c) (3 points) Using the result above, compute the derivative of the Binary Cross Entropy Loss (of a single sample) w.r.t. the score z.

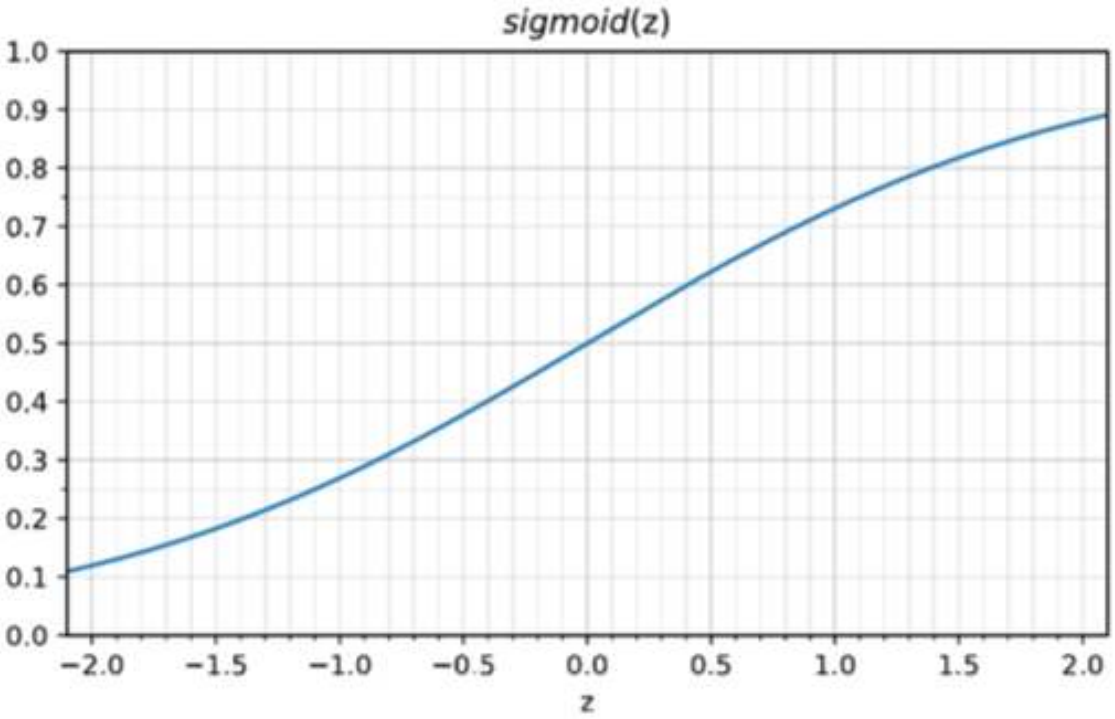
Solution:

$CE(p) = -t \cdot log(p) - (1-t)log(1-p)$

$\frac{\partial CE(p)}{\partial p} = -\frac{t}{p} - \frac{1-t}{1-p} \cdot (-1) = -\frac{t}{p} + \frac{1-t}{1-p}$
(1 pt. for differentiation of $CE(p)$)

$\frac{\partial p}{\partial y} = \frac{\partial \sigma(y)}{\partial y}$ is known from the problem above as $\sigma(y)(1-\sigma(y))$ i.e. $p(1-p)$.

$\frac{\partial CE(p)}{\partial y} = \frac{\partial CE(p)}{\partial p}\frac{\partial p}{\partial y} = (-\frac{t}{p} + \frac{1-t}{1-p}) \cdot p(1-p) = -t(1-p) + p(1-t)$
$= -t + tp + p - pt = p - t$
(1pt. for chain rule, 1 pt. for correct rearrangement)



(d) (5 points) Assume you are training our logistic regression classifier on a single picture of a dog ($t = 1$), with features $x = [0.8, 0.5, -1]^T$. The current model parameters are $w = [0.5, 0.4, -0.3]^T$ and $b = 0.2$. Perform a gradient descent iteration with learning rate $\alpha = 0.1$ and update the weight w_1 (indexing starts with 1) with respect to the Binary Cross Entropy Loss. You can use the plot above to estimate values of the sigmoid function.

Solution:

Score: $y = x^T w + b = 0.9 + 0.2 = 1.1$ (0.5 pt)

Probability (from plot) $p = \sigma(y) = \sigma(1.1) \approx 0.75$ (1 pt.)

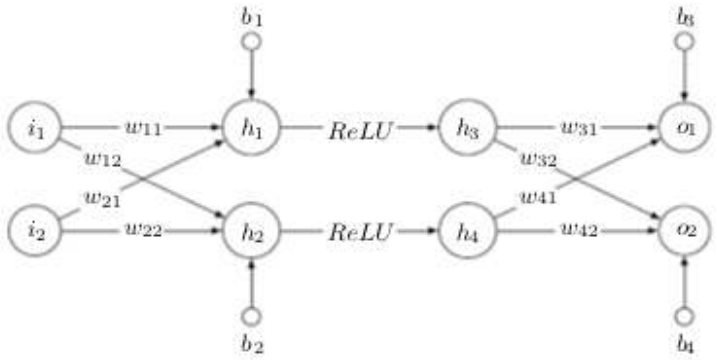
Inserting into result from above: $\frac{\partial CE(p)}{\partial y} = p - t = 0.75 - 1 = -0.25$ (1 pt.)

Applying chainrule to get derviative w.r.t. w_1: $\frac{\partial CE(p)}{w_1} = \frac{\partial CE(p)}{\partial y}\frac{y}{\partial w_1} = -0.25 \cdot x_1 = -0.25 \cdot 0.8 = -0.2$ (1 pt.)

Update the weight: $w_1^{(t+1)} = w_1^{(t)} - \alpha \frac{\partial CE(p)}{w_1} = 0.5 - 0.1 \cdot (-0.2) = 0.52$ (1.5 pt.)

Part IV: Backpropagation (12 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) (4 points) Write down your calculations about the forward pass, i.e., get the mean squared loss. (Please write the computations of each step in the forward pass.)

(a) (4 points) Write down your calculations about the forward pass, i.e., get the mean squared loss. (Please write the computations of each step in the forward pass.)

Solution:

Forward pass:

$$h_1 = i_1 \times w_{11} + i_2 \times w_{21} + b_1 = 2.0 \times 1.0 - 1.0 \times 0.5 + 0.5 = 2.0$$

$$h_2 = i_1 \times w_{12} + i_2 \times w_{22} + b_2 = 2.0 \times -0.5 + -1.0 \times -1.0 - 0.5 = -0.5$$

$$h_3 = max(0, h_1) = h_1 = 2$$

$$h_4 = max(0, h_2) = 0$$

$$o_1 = h_3 \times w_{31} + h_4 \times w_{41} + b_3 = 2 \times 0.5 + 0 \times -0.5 - 1.0 = 0$$

$$o_2 = h_3 \times w_{32} + h_4 \times w_{42} + b_4 = 2 \times -1.0 + 0 \times 1.0 + 0.5 = -1.5$$

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

(b) (5 points) Please 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:

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

2. (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, 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.)

Part V: Layers (12 points)

1. (8 points) You are given the following kernel W of a convolutional layer:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

W

(a) (2 points) Apply the kernel W with a stride of $s = 2$ and a padding of $p = 1$ to the input X below. Fill in the missing values (red boxes) of the output $Z = X * W$.

3	1	6	8	0
2	3	2	7	-6
-8	0	6	-5	-8
1	2	5	-2	-4
8	7	-2	-1	7

X

1	3	1
0	2	-2
2	1	0

Z

Solution:

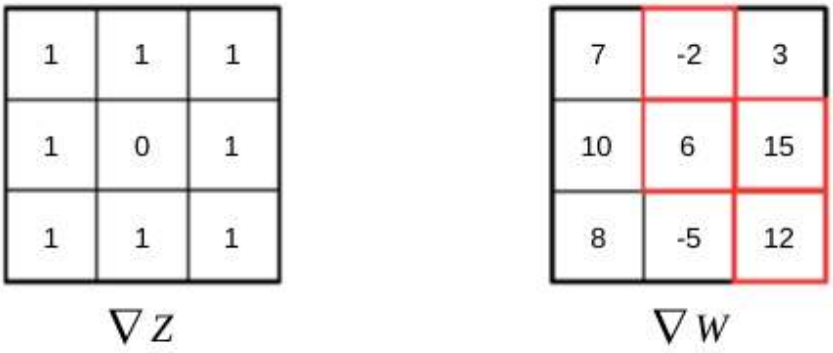
top middle: $(1 + 6 + 8 + 3 + 2 + 7) * \frac{1}{9} = 27/9 = 3$

center: $(3 + 2 + 7 + 0 + 6 + (-5) + 2 + 5 + (-2)) * \frac{1}{9} = 18/9 = 2$

middle right: $(7 + (-6) + (-5) + (-8) + (-2) + (-4)) * \frac{1}{9} = -18/9 = -2$

bottom left: $(1 + 2 + 8 + 7) * \frac{1}{9} = 18/9 = 2$

(b) (4 points) Given input X, and the gradients ∇Z of the output, calculate the gradients ∇W of the kernel. Fill in the missing values (red boxes) in the figure below.



Solution:

top middle: $2 + 0 * 2 + (-6) + 1 + 5 + (-4) = -2$

center: $3 + 6 + 0 + (-8) + 0 * 6 + (-8) + 8 + (-2) + 7 = 6$

middle right: $1 + 8 + 0 + 0 * (-5) + 7 + (-1) = 15$

bottom right: $3 + 7 + 2 + 0 * (-2) = 12$

Explanation: For a given position in ∇W, we sum up all values in X, which were multiplied with the corresponding position of W during the forward pass, except for the one belonging to the center piece, since the center of $\nabla Z = 0$.

(c) (2 points) Is this convolutional layer the same as a 3×3 average pooling layer? Do they compute the same function? Do they behave similarly during train and test?

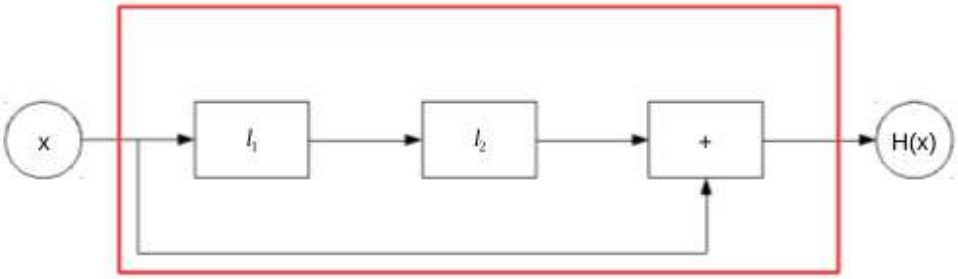
Solution:

Currently, they both compute the same function. (1 pt.)

However, the convolutional layer has trainable weights, so it can update it's kernel. Thus, the two layers are NOT the same. (1 pt.)

2. (4 points) As you learned in the lecture, the use of skip connections in ResNets has substantially improved the Deep Learning state-of-the-art by allowing us to train much deeper networks.

   (a) (2 points) Draw a residual block together with a skip connection below. The residual block should contain two layers, which you can represent by l_1 and l_2. For simplicity, you don't need to draw in ReLU-activations.



   (b) (1 point) For your above drawing, given the partial derivative of the residual block $R(x) = l_2(l_1(x))$ as $\frac{\partial R(x)}{\partial x} = r$, calculate $\frac{\partial H(x)}{\partial x}$.

Solution:

$$\frac{\partial H(x)}{\partial x} = \frac{\partial x + R(x)}{\partial x} = \frac{\partial x}{\partial x} + \frac{\partial R(x)}{\partial x} = 1 + r$$

   (c) (1 point) Why can we build deeper networks with residual blocks?

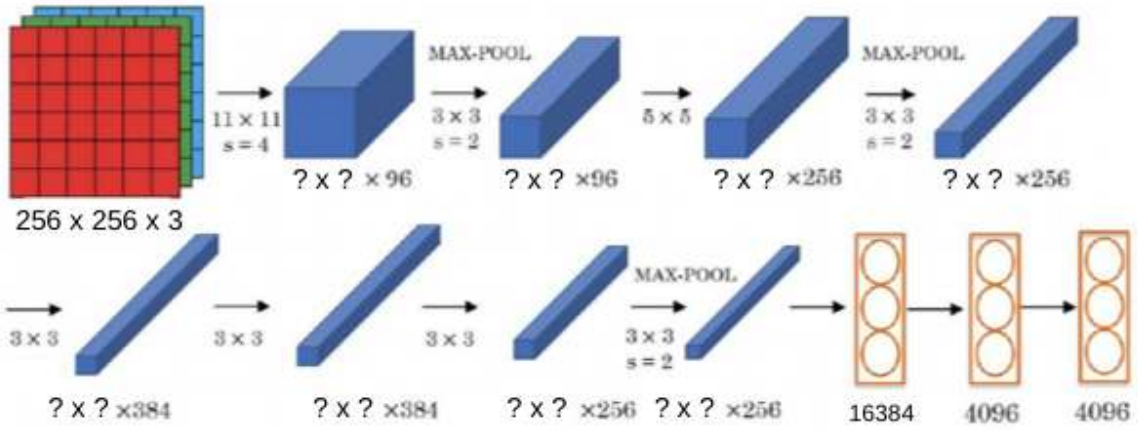
Solution:

Skip connections directly pass gradients through the network, and thereby alleviate the vanishing gradient problem.

("It's easier to learn the identity function" is also a valid argument.)

Part VI: CNN Architectures (12 points)

1. (12 points) After having obtained strong programming experience in I2DL, you want to demonstrate your acquired deep learning skills by competing in a renowned image classification contest. In this contest, you have to classify images of size 256×256 into one of 1000 classes. Since you learned that CNNs are great at such tasks, you decide to start out with the following CNN architecture:



Notes:

- The values directly below the arrows indicate the filter sizes f of the corresponding convolution and pooling operations.
- s stands for stride. If no stride is specified, a stride of $s = 1$ is used.
- All convolution and pooling layers use a padding of $p = \frac{f-1}{2}$, for corresponding filter size f.

(a) (1 point) Which classic network architecture is similar to the architecture above?

Solution:

AlexNet

(b) (1 point) In the figure above, the output layer is missing. Explain which type of layer you would use there, what it's dimensions would be, and which output and loss functions you would choose.

Solution:

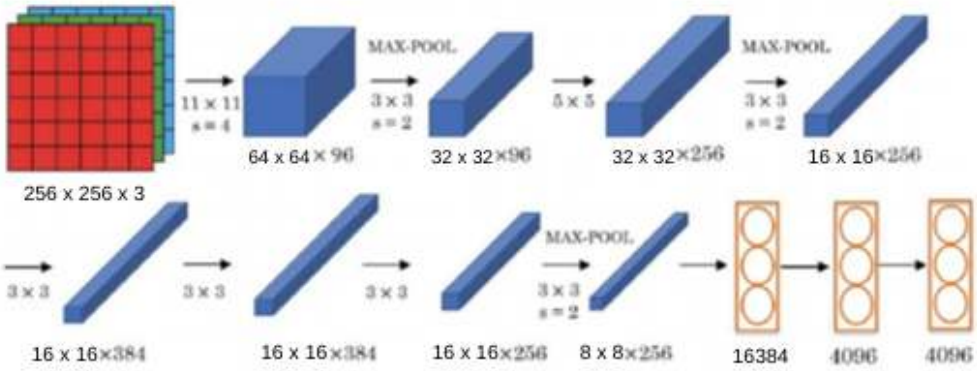
We use a fully-connected layer with 1000 neurons, softmax activation and crossentropy loss.

(c) (3 points) Explain how to calculate output sizes of convolutional layers. Then, apply your formula to find the heights and widths of all layers in the network, and fill in your results in the figure below.

Solution:

$out = \frac{in - f + 2*p}{s} + 1$

(1 pt. for formula, 2 pt. for values in figure)



(d) (3 points) Calculate the size of the combined receptive field of the first three layers. This corresponds to the area of pixels in the input image that each neuron after the third layer (before the second MAX-POOL) "sees".
Hint: Strides affect the total receptive field sizes of subsequent layers.

Solution:

Size of total receptive field after layer $k > 1$: $r_k = r_{k-1} + (\prod_{i=1}^{k-1} s_i) \cdot (f_k - 1)$

- layer 1: 11
- layer 2: $11 + (4 \cdot (3 - 1)) = 11 + 8 = 19$
- layer 3: $19 + (4 \cdot 2 \cdot (5 - 1)) = 19 + 32 = 51$

Alternative Solution ("from right to left"): $n_k = f_k + (r_{k+1} - 1) \cdot s_k$, with $(r_4 = 1)$

- 3rd to 2nd layer: 5x5 (5x5 conv, $r_3 = 5$)
- 2nd to 1st layer: 11x11 (3x3 conv; $r_2 = 3 + (5 - 1) \cdot 2$)
- 1st layer to input: 51x51 px (11x11 conv; $r_1 = 11 + (11 - 1) \cdot 4$)

(e) (1 point) Calculate the number of parameters in the first convolutional layer.

Solution:

$11 * 11 * 3 * 96 + 96 = 34944$

(f) (1 point) If we apply this network architecture to larger input images, would the total amount of parameters in the network change? Justify your answer.

Solution:

Yes, the total amount of parameters would increase, because our network is not fully-convolutional. As a result, the first fully-connected layer would have more input neurons and, thus, more parameters.

(g) (2 points) Unfortunately, the dataset of the competition is rather small. Online you found a much larger image classification dataset containing almost similar images, but the set of classes is different. Explain how you could make use of the larger dataset, and how each layer in your network would be affected in the process.

Solution:

We can use the larger dataset for transfer learning, by first pretraining the model on the large dataset, and then finetuning it to the competition dataset.
(1 pt. for transfer learning)

After pretraining, we freeze the convolutional and pooling layers, and replace and retrain the fully connected layers.
(1 pt. for replacing output layer; which other layers to freeze/train/retrain is merely a design choice, which depends on the specifics of the two datasets and cannot be specified precisely given only the little info we have.)

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