Exam ws20 21 solution

Introduction to Deep Learning (Technische Universität München)

Chair of Visual Computing & Artificial Intelligence
Department of Informatics
Technical University of Munich

Esolution
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:**	Sunday 21st February, 2021
Examiner:	Prof. Dr. Matthias Nießner	**Time:**	08:00 – 09:30

final grade 35.5

	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9	P 10	P 11
I											

Working instructions

- This exam consists of **22 pages** with a total of **11 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: None
- This is the blackened exam. Please fill out the boxes here.

Left room from _____ to _____ / Early submission at _____

Problem 1 Multiple Choice (16 credits)

- 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.

How to Check a Box Using Pen and Paper:

- 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, put a **cross** next to the filled box.

a) How will the Validation Loss behave, if you shuffle the validation set, compared to, if you don't shuffle it?

- ☐ 1) The validation loss will be higher if shuffled.
- ☐ 2) The validation loss will be lower if shuffled.
- ☒ 3) The validation loss will be the same if shuffled.
- ☐ 4) ~~It's not possible to predict how shuffling will affect the Validation Loss~~.

b) Which one of the following layers have the same train and test time behavior?

- ☐ 1) ~~Batch Normalization.~~

because in the drop out half the neurons are turned off in the training but in the test they;re not for batch normaliz the running mean and variance vs the mean and variance of each batch in the train

- ☒ 2) Linear Layer.
- ☐ 3) Dropout.
- ☒ 4) Sigmoid.

c) Which of the following layers does not have trainable parameters?

- ☐ 1) Parametric ReLU.
- ☒ 2) ~~Global Average Pooling~~.
- ☐ 3) Convolution.
- ☒ 4) ~~Dropout~~.

d) The following is true about the L1 and L2 parameter norm regularization techniques:

- ☒ 1) ~~They introduce additional residuals in the cost function.~~
- ☒ 2) ~~The L2 norm encourages a mean distribution over the weight values.~~
- ☐ 3) They improve training accuracy.
- ☐ 4) They aim to reduce the overall cost of the optimization.

e) Which of the following statements are true regarding second order derivative based optimization techniques

- ☒ 1) They converge to the minimum in lesser number of iterations when compared to first order techniques.
- ☐ 2) Each iteration step of second order technique are usually more faster than the first order technique.
- ☐ 3) RMS Prop is a second order optimization technique.
- ☒ 4) Each iteration step of second order technique are usually slower than the first order technique.

f) You start training your Neural Network but the train loss is almost completely flat and not decreasing. What could be the cause?

- [ ] 1) Class distribution is too even in the dataset.
- [x] 2) Bad initialization.
- [x] 3) Learning rate is too small.
- [ ] 4) Regularization strength is too small.

g) What can 1 × 1 convolutions be used for?

- [ ] 1) To perform feature selection.
- [x] 2) To reduce the number of channels before a costly operation.
- [x] 3) To make a network more complex when coupled with non-linearities.
- [x] 4) To replace fully-connected layers in order to make a network fully- convolutional.

h) What is true about Dropout?

it's a bagging (diff dataset same model) method not ensemble (different loss function and .. check notess)

- [ ] 1) When using dropout, our network can be seen as an ensemble of multiple independent smaller networks.
- [ ] 2) Dropout makes training generally faster.
- [x] 3) Dropout acts as regularization.
- [ ] 4) Dropout can also be applied at test time, but the output has to be scaled by the number of training samples.

MCQ 6/16

Problem 2 Case Study: Road Signs (7 credits)

Recently one of your friends wants to get a driving license in Germany, and since it is still in lock-down period, he would like to start to study the theoretical aspect first. He tries to memorize the meaning of each road signs, and after a while he feels tried and wants to have some fun with it, i.e., develop a deep learning model to classify the road signs and compete against it with his own memory to see whether he or the model can reach higher accuracy. So he approaches you for your deep learning expertise. He gets a hand-labelled dataset which has around 5,000 RGB images that contains both road signs taken at daytime and at night as well as 20 classes of different road signs.

80-20-20

0 ☑
1 ☐

a) Before building up the model, you need to split the dataset. How would you do this?

Train + Validation + Test set with 60%, 20%, 20% or 80%, 10%, 10%. (0.5 point for resonable percentages)
Important: answer should mention that the split needs to cover both daytime and nighttime data. (0.5 point)

I disagree I think there is a problem with the model's parameters like the training rate and possibly also the network architecture. I would first increase the network architecture to try and fit these images better then try to play with the learning rate

0 ☐
1 ☐
2 ☑

b) You first trained a very simple model only on 200 samples, the train loss converges to a relative big value. Your friend thinks that if you train on more data and train longer, this issue can be solved. Do you agree? If not, what would you do?

No, I don't agree. (1 point)
The simple model has high training error which means the capacity of the model is too small such that this is a bias problem. Training on larger dataset will not help with this situation.
In this case, it would be better by adding more learnable parameters to increase the capacity of the model.
(1 point for increase model capacity, if only mention other possible solutions then 0.5 point)

Cross validation (brute force), and random search. We could instead of trying all possible combinations just do a couple of iterations each time with a different combination of hyperparameters

0 ☐
1 ☑
2 ☐

c) After you solved the high training loss problem, you are training your model from scratch and getting a reasonable result. However, you would like to search for better hyperparameters. Name two hyperparameter search strategies and shortly discuss which procedure you would use to combine those.

Grid Search, Random Search. (0.5 point for each method)
Random search first to find reasonable region, then use grid search to refine. (1 point, if the order is the other way around then get 0.5 point)

Using mirroring we could augment the data by flipping the image so objects on the right would be on the left and we could also do it horizontally so we flip top and bottom

blurring, salt and pepper, rotation

d) Even after all this optimization, you accept that the dataset itself is not big enough. Here are some more data examples. To make your model generalize better, you decide to perform data augmentation.

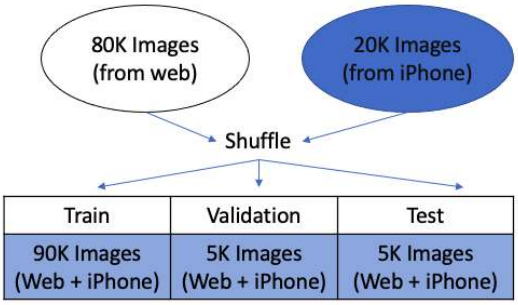
0
1
2

1. Explain how you could augment the data using mirroring (1p).
2. Give 3 examples of further data augmentation techniques that you can use in addition (1p).

It's important to note that some classes are mirror-inverted (e.g. left- and right-only-signs). Therefore, we must not only mirror the images X, but also need to adjust the label y. (1.0 point for this answer, it is essential to mention that mirroring effect of symmetric sign.)
Other types of regularization are: adding noise, small random rotations, translations, etc. (1.0 point to mention all 3 correct methods, otherwise no points.)

Problem 3 Data Analysis (8 credits)

You want to build an iPhone app that can classify dog images taken from the iPhone-camera. To collect data, you walk through Munich and take pictures of dogs (as well as some non-dog-objects for the negative class). This way you can collect 20.000 samples. However, according to your experience you need much more images to train a good model. You get the idea to simply write a Python script to download 80.000 images with a similar dog/non-dog ratio from the web. You then combine both datasets, randomly shuffle the data, and split the resultig dataset into train (90%), validation (5%) and test (5%) splits.



Because it could easily create a bias in the data where all the validation or test data are from the downloaded images or the collected samples or most of the training data is from the downloaded images.

0 ☑
1 ☐

a) Why is it a bad idea, especially in terms of evaluation of your model, to first mix all the data and then create the training split from the mixed data?

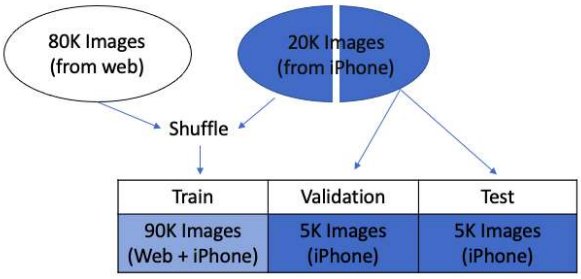
The data comes from 2 different distributions (0.5pt). By the above approach, **only $\frac{1}{5}$ of the images in the validation (and test) set come from the target distribution** (iPhone images). Therefore, we won't be able to assess how well the model is actually performing in the task we're interested in (0.5pt).

Note: no points for shuffling-related answers or different ratio between web and iPhone images among training/val/test set.

0 ☐
1 ☑
2 ☐

b) You now decide to combine the datasets differently: You build a validation and test set that only consist of images from the target-distribution (5,000 images both) and mix the remaining 10,000 iPhone images with the internet-images to form the training data. Why will this approach lead to problems in the model evaluation as well?

Because the validation set is not a good representative of the data we trained on



Our validation set now has a good assessment about the performance of our model on the target data. However: Training distribution is different from the validation/test distribution. (1pt).

Therefore, if the validation error is high, we can't tell if the model performs bad because a) the model is overfitting, or b) there is a data mismatch error (1pt).

Note: 90-5-5 datasplit is not really problematic here, as we just have a simple binary classification problem. This answer didn't get any points.

c) Explain a better way to split the data by filling out the table analogous to the question b).
Hint: Think about using a 4th set to take into account the bridge between both distributions. Fill in their name, the distribution(s) they come from, and the number of samples.

0 1 2

Train	Bridge / Train-Dev	Validation	Test
85K Images (Web + Target)	5K Images (Web + Target)	5K Images (Target data only)	5K Images (Target data only)

The "bridge" (also: "train-dev") set is used, to evaluate whether the model is generalizing well to unseen samples or overfitting to the training data. Therefore, the data in this set comes from the same distribution as the data in the training-set. The validation- and test- set come from the target-distribution. This way we can evaluate, how well the model fits the target-distribution. (2 pt if all correct.)

"train (mixed) | val (web) | val (phone) | test (phone)": also 2 pt, even though val (web) should better be from same distr. as training to have more accurate information about overfitting.

"web | phone | phone | phone" IF mentioned that they first do pre-training on web-images and then transfer learning: also received 2 points.

"train (web) | val (web) | val (phone) | test (phone)": 1.5 pt - because not using any phone images for training.

"train (mixed) | val (mixed) | test (web) | test (phone)": 1.5 pt - not able to evaluate phone performance.

"mixed" in every set gave 0 points, as we want to train a classifier of phone images, web-data should not be in val/test.

d) You now train the network. You notice that the variance of the model is high. Fill in the table below, what sets will have a high and a low loss. Leave boxes empty for sets that aren't used during the whole training and optimization process.

0 1

Train: low
bridge: low
validation: high
test: high

Train	Bridge / Train-Dev	Validation	Test
Low	High	High	(Not used)

As the model doesn't generalize to unseen samples, the loss will be high for all sets except the training data (to which we are overfitting) (1pt)
Only get point if train-set has low loss and all non-train sets have high loss.

e) You now improve your model and solve the problem of high variance. However, you notice that there is a data-mismatch problem. How will the losses in this case look like? Leave boxes empty for sets that aren't used during the whole training and optimization process. (Fill in "high or "low")

0 1

train:low, bridge: low, val: low, test: high

Train	Bridge / Train-Dev	Validation	Test
Low	Low	High	(Not used)

As we could reduce the variance error (e.g. by regularization), the model performs well on unseen samples that come from the same distribution, as the training data. However, due to the data mismatch problem, it performs poorly on data from different distributions.

The grading of this subproblem depends on the student's solution to subproblem c). Only get point if for at least 3 datasets a loss was provided and datasets that have a different distribution than training set must have high loss.

Note: data mismatch *problem* implies that the loss for the other distribution from training is higher. If it would be lower (i.e., we would perform better on the target distribution in val/test than on training distribution), this would not be a problem but instead show that training with the web-data works great.

0 ☐
1 ☑

f) How could you solve the data mismatch problem? Collect more data for the mismatched objects and train the network on them

We need to incorporate characteristics of the target distribution into the training set.

- collect more data from target distribution
- analyze the error to find out what are the difference in the datasets
- synthesize data, etc.

Problem 4 Optimization (9 credits)

a) Write down the formula for the binary cross-entropy loss. Make sure to define all the variables in your equation.

0 1

$$\mathcal{L}(\hat{y}_i, y_i) = -(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)) \quad \text{or}$$

$$BCE = -\frac{1}{N} \sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)$$

Where y_i is the binary label and $\hat{y}_i$ the prediction/score/probability of class i.
Formula: (0.5 points) (Also get the points without the minus at first place, since it's also in our slide.)
Explanation of variables: (0.5 points). (No point for: $\hat{y}_i$ is the predicted label.)

because of the compute and memory capacity. we do batch gradient descent

b) Why is computing the full-batch gradient often impractical during gradient descent? What do you do instead?

0 1

Impractical: update only after loop over entire data set / extremely expensive to compute / too slow / GPU limitation (0.5 points) (No point for: Only say the dataset size is too large.)
Use instead: Update over mini-batch / SGD (0.5 points)

c) What is a saddle point? What is the advantage/disadvantage of Stochastic Gradient Descent (SGD) in dealing with saddle points?

A saddle point is a point where it's neither a local min or max. SGD is able to escape it

0 1 2

Saddle point - The gradient is zero (0.5 points), but it is neither a local minima nor a local maxima. (or:the gradient is zero and the function has a local maximum in one direction, but a local minimum in another direction). (0.5 points) SGD has noisier updates and can help escape from a saddle point. (1 point)
(- 0.5 point for : only say SGD helps escape from saddle point but without giving a reason.)

d) Given a learning rate of α, gradient ∇, and velocity ν, write down the formula(s) for momentum update in SGD.

0 1 2

$$\nu^{k+1} = \beta \cdot \nu^k + \nabla_\Theta L(\Theta^k) \qquad \Theta^{k+1} = \Theta^k - \alpha \cdot \nu^{k+1}$$

or

$$\boldsymbol{v}^{k+1} = \beta \cdot \boldsymbol{v}^k - \alpha \cdot \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}^k\right) \qquad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \boldsymbol{v}^{k+1}$$

(1 point for each formula)
(- 0.5 points: only write ∇ without the loss term, index wrong/missing)

faster convergence. more computationally heavy

e) What would be an advantage of a second order optimization method such as the Newton method besides taking less iteration steps? Why is it not commonly used in the context of neural networks?

0 1 2

Avoid choosing the learning rate(1P) Computing the inverse Hessian is expensive as it depends on the number of parameters (millions) / estimating the Hessian in a stochastic setting is difficult. (1P)

0 ☑
1 ☐

f) What is the difference between Newton and quasi-Newton methods, e.g., Broyden–Fletcher–Goldfarb–Shanno (BFGS)?

Quasi-Newton Methods do not compute the Hessian explicitely but find and approximation in $\mathcal{O}(n^2)$ or $\mathcal{O}(n)$.

Problem 5 Batch Normalization (6 credits)

For an input $z = (z^{(1)}, z^{(2)}, ..., z^{(b)}) \in \mathbb{R}^b$ with $z^{(i)} \in \mathbb{R}^n$ (b batch size, n feature dimension) the output of a batch normalization layer $y \in \mathbb{R}^{b \times n}$ is given by

$$z_{\text{norm}} = \frac{z - \mu}{\sigma}$$

$$y = \gamma \cdot z_{\text{norm}} + \beta$$

where

$$\mu = \frac{1}{b} \sum_{i=1}^{b} z^{(i)}$$

$$\sigma = \sqrt{\frac{1}{b} \sum_{i=1}^{b} \left(z^{(i)} - \mu\right)^2}$$

Here, μ is the mean and σ is the standard deviation of z. All additions and multiplications are elementwise and the trainable parameters $\gamma, \beta \in \mathbb{R}^n$ are replicated along the sample dimension.

This way we allow the neural network to change its mind if we do a normalization that is not optimal the network can update these parameters and eliminate the normalization

a) Explain shortly the purpose of the learnable parameters in the batch normalization layer.

0
1

They allow the network to undo/revert the normalization. (1pt)

b) Now you want to use a neural network with batch normalization layers to build a classifier that can distinguish a cat from a dog. In one of the batch normalization layers, you forward propagate a batch of b examples in your network. The input $z = [z_{(1)}^T, z_{(2)}^T, ..., z_{(b)}^T]^T$ of the batch normalization layer has shape ($b = 4, n_h = 3$), where n_h represents the number of neurons in the pre-batchnorm layer:

0
1
2

$$z = \begin{bmatrix} 12 & 0 & -5 \\ 14 & 10 & 5 \\ 14 & 10 & 5 \\ 12 & 0 & -5 \end{bmatrix}$$

Calculate μ, σ and z_{norm}.

μ ~~correct:~~ (0.5 points)
σ correct: (1 points)
z_{norm} correct: (2 points)

$$\mu = \begin{bmatrix} 13 \\ 5 \\ 0 \end{bmatrix} \qquad \sigma = \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix} \qquad z_{\text{norm}} = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

0 ☑
1 ☐

c) Suppose the learnt parameters for scaling $\gamma = (1, 1, 1)$ and learnt parameters for offsetting $\beta = (0, -10, 10)$. What is the final result y after applying the learnable parameters?

$$y = \begin{bmatrix} -1 & -11 & 9 \\ 1 & -9 & 11 \\ 1 & -9 & 11 \\ -1 & -11 & 9 \end{bmatrix}$$ (0.5 points for each correct column)

0 ☑
1 ☐

d) The above given definition of a batch normalization is applied in combination with fully-connected layers. Using a batch normalization layer in combination with a convolutional layer implies some changes to the original definition. Instead of an input dimension of $b \times n$, we have an input dimesnion of $N \times C \times H \times W$ where N represents the number of samples, C the number of channels and $H \times W$ the size of the input image.

What is the name of the altered layer? Explain shortly the difference to the original defined batch normalization procedure.

1x1 convolutional layer. This will allow us to shrink the dimensions of the image before further processing

1. Spatial Batch Normalization (0.5 point), BatchNorm2d could be accepted as well
2. Instead of calculating the mean and variance across all input samples (0.5pt), the mean and variance will be calculated per channel of the input (0.5pt).

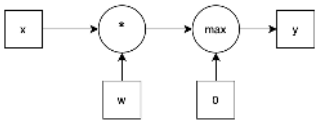
Problem 6 Skip Connections and Computation Graph (6 credits)

Suppose you and your team want to train a very deep network, but the optimization gets more difficult as you go deeper. Thus, you decided to use skip-connections in your model. However, your team members are not convinced that skip-connections help with the optimization and they think it is a waste of time. You want to show them how skip-connections help with the gradient flow.

a) First, let's start with a simple network layer. Assume x, w and y are scalars and $*$ is the simple multiplication. Draw the computation graph of the function $y = max(w * x, 0)$ and write down the derivative **dy** := $\frac{dy}{dx}$.

0 1 2 ☑

Computational Graph correct: 1 point
derivative correct: 1 point

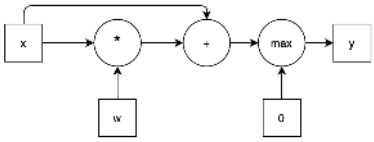


$dy/dx = 1(y > 0) * w$

b) Now, let's explore a simple skip-connection. Draw the computation graph of the operation $y = max(w * x + x, 0)$ and write down the the derivative **dy** := $\frac{dy}{dx}$.

0 1 2 ☑

Computational Graph correct: 1 point
Derivative correct: 1 point



$dy/dx = 1(y > 0) * (w + 1)$

c) Evaluate both derivatives at $w = 0.001$, assuming $y > 0$. What are the values you get? Which problem did skip-connections solve?

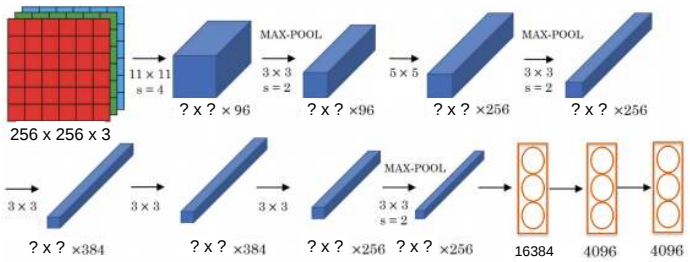
vanishing gradient problems

0 ☑ 1 2

a) dy/dx = 0.001 (0.5 point)
b) dy/dx = 1.001 (0.5 point)
The skip connection solved the vanishing gradient problem by allowing a larger gradient flow (1 point).

Problem 7 Convolutional Neural Networks (7.5 credits)

Your friend needs your expertise in classifying a set of RGB Images of size 256 x 256 pixels into a total of 1000 classes. Can you help him out?
Since you learned that CNNs are great to tackle these sort of 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 convolutional and pooling layers use a padding of $p = \frac{f-1}{2}$ for a corresponding filter size of f.
- For each convolutional filter, we include a bias.

0 ☐
1 ☑

a) In the figure above, the output layers for classification are missing. Explain:

1. Which type of last layer you would use there and how would you choose it's dimension? Fully connected layer. dimensions based on the no. of classes
2. Which output function and loss function would you choose for this task? Sigmoid. Cross Entropy loss

1. Fully-connected layer (0.5 points) with 1000 neurons (0.5 points)
2. Softmax function combined with Cross-Entropy Loss (0.5 point)

0 ☐
1 ☑

b) Calculate the output dimension of the image after passing through the first convolutional layer. Make sure to include your calculations in the solution.

Formula to calculate the output dimension: $\dim_{out} = \frac{\dim_{in} - f + 2 \cdot p}{s} + 1$ Output Dimension $(\frac{256-11}{4} + 1) = 64 \times 64 \times 96$ (1 Point)

0 ☐
1 ☑

c) Calculate the number of learnable parameters in the first convolutional layer. Make sure to include your calculations in the solution. Un-multiplied answers are accepted (e.g., $3 * 3 * 16$)

Number of Parameters $(11 * 11 * 3 + 1) * 96 = 34944$ parameters (1 Point)

d) Calculate the size of the combined receptive field of the first two and of the first three layers. This corresponds resp. to the area of pixels in the input image that each neuron

0 1 2

1. after the second layer (right after the first MAX-POOL layer)
2. after the third layer (before the second MAX-POOL layer)

"sees".
Hint: Strides affect the total receptive field sizes of subsequent layers.

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

- layer 1: $r_1 = 11$ (11×11 filter)
- layer 2: $r_2 = 11 + (4 \cdot (3 - 1)) = 11 + 8 = 19$ (0.5p answer, 0.5p calculation)
- layer 3: $r_3 = 19 + (4 \cdot 2 \cdot (5 - 1)) = 19 + 8 \cdot 4 = 19 + 32 = 51$ (0.5p answer, 0.5p calculation)

Alternative solution for r_3 (from right to left, needs separate calculation for r_2!): $r_{i-j} = f_j + (r_{i-(j+1)} - 1) * s_j$)

- 3rd to 2nd layer: $r_{3-2} = 5$ (5×5 filter)
- 3rd to 1st layer: $r_{3-1} = 3 + (5 - 1) \cdot 2 = 11$ (3×3 filter, stride 2)
- 3rd layer to input: $r_3 = r_{3-0} = 11 + (11 - 1) \cdot 4 = 51$ (11×11 filter, stride 4)

After a series of convolutional layers, the architecture has 3 fully connected layers that help process the spatial information obtained by the convolutions and prepare it for the output layer. Our friend just found an article about image segmentation and gets really excited about this. He decides to forget about the classification task and wants to work on image segmentation.

We don't need to reject our architecture: We first convert our architecture to a fully-convolutional network by ditching the fully connected layers. Next, we want to produce an output similar to the input size from this bottleneck where we would like to mirror the current architecture.

e) How would you replace the convolutional layers in the mirrored architecture to increase the the image size from our bottleneck onwards?

0 1

Accepted answers: upsampling/unpooling + convolution, transposed convolution.

Grading notes: upsampling alone (without conv) is 0.5p, "inverse convolution", "transformed convolution", and all other misspellings are 0.5p.

f) The new architecture is able to process an image of any input size. How about the original architecture that we started with, was it able to handle images of arbitrary sizes as input, too? Give an explanation for your answer.

0 1

No, the original architecture was not able to handle images of arbitrary size (0.5p).

Explanation: Fully-connected layers require fixed size and cannot handle variable output size of convolutional layers (0.5p).

Problem 8 General Training (8 credits)

No. We shouldn't do that because we need to normalize the data after it has been partially processed by the neural network to modify its variance and mean again and maintain the gradient

0 ☑
1 ☐

a) When we normalize the data as a preprocessing step is the test set also normalized? If no, explain why you shouldn't do it. If yes, describe how do we normalize it.

Yes (0.5p), the test set is normalized with statistics calculated from the training set (0.5p)

Grading note: only explicit mentioning of training statistics gives full points, e.g., "yes, the test set is normalized similarly as train" is 0.5p.

The forward pass calculates the value of the compute graph from left to right at each step then passes this value to the step after it. The arguments for the forward step are the inputs from the previous layer and the weights.

The backward pass computes the partial gradient up to each step and passes that the layer before it (from right to left) until we can compute the gradient with respect to each weight. Its arguments are whatever the gradient requires for computation and the dout from the next layer

0 ☐
1 ☐
2 ☑

b) 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.

Forward Function: takes output from previous layer, performs operation, returns result (0.5 pt.), caches values needed for gradient computation during backprop (0.5 pt.)
Backward Function: takes the upstream gradient (0.5 pt.), returns partial derivatives or gradients (0.5 pt.)

If the answer has been written with respect to forward and backward pass through the network, instead of forward() and backward() functions of a layer, partial points have been provided if the answer contains the above key points.

0 ☐
1 ☑

c) You are training a neural network with 15 fully-connected layers using a *tanh* nonlinearity.

it will lead to a dying gradient because the tang saturates for very large numbers to 1 and gradient becomes 0

1. Explain the behavior of the gradient of the non-linearity with respect to very large positive inputs.

Relu because it always has a positive gradient for positive numbers and it doesnt saturate

2. Suggest another non-linear activation function that does not have this behaviour.

1. Because the tanh is almost flat for very large positive values, its gradient will be almost 0. (or) vanishing gradients (or) gradients die off (0.5pt)
2. ReLu and it's variants (0.5pt)

No points only if saturation of tanh is mentioned without stating the gradients becoming zero.

we redo the train, validation splits k times and calculate the validation each time and the final validation is the average of all scores

0 ☐
1 ☑
2 ☐

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

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)

unbalanced data would cause a bias in the model where it will tend to classify images more as cats than dogs. Two solutions are data augmentation for the dog class or using less cat images or collecting more dog data

0 ☐
1 ☐
2 ☑

e) You have 4000 cat and 100 dog images and want to train a neural network on these images to do binary classification. What problem(s) do you foresee with this dataset distribution? Name two possible solutions.

Network prefers cats as they are more likely or imbalance between classes (1pt)
leave out pics/reweight dataloader/reweight loss function/collect more dog images/data augmentation on dog images (0.5pt/sol)
No points for: dropout, regularization, batch norm, transfer learning, "get more data" or "data augmentation" without specifying which

Problem 9 Unsorted Short Questions (8.5 credits)

tanh: 1/1024
relu: 2/1024

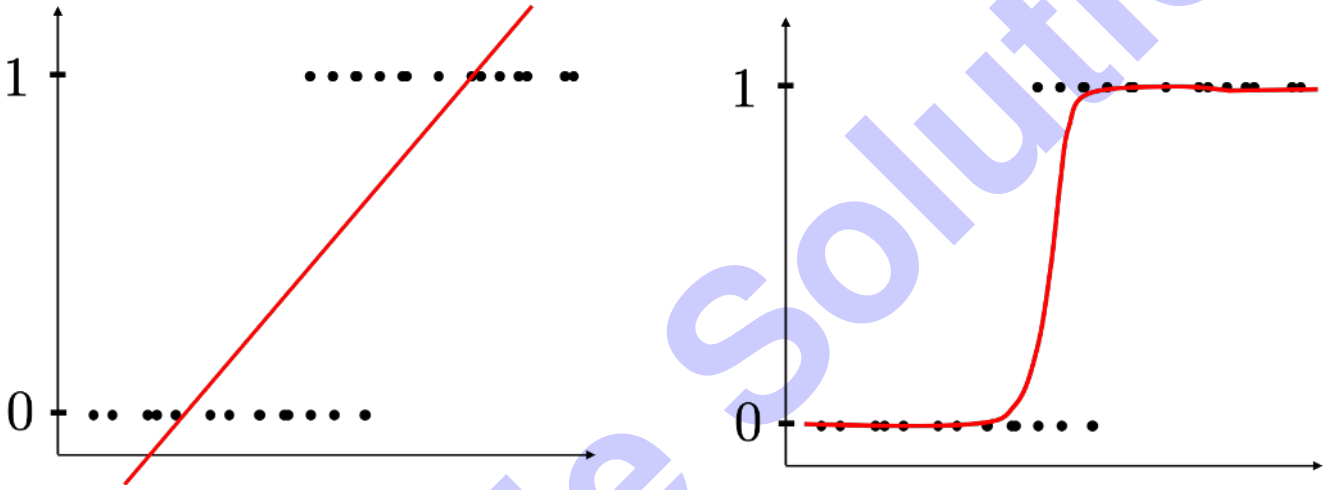
a) Given an input image with 50×50 pixels followed by a fully connected layer with 1024 neurons and a tanh activation. Compute the variance normalization coefficient of the Xavier initialization for the fully connected layer. How should the coefficient change if the tanh is replaced by a ReLU?

☑ 0
☐ 1
☐ 2

Var = 1/2500 (1 point)
ReLU: multiplied by 2, i.e. *Var* = 1/1250 (1 point if mentioning the double relationship between 2 variance correctly; 1 point if all results are correct.)

b) Consider the following data points. Sketch a linear (0.5p) and logistic (0.5p) regression line into the figures. Which model is more suitable for this task (0.5p)?

☐ 0
☑ 1



Logistic regression (0.5 pt) For each drawing 0.5 pt. The logistic regression line needs to be a function and must not have multiple y values for one x value. The linear regression line must not have any bends or non-linear points.

0 ☑
1 ☐
2 ☐
3 ☐

c) Design a 2-layer neural network with 3 neurons to implement a ***XOR*** function. For each neuron,

$$y = f(wx + b), f(x) = \begin{cases} 0, x \leq 0 \\ 1, x > 0 \end{cases}$$

specify the proper weight and bias, where both weight and bias value must be chosen from $[-1, 0, 1]$.
Hint: ***XOR =*** $(A \cap \bar{B}) \cup (\bar{A} \cap B)$)

W_{11}		b_{11}	
W_{12}		b_{12}	
W_{21}		b_{21}	

Layer 1:

1. $A \cap \bar{B} : AND(A, \bar{B}) = A - B - 0 \rightarrow S_1, \therefore W_{11} = [1, -1], b_{11} = 0$
2. $\bar{A} \cap B : AND(\bar{A}, B) = -A + B - 0 \rightarrow S_2, \therefore W_{12} = [-1, 1], b_{12} = 0$

Layer 2:

1. $S_1 \cup S_2 : OR(S_1, S_2) = S_1 + S_2 + 0 \rightarrow \therefore W_{21} = [1, 1], b_{21} = 0$

0.5pt if you only have $b_{21} = 0$ correct and 0.5pt if you only have $b_{11} = 0$ and $b_{12} = 0$ correct.

0 ☑
1 ☐
2 ☐

d) Suppose we have a grayscale image represented as an array, where larger values denote lighter pixels. What is the effect when we convolve it with the following kernel?

it will detect if there are points in the image where there is more light in the center and then it starts to get darker as we go further from it

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & -4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (1)$$

If the input image is encoded as numbers between [0 and 255] or [0 and 1], the output will be completely black. 2pt (only 1pt for "darker") Or: If the input image is encoded as positive and negative numbers, such as numbers in [-1,1], the kernel will invert the image (1pt), blur it (1pt), and increase the inverted values. (1pt) NOT edge detection. NOT sharpening. NOT cross detection.

Problem 10 Recurrent Neural Networks and Backpropagation (8 credits)

Recurrent neural networks, also known as RNNs, are a class of neural networks that allow an arbitrary number of inputs and, thus, are often used for sequences of data, e.g., in the fields of natural language processing and speech recognition.

a) Mathematically explain the reason for exploding and vanishing gradients when using a classic RNN, i.e., $A_t = \theta_c A_{t-1} + \theta_x x_t$, where both θ_c and θ_x are orthogonal. (2 points)

☑ 0 ☐ 1 ☐ 2

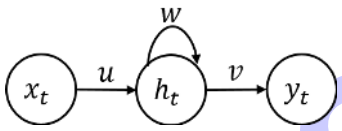
1. Show backpropagation explicitly to calculate gradients (1 point):

$$\frac{\delta h^{(t)}}{\delta h^{(1)}} = \frac{\delta h^{(t)}}{\delta h^{(t-1)}} \cdots \frac{\delta h^{(2)}}{\delta h^{(1)}}$$

2. After eigen-decomposition of θ_c, the **largest** eigenvalue of $\theta_c > 1$ means explosion(0.5 point) and < 1 means vanishing (0.5 point).
   P.s. if the answer discussed eigenvalues of A_t instead of θ_c: 0 point
   if the answer discussed two cases of eigenvalues but didn't show explicitly which matrix should be decomposed: 0.5 point

b) Now consider the following RNN

☑ 0 ☐ 1 ☐ 2 ☐ 3



which uses the one-dimensional ReLU-RNN cell

$$h_t = ReLU(u * h_{t-1} + w * x_t).$$

Compute the forward propagation y_2, h_2 and the gradient $\mathbf{dy}_2 := \frac{\delta y_2}{\delta u}$ where

$$h_0 = 3,\ w = 2,\ v = -1,\ u = 3,\ x_1 = 1,\ x_2 = 2.$$

Since there is a mismatch between the graph and the given equation, answers based either on the graph or on the equation are accepted. Solution based on the graph:

$$h_1 = ReLU(u * x_1 + w * h_0) = 9$$
$$h_2 = ReLU(u * x_2 + w * h_1) = 24(1p)$$
$$y_2 = v * h_2 = -24(1p)$$
$$\frac{\delta y_2}{\delta u} = v * w * x_1 + v * x_2 = -4(1p)$$

Solution based on the equation:

$$h_1 = ReLU(u * h_0 + w * x_1) = 11$$
$$h_2 = ReLU(u * h_1 + w * x_2) = 37(1p)$$
$$y_2 = v * h_2 = -37(1p)$$
$$\frac{\delta y_2}{\delta u} = v * (h_1 + u * h_0) = -20(1p)$$

Each equation and final result counts 0.5 points.

0 ☐
1 ☐
2 ☑

c) To circumvent the vanishing gradient problem, the Long-Short Term Memory (LSTM) unit was proposed. It is defined as

$$
\begin{aligned}
g_1 &= \sigma\left(W_1 \cdot x_t + U_1 \cdot h_{t-1}\right), \\
g_2 &= \sigma\left(W_2 \cdot x_t + U_2 \cdot h_{t-1}\right), \\
g_3 &= \sigma\left(W_3 \cdot x_t + U_3 \cdot h_{t-1}\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot x_t + u_c \cdot h_{t-1}\right), \\
c_t &= g_2 \circ c_{t-1} + g_3 \circ \tilde{c}_t, \\
h_t &= g_1 \circ c_t,
\end{aligned}
$$

g2 - forget
g3- update
g1 - output

where g_1, g_2, and g_3 are the gates of the LSTM cell.
1) Assign these gates correctly to the **forget** f, **update** u, and **output** o gates. (1p)
2) What does the value c_t represent in a LSTM? (1p) cell state

g_1 = output gate
g_2 = forget gate
g_3 = update gate/input gate
(1 pt)
c_t: cell state
(1 pt)

0 ☐
1 ☑

d) Why does the LSTM unit solve the vanishing gradient problem that is present in the default definition of an RNN cell?
because it provides a high way for the previous cell state that doesn't have to go through the activation functionss. It also allows not to take previous cell state into consideration

gradient highway through the cell state (0.5pt) and gate system to remove or add information to the cell state (0.5pt)
(another alternative: from the perspectives of weights and activation functions as in the slides)

Problem 11 Advanced Deep Learning (6 credits)

to update the optimal next action to take at each state. They are needed to be able to choose an action at each given state

a) Where do we use neural networks in Q-learning (1p) and why are they needed (1p)?

0 ☑ 1 ☐ 2

Function approximator for Q values: $Q^*(s, a) = Q(s, a; \theta)$ if θ are the network parameters. (1 point to mention that it is used to estimate Q values.) For even semi big problems, calculating all state action pairs is not tractable, but we can approximate it using neural networks. (1 point to mention that it is not tractable to build such huge look-up table.)

b) State the Markov assumption in reinforcement learning both in words and as a formula for probability states s_t at time t.

current state only depends on the previous state C_t = C_t-1 + x_t

0 ☐ 1 ☑ 2

Multiple formulations possible including (1p):

$$\mathbb{P}(s_t|s_{t-1}) = \mathbb{P}(s_t|s_1, \dots, s_{t-1})$$

or

$$\mathbb{P}(s_t|s_{t-1}) = \mathbb{P}(s_t|s_{t-1}, \dots, s_1)$$

or

$$\mathbb{P}(s_{t+1}|s_t) = \mathbb{P}(s_{t+1}|s_1, \dots, s_t)$$

and: The state at time t only depends on the previous state. / The next future state only depends on the present state. (1p) - Not mentioning the "state" subtracts 0.5 points unless the formula is also present.

Complexity, ??

c) What are the two key challenges when trying to define a neural network on graphs?

0 ☑ 1 ☐ 2

Variable sized inputs or variable number of nodes and edges (1P)
permutation invariance(1P)

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