Chair of 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:**	Wednesday 19th February, 2020
Examiner:	Prof. Dr. Matthias Nießner	**Time:**	13:30 – 15:00

	P 1	P 2	P 3	P 4	P 5	P 6	P 7
I							

Working instructions

- This exam consists of **20 pages** with a total of **7 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 89 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources: **none**
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.
- If you need additional space for a question, use the additional pages in the back and properly note that you are using additional space in the question's solution box.

Left room from _____ to _____ / Early submission at _____

Problem 1 Multiple Choice (18 credits)

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 put an additional **cross** besides the box.

a) You train a neural network and the loss diverges. What are reasonable things to do?

- ☒ Try a different optimizer.
- ☐ Add dropout.
- ☒ Decrease the learning rate.
- ☐ Increase the number of parameters.

b) Use a neuron to regress the **AND function**. The activation function of the neuron is $f(x) = 0$ if $x < 0$ else 1. What is the weight and bias of the neuron?

x_1	x_2	x_1 AND x_2
1.0	1.0	1.0
1.0	0.0	0.0
0.0	0.0	0.0
0.0	1.0	0.0

- ☐ Bias: 1.5, w_1: 2.0, w_2: 2.0
- ☐ Bias: -1.0, w_1: 1.5, w_2: 1.5
- ☒ Bias: -1.5, w_1: 1.0, w_2: 1.0
- ☒ Bias: -2.0, w_1: 1.4, w_2: 1.2

c) You want to train an autoencoder to overfit a single image with a fixed learning rate. Setting numerical precision aside, which loss function **is able to** reach **zero** loss after training with gradient descent.

- ☒ L2
- ☐ L1

d) Regularization:

- ☒ Is a technique that aims to reduce the generalization gap.
- ☐ Dropout, the use of ReLU activation functions, and early stopping can all be considered regularization techniques.
- ☒ Weight decay (L^2) is commonly applied in neural networks to spread the decision power among as many neurons as possible.
- ☐ Is a technique that aims to reduce your validation error and increases your training accuracy.

e) Which statements are correct for a Rectified Linear Unit (ReLU) applied in a CNN?

- [ ] Despite a small learning rate, without saturation the gradients are very likely to explode.
- [x] A large negative bias in the previous layer can cause the ReLU to always output zero.
- [x] Large and consistent gradients allow for a fast network convergence.
- [ ] Max pooling must always be applied after the ReLU.

f) You want to use a convolutional layer to decrease your RGB image size from 2545x254 to 127x127. What parameter triplets achieve this?

- [ ] Kernel size 3, stride 2, padding 1
- [x] Kernel size 6, stride 2, padding 2
- [x] Kernel size 2, stride 2, padding 0
- [ ] Kernel size 5, stride 2, padding 2

g) Your train and val loss converge to about the same value. What would you do to increase the performance of your model?

- [x] Add more input features.
- [x] Increase the capacity of your model.
- [ ] Add more training data.
- [ ] Add regularization.

h) An autoencoder

- [ ] has no loss function.
- [x] has a bottle neck layer.
- [ ] is often used for fine-tuning pre-trained models.
- [ ] requires class labels for training.

i) What is the correct order of operations for an optimization with gradient descent?

(a) Update the network weights to minimize the loss.

(b) Calculate the difference between the predicted and target value.

(c) Iteratively repeat the procedure until convergence.

(d) Compute a forward pass.

(e) Initialize the neural network weights.

- [ ] bcdea
- [ ] ebadc
- [ ] eadbc
- [x] edbac

Problem 2 Short Questions (22 credits)

0 1 2

a) Kaiming initialization corresponds to Xavier initialization with the variance multiplied by two. In which case (1p) and why (1p) would you chose this initialization?

ReLU activation. (1p) → Half of the input. (1p)

0 1 2

b) You are given a convolutional layer with kernel size 3, number of filters 3, stride 1 and padding 1. Compute the shape of the weights (0.5p) and write them down explicitly such that this convolutional layer represents the identity for an RGB image input (1.5p).

Shape of W=(3,3,3,3) (0.5 p)
W[0,0,1,1] = W[1,1,1,1] = W[2,2,1,1] = 1, else 0 (1.5p) i.e. W =

$$\begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

0 1 2

c) Name two reasons to use an inception layer in favor of a standard convolutional layer.

Avoid choosing kernel size / dimensionality reduction leads to lower computational cost

0 1 2

d) What are the definitions of *bias* and *variance* in the context of machine learning?

Bias: underfitting, error caused by model being too simple
Variance: overfitting, high variance in the model means its pays more attention to training data and doesnt generalize on unseen data (val / test)

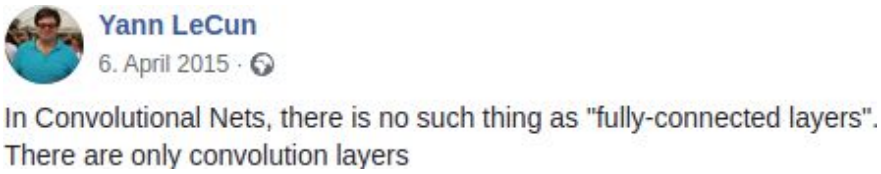
e) In Generative Adversarial Networks the generator and discriminator play a two player min-max game. Why is this hard to optimize and what heuristical method is used instead of the default min-max formulation.

0 1 2

G can not learn when D rejects all generator samples
G maximizes the log-probability of D being mistaken

f) Consider the quote below. Demonstrate how a fully connected layer with an input size of 512 neurons and an output of 10 neurons can be modeled as a convolutional layer.

0 1 2



$512 \xrightarrow{unsqueeze} 512x1x1 \xrightarrow{\text{apply 1x1 conv, weight shape (512, 10, 1, 1)}} 10x1x1$

g) Explain the Markov Assumption in reinforcement learning.

0 1 2

$P[s_{t+1}|s1, ..., s_t] = P[s_{t+1}|s_t]$ or
The future is independent of the past given the present

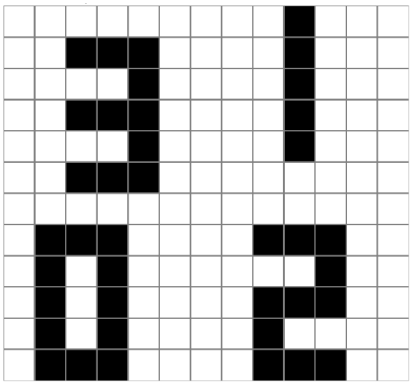
h) Where do we use neural networks in Q-learning (1p) and why are they needed (1p)?

0 1 2

Function approximator for Q values: $Q^*(s, a) = Q(s, a; \theta)$ where θ are the network parameters.
For even semi big problems, calculating all state action pairs is not tractable, but we can approximate them using neural networks.

0
1
2

i) The following image shows a rectangular image of size 16×16. Design a 5×5 convolutional filter that is maximally activated when sliding over the '3' in the image below (Black pixels are 1s and white are -1. For simplicity use only -1s and 1s in your designed filter).



$$W = \begin{bmatrix} -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & 1 & 1 & 1 & -1 \end{bmatrix}$$

1's could be shifted to the left or right
Grading mostly binary (0 or 2 points) with exceptions being $\cdot(-1)$ or a single wrongly placed operator.

0
1
2

j) Name one advantage and one disadvantage of Recurrent Neural Networks in general.

Advantage: can model series of variable lengths
Disadvantage: vanishing/exploding gradients

0
1
2

k) Long-Short Term Memory Units suffer less from the vanishing gradient problem than vanilla RNNs. What two design changes make this possible?

Gate system, esp. the forget gate (1 pt) .
Highway for gradients through the cell state. (1 pt)

Problem 3 House Prices and Backpropagation (11 credits)

You are tasked to predict house prices for your first job, i.e. for a given input vector of numbers you have to predict a single floating point number that indicates the house price (e.g., between 0 and 1m euros).

a) What network loss function would you suggest for this problem (0.5p) and why (0.5p)?

0
1

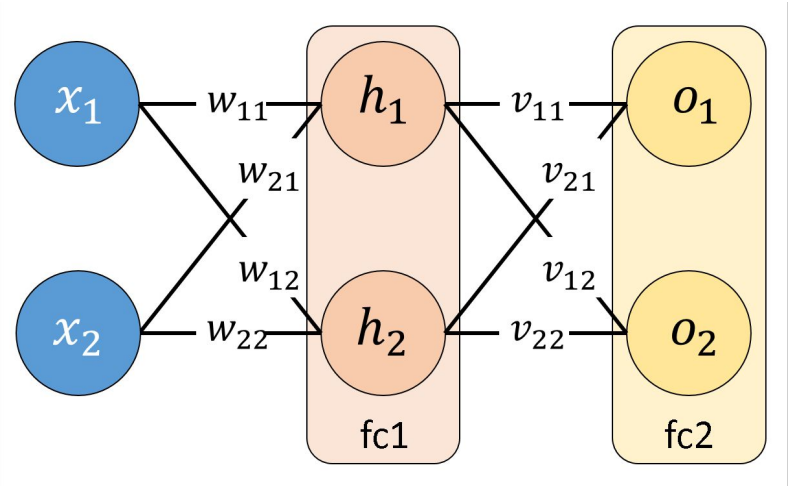
Loss: MSE, floating point regression

b) How would you approximate the task as a classification problem (0.5) and which loss function would you propose in this situation (0.5)?

0
1

Bucket prices into price ranges
Cross Entropy

After collecting some data you start off with a simple architecture



Your architecture is composed of two fully-connected layers (fc1 and fc2) which both contain

- a linear layer with weights and biases as outlined on the next page,
- followed by a Dropout with a probability parameter of 0.5,
- as well as a Leaky ReLu with a parameter of 0.5 as your non-linearity of choice at the end.

c) How many trainable parameters does your network have?

0
1

12

Weights and biases of the linear layers:

Layer	fc1						fc2					
Nodes	h_1			h_2			o_1			o_2		
Variable	w_{11}	w_{21}	b_{h_1}	w_{12}	w_{22}	b_{h_2}	v_{11}	v_{21}	b_{o_1}	v_{12}	v_{22}	b_{o_2}
Value	1.0	-1.0	1.0	-1.0	1.0	-1.0	0.5	-1.0	1.0	1.0	1.0	0.0

0 1 2 3 4

d) You are experiencing difficulties during training and thus decide to check the network in test mode manually. In your test case the input x values are

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

What is the output of your network for that input? Write down all computations for each step.

h_1 linear$= 1 \cdot 1 + 0 \cdot -1 + 1 = 2$ (0.5) $-Dropout > 1$ $-LReLu > 1$ (0.5),
h_2 linear$= 1 \cdot -1 + 0 \cdot 1 - 1 = -2$ (0.5) $-Dropout > -1$ $-LReLu > -0.5$ (0.5),
o_1 linear$= 1 \cdot 0.5 + -0.5 \cdot -1 + 1 = 2$ (0.5) $-Dropout > 1$ $-LReLu > 1$ (0.5),
o_2 linear$= 1 \cdot 1 + -0.5 \cdot 1 + 0 = 0.5$ (0.5) $-Drop > 0.25$ $-LReLu > 0.25$ (0.5),
$=> o = \begin{pmatrix} 1 \\ 0.25 \end{pmatrix}$
0.5 for linear and 0.5 for dropout + leaky relu

0 1

e) As you were unsure of your loss function, you phrase the task as a binary classification problem for each of your two outputs independently. Calculate the binary cross-entropy with respect to the natural logarithm for the labels

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

You may assume $0 \cdot \ln(0) = 0$. (Write down the equations and keep the simplified logarithm.)

BCE(x, y) $= -(y_1 \cdot \log(o_1) + (1 - y_1) \cdot \log(1 - o_1)) - (y_2 \cdot \log(o_2) + (1 - y_2) \cdot \log(1 - o_2))$
$= -\ln(1) - \ln(1 - 0.25) = -\ln(0.75))$ (1p)
-0.5 for minus
BCE for both components: $-(y \cdot \log(o) + (1 - y) \cdot \log(1 - o))$

f) Please update the weight v_{21} using gradient descent with a learning rate of 1.0 with respect to the binary cross-entropy loss as well as labels from e) and the forward computation from d). (Please write down all your computations. Writing down formulas and values in general form is encouraged)

0 1 2 3

Layers: o_{lin}, o_{drop}, o_{lrelu}, BCE.

$$\begin{aligned}\frac{\partial BCE}{\partial v_{21}} &= \frac{\partial(-y_1 \ln(o_{lrelu}))}{\partial o_{lrelu}} \cdot \frac{\partial o_{lrelu}}{\partial o_{drop}} \cdot \frac{\partial o_{drop}}{\partial o_{lin}} \cdot \frac{\partial o_{lin}}{\partial v_{21}} \\ &= -\frac{1}{o_{lrelu}} \cdot 1 \cdot 0.5 \cdot \frac{\partial(h_{afterlrelu2} \cdot v_{21} + b_{o_2})}{\partial v_{21}} \\ &= -1 \cdot 0.5 \cdot h_{2afterlrelu} = -0.5 \cdot -0.5 = 0.25\end{aligned}$$

$$v_{21}^{+} = v_{21} - lr \cdot \frac{\partial BCE}{\partial v_{21}} = -1.0 - 1.0 \cdot 0.25 = -1.25$$

Split in layers: 1p,
ln derivative 0.5p,
final derivative result 0.5p,
update 1p

Problem 4 Optimization (11 credits)

0 1

a) Why can't we expect to find a global minima while training neural networks?

Neural networks are non-convex

- many (different) local minima
- no (practical) way to say which is globally optimal

0 1

b) Why is finding a local minimum often enough?

The idea of the overparameterization in neural networks suggests that many local minimia provide equivalent performance (Note: Similar performance does not mean similarity in parameter space)

0 1 2

c) What is a saddle point (1p)? What is the advantage/disadvantage of Stochastic Gradient Descent (SGD) in dealing with saddle points (1p)?

Saddle point - The gradient is zero (0.5p), but it is neither a local minima nor a local maxima.
SGD has noisier updates and can help escape from a saddle point. Disadvantage: no momentum

0 1

d) Explain the concept behind momentum in SGD.

Avoid getting stuck in local minima. or accelerate optimization. Uses *exponentially* weighted averages of past gradients.

0 1

e) Which optimizer introduced in the lecture uses second but not first order momentum?

RMSProp

0 1 2

f) Explain the beta (β_1 and β_2) hyperparameters of Adam with respect to the gradients.

These two are exponential decay rates for the moment estimates. β_1 is related to bias correction for first moment estimate (decays the running average of the gradient), while β_2 is used in bias correction of second moment estimate (decays running average of square of gradient).

g) What would be an advantage of a second order optimization method such as the Newton method (1p) ? Why is it not commonly used in the context of neural networks?

0
1

Faster / second order convergence in fewer iterations (1p).
Estimating the Hessian in a stochastic setting is difficult - stochasticity doesnt work well with 2nd order methods. Note: Second part of the question isn't graded - the question is worth 1p instead of 2p originally

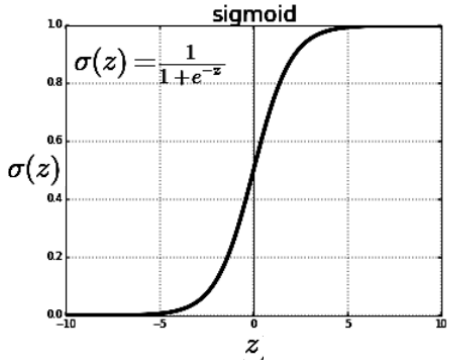
h) Name the key idea of Newton's method (1p) and write down the update step formula (1p).

0
1
2

Approximate by second order taylor expansion
$\theta* = \theta_0 - H^{-1}\nabla_\theta L(\theta)$

Problem 5 Network Architectures and Training (10 credits)

You are training a neural network with 10 convolutional layers and the activation function shown below:



0 1 a) What is the purpose of activation functions in neural networks?

Introduce non-linearity or otherwise network is only linear / prevents layer collapse (1pt).

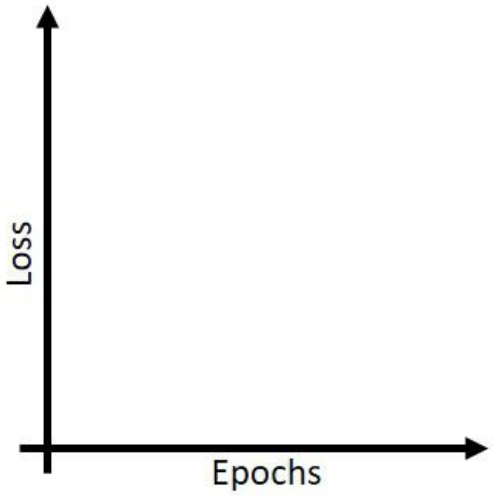
0 1 b) You find that your network still is not training properly and discover that your network weights have all been default initialized to 1. Why might this cause issues for training?

Weights all 1 means everything computes the same function, gradients are all the same -> network capacity

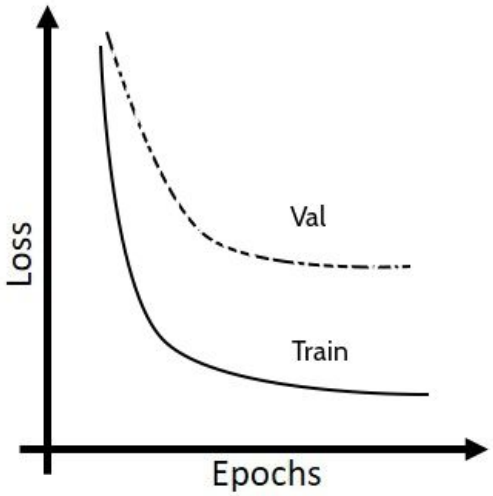
0 1 2 c) How can you resolve the problems with weight initialization and provide stable training in most scenarios for the proposed network? Explain your solution.

Xavier / Kaiming initialization (or their equation) (1p); aims to maintain same variance of output as input (1p).

0 1 2 d) After employing your solutions, you are ready to train your network for image segmentation. After 50 epochs, you come to the conclusion that the network is too large for such a task. What is this effect called? How do you make this observation? Make a plot of the corresponding training (regular line) and validation losses (dashed line) and name them appropriately.



Overfitting (1p); Plot (1p).



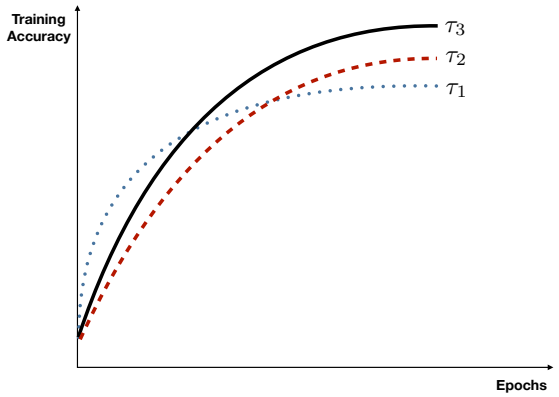
e) Without changing the convolutional layers of your network, name two approaches to counteract the problems encountered in (f).

0
1
2

weight regularization / weight decay, data augmentation, dropout, more data, early stopping (1p each)

f) You adapt your network training accordingly, and now are performing a grid search to find the optimal hyperparameters for vanilla stochastic gradient descent (SGD). You try three learning rates τ_i with $i \in \{1, 2, 3\}$, and obtain the following three curves for the training accuracy. Order the learning rates from larger to smaller.

0
1
2



$\tau_1 > \tau_3 > \tau_2$. (2p for all correct, no half points given)

Problem 6 Batchnormalization (5 credits)

A friend suggested you to use Batchnormalization in your network. Recall the batch normalization layer takes values $x = (x(1), ..., x(m))$ as input and computes $x = (x_{norm}^{(1)}, ..., x_{norm}^{(m)})$ according to:

$$x_{norm}^{(k)} = \frac{x^{(k)} - \mu}{\sqrt{\sigma^2}} \qquad \text{where } \mu = \frac{1}{m}\sum_{k=1}^{m} x^{(i)}, \quad \sigma^2 = \frac{1}{m}(x^{(k)} - \mu)^2.$$

It then applies a second transformation to get $y = (y^{(1)}, ..., y^{(m)})$ using learned parameters $\gamma^{(k)}$ and $\beta^{(k)}$:

$$y^k = \gamma^{(k)}(x)_{norm}^k + \beta^{(k)}.$$

a) How would you make the formulation above numericly stable?

0
1

Replace $\frac{x^{(i)}-\mu}{\sqrt{\sigma^2}}$ with $\frac{x^{(i)}-\mu}{\sqrt{\sigma^2+\epsilon}}$.
Prevents division by zero if variance is zero
Other answers: add a small constant / noise to σ^2 / denominator

b) How can the network undo the normalization operation of Batchnorm? Write down the exact parameters.

0
1
2

$\gamma = \sqrt{var(x^k)}$, $\beta^k = E(x^k)$

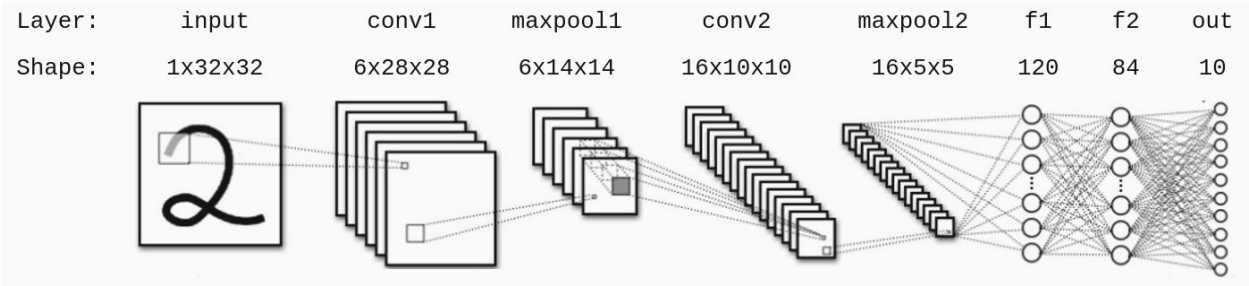
c) Name the main difference of Batchnormalization during training or testing and note down eventual parameters that need to be stored.

0
1
2

Train: compute statistics from minibatch. To store: mean and variance
Test: use stored mean and variance and don't compute any batch statistics

Problem 7 Convolutional Neural Networks (12 credits)

You are contemplating design choices for a convolutional neural network for the classification of digits. LeCun et. al suggest the following network architecture:



For clarification: the shape **after** having applied the operation 'conv1' (the first convolutional layer in the network) is 6x28x28.
All operations are done with stride 1 and no padding. For the convolution layers, assume a kernel size of 5x5.

0 1 2 3

a) Explain the term 'receptive field' (1p). What is the receptive field of one pixel after the operation 'maxpool1'(1p)? What is the receptive field of a neuron in layer 'f1' (1p)?

Receptive field is the size of the region in the input space that a pixel in the output space is affected by.
maxpool1: 6x6. One pixel after maxpool1 is affected by 4 pixels (2x2) in conv1. with 5x5 kernel and stride 1, a 2x2 output comes from a 6x6 grid.
(0.5p if only maxpool1 wrt to conv1(= 2x2) specified)
f1: whole image (32x32)

0 1

b) Instead of digits, you now want to be able to classify handwritten alphabetic characters (26 characters). What is the **minimal** change in network architecture needed in order to support this?

Change no. of output neurons from 10 to 26
(0.5p if only "change number of output neurons" specified)

0 1 2

c) Instead of taking 32×32 images, you now want to train the network to classify images of size 68×68. List two possible architecture changes to support this?

- Resize layer to downsample images to 32x32 / Downsample images to 32x32 (preprocess)
- conv 5x5 ($68 \rightarrow 64$) + maxpool 2x2 ($64 \rightarrow 32$) before the current architecture (0.5p if only specified conv+maxpool without parameters)
- Change input dimension of layer f1 to 16x14x14 (= 3136) (0.5p if only suggested changing input dimension layer without new dim)
- fully convolutional layers + global average pooling (0.5p if only fully conv layer suggested without global average pooling)

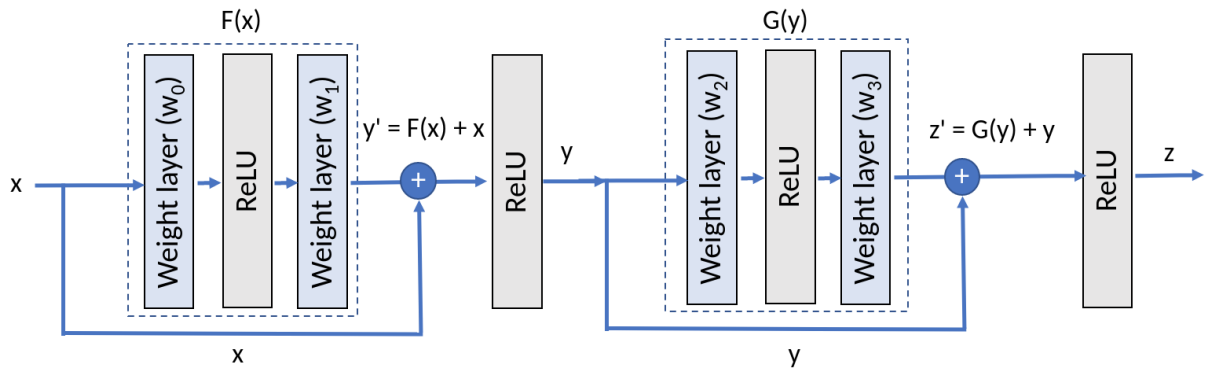
d) Your architecture works and you manage to classify characters fairly well. After reading many online blogs, you decide to try out a much deeper network to boost the network's capacity. Name 2 problems that you might encounter when training very deep networks.

0 1 2

- Vanishing gradients
- Memory issues
- Overfitting
- Increased training time

e) You read that skip connections are beneficial for training deep networks. In the following image you can see a segment of a very deep architecture that uses skip connections. How are skip connections helpful? (1p). Demonstrate this mathematically by computing gradient of output z with respect to 'w_0' for the network below in comparison to the case without a skip connection (3p). For simplicity, you can assume that gradient of ReLU, $\frac{d(ReLU(p))}{dp} = 1$.

0 1 2 3 4



Help prevent vanishing gradients / Provides highway for the gradients in backward pass (1p)

Let,

$$z' = G(y) + y$$
$$G(y) = ReLU(w_2 y) w_3$$
$$z = ReLU(z')$$
$$y' = F(x) + x$$
$$F(x) = w_1 ReLU(w_0 x)$$
$$y = ReLU(y')$$
$$\frac{dz}{dw_0} = \frac{dz}{dz'}\frac{dz'}{dy}\frac{dy}{dy'}\frac{dy'}{dw_0}$$
$$\frac{dz}{dw_0} = \frac{d(ReLU(z'))}{dz'}\left(\frac{dG(y)}{dy} + 1\right)\frac{d(ReLU(y'))}{dy'}\frac{dF(x)}{dw_0}$$
$$\frac{dz}{dw_0} = \frac{d(ReLU(z'))}{dz'}\left(w_3 w_2 \frac{d(ReLU(w_2 y))}{d(w_2 y)} + 1\right)\frac{d(ReLU(y'))}{dy'} w_1 x \frac{d(ReLU(w_0 x))}{d(w_0 x)}\Big)$$

Putting ReLU derivatives to 1

$$\frac{dz}{dw_0} = (w_3 w_2 + 1) w_1 x$$ (2 points for full expansion, 1pt if $\frac{dG(y)}{dy}$ and $\frac{dF(x)}{dw_0}$ are not expanded)

Comparing this to derivative without skip connection, which is

$$\frac{dz}{dw_0} = (w_3 w_2) w_1 x$$ (1 point / 0.5p if not expanded)

The extra '+1' term in the skip connection derivative help propagation of gradient flow, preventing vanishing gradients

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