Chair for Computer Vision and 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:**	Thursday 8th August, 2019
Examiner:	Prof. Dr. Leal-Taixé, Prof. Dr. Nießner	**Time:**	08:00 – 09:30

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

Working instructions

- This exam consists of **20 pages** with a total of **6 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
- 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. This includes calculators.

Left room from __________ to __________ / Early submission at __________

Problem 1 Multiple Choice (18 credits)

Mark your answer clearly by a cross in the corresponding box. Multiple correct answers per question possible.

a) Your network is overfitting. What are good ways to approach this problem?

- [ ] Increase the size of the validation set
- [x] Increase the size of the training set
- [x] Reduce your model capacity
- [ ] Reduce learning rate and continue training

b) A sigmoid layer

- [ ] has a learnable parameter.
- [ ] cannot be used during backpropagation.
- [x] is continuous and differentiable everywhere.
- [ ] maps to values between -1 and 1.

c) Training error does not decrease. What could be a reason?

- [x] Too much regularization.
- [ ] Too many weights in your network.
- [x] Bad initialization.
- [x] Learning rate is too high.

d) How many network parameters are in ResNet-152?

- [ ] 1,337,337.
- [x] 60,344,232.
- [ ] more than a billion.
- [ ] 152.

e) 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

f) Dropout

- [ ] has trouble with tanh activations.
- [x] is an efficient way for regularization.
- [x] can be seen as an ensemble of networks.
- [ ] makes your network train faster.

g) Consider a simple convolutional neural network with a single convolutional layer. Which of the following statements is true about this network?

- [ ] All input nodes are connected to all output nodes.
- [ ] It is scale invariant.
- [ ] It is translation invariant.
- [ ] It is rotation invariant.

h) You are building a model to predict the presence (labeled 1) or absence(labeled 0) of a tumor in a brain scan. The goal is to ultimately deploy the model to help doctors in hospitals. Which of these two metrics would you choose to use?

- [x] Recall $= \frac{\text{True positive examples}}{\text{Total positive examples}}$.
- [ ] Precision $= \frac{\text{True positive examples}}{\text{Total predicted positive examples}}$.
- [ ] Average Precision $= \frac{\text{True positive examples + True negative examples}}{\text{Total examples}}$.

i) Why you would want use 1×1 convolutions? (check all that apply)

- [ ] Predict binary class probabilities.
- [x] Collapse number of channels.
- [x] Learn more complex functions by introducing additional non-linearities.
- [ ] To enforce a fixed size output.

Problem 2 **Short Questions (24 credits)**

0 1 2

a) You are training a neural network with 15 fully-connected layers with a *tanh* nonlinearity. Explain the behavior of the gradient of the non-linearity with respect to very large positive inputs.

Because the tanh is almost flat for very large positive values (1pt), its gradient will be almost 0. (1pt)
Comment: Points deducted for saying "gradient saturates" but not mentioning the small value of the gradient, a neuron saturates but not the gradient.

0 1 2

b) Why might this be a problem for training neural networks? Name and explain this phenomenon.

Vanishing gradient (1p), during backprop gradient of non-linearity is close to zero, makes training/parameter updates much much slower. (explanation is another 1p)

0 1 2

c) In modern architectures, another type of non-linearity is commonly used. Draw and name this non-linearity (1p) and explain why it helps solve the problem mentioned in the previous two questions (1p).

Rectified Linear Unit (0.5p) + drawing (0.5p)
Because ReLU activations are linear, they do not saturate for large (positive) values, and hence freely allow gradients to change weights in the network. (1p) *Comment: Saturation was enough*

0 1 2

d) Why do we often refer to $L2$-regularization as “weight decay”? Derive a mathematical expression that includes the weights W, the learning rate η, and the $L2$-regularization hyperparameter λ to explain your point.

Weight update with objective function J incl. weight decay:
$W = W - \eta \nabla_W \left(J + \frac{1}{2}\lambda \sum_i W_i^2\right)$
$W = W(1 - \eta\lambda) - \eta \nabla_W J,$
where η = learning rate and λ = regularisation parameter with $\eta\lambda << 1$.
Value of W is pushed towards zero in each iteration.

Points: Qualitative answer: 0.5. Mathematical part: l2 loss 0.5, weight update formula 0.5, final result 0.5

e) You are solving the binary classification task of classifying images as cars vs. persons. You design a CNN with a single output neuron. Let the output of this neuron be z. The final output of your network, $\hat{y}$ is given by:

0 1

$$\hat{y} = \sigma\left(ReLU(z)\right)$$

You classify all inputs with a final value $\hat{y} \geq 0.5$ as car images. What problem are you going to encounter?

Using *ReLU* then sigmoid will cause all predictions to be positive (0.5p)
$\sigma(ReLU(z)) \geq 0.5 \quad \forall z.$ (0.5p)
Writing "all predictions are 'cars' is enough

f) Suppose you initialize your weights **w** with uniform random distribution $U(-\alpha, \alpha)$. The output **s** for given input vector **x** is given by

0 1 2

$$s_i = \sum_{j=0}^{n} w_{ij} \cdot x_j,$$

where n is the number of input values.
Assume that the input data **x** and weights are independent and identically distributed. How do you have to choose α such that the variance of the input data and the output is identical, hence $\mathrm{Var}(s) = \mathrm{Var}(x)$.
Hint: For two statistically independent variables X and Y holds:

$$\mathrm{Var}(X \cdot Y) = \left[\mathrm{E}(X)\right]^2 \mathrm{Var}(Y) + \left[\mathrm{E}(Y)\right]^2 \mathrm{Var}(X) + \mathrm{Var}(X)\mathrm{Var}(Y)$$

Furthermore the PDF of an uniform distribution $U(a, b)$ is

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise.} \end{cases}$$

The variance of a continuous distribution is calculated as

$$\mathrm{Var}(X) = \int_{\mathbb{R}} x^2 f(x)\, dx - \mu^2,$$

where μ is the expected value of X.

$\mathrm{Var}(s_i) = \mathrm{Var}\left(\sum_{j=0}^{n} w_{ij} \cdot x_j,\right) = \sum_{j=0}^{n} \mathrm{Var}(w_{ij})\mathrm{Var}(x_j) = n \cdot \mathrm{Var}(w)\mathrm{Var}(x)(1p)$

$\mathrm{Var}\left(U(-\alpha, \alpha)\right) = \frac{1}{3}\alpha^2$ (0.5)

$\alpha = \sqrt{\frac{3}{n}}$(0.5)

Correct result: 2p
If only $Var(w) = \frac{1}{n}$ is written then 1p.

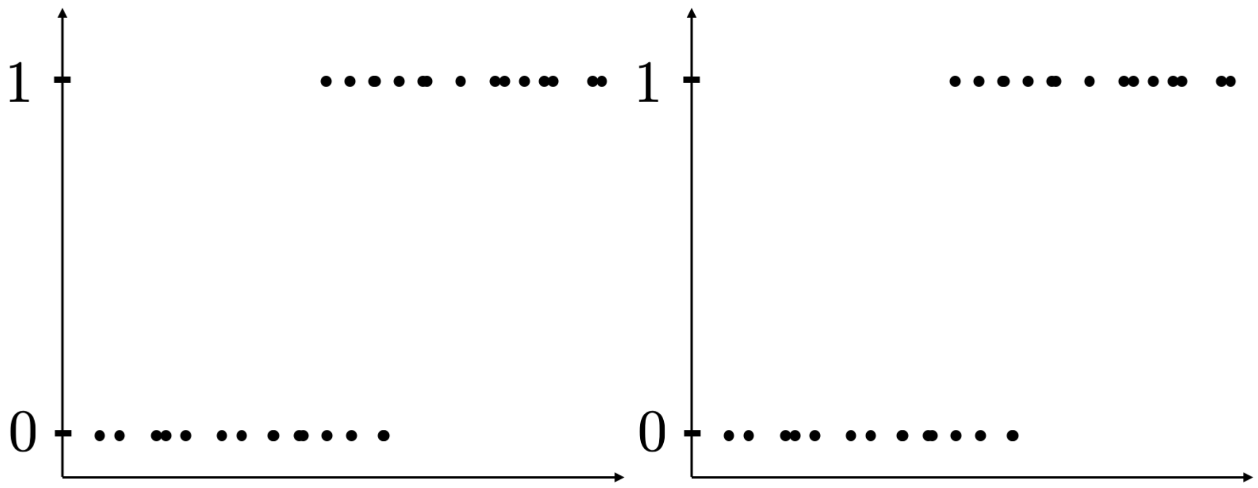
g) Consider 2 different models for image classification of the MNIST data set.
The models are: (i) a 3 layer perceptron, (ii) LeNet.
Which of the two models is more robust to translation of the digits in the images? Give a short explanation why.

0 1 2

LeNet (0.5p), Convolutional layers (1.5p)

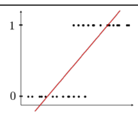
2p: lenet mentioned and convolutional layers as reason
1.5: lenet mentioned, convolutional layers are mentioned but students wrote too much text which included wrong statements

h) Consider the following one-dimensional data points with classes $\{0, 1\}$. Sketch a linear (0.5p) and logistic (0.5p) regression into the figures. Which model is more suitable for this task (1p)?

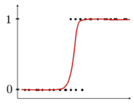


Plot linear regression (left) and logistic regression (right).

Logistic regression. (1pt)
an "S" is not a function



(0.5pt)



(0.5p) *line should go through points not 0,*

i) What is the mean and standard deviation of Xavier initialization? What changes to this initialization would you propose when used with ReLU non-linearities?

Mean=0, Var= $\frac{1}{n}$. With relus: Var= $\frac{2}{n}$ (1p each) *Writing variance instead of stddev was fine, both solutions accepted*

j) You have 4000 cat and 100 dog images and want to train a neural network on these images to do binary classification. What problems 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 images/reweight dataloader/reweight loss function/collect more dog images/data augmentation for dogs (0.5pt/sol) No points for: dropout, regularization, batch norm, transfer learning, "get more data"

k) Why is initializing all the weights of a fully connected layer to the same value problematic during training?

If all weights are equal, nodes will learn the same thing during backpropagation, and this limits the capacity. (2 if correct)
If there is no mention of gradients/weight updating, e.g. by only saying "the network will not learn", -> 1.5p

l) What is the difference between dropout for convolutional layers compared to dropout for fully connected layers? Explain both behaviours.

0
1
2

Conv: drop feature map at random, fully connected: drop weights at random (1p each)

Problem 3 **Optimization (12 credits)**

0 1 2

a) Explain the concept behind RMSProp optimization. How does it help converging faster?

Mitigate step size in directions with high-variance gradients (1). Can increase learning rate (1).

0 1

b) Which SGD variation uses first and second momentum?

Adam.

0 1

c) Why is it common to use a learning rate decay?

When far away (0.5p), one want higher gradients to get closer to solution; the closer you get, the less jitter/overshooting you want. (0.5)

0 1 2

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

Saddle point - The gradient is zero (0.5p), but it is neither a local minima nor a local maxima (0.5p) (or:the gradient is zero and the function has a local maximum in one direction, but a local minimum in another direction).
SGD has noisier updates and can help escape from a saddle point (1p)

0 1

e) Why would one want to use larger mini-batches in SGD?

Make the gradients less noisy.

0 1

f) Why do we usually use small mini-batches in practice?

Limited GPU memory / faster compute (for each batch), so faster update

0 1

g) Your network's training curve diverges (assuming data loading is correct). Name one way to address the problem through hyperparameter change.

reduce learning rate (1 point each)

h) What is an epoch?

0
1

full run through the entire train set

i) When is SGD guaranteed to converge to a local minima (provide formula)?

0
1
2

Robbins-Monro condition; $\sum_{i=1}^{\infty} \alpha_i = \infty$ (1p) and $\sum_{i=1}^{\infty} \alpha_i^2 < \infty$ (1p)

Problem 4 Convolutional Neural Networks and Advanced Architectures (12 credits)

In the following we assume that the input of our network is a $224 \times 224 \times 3$ color (RGB) image. The task is to perform image classification on 1000 classes. You design a network with the following structure [CONV - RELU] x 20 - FC - FC. That is, you place 20 consecutive convolutional layers (including non-linear activations), followed by two fully-connected layers. Each layer will have its own number of filters and kernel size.

a) The first 3 convolutional layers have each 5 filters with kernels of size 3×3, applied with stride 1 and no padding. How large is the receptive field of a feature after the 3 convolutional operations?

1x1$->$ 3x3$->$ 5x5$->$ 7x7 (1p)

b) What are the dimensions of the feature map after the 3 convolutional operations from (a) ?

224 - 2 (first conv layer) - 2 (second conv layer) - 2 (third conv layer) = 218x218x5 (number of filters) (1p spatial size, 1p kernel size)

c) What are the dimensions of the weight tensor of the first convolutional layer? (1p) What does each dimension represent? (1p)

Shape: (3, 5, 3, 3) (1pt)
Reasoning: input channels (RGB), output channels/number of filters, kernel size = 3x3 (1p)
(no points when only 3dims are mentioned)

d) After the 10th convolutional layer your feature map has size 100x100x224. You realize the next convolutional filter operation will involve too many multiplications that make your network training slow. However, the next layer requires identical *spatial size* of the feature map.
Propose a solution for this problem (1p) and demonstrate your solution with an example (1p).

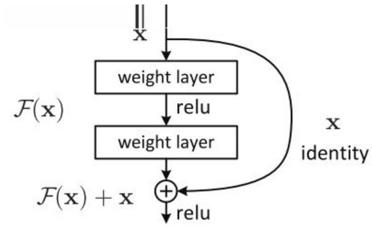
1x1 convolutions (1p)
If you use 25 convolutional filters of 1x1x256, we would reduce the feature map to 100x100x25, making the next operation cheaper. (1p)
Comment: Any output larget than 100x100x224 is wrong.

e) Your network is now trained for the task of image classification. You now want to use the trained weights of this network for the task of *image segmentation* for which you need a pixel-wise output. Which layers of your original network described above can you *not* reuse for the image segmentation task? (1p) Describe briefly how you would adapt the network for image segmentation given *any input image size*? (1p)

The FC layers, because they take a fixed input size (1p) Make it fully convolutional (1pt). *Comment: mentioning only upscaling: 0.5p*

f) You decide to increase the number of layers substantially and therefore you switch to a ResNet architecture. Draw a ResNet block (1p). Describe all the operations inside the block (1pt). What is the advantage of using such a block in terms of training (1p)?

0
1
2
3



(1pt)

Final **summation** of passed features through convolutional layers and skipped initial features. $F(x) + x$. (1p)

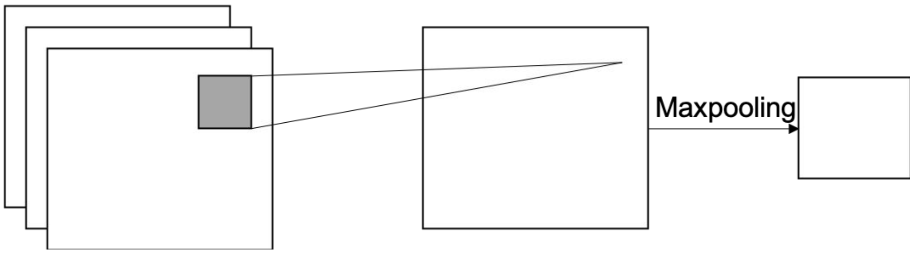
One of multiple solutions: Skip-connections

- provide highways for gradients and make network easier to train
- resolve vanishing gradient problem (1p)

Problem 5 Backpropagation and Convolutional Layers (12 credits)

Your friend is excited to try out those "Convolutional Layers" you were talking about from your lecture. However, he seems to have some issues and requests your help for some theoretical computations on a toy example.

Consider a neural network with a convolutional (without activation) and a max pooling layer. The convolutional layer has a single filter with kernel size $(1,1)$, no bias, a stride of 1 and no padding. The filter weights are all initialized to a value of 1. The max pooling layer has a kernel size of $(2,2)$ with stride 2, and 1 zero-padding.



You are given the following input image of dimensions $(3,2,2)$:

$$x = \left(\begin{bmatrix} 1 & -0.5 \\ 2 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 1 \\ -1.5 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \right)$$

a) Compute the forward pass of this input and write down your calculations.

Forward pass

$$\begin{bmatrix} 1 & -0.5 \\ 2 & -2 \end{bmatrix} + \begin{bmatrix} -2 & 1 \\ -1.5 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 \\ 0.5 & -1 \end{bmatrix} (1p)$$

After max pooling,

$$\begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix} (1p)$$

b) Consider the corresponding ground truth,

$$y = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Calculate the binary cross-entropy with respect to the natural logarithm by summing over all output pixels of the forward pass computed in (a). You may assume $\log(0) \approx -10^9$. (Write down the **equation** and keep the logarithm for the final result.)

$$\begin{aligned} BCEloss &= -\sum_i t_i \log s_i \qquad \text{(0.5}p\text{ for either his or the line below)} \\ &= -\log(2w_1 - 1.5w_2) - \log(-0.5w_1 + w_2) \\ &= -\log(0.5) - \log(0.5) = 2\log 2 \qquad (1p) \end{aligned}$$

c) You don't recall learning the formula for backpropagation through convolutional layers but those 1×1 convolutions seem suspicious. Write down the name of a common layer that is able to produce the same result as the convolutional layer used above.

Fully-connected layer

d) Update the kernel weights accordingly by using gradient descent with a learning rate of 1. (Write down your calculations!)

0
1
2
3
4
5

Partial derivatives for w1/w2 (2p),

$$\frac{\partial BCE}{\partial w_1} = -\frac{\partial \ln(2w_1 - 1.5w_2) + \ln(-0.5w_1 + w_2)}{\partial w_1} = -\frac{2}{2w_1 - 1.5w_2} - \frac{-0.5}{-0.5w_1 + w_2} = -4 + 1 = -3$$

$$\frac{\partial BCE}{\partial w_2} = -\frac{\partial \ln(2w_1 - 1.5w_2) + \ln(-0.5w_1 + w_2)}{\partial w_2} = -\frac{-1.5}{2w_1 - 1.5w_2} - \frac{1}{-0.5w_1 + w_2} = 3 - 2 = 1$$

Update using gradient descent for w1/w2 (2p),

$$w_1^+ = w_1 - lr * \frac{\partial BCE}{\partial w_1} = 1 - 1 \times (-3) = 4$$

$$w_2^+ = w_2 - lr * \frac{\partial BCE}{\partial w_2} = 1 - 1 \times 1 = 0$$

Derivate and update for w3 (1p total):

$$\frac{\partial BCE}{\partial w_3} = 0$$

$$w_3^+ = w_3 - 0 = 1$$

1p if the person only wrote at least the gradient descent update rule

e) After helping your friend debugging, you want to showcase the power of convolutional layers. Deduce what kind of 3×3 convolutional filter was used to generate the output (right) of the grayscale image (left) and write down its 3×3 values.

Vertical edge detector (1p)

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

Flipping & Scaling are OK

f) He finally introduces you to his real problem. He wants to find 3×3 black crosses in grayscale images, i.e., each pixel has a value between 0 (black) and 1 (white).

You notice that you can actually hand-craft such a filter. Write down the numerical values of a 3×3 filter that maximally highlights on the position of black crosses.

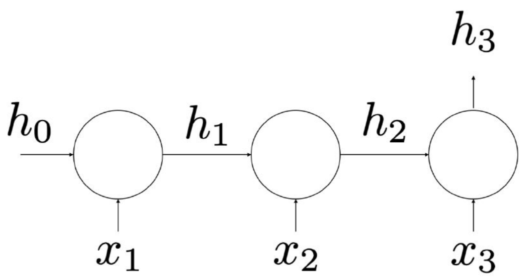
$$\begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix} (2p)$$

Flipping & Scaling are OK, even though pixel values were given

Problem 6 Recurrent Neural Networks and LSTMs (12 credits)

a) Consider a vanilla RNN cell of the form $h_t = \tanh(V \cdot h_{t-1} + W \cdot x_t)$. The figure below shows the input sequence x_1, x_2, and x_3.

0
1
2



Given the dimensions $x_t \in \mathbb{R}^4$ and $h_t \in \mathbb{R}^{12}$, what is the number of parameters in the RNN cell? Neglect the bias parameter.

$4 \times 12 + 12 \times 12$ (1 pt) $= 48 + 144 = 192$ (1 pt)

b) If x_t is the 0 vector, then $h_t = h_{t-1}$. Discuss whether this statement is correct.

0
1
2

False: (1 pt)
After transformation with V and non-linearity $x_t = 0$ does not lead to $h_t = h_{t-1}$ (1 pt) . Full points require explanation, solely equation not sufficient.

0
1
2
3

c) Now consider the following **one-dimensional** ReLU-RNN cell.

$$h_t = \text{ReLU}(V \cdot h_{t-1} + W \cdot x_t)$$

(Hidden state, input, and weights are scalars)
Calculate h_1, h_2 and h_3 where $V = 1$, $W = 2$, $h_0 = -3$, $x_1 = 1$, $x_2 = 2$ and $x_3 = 0$.

$h_0 = -3$
$h_1 = \text{relu}(1 \cdot (-3) + 2 \cdot 1) = 0$ (1 pt)
$h_2 = \text{relu}(1 \cdot 0 + 2 \cdot 2) = 4$ (1 pt)
$h_3 = \text{relu}(1 \cdot 4 + 2 \cdot 0) = 4$ (1 pt)

d) Calculate the derivatives $\frac{\partial h_3}{\partial V}$, $\frac{\partial h_3}{\partial W}$, and $\frac{\partial h_3}{\partial x_1}$ for the forward pass of the ReLU-RNN Cell of (c). Use that $\frac{\partial}{\partial x}\mathrm{ReLU}(x)\Big|_{x=0} = 1$.

0 1 2 3

$$h_t = \mathrm{ReLU}(V \cdot h_{t-1} + W \cdot x_t) = \mathrm{ReLU}(z_t)$$

$$\frac{\partial h_3}{\partial V} = \frac{\partial}{\partial x}\mathrm{ReLU}(x)\Big|_{x=z_3} \cdot h_2 + \frac{\partial}{\partial x}\mathrm{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot h_1 + \frac{\partial}{\partial x}\mathrm{ReLU}(x)\Big|_{x=z_1} \cdot V^2 \cdot h_0 =$$

$$= 1 \cdot 4 + 1 \cdot 1 \cdot 0 + 0 \cdot 1 \cdot (-3) = 4 \qquad \text{(1 pt)}$$

$$\frac{\partial h_3}{\partial W} = \frac{\partial}{\partial x}\mathrm{ReLU}(x)\Big|_{x=z_3} \cdot x_3 + \frac{\partial}{\partial x}\mathrm{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot x_2 + \frac{\partial}{\partial x}\mathrm{ReLU}(x)\Big|_{x=z_1} \cdot V^2 \cdot x_1 =$$

$$1 \cdot 0 + 1 \cdot 2 + 0 \cdot 0 = 2 \qquad \text{(1 pt)}$$

$$\frac{\partial h_3}{\partial x_1} = \mathrm{ReLU}(x)\Big|_{x=z_3} \cdot V \cdot \mathrm{ReLU}(x)\Big|_{x=z_2} \cdot V \cdot \mathrm{ReLU}(x)\Big|_{x=z_1} \cdot W = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 0 \cdot 2 = 0 \qquad \text{(1 pt)}$$

Only correct and calculated result gives point.

0
1
2

e) A Long-Short Term Memory (LSTM) unit is defined as

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

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)

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

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