UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Exam in:	INF5860/INF9860 — Machine Learning for Image Analysis
Day of examination:	11th June 2018
Examination hours:	14:30 – 18:30

This exercise set consists of 0 pages.

Appendices:	None
Permitted aids:	Certified calculator

Read the entire exercise text before you start solving the exercises. Please check that the exam paper is complete. If you lack information in the exam text or think that some information is missing, you may make your own assumptions, as long as they are not contradictory to the "spirit" of the exercise. In such a case, you should make it clear what assumptions you have made.

You should spend your time in such a manner that you get to answer all exercises shortly. If you get stuck on one question, move on to the next question.

Your answers should be short, typically a few sentences and / or a sketch should be sufficient.

Every subtask has equal weight in the evaluation.

(Continued on page 2.)

Exercise 1 Dense neural networks

Suppose we have a small dense neural network as is shown in **??**. The

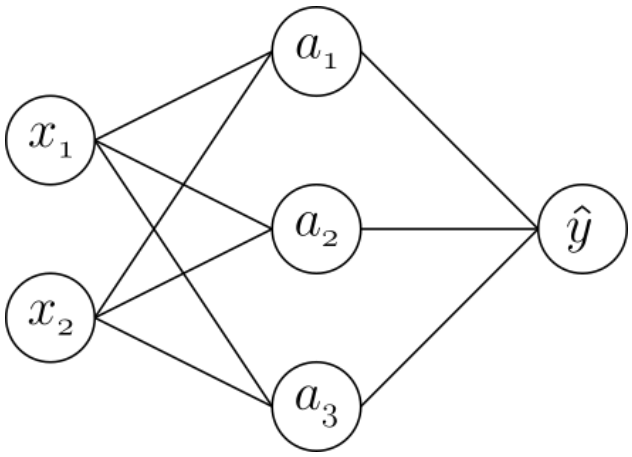

Figure 1: A small dense neural network

input vector $x = [x_1, x_2]^\mathsf{T}$ and the associated ground truth y, are

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \quad y = 32.$$

In the first layer we have the following weight and bias parameters

$$\begin{pmatrix} w_{11}^{[1]} & w_{12}^{[1]} & w_{13}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} & w_{23}^{[1]} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 3 \\ 2 & -1 & 1 \end{pmatrix}, \quad \begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

In the second layer we have the following weight and bias parameters

$$\begin{pmatrix} w_{11}^{[2]} \\ w_{21}^{[2]} \\ w_{31}^{[2]} \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}, \quad b_1^{[2]} = 1.$$

For clarity,

- $w_{jk}^{[l]}$ is the value of the weight parameter from node j in layer $l-1$ to node k in layer l.
- $b_k^{[l]}$ is the value of the bias parameter at node k in layer l.

(Continued on page 3.)

1a

Compute the value of the network output, $\hat{y}$, when the activation functions in the first and second layer are identity functions.

For the activations in the first layer, with g as the identity function, we have

$$\begin{aligned} a_1 &= g(w_{11}^{[1]} \cdot x_1 + w_{21}^{[1]} \cdot x_2 + b_1^{[1]}) \\ &= g(2 \cdot 1 + 2 \cdot 3 + 1) \\ &= 9 \\ a_1 &= g(w_{12}^{[1]} \cdot x_1 + w_{22}^{[1]} \cdot x_2 + b_2^{[1]}) \\ &= g(1 \cdot 1 - 1 \cdot 3 + 0) \\ &= -2 \\ a_1 &= g(w_{13}^{[1]} \cdot x_1 + w_{23}^{[1]} \cdot x_2 + b_3^{[1]}) \\ &= g(3 \cdot 1 + 1 \cdot 3 - 1) \\ &= 5 \end{aligned}$$

We then get the following activation in the final layer

$$\begin{aligned} \hat{y} &= w_{11}^{[2]} \cdot a_1 + w_{21}^{[2]} \cdot a_2 + w_{31}^{[2]} \cdot a_3 + b_1^{[2]} \\ &= 3 \cdot 9 - 1 \cdot 2 + 2 \cdot 5 + 1 \\ &= 36 \end{aligned}$$

1b

Compute the value of the network output, $\hat{y}$, when the activation functions in the first and the second layer are ReLU functions.

For the activations in the first layer, with g as the ReLU function, we have

$$\begin{aligned} a_1 &= g(w_{11}^{[1]} \cdot x_1 + w_{21}^{[1]} \cdot x_2 + b_1^{[1]}) \\ &= g(2 \cdot 1 + 2 \cdot 3 + 1) \\ &= 9 \\ a_1 &= g(w_{12}^{[1]} \cdot x_1 + w_{22}^{[1]} \cdot x_2 + b_2^{[1]}) \\ &= g(1 \cdot 1 - 1 \cdot 3 + 0) \\ &= 0 \\ a_1 &= g(w_{13}^{[1]} \cdot x_1 + w_{23}^{[1]} \cdot x_2 + b_3^{[1]}) \\ &= g(3 \cdot 1 + 1 \cdot 3 - 1) \\ &= 5 \end{aligned}$$

(Continued on page 4.)

We then get the following activation in the final layer

$$\begin{aligned}\hat{y} &= g(w_{11}^{[2]} \cdot a_1 + w_{21}^{[2]} \cdot a_2 + w_{31}^{[2]} \cdot a_3 + b_1^{[2]}) \\ &= g(3 \cdot 9 + 1 \cdot 0 + 2 \cdot 5 + 1) \\ &= 38\end{aligned}$$

1c

Given a squared error loss function $J = (\hat{y} - y)^2$, and *identity functions* as activations in all nodes (as in task **1a** above), compute the value of the following four partial derivatives

$$\frac{\partial J}{\partial b_1^{[2]}},$$
$$\frac{\partial J}{\partial w_{21}^{[2]}},$$
$$\frac{\partial J}{\partial b_2^{[1]}},$$
$$\frac{\partial J}{\partial w_{13}^{[1]}}.$$

For our simple network, we can write out the loss function explicitly as

$$\begin{aligned}J &= (\hat{y} - y)^2 \\ &= \left(\left(\sum_{k=1}^{3} w_{k1}^{[2]} a_k + b_1^{[2]} \right) - y \right)^2 \\ &= \left(\left(\sum_{k=1}^{3} w_{k1}^{[2]} \left(\sum_{j=1}^{2} w_{jk}^{[1]} x_j + b_k^{[1]} \right) + b_1^{[2]} \right) - y \right)^2\end{aligned}$$

(Continued on page 5.)

With this, a general solution is then

$$\frac{\partial J}{\partial b_1^{[2]}} = 2(\hat{y} - y) \cdot 1$$
$$\frac{\partial J}{\partial w_{k1}^{[2]}} = 2(\hat{y} - y) \cdot a_k$$
$$\frac{\partial J}{\partial b_k^{[1]}} = 2(\hat{y} - y) \cdot w_{k1}^{[2]}$$
$$\frac{\partial J}{\partial w_{jk}^{[1]}} = 2(\hat{y} - y) \cdot w_{k1}^{[2]} x_j$$

We therefore get the following values of the partial derivatives

$$\begin{aligned}
\frac{\partial J}{\partial b_1^{[2]}} &= 2(\hat{y} - y) \cdot 1 \\
&= 2 \cdot (36 - 32) \cdot 1 \\
&= 8 \\
\frac{\partial J}{\partial w_{21}^{[2]}} &= 2(\hat{y} - y) \cdot a_2 \\
&= 2 \cdot (36 - 32) \cdot (-2) \\
&= -16 \\
\frac{\partial J}{\partial b_2^{[1]}} &= 2(\hat{y} - y) \cdot w_{21}^{[2]} \\
&= 2 \cdot (36 - 32) \cdot 1 \\
&= 8 \\
\frac{\partial J}{\partial w_{13}^{[1]}} &= 2(\hat{y} - y) \cdot w_{31}^{[2]} x_1 \\
&= 2 \cdot (36 - 32) \cdot 2 \cdot 1 \\
&= 16
\end{aligned}$$

1d

Given everything as in task **1c**, perform one step with gradient descent with learning rate 2, and compute the updated values of the two parameters

$$b_2^{[1]},$$
$$w_{13}^{[1]}.$$

(Continued on page 6.)

With learning rate λ, the parameters are updated as

$$b_k^{[l]} \leftarrow b_k^{[l]} - \lambda \frac{\partial J}{\partial b_k^{[l]}}$$
$$w_{jk}^{[l]} \leftarrow w_{jk}^{[l]} - \lambda \frac{\partial J}{\partial w_{jk}^{[l]}}.$$

Inserting the values above gives

$$\begin{aligned} b_2^{[1]} &= 0 - 2 \cdot 8 \\ &= -16 \\ w_{13}^{[1]} &= 3 - 2 \cdot 16 \\ &= -29 \end{aligned}$$

Exercise 2 Convolutional neural networks

2a

You are given an input image (**??**) and a kernel (**??**). Your task is to evaluate the shaded pixel in the image after the convolution. The origin of the kernel is the shaded pixel. Assume we use zero padding.

3	4	5	1	0
2	0	7	3	4
1	3	1	0	8
5	5	3	8	1
1	2	5	9	9

(a) Input image

1	5	1
2	3	2
1	5	1

(b) Kernel

Figure 2

Answer = 49

(Continued on page 7.)

2b

Explain the dimensions of the filter bank (learnable convolution kernel weights) in a typical convolutional layer.

The order is not important, but the answer should include the elements in W:

$$\texttt{shape}(W) = \texttt{[filter\_out, hx, hy, filter\_in]}$$

2c

What is the receptive field after the following layers?

Layer #	filter size	stride
Layer 1	3×3	1
Layer 2	5×5	1
Layer 3	2×2	2
Layer 4	3×3	1

Receptive field, r

$$\begin{aligned} r &= 3 + (5-1) + (2-1) + 2 \cdot (3-1) \\ &= 12 \end{aligned}$$

2d

Given two models:

- Model A has a single 7×7 convolutional layer.
- Model B has multiple subsequent 3×3 convolutional layers. The number of layers is such that the size of the receptive field is the same as in model A.

What is the ratio of the number of parameters involved in the two different models? Assume that the number of input and output feature maps are equal in the two models (and for all layers). Ignore bias parameters and use stride = 1.

Model B needs to have 3 successive 3×3 convolutions in order to get a receptive field with the same size as that of model A. The parameters are

(Continued on page 8.)

the weights in the convolution filters, and therefore the ratio between the number of parameters is

$$\frac{Model\,A}{Model\,B} = \frac{7 \cdot 7}{3 \cdot 3 \cdot 3} = \frac{49}{27} \tag{1}$$

Exercise 3 Generalization

3a

Explain briefly the effect of the regularization loss on the hypothesis set.

The regularization loss will reduce the size of the hypothesis set and penalize complex decision boundaries.

3b

Splitting the available dataset is a common way to estimate the out-of-sample error. If you check multiple models on the test set and select the best performing model, will this be a good indicator of the out-of-sample error? Justify your answer.

In the example, the model is tuned on the test set. For this reason, the test error will be over-optimistic.

Exercise 4 Object localization

You are to design an image classifier that in addition predicts where in the image the (single) object of interest is. You can assume that there is only one interesting object in the image, if any.

4a

Explain briefly how you would set up the target vector, y, (and the output vector), in this case.

In this case, we are only localizing one single object per image, so it would suffice to extend an ordinary classification target vector with bounding box coordinates. As an example where we are to classify images into n

(Continued on page 9.)

classes and localizing objects, we could have a target vector

$$y = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \\ b_x \\ b_y \\ b_h \\ b_w \end{pmatrix}$$

where c_0 indicates if there is an object present (or if there is only background), c_i indicates what class in $i = 1, \ldots, n$. This is a regular classification target vector such that $\sum_{i=0}^{n} c_i = 1$. (b_x, b_y) could be the top left corner of a bounding box, b_h the height, and b_w the width of the bounding box.

4b

Explain briefly how an associated loss function could look like.

As a loss function, we could have a cross entropy loss on the class labels, and a mean squared error on the bounding box coordinates. The bounding box loss would only be invoked if $c_0 \neq 1$. A detailed answer could therefore be

- Partition y into $y = [c, b]$, with
  - $c = [c_0, c_1, \ldots, c_n]$
  - $b = [b_r, b_c, b_h, b_w]$
- L_2 loss for object bounding box location b

$$L_b(\hat{b}, b) = \sum_{i \in \{x,y,h,w\}} (\hat{b}_i - b_i)^2$$

- Cross entropy loss for object categories c

$$L_c(\hat{c}, c) = -\sum_{i=0}^{n} c_i \log \hat{c}_i$$

- The total loss can be written as

$$L(\hat{y}, y) = L_c + I[\hat{c}_0 > 0] L_b$$

- Notice that we only compare the bounding box if there is an object in the image

(Continued on page 10.)

Exercise 5 Recurrent neural networks

Recurrent neural networks are powerful models for processing sequential data.

5a

Show and describe the most general recurrence formula for a recurrent neural network.

$$h_t = f(h_{t-1}, x_t)$$

h: hidden state

x: input

t: time step

5b

Why are long range dependencies difficult to learn in a recurrent neural network?

Vanishing gradients, hidden state size equal to one:

$$h_t = \tanh\left(W_{hh}h_{t-1} + W_{hx}x_t\right)$$
$$\frac{\partial h_t}{\partial h_{t-1}} = \left[1 - \tanh^2(W_{hh}h_{t-1} + W_{hx}x_t)\right] \cdot W_{hh}$$

If W_{hh} is smaller than one, the recursive use of $\frac{\partial h_t}{\partial h_{t-1}}$ will force the gradient to zero as $[1 - \tanh^2(W_{hh}h_{t-1} + W_{hx}x_t)] \leq 1$.

5c

Why is Gated Recurrent Units (GRU) more efficient in preserving long range dependencies than vanilla RNNs?

Uses gates to control information flow. (The hidden state is not required to undergo transformation for each time step)

(Continued on page 11.)

5d

What is the advantage and disadvantage with using Truncated Back-propagation Through Time (TBTT)?

Advantage: Faster parameter updates.

Disadvantage: Not able to capture longer dependencies than the truncated length.

5e

Explain *briefly* how a convolutional neural network and a recurrent neural network can be used to generate descriptive natural language from an image?

We can use a CNN to encode information of an image into a feature vector. We can take this feature vector as an input to the first hidden state in a RNN. During training, we can feed the caption as a sequence of words into the RNN and ask the RNN to predict the next word in the caption. During caption generation, we can use the predicted word at time t, as an input at time $(t+1)$.

Exercise 6 Unsupervised learning

6a

Explain briefly the difference between supervised learning, unsupervised learning and reinforcement learning.

Supervised learning: The goal is to create a function f that "approximates" the mapping: $f(x) \approx y,\ \forall(x,y)\ in\ \Omega_{train}$

Unsupervised learning: We do not have any labeled data. Our goal is to find an underlaying structure of the data

Reinforcement learning: In RL an agent learns from the experiences it gains by interacting with the environment. The goal is to maximize an accumulated reward given by the environment.

(Continued on page 12.)

6b

Describe how you could use an autoencoder to denoise images with gaussian white noise.

First, we would have to train the autoencoder to remove noise from images. If f is the autoencoder, we would train it to approximate $f(x + w) \approx x$ where w is the added white noise. This trained autoencoder could then be used to remove white gaussian noise from new images (from the same distribution).

6c (PhD candidates only)

Describe the concepts of a variational autoencoder and what it is most commonly used for.

A variational autoencoder is trained to encode a signal into a mean μ and variance σ^2 in the latent space. The decoder then reconstructs the signal from a random vector $z = \mu + w\sigma$, where $w \sim \mathcal{N}(0, 1)$. Because of this, the decoder learns a "continuous" latent space, that is, vectors that are close in the latent space should also be close after decoding.

We can guide the solutions by restricting the generative distribution q to approximate some distribution p In that way, the latent vectors, even for different categories, will be relatively close. The desired distribution used in variational autoencoders is the standard normal distribution $p = \mathcal{N}(0, 1)$. We use the familiar KL-divergence between the desired and the generated distribution as a regularizer in the loss function. With this, the total loss, J, for an input example x is something like

$$J(x) = ||x - f(x)|| + D_{KL}(p||q_{\mu,\sigma})$$

That is, the total loss is the sum of what we call the *reconstruction loss* (just some distance metric between the input and the output) and the *latent loss*. The latent loss for a single input x can be shown to be equal to

$$D_{KL}(p||q_{\mu,\sigma}) = \frac{1}{2}(\mu^2 + \sigma^2 - \log \sigma^2 - 1)$$

It is enough to mention the composition of the total loss, the exact expression is not necessary.

Variational autoencoders are used to generate signals, and are especially useful when you want to have more control over what you are generating.

(Continued on page 13.)

Exercise 7 Miscellaneous

7a

Why will the TensorFlow code in **??** throw an error?

```
import tensorflow as tf

x = tf.Variable(initial_value=1)
y = tf.Variable(initial_value=2)
z = tf.add(x,y)

with tf.Session() as sess:
    z_val = sess.run(z)
```

Figure 3: TensorFlow code snippet

```
FailedPreconditionError:  Attempting to use uninitialized value Variable
3.
```

7b

Explain briefly the concept of Policy gradient methods in reinforcement learning.

Learning the probability that an action is good directly. Idea:

- If a trajectory achieves a high reward, the actions were good
- If a trajectory achieves a low reward, the actions were bad
- We will use gradients to enforce more of the good actions and less of the bad actions. Hence the method is called Policy Gradients.

7c (PhD candidates only)

Explain briefly the normalization done by the batch-norm algorithm during training and testing.

During training:

- Compute μ and σ as the minibatch mean and variance of x (usually the activation of a layer)

(Continued on page 14.)

- Normalize to zero mean and unit variance:

$$x_0 = (x - \mu)/(\sigma + \epsilon)$$

- Allow a shift to trainable scaling:

$$= \gamma x_0 + \beta$$

During testing:
After training, μ and σ are recomputed for the entire training data set (using running average estimates). These values are stored and used during testing.

(Continued on page 15.)