EXAMINATION PAPER

Home exam number 2

Home exam in:	**FYS-3033 - Deep Learning**
Hand-out:	**Tuesday March 24, 2020, 08:00**
Hand-in:	**Thursday April 23, 2020, 14:00**

The exam contains 9 pages including this cover page

Contact person:	**Michael Kampffmeyer**
Email:	**michael.c.kampffmeyer@uit.no**

Before You Start

Module examination

This is the second portfolio assignment for FYS-3033 Deep Learning. Portfolio assessment of project assignments counts about 40 % of the final grade in the course. All modules in the portfolio are assessed as a whole and one combined grade is given. Note that access to the final examination requires submission and approval of project assignments.

Portfolio instructions

Your code should be submitted together with your report (see instructions below). **Further, please include a discussion of the results obtained in your report.**

Make sure your report shows that you understand what you are doing. The code should be commented in such a way that any person with programming knowledge should be able to understand how the program works. Like your report, the code must be your own individual work.

You are permitted to use deep learning frameworks such as Pytorch and Tensorflow. As there is a lot of code available online, please make sure that your report and code clearly show that you understand what you are doing.

Hand-in format

Please submit *one* single `.zip` file that contains two folders, one called `doc` that contains your report, and another one called `src` containing the code. The file name of the `.zip` file should follow the format `homeexam2_candidate`*`XX`*`.pdf` (replace *XX* with your candidate number obtained from StudWeb) for anonymity.

Follow the hand-in instruction in WiseFlow and make sure to submit before the WiseFlow room closes.

Problem 1

In this problem you will derive some of the theoretical results that are found in unsupervised deep learning methodology.

(1a) The goal of generative models is to sample from the underlying distribution, p_{data}, of data $\mathbf{x}$. Generative Adversarial Networks (GANs) achieve this goal by constructing a generator G, which tries to fool a discriminator D. The generator transforms a sample $\mathbf{z}$ from a prior noise distribution, $p_{\mathbf{z}}$, into a new distribution p_g. Given the loss that the discriminator is tasked to maximize

$$V(G,D) = \int_x p_{\text{data}(x)} \log(D(x))dx + \int_z p_z(z)\log(1 - D(g(z)))dz \tag{1}$$

and assuming that the optimal solution has been reached ($p_g = p_{\text{data}}$). Show that the optimal solution for the discriminator given a fixed generator g is

$$D(x) = \frac{1}{2} \; . \tag{2}$$

Finally, illustrate that this optimum corresponds to a value of $-\log(4)$.

(1b) The reparameterization trick is a key component of VAEs. What is its purpose in VAEs? For VAEs the latent variable is sampled from a normal distribution with parameters $\mu_{z|x}$ and $\sigma_{z|x}$, where the parameters are learned by the encoder of the VAE. For the reparameterization trick we perform our sampling operation as

$$z = \mu_{z|x} + \sigma_{z|x}\epsilon \; , \tag{3}$$

where $\epsilon \sim \mathcal{N}(0,1)$. Show that z follows a normal distribution with parameters $\mu_{z|x}$ and $\sigma_{z|x}$.

(1c) Show that the KL divergence between two univariate gaussians $p(x) = N(\mu_1, \sigma_1)$ and $q(x) = N(\mu_2, \sigma_2)$ can be expressed as

$$KL(p,q) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \; . \tag{4}$$

Discuss how this result is used in Variational Autoencoders.

Problem 2

In this problem you will perform the task of semantic segmentation. Semantic segmentation is the task of classifying each pixel in an image to a specific class and crucial for applications such as self-driving cars. One of the first end-to-end learnable approaches was proposed by Long et al. in 2015 and is called the Fully Convolutional Network (FCN) (1). Figure 1 displays the architecture. In this exercise, you will implement the network below and test it on the Cambridge-driving Labeled Video Database. Note that the FCN is based on the VGG-16 (2), but with an included upsampling layer. Furthermore, the two fully connected layers at the end of the VGG-16 are converted to convolutional layers. For details about the VGG-16 architecture, see column D of Table 1 in Simonyan and Zisserman (2). A script for loading the data is provided on Canvas.

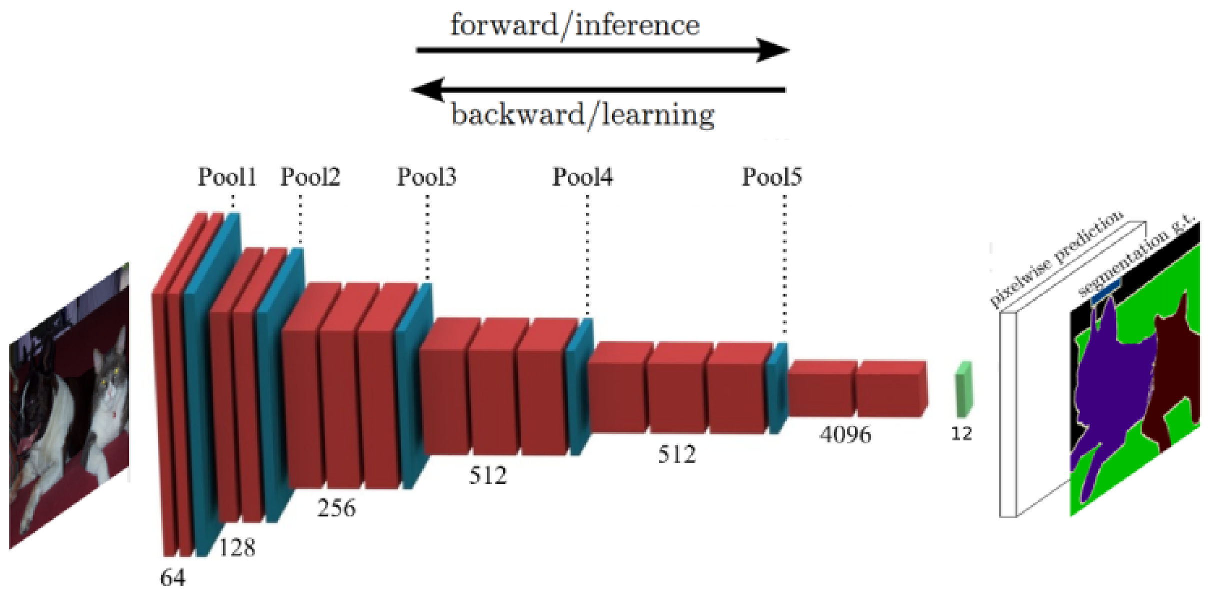

Figure 1: *Illustration of the FCN architecture (1).*

(2a) Explain how fractional strided convolution (also called deconvolution or transposed convolution) can be used to build a segmentation network and perform upsampling.

(2b) Implement the Fully Convolutional Network illustrated in Figure 1 (FCN-32s) with bi-linear upsampling. Report the mean accuracy and mean Intersection over Union (IOU) overall and per class.

(2c) Implement the Fully Convolutional Network illustrated in Figure 1 (FCN-32s) with learned upsampling (with fractional strided convolution). Report the mean accuracy and mean Intersection over Union (IOU) overall and per class. Compare and discuss the results from this implementation from the network in (2b).

(2d) Explain Batch Normalization (BN) (3) and Dropout (4).

Bonus exercise:

Note, this exercise will give you bonus points, but is not required to get a a full score in this home exam.

(a) Include Batch Normalization in the network from problem (2c) before each ReLU-nonlinearity. Plot the cost and accuracy with and without BN, discuss results.

(b) Include Dropout in the network from problem (2c) with a dropout rate of 0.5 after each of the two layers that contain 4096 filters, discuss results.

Problem 3

In this problem you will perform the task of unsupervised domain adaptation by considering a labeled source dataset (The Street View House Numbers (SVHN) Dataset) and an unlabeled target dataset (MNIST). The task is to use the label information in SVHN to design a MNIST classifier based on the Adversarial Discriminative Domain Adaptation (ADDA) (5) approach illustrated in Figure 2. Scripts for loading both datasets is provided on Canvas.

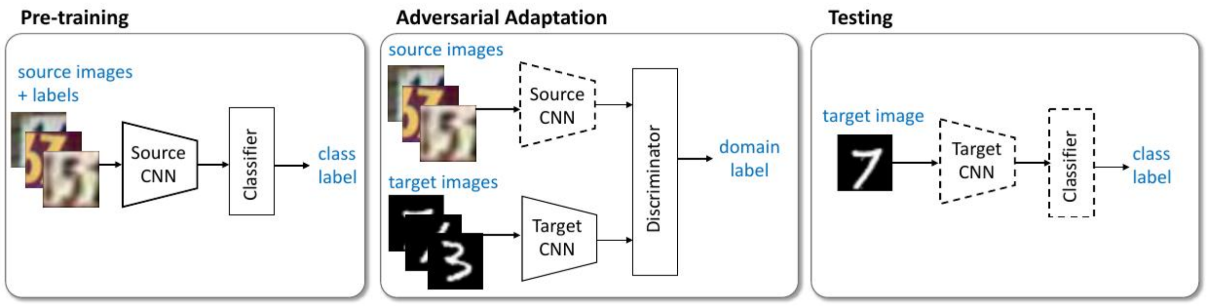

Figure 2: *Domain adaptation framework ADDA (5). The source encoder CNN is first pre-trained using labeled source image examples. Adversarial adaptation is performed by learning a target feature extractor CNN such that a discriminator that sees source features and target features cannot reliably predict their domain label. During testing, features are extracted for target images and classified by the source classifier. Dashed lines indicate fixed network parameters.*

(3a) Implement and train a network to classify the SVHN images. The network architecture is given in Figure 3. Note, this is a wider version of the model that you implemented in Home Exam 1.

(3b) Use the network trained in Problem (3a) directly to classify the images of the MNIST dataset. Report the achieved accuracy.

(3c) In the ADDA framework, a discriminator is used to train the target feature extractor by aligning its extracted features with the features from the source feature extractor (the CNN trained in Problem 3a without the output layer). In Figure 4, the discriminator in the framework has been replaced with a mean squared error loss. Discuss, why this is not a good idea.

(3d) Train the target feature extractor by aligning its extracted features with the features from the source feature extractor using a discriminator. Note, both feature extractors should have an identical architecture and it is recommended to initialize the target feature extractor with the weights of the source feature extractor. The discriminator should consist of two layers with 500 units, each followed by ReLU-nonlinearities, and the final output layer.

Test the learned feature extractor (followed by the source classifier) on the MNIST dataset. Compare the results to the results from Problem (3b) when no domain adaptation is performed.

Disclaimer: As mentioned in the lecture, training generative adversarial networks can be challenging.

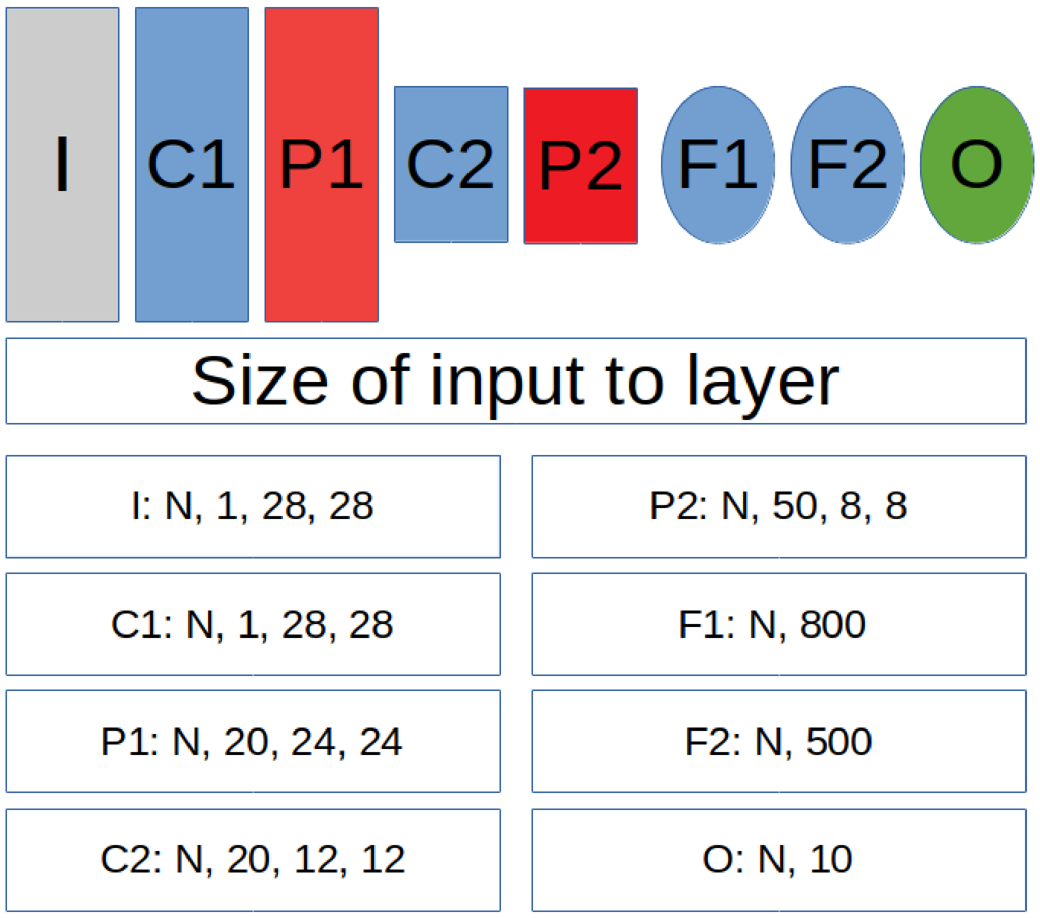

Figure 3: *Illustration of a wide LeNet-5 architecture. Layers labeled with a 'C' refers to a convolutional layer, which performs a convolution operation followed by an activation function. Layers labeled with a 'P' refers to a pooling layer, which performs a max pooling operation. Layers labeled with a 'F' refers to a fully connected layer, which performs matrix multiplication followed by an activation function. The 'I' and 'O' layer indicate the input and the output layer, respectively. Below the architecture itself is a description of the size of the input to each layer. There is no padding in the convolutional layers, stride is set to 1, and the filters are of size* 5×5*. The pooling layers apply* 2×2 *filters with a stride of 2. Assume a ReLU activation function.*

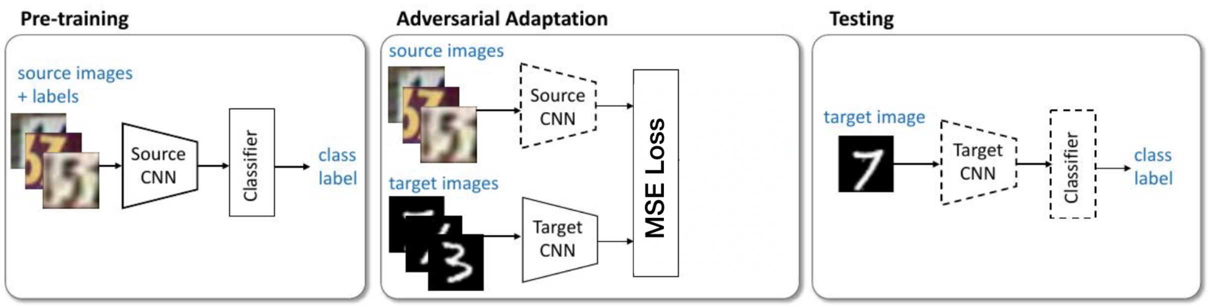

Figure 4: *Modified version of the domain adaptation framework ADDA (5), where the target CNN is trained with a mean squared error loss.*

References

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[5] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.