Deep Learning for Image Analysis 2021: Assignment 3

Joakim Lindblad

April 21, 2021

Due: May 5, 2021

On this assignment you may collect *extra points* (which are, together with the group assignment, the seminar, and the final project task and its presentation, counted towards the overall grade of the assignments part of the course) by handing in a nice looking report which also includes solutions to the optional Exercises 2.3 and 2.4. (In other words, if you are not aiming for top grades, you may skip Exercises 2.3 and 2.4.)

1 Classification of hand-written digits using a Convolutional Neural Network

The most common deep learning approach for ***image classification*** *is the use of Convolutional Neural Networks (CNNs). They reduce the number of needed parameters over fully connected neural networks, and provide reduced translation dependence which is welcome in many applications. In a typical (classic) setup a CNN architecture will contain a sequence of convolutional layers, each followed by an activation function such as ReLU or sigmoid, interspersed with spatial pooling (most often maxpooling) layers in order to enlarge the field of view. Towards the end of the network, after several convolutional and pooling layers, the high-level image features are classified by fully-connected (also called dense) layers, were all neurons or nodes are connected to all activations of the previous layer. In the final layer a softmax function maps the non-normalized output of the network to a probability distribution over predicted output classes.*

1.1 Classification of MNIST

In this assignment task you will again perform classification of the MNIST dataset, this time using a convolutional neural network instead of the fully connected network used in assignment 2. Since it is a bit cumbersome to implement backpropagation for the convolution operations, we will here leave the do-it-yourself coding and move to the PyTorch deep learning framework.

Exercise 1.1 *Start by implementing exactly the same (fully connected) network as you designed for Assignment 2 using built-in PyTorch functions. Try to set learning parameters to imitate your code from Assignment 2; it is fine to use the default weight initialization scheme of the framework. Compare the performance with your Assignment 2 results; do you observe similar accuracy? How many times faster/slower was your own implementation? (Make sure to enable GPU support if a suitable graphics card is available!).*

Exercise 1.2 *Let us now exchange the Fully connected architecture for a Convolution based one. Construct the network specified below in PyTorch. Train the network using standard Stochastic Gradient Descent (SGD) on the MNIST dataset. Choose a suitable mini-batch size and number of training epochs (observing a reasonable convergence) and a learning rate which gives you a good convergence without waiting forever. You are expected to reach above 98% accuracy on the test data partition. How many learnable weights does this network contain? Compare with how many weights you had in the previous exercise.*

1) Convolution	*8 times 3x3x1 convolutions with stride 1 and padding 1*
2) ReLU	*Non-linearity*
3) Max Pooling	*2x2 max pooling with stride 2*
4) Convolution	*16 times 3x3x8 convolutions with stride 1 and padding 1*
5) ReLU	*Non-linearity*
6) Max Pooling	*2x2 max pooling with stride 2*
7) Convolution	*32 $\times$ 3x3x16 convolutions with stride 1 and padding 1*
8) ReLU	*Non-linearity*
9) Fully Connected	*10 fully connected layer*
10) Softmax	*Softmax layer*
11) Classification Output	*Crossentropy with 10 classes*

Exercise 1.3 *In the previous exercise we first applied the activation function (ReLU in this case) and then performed Max pooling. What would change (accuracy, speed) if you swap the order of these two operations? Test your hypothesis empirically by exchanging layers 2 and 3, resp. layers 5 and 6, such that the pooling operation is performed before the ReLU. The difference is probably more distinct for the hyperbolic tangent activation function, why?*

Exercise 1.4 *Now change the optimizer, training the network with ADAM instead of SGD. It is fine to pick the default parameters proposed by PyTorch. (Commonly appearing default values are: GradientDecayFactor (β_1): 0.9000, SquaredGradientDecayFactor (β_2): 0.9990, ϵ: 10^{-8}.) Do you manage to get better results faster than when using the plain SGD optimization?*

Exercise 1.5 *Try a few (at least 3) variations based on what you have learned in the course so far; this could be architectural changes, various regularization approaches, change of optimization method, learning-rate scheme, change of activation function, etc. How good performance do you manage to reach?*

For each of the exercises above, **include in the report:**

1. a plot of the training loss,
2. the classification results (on the test data partition) using classification accuracy as performance measure,
3. also, provide a short comment for each case if you find that the network is over- or under-fitting the data.

For the best performing model, also include a *confusion matrix* of the classification results on the test data partition. Provide a few examples of misclassified images (image, including a note about the real class and predicted class). Make sure to list all your used hyperparameters in the report (including default values of your chosen framework) such that someone else can try to repeat your work (reproducible research).

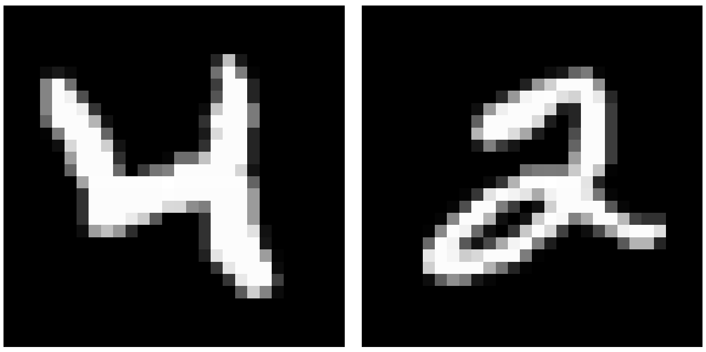
(a) 4 (b) 2

Figure 1: Two example images from the MNIST digit recognition dataset.

Information about the MNIST dataset: The MNIST dataset, illustrated in Fig. 1, consists of 60000 training images, and 10000 test images. The dataset has become a classic benchmark dataset for machine learning; do checkout the `http://yann.lecun.com/exdb/mnist/` page for some context. The version you have received is prepackaged into subfolders 'Train' and 'Test', and further into subfolders '0'-'9'. containing a number of images on the form 0000.png-xxxx.png. (Note: Each class does not have exactly the same number of images.) All images are 28x28 pixels and stored in the png-file format.

2 Semantic segmentation of Biomedical images

In ***semantic image segmentation*** *the objective is to assign an object class to each pixel in the image. Such object classes may be foreground and background, or different objects as well as object boundaries. Unlike image classification where we are interested in one global label for the entire image, in segmentation we wish to perform classification of each pixel to get dense pixel-wise predictions and labels from our model.*

One problem of re-purposing standard CNN architectures, which were developed for classification tasks, for dense predictions are the fully connected layers present in many architectures as we have to preserve the spatial information for the class maps. In 2014 Long et al. introduced Fully Convolutional Networks which allowed dense predictions by avoiding all fully connected layers.

Another problem in using CNNs for segmentation lies in the pooling layers. They increase the field of view to be able to aggregate context within an image, but discard a lot of the spatial information. In segmentation however, we need an exact alignment of the object class maps and our image in full spatial resolution as we want to label every pixel in it. One way to overcome this problem is either to replace pooling layers altogether and aggregate context in a bigger field of view with special convolutional layers which are often referred to as 'dilated' or 'atrous' convolutions. Another way to handle the effect of pooling layers is to recover object details in the needed spatial dimensions by so-called decoders which upsample the information which was encoded in the first part of the CNN. Instead of using simple bilinear interpolation, the interpolation can be learnt by special convolutional layers. There exist different approaches for doing this, transposed convolution (also (incorrectly) known as deconvolution) and pixel shuffle, (a.k.a. sub-pixel convolution) are two commonly used ones. In this assignment we will use transposed convolution.

2.1 Segmentation of the Warwick Biomedical Dataset

In this assignment task you will segment images of Hematoxylin and Eosin (H&E) stained microscopy slides of colorectal cancer (see example in Fig. 2). The images show a group of cells that synthesize substances in the human body – so called *glands*. These are important histological structures which are present in most organ systems as the main mechanism for secreting proteins and carbohydrates. The morphology (shape) of glands is used routinely by pathologists to assess the degree of malignancy of several adenocarcinomas, including prostate, breast, lung, and colon cancers.

The dataset (WARWICK.zip) is provided including ground truth annotations by expert pathologists. To provide improved performance, the stains have been unmixed resulting in two-channel images (the blue component in the images is constant equal to zero). Your task is to, given an input image, compute an output label-image which has the value one (1) where there is a gland in the image, and zero (0) otherwise.

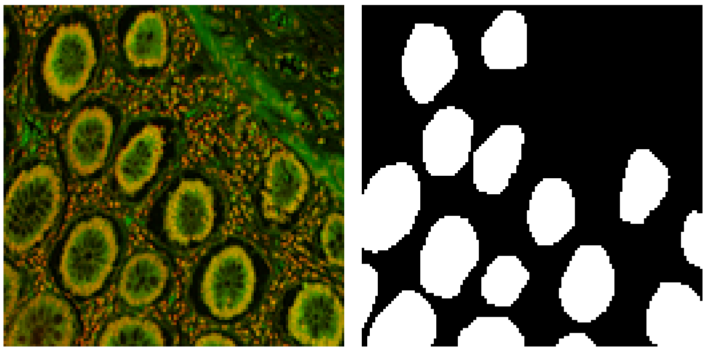
(a) Image 20 (b) Labelling 20

Figure 2: An example image from the Warwick biomedical dataset.

In order to re-purpose a CNN which was developed for a classification task (e.g. the one used for the MNIST task), a few modifications have to be considered:

- The spatial dimensions of the input which have been reduced by pooling layers have to be recovered by transposed convolution layers to gain exact alignment of class maps and the image.
- Do not include any fully-connected layers. The very last fully-connected layer may be replaced by a 1×1-convolution.
- The softmax has to be taken in the channel-direction to give a class prediction per pixel.
- The loss function has to be adapted to handle a ground truth given by a binary image of the same spatial dimensions as the input image, not a single label for the entire image as in classification.

Note, to avoid checkerboard artifacts from the transposed convolutions, it is important that the filter size is an integer times the stride.

Exercise 2.1 *Design a network for semantic segmentation of the provided data set, e.g., by modifying the network you used for the MNIST classification task as specified above.*

1. *Evaluate and report the results of the segmentation on the test-set using the Sørensen–Dice coefficient as performance measure (you are expected to reach an average DSC score above 0.7):*

$$DSC(A, B) = \frac{2|A \cap B|}{|A| + |B|},$$

   for A and B being the gland segmentation maps of the network and the ground truth annotation, respectively.

2. *Visually inspect one of the cases (in the test-set) where the network did not perform well in terms of the evaluation metric, and try to describe what you think makes that particular instance challenging in comparison to the other instances. Include the examples as images in your report.*

Exercise 2.2 *Evaluate on the same segmentation task three different techniques you have learnt in the course on how to improve the performance on unseen data. Here you probably want to split the training data into training+validation when performing parameter tuning. Complement in the report raw performance figures with your own subjective opinions regarding the different techniques. Include comments about over/underfitting.*

Exercise 2.3 ***Not mandatory exercise****, but counted towards getting a higher grade. Implement skip connections in your architecture, bringing higher resolution information into the upsampling path. Evaluate if this improves performance. In particular, check (and visualize in the report) if the use of skip connections leads to better accuracy at the boundary of the glands.*

Exercise 2.4 ***Not mandatory exercise****, but counted towards getting a higher grade. Implement residual connections in your architecture. Evaluate if this improves performance. In particular, check if the inclusion of residual connections allows training of deeper networks by replacing each Convolution+activation pair in your architecture with a block of two or three similar pairs, where the residual connection bridges over each such block. Check the speed of learning with and without the residual connections in place (for otherwise identical architectures).*

Information about the WARWICK dataset: The WARWICK dataset, illustrated in Fig. 2, as presented for this assignment consists of 85 training images, and 60 test images, each of size 128×128 and with a binary mask indicating Glandular regions in the images. This is a selected and pre-processed part (colors are unmixed and images are resized) of the Warwick-QU dataset (released with the MICCAI'2015 GlaS contest). See `https://warwick.ac.uk/fac/sci/dcs/research/tia/glascontest/download/` for information about the original data.

Use the 'Assignment template' (on the Course homepage) and submit a pdf with answers to all (mandatory) exercises. Additionally, submit a zip file with your well commented code.