# Deep Learning for Computer Vision
## Assignment Sheet 2 - Due 03.12.2017 23:59

**Q & A: 28.11.2017, last third of lecture**
**Next exercise group: 5.12.2017, 13:30 (instead of lecture)**

We will train classifiers to recognize Japanese characters. You will work with a dataset with 19909 training and 3514 testing images, see figure 1. The dataset contains 10 different characters therefore we have to train 10 classes.

`http://codh.rois.ac.jp/char-shape/software/pmjt_sample_20161116.tar.gz`

し　に　の　て　り　を　か　く　き　も

Figure 1: Example images from the dataset

**Exercise 2.1: multi-class logistic regression (10 + 30 + 5 points)**

(a) Normalize the dataset such that each image has zero mean and unit norm. Transform the desired outputs from training and testing data into one-hot format compatible with Tensorflow[1]

(b) Train a linear classifier with logistic regression. The probability that a datum $\mathbf{x}$ belongs to class $k$ is

$$P\left(y=k|\mathbf{x}\right)=\exp\left(\mathbf{w}_k^T\mathbf{x}\right)/\sum_j\exp\left(\mathbf{w}_j^T\mathbf{x}\right)=\text{softmax}(W\boldsymbol{x})_k,\ \text{with } W=\begin{bmatrix}\boldsymbol{w}_1^T\\ \vdots\\ \boldsymbol{w}_K^T\end{bmatrix}.$$

The class which has the highest probability is assigned to the datum $\mathbf{x}$. There is a different input-output relationship, therefore, the $L2$ loss does not work for us and you need to use softmax cross entropy with logits. A simple step-by-step tutorial how to build a classifier can be found here:
`https://www.tensorflow.org/get_started/mnist/beginners`

(c) Reshape the ten vectors $\boldsymbol{w}_k^T$ into images, visualize and compare to a few examples of that particular class (e.g. in a nice table of images).

**Exercise 2.2: training a convolutional neural network (20+10+5+10+10 points)**

(a) Build a convolutional neural network with at least two convolution layers to train features and one fully connected layer to classify the features. The last fully connected layer must have 10 units to match number of classes, and use softmax activation (corresponding to the model of exercise 2.1).

(b) Train the prosed architecture on the normalized dataset from Exercise 2.1. Proceed in small mini-batches instead training on the entire dataset at once.

[1] See https://www.tensorflow.org/api_docs/python/tf/one_hot. One-hot encoding encodes a desired output $c \in C$ into a vector of length card $(C)$ where only the element that encodes the class $c$ is one.

(c) Evaluate accuracy on the test and training data on the network (the percentage of correctly classified digits).

(d) Remove one convolution layer and describe how this influences training and testing accuracy.

(e) Try different step sizes for the training (assuming you have chosen gradient descent as in the tutorial example). Explain what happens if the step-size is too large and too small. Draw a graph how the accuracy changes over number of iterations for different step sizes.

(f) (*Bonus*) Perform a hyper-parameter tunning (e.g. number of layers, number of feature maps per layer, kernel size ...). Play with the architecture and evaluate accuracy for different hyperparameter settings using $k$-fold cross-validation. Describe how the changes influence the overall accuracy on test and training data.

`https://www.tensorflow.org/get_started/mnist/pros`

**Exercise 2.3: retraining a network for new tasks (20 points)**
This exercise teaches you how to retrain the final layer of a network for new classification tasks. This is useful for example if you want to use a state-of-the-art network from the internet which was pre-trained on millions of images, and which you cannot fully train yourself.

(a) Create a new network with exactly the same architecture as you have used in exercise 2.2. From your network which was fully trained on the japanese character dataset, copy over the weights and biases of all the layers except for the final classificaiton layer, and mark all of the copied layers as untrainable (extra parameter `trainable=False` at variable creation).

(b) Retrain the final classification layer on MNIST (see tutorials above). Compare classification accuracy with logistic regression on MNIST and a convnet of the same architecture where all layers have been fully trained on MNIST.

**Exercise 2.4: what does a unit respond to? (Bonus, 20 points)**
This exercise is somewhat tricky to perform, and therefore a bonus exercise. The idea is to find an image such that the response of a single unit in the network is maximal. Intuitively, this is the kind of pattern which the unit detects.

- Pick a unit in the network, e.g. a single class score of the final layer. Its output is a function $\varphi(\boldsymbol{x})$ of the input image.
- Set up a Tensorflow graph to solve the optimization problem

$$\underset{\mathbf{x}}{\operatorname{argmin}}\{\lambda \|\mathbf{x}\|_2^2 \; - \; \varphi(\mathbf{x})\}$$

  and perform gradient descent to optimize for the input image. Pick a value of $\lambda$ which gives a sensible result (it probably does not matter much). You have to be careful to mark all variables except for the input image as untrainable. Note: the input image will not be a placeholder in this case, it must be a trainable variable.
- Repeat for units in different layers, and visualize the corresponding response-maximizing images. You probably have to normalize in some way so that you can see them.

**Exercise 2.5: statistics (20 points)**
Let $x_1, \ldots, x_L$ be random samples drawn from a uniform distribution on the interval $[0, \theta]$. The upper bound $\theta$ of the interval is the unknown parameter of the distribution. Let

$$\bar{x} = \frac{1}{L} \sum_{l=1}^{L} x_l$$

denote the sample mean.

(a) Show that $\bar{x}$ is a biased estimator of $\theta$.

(b) Suggest a simple modification so that it becomes unbiased.

(c) Compute the standard error of your estimator in (b), and show that the limit is zero for $L \to \infty$.

(d) Find a different, biased estimator for $\theta$ whose standard error goes to zero with $L \to \infty$.