ML Laboratory 04: Convolutional Neural Networks (CNNs)

1. Objective

Students should understand and be able use some basic pretrained CNN models available in Matlab

2. Theoretical aspects

CNNs are multilayer networks adapted for image processing, which use the **convolution** operation extensively.

Starting points

1. The layers in a normal multilayer perceptron network (MLP) are **fully-connected**: each output value is a combination of all inputs
2. Each full-connected layer is a **full (dense) matrix**

The number of parameters in fully-connected layers is **huge**.

Example Consider a layer with input size = 300 x 300 color pixels, and output size equivalent to 150 x 150 color pixels. How many parameters does this layer have?

1. Images are large: 1 Mexapixel color image = 3 million values
2. Fully-connected layers have huge size

Convolution

DSP deja-vu vibes:

$$y[n] = \sum_k x[n-k]h[k]$$

- Some videos here: a **kernel** with fixed coefficients $h[k]$ is slided over the input $x[n]$ and computes the output as a linear combination of the surrounding input samples

Key points:

- Conolution has been used for ever in signal and image processing for **extracting features** (edges, frequency bands, etc)
- Convolution is a kind of matrix multiplication, with an almost sparse matrix of a special form ("circulant" or "Toeplitz")
- Each output value depends only on the surrounding pixels

Convolutional Neural Networks

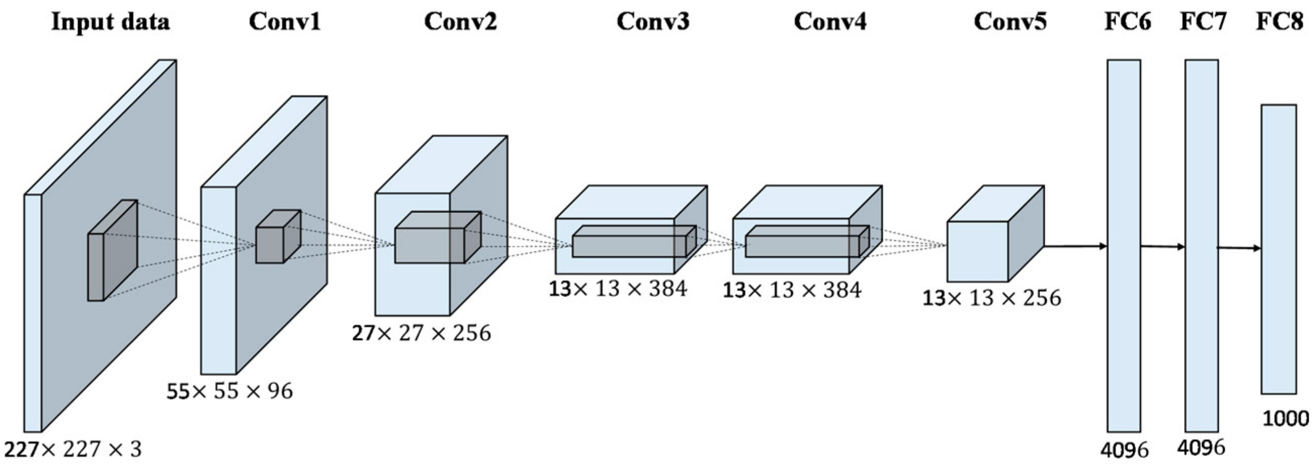
- The basic unit is now **a layer**
- The data is viewed as **tensors**: 3D cubes of data (like a three-dimensional matrix)
- Each layer takes as input an $M_1 \times N_1 \times C_1$ tensor and produces an output $M_2 \times N_2 \times C_2$ tensor
- We don't think of individual neurons anymore. Each neuron in a convolutional layer does exactly the same operation as the others, with the same weights, but "sees" just one small part of the input image

Architecture of a CNN

AlexNet:

[1] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). "ImageNet classification with deep convolutional neural networks" (PDF). Communications of the ACM. 60 (6): 84–90. doi:10.1145/3065386.

ISSN 0001-0782.



Layer types

Open the AlexNet model in Matlab and look at the architecture directly

```
In [ ]: net = alexnet('Weights','imagenet')
        analyzeNetwork(net)   % or double-click `net` in the Workspace
```

Layer types:

- Convolutional
- Activation
- Pooling (MaxPooling)
- Fully connected (e.g. like in multilayer perceptron)
- Softmax activation to get probability-like scores (like in multilayer perceptron)
- Other optional stuff: normalization, dropout, batch normalization etc

A CNN can be viewed as a feature extractor + classificator:

- The output layers are very similar to multilayer perceptron: fully-connected layers + softmax
- The first part, with the convolution layers, is a kind of **feature extractor**

The model

A CNN has a fixed arcihtecture composed of several layers. For typical networks, the architecture is described in the accompanying papers.

Inputs:

- a color image represented as a tensor X of size $L_1 \times L_2 \times 3$

Outputs (assuming one-hot encoding):

- a vector $\hat{y}$ which should be understood as scores/probability of belonging in each class
- the **location of the maximum** value gives the predicted class

The model parameters

The model parameters are the parameters of the layers:

- the filter coefficients in the convolutional layers
- the weights in the fully-connected layers

  The number of parameters of the convolutional layers is **much smaller** than for fully-connected layers.

The cost function

For classification, the cross-entropy is typically used. This is exactly similar to how it is used in MLP networks.

For a single input:

$$L(y, \hat{y}) = -y_1 \log \hat{y_1} - \dots - y_n \log \hat{y_n} = -\log \hat{y_{class}},$$

where $\hat{y_{class}}$ is the model's predicted probability for the true class of the input.

For multiple inputs: do the average of all

$$J = \frac{1}{N} \sum_i L(y^i, \hat{y}^i)$$

Training

Training is done with **backpropagation** and gradient descent (or some variant of it).

Backpropagation = the technique to compute the derivatives of J with respect to all parameters in the network.

Same story as for multilayer perceptron (MLP) networks

Matlab functions for working with CNNs

Lots of new functions, there is a massive push in this direction in the last 2-3 years. Search the docs, there are many nice tutorials.

Other frameworks besides Matlab

The most used deep learning frameworks are written in Python:

- Tersorflow + Keras
- Pytorch
- ...

3. Practical work

Reference: https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html (https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html)

1. Load the AlexNet model with pretrained weights on the ImageNet database. Open the model and examine the architecure.

   - How many parameters does the first convolutional layer have?
   - How many parameters does each trainable layer have? (trainable = convolutional or fully-connected)
   - What is the share of the fully-connected layers in the total number of parameters?

1. Play with AlexNet. Download an image from the Internet and classify it. Does it work?

   The ImageNet class names can be found here: https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a (https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a)
2. Perform **Transfer Learning** with AlexNet following the tutorial in here: https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html (https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html)
3. Use a different model:

   - Check out the available pretrained CNN models in Matlab: https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html (https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html)
   - Try one of the smaller, but more accurate ones: GoogLeNet, Resnet-18, Mobile-net v2

1. Compare other networks (GoogLeNet, Resnet-18, Mobile-net v2) with AlexNet:

   - do they have more or less layers?
   - do they have smaller or bigger fully-connected layers a the end?

1. Add one extra convolutional layer in the middle part of AlexNet. Train the network (warning: it may take a long long time)

4. Final questions

TBD