USING DEEP BELIEF NETWORK IN OBJECT RECOGNITION

(Milan Halabuk)
Objectives

- Implement DBN
- Generate dataset of input images
- Test performance of DBN in object classification tasks
- Find optimal topology
What is Deep Belief Network?

- DEEP BELIEF NETWORK [1]
- Unsupervised Greedy-Layer Wise Pre-Training
- Supervised fine tuning – add final layer of classification neurons
Advantages and disadvantages

+ Efficient usage of hidden layers (higher performance gain by adding layers compared to Multilayer perceptron) [7]

+ Robustness in classification (size, position, color, view angle – rotation)

- Hardware requirements (?)
Recent success of DBN

- Object classification
- Alex Krizhevsky achieved with CIFAR-10 data classification success of 78.9% [2] (convolutional DBN)
- Convolutional multilayer perceptron achieved 58% [3]

Example of CIFAR-10 data
Implementation

- Based on lectures from deeplearning.net
- Python
- Using fast Theano library
Theano

- Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

- Dynamic C code generation [8]
Creating Dataset

- Using webGL (based on OpenGL ES 2.0)
- Various scale, rotation (3D) and color
- Generate greyscale images (perspective view)
Dataset examples - CROSS
Dataset examples - SQUARE
Dataset examples - TRIANGLES
Dataset specification

- 3 classes – SQUARE, TRIANGLE, CROSS
- 14526 input images (4752 for each class)
- Input image resolution: 32x32 pixels
- Size variation: 7 – 20 px
- Brightness (color) variation: 0.5 – 1
- View angle (rotation)
  - Z → small random value (step 4-5 deg.)
  - X,Y → random value (range <-60, 60> deg.)
Net performance (1)

- TOPOLOGY: 1024 – 500 – 500 – 500 – 3

  Train data: 10000 (random selection)
  Validation data: 2000 (random selection)
  Test data: 2256 (random selection)

  Number of pretraining epoch (for each hid. Layer): 100 (2844 sec.)

  Number of finetuning epoch: 44 (334 sec.)

- Validation error after pretraining: 34.35%
- Validation error after finetuning: 0.4%
- Test error after finetuning: 0.57% (13 from 2256)
Net performance (2)

Validation error during fine-tuning

Error in %

Epoch
Finding optimal topology

- Topology: 1024-50-50-50-3
  - Validation error after pretraining: 54.05%
  - Test error after finetuning: 2.75%

- Topology: 1024-100-100-100-3
  - Validation error after pretraining: 52.65%
  - Test error after finetuning: 1.69%

- Topology: 1024-200-200-200-3
  - Validation error after pretraining: 33.2%
  - Test error after finetuning: 0.53%

- Topology: 1024-500-500-500-3
  - Validation error after pretraining: 34.35%
  - Test error after finetuning: 0.57%
Conclusion

Questions?

References:
[1] - online: http://deeplearning.net/tutorial/DBN.html, Montreal, Canada
[3] - A. Krizhevsky, Convolutional Neural Networks for Object Classification in CUDA, University of Toronto, 2010