USING DEEP BELIEF NETWORK IN OBJECT RECOGNITION

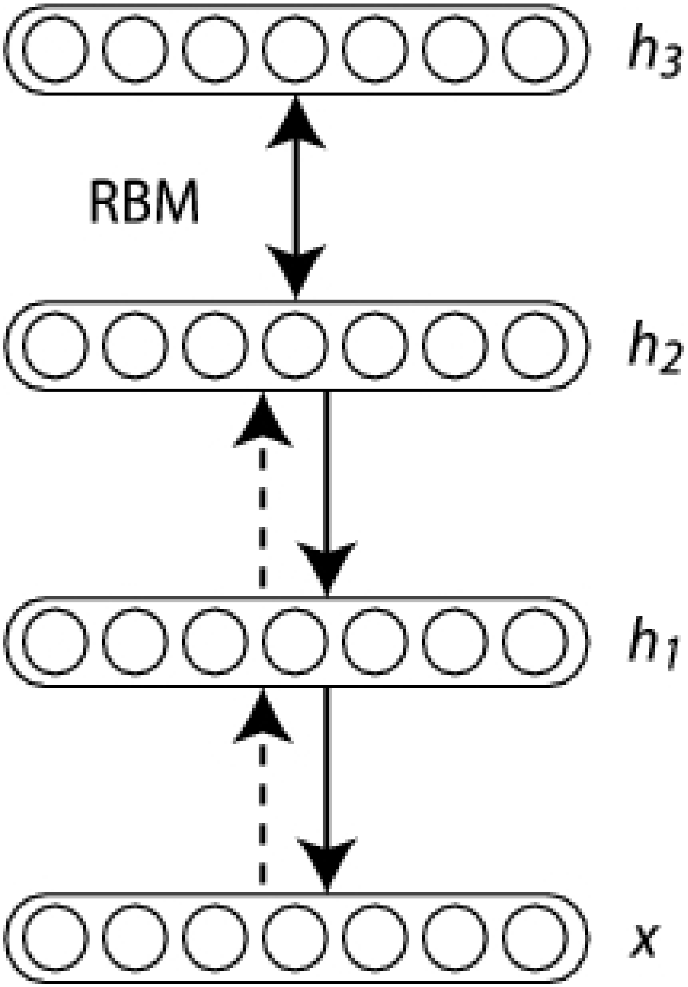
(Milan Halabuk)

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 tunning – add final layer of classification neurons



Advantages and disadvantages

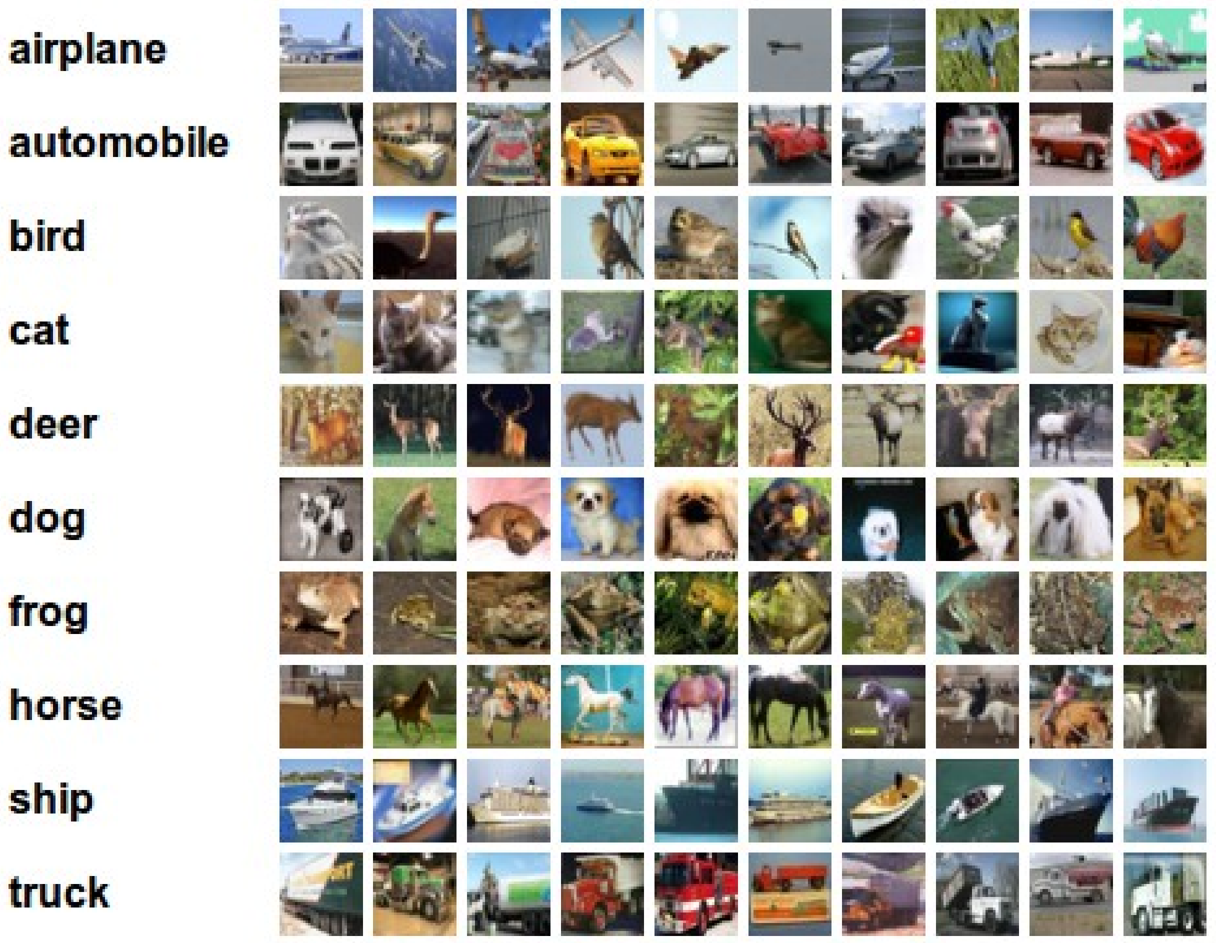
+ Efficient usage of hidden layers (higher performance gain by adding layers compared to Multilayer perceptron) [7]

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

- Hardware requirements (?)

Recent succes of DBN

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



Exampel 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]

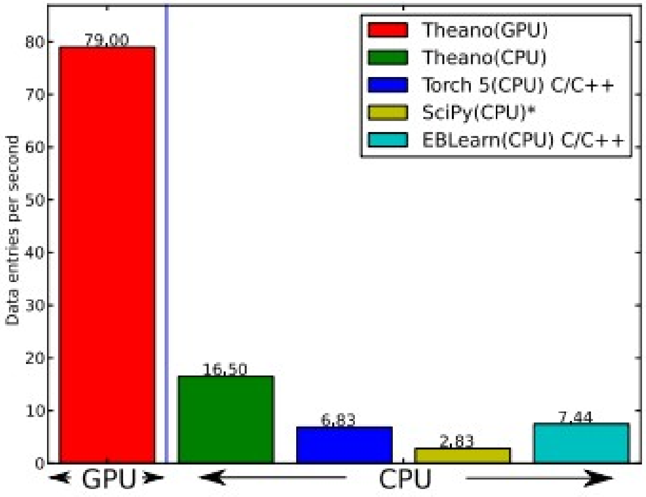
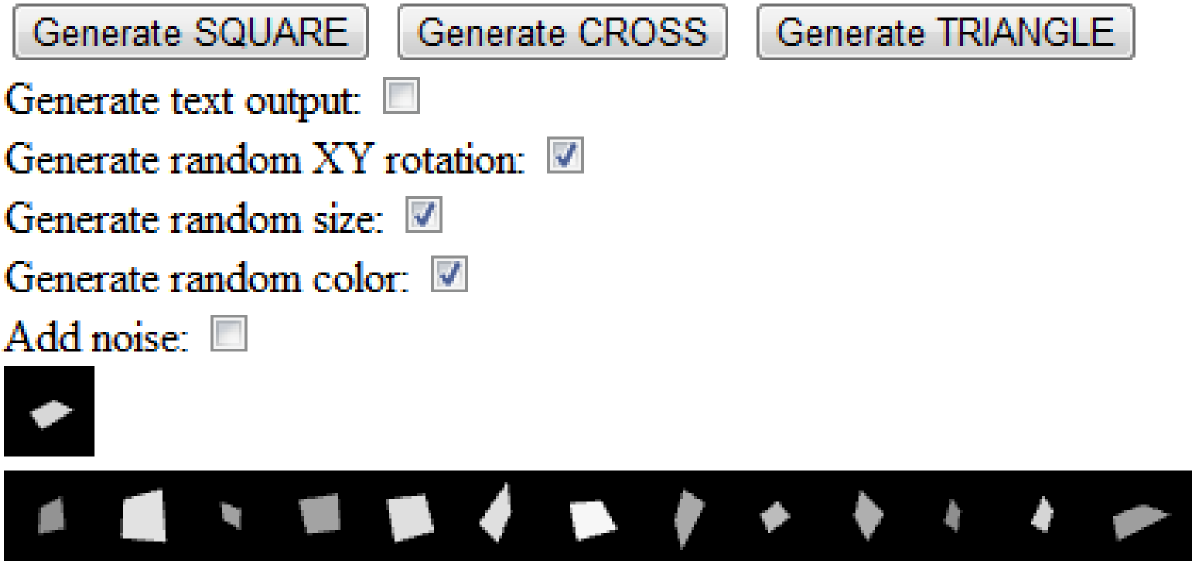


Figure 6: *Fitting a convolutional network using different software. The benchmark stresses convolutions of medium-sized (256 by 256) images with small (7 by 7) filters.*

Creating Dataset

- Using webGL (based on OpenGL ES 2.0)
- Various scale, rotation (3D) and color
- Generate greyscale images (perspective view)



Generate text output: ☐

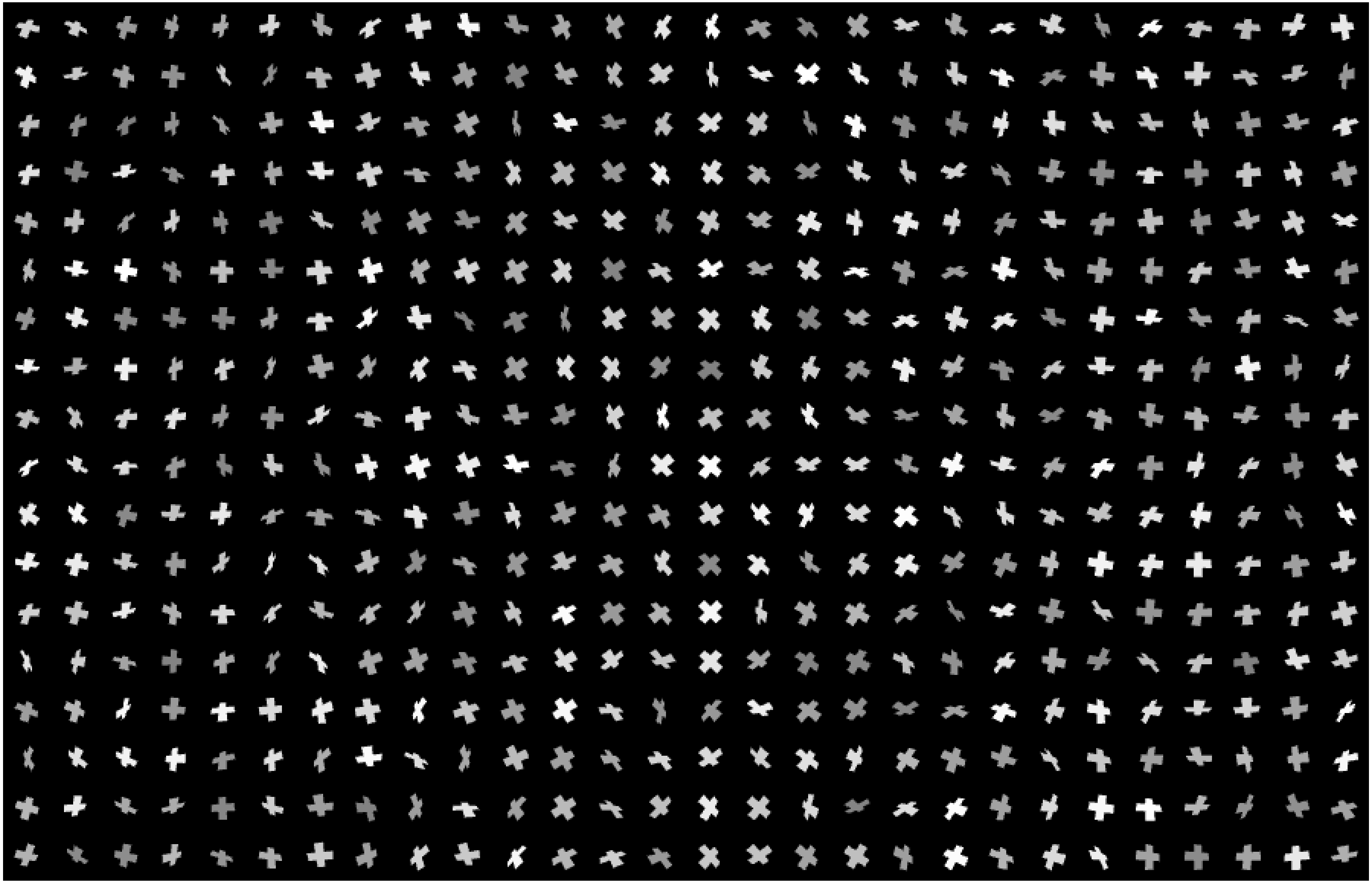
Generate random XY rotation: ☑

Generate random size: ☑

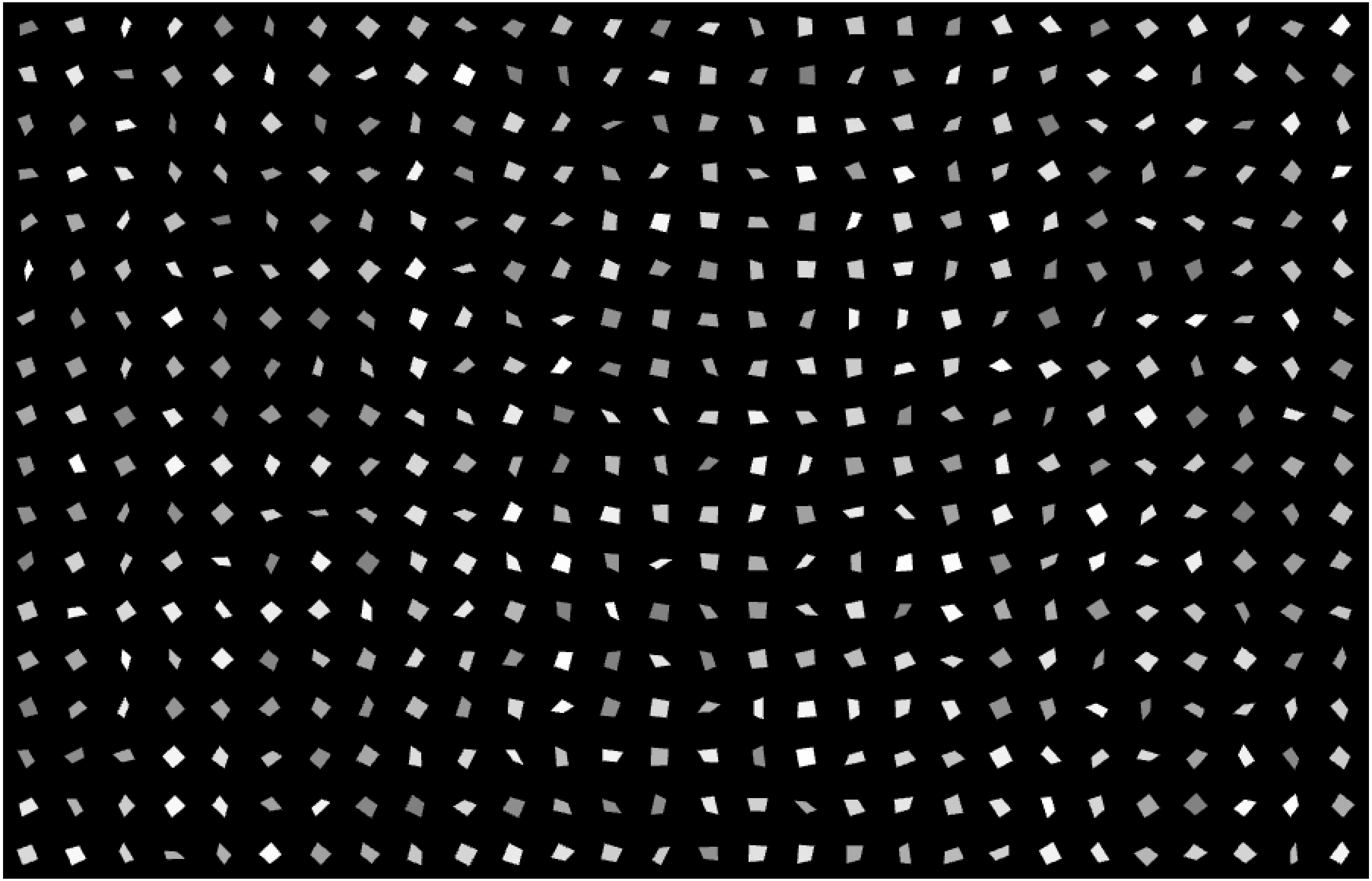
Generate random color: ☑

Add noise: ☐

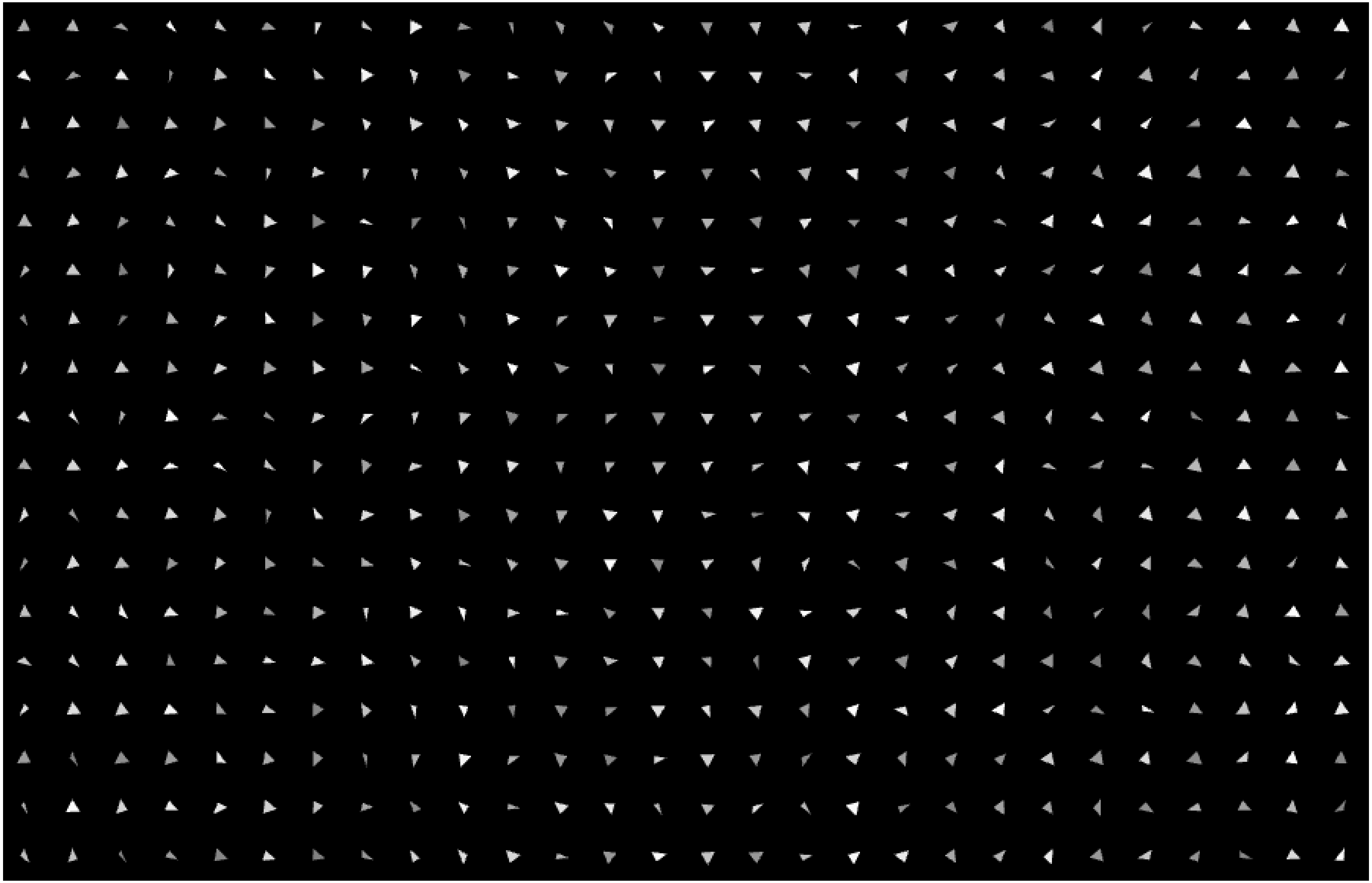
Dataset examples - CROSS



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 angel (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

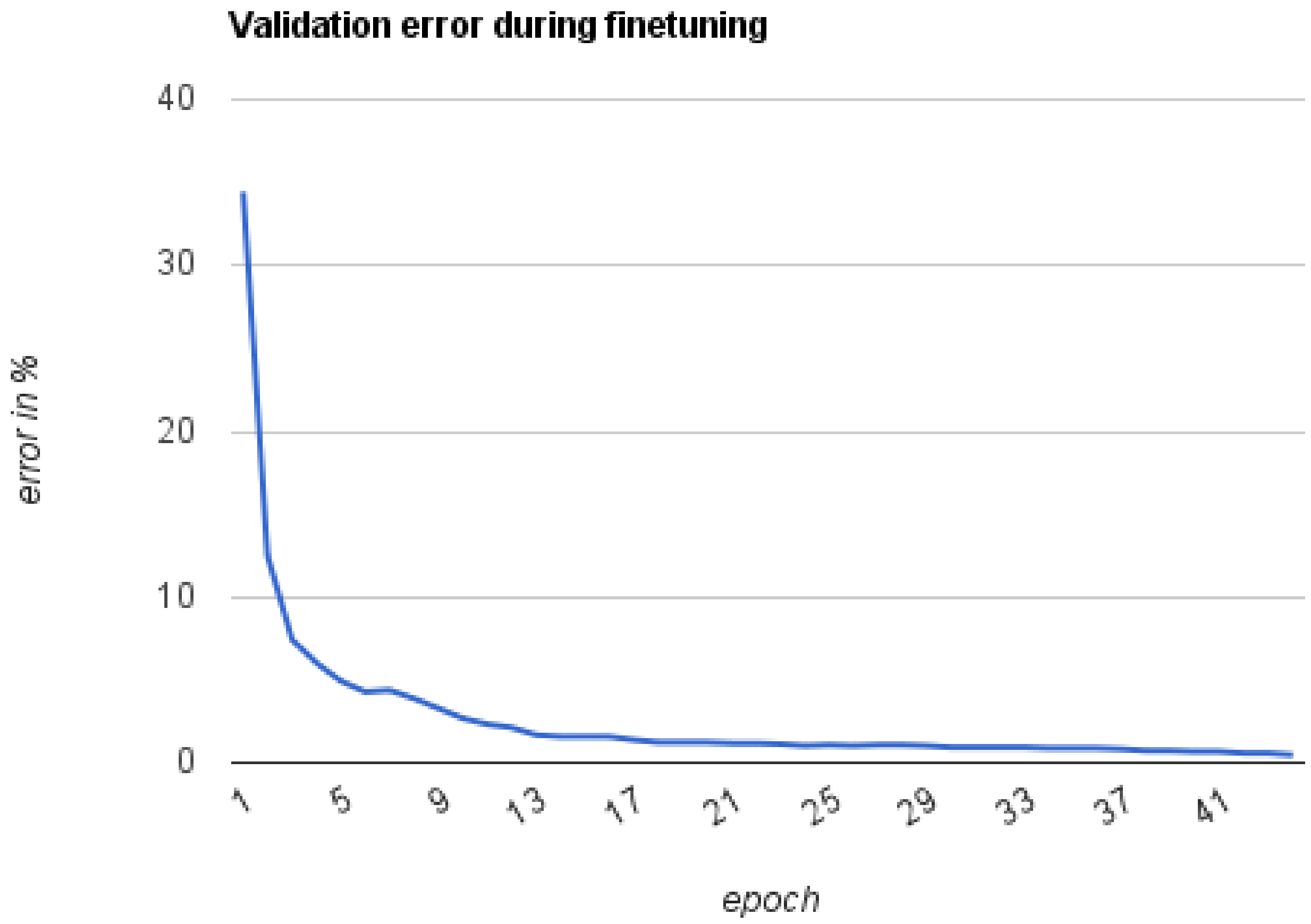
  Train data: 10000 (random selection)
  Validation data: 2000 (random selection)
  Test data: 2256 (random selection)

  Number of pretrainich 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)



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

[2] - A. Krizhevsky, CONVOLUTIONAL DEEP BELIEF NETWORKS ON CIFAR-10, University of Toronto, 2010

[3] - A. Krizhevsky, Convolutional Neural Networks for Object Classifcation in CUDA, University of Toronto, 2010

[4] - I. Arel, D. C. Rose, T. P. Karnowski, DEEP MACHINE LEARNING - A NEW FRONTIER IN ARTIFICIAL INTELLIGENCE RESEARCH, The University of Tennessee, USA

[5] - G. E. Hinton, R. R. Salakhutdino, REDUCING THE DIMENSIONALITY OF DATA WITH NEURAL NETWORK, 2006

[6] – G. E. Hinton ,A PRACTICAL GUIDE TO TRAINING RESTRICTED BOLTZMANN MACHINES, 2010, University of Toronto

[7] – G. E. Hinton ,DEEP BELIEF NETWORKS, sept. 2009, University of Toronto Online: http://videolectures.net/mlss09uk_hinton_dbn/

[8] - James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, Yoshua BengioTheano: A CPU and GPU Math Compiler in Python, 2010, PROC. OF THE 9th PYTHON IN SCIENCE CONF.