Prazo limite para entrega (grupos de até 4 alunos): 13/04/2022

Enviar o arquivo **pdf** do trabalho E códigos-fonte em Matlab (Python, R ou outra linguagem) compactados todos juntos em arquivo **.zip** por email para: leandro.coelho@ufpr.br **ASSUNTO**: TE941 - Work 02

Work 02: Convolutional neural networks (CNNs)

Leandro dos Santos Coelho

Pontifícia Universidade Católica do Paraná (PUCPR), Escola Politécnica

Pós-Graduação em Engenharia de Produção e Sistemas (PPGEPS), Graduação em Engenharia de Controle e Automação (Mecatrônica)

Rua Imaculada Conceição, 1155, CEP 80215-901, Curitiba, PR, Brasil

Universidade Federal do Paraná (UFPR), Graduação e Pós-Graduação em Engenharia Elétrica, Campus Centro Politécnico

Av. Cel. Francisco H. dos Santos, 100, CEP 81530-000, Curitiba, PR, Brasil

e-mail: **leandro.coelho@pucpr.br; lscoelho2009@gmail.com; leandro.coelho@ufpr.br**

Currículo Lattes: *http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4792095Y4*

Google Scholar: *https://scholar.google.com/citations?user=0X7VkC4AAAAJ&hl=pt-PT*

Linkedin: *https://www.linkedin.com/in/leandro-dos-santos-coelho-07a08893/*

Prazo limite para entrega (grupos de até 4 alunos): 13/04/2022
Enviar o arquivo **pdf** do trabalho E códigos-fonte em Matlab (Python, R ou outra linguagem) compactados todos juntos em arquivo **.zip** por email para: leandro.coelho@ufpr.br **ASSUNTO**: TE941 - Work 02

Work 02: 2 tasks

✓ Part 01: *Theoretical*

✓ Part 02: *Practical*

Part 01: *Theoretical*

General

1. What is deep learning?

Convolutional neural networks (CNNs)

2. Why do we prefer CNNs over shallow artificial neural networks for image data?
3. Explain the role of the convolution layer in a CNN design.
4. What is the role of the fully connected (FC) layer in CNN?
5. Why do we use a pooling layer in a CNN?
6. Explain the characteristics of the following pooling approaches: max pooling, average pooling, and sum pooling.

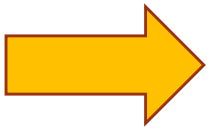
Part 02: *Practical*

❶ Test 5 **setups** for the CNNs with focus in the accuracy improve values (during training and test phases) for **20** (or more) learning epochs.

❷ What was the best setup in terms of performance metrics (accuracy or MSE, Mean Squared Error) to the evaluated CNNs the the **MNIST and MNIST-like fashion product database** ?

5% **(or more)**

Full databases
- 60,000 training examples
- 10,000 testing examples
- 10 classes



Setup in both databases
- **3,000** training examples
- **500** testing examples
- 10 classes

Dataset: MNIST

The MNIST database of handwritten digits, available from this page, has a training set of **60,000 examples**, and a test set of **10,000 examples**. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:
train-images-idx3-ubyte.gz: training set images (9912422 bytes)
train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

http://yann.lecun.com/exdb/mnist/

Dataset: MNIST

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

Size-normalized examples from the MNIST database.

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

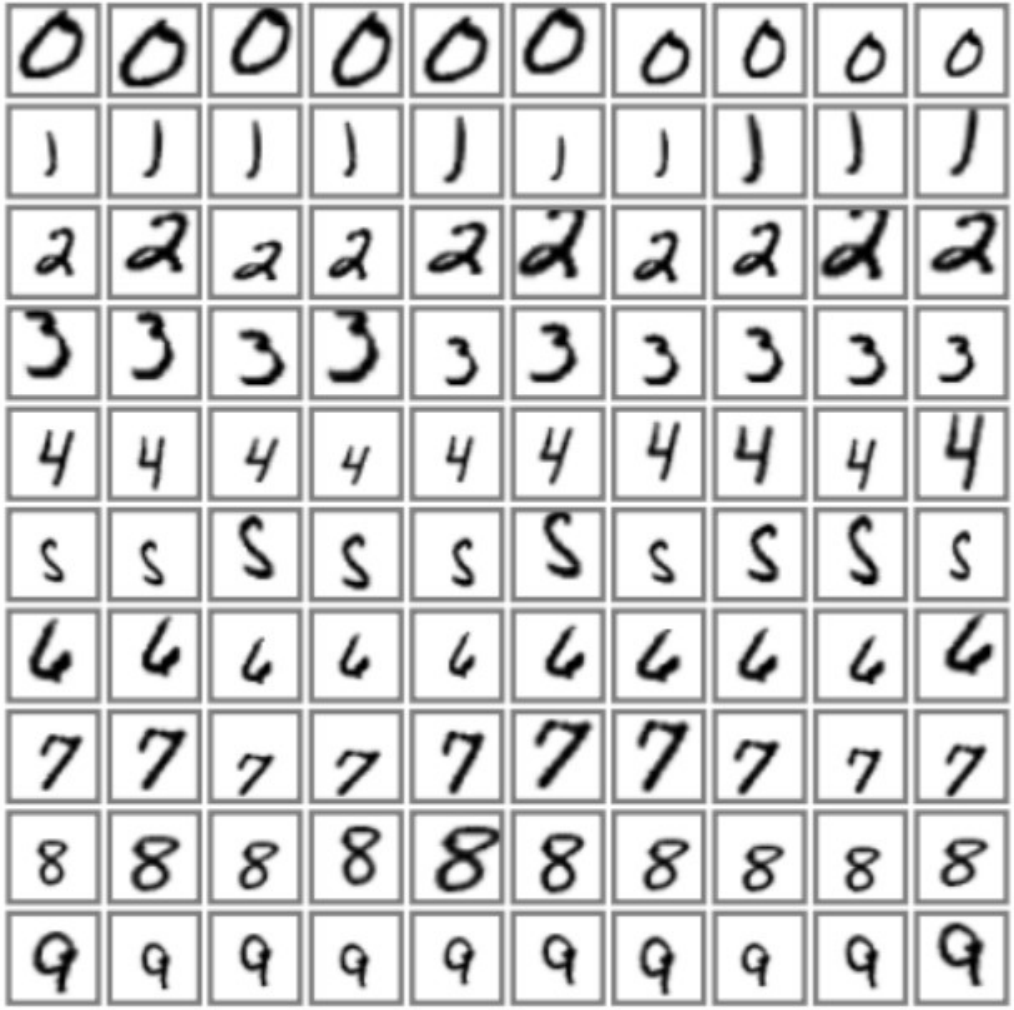
Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

Dataset: MNIST

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond



Examples of distortions of ten training patterns.

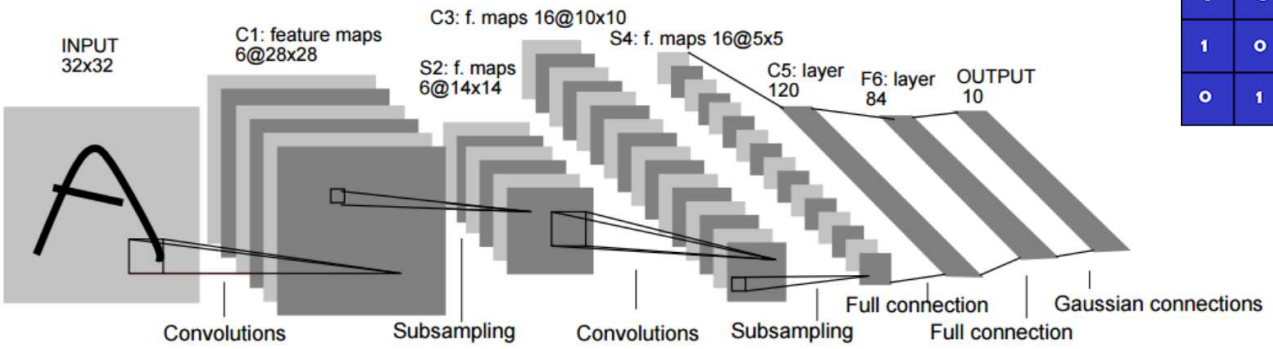
PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

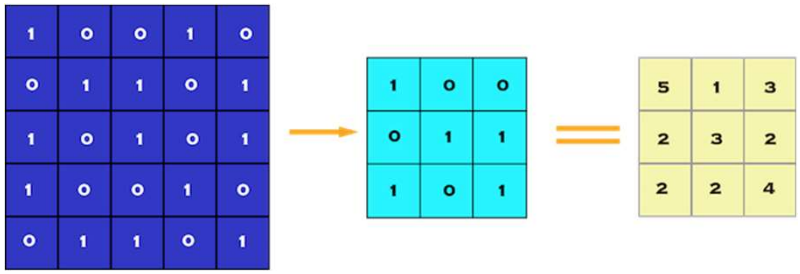
Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Dataset: MNIST

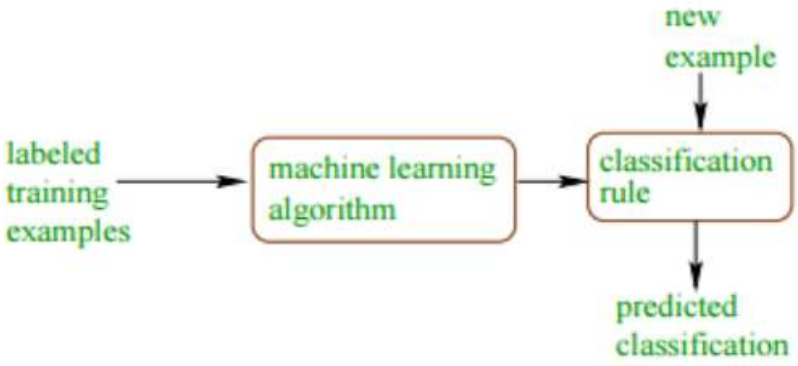


Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.



LeNet-5

The 82 test patterns misclassified by LeNet-5. Below each image is displayed the correct answers (left) and the network answer (right). These errors are mostly caused either by genuinely ambiguous patterns, or by digits written in a style that are underrepresented in the training set.



THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

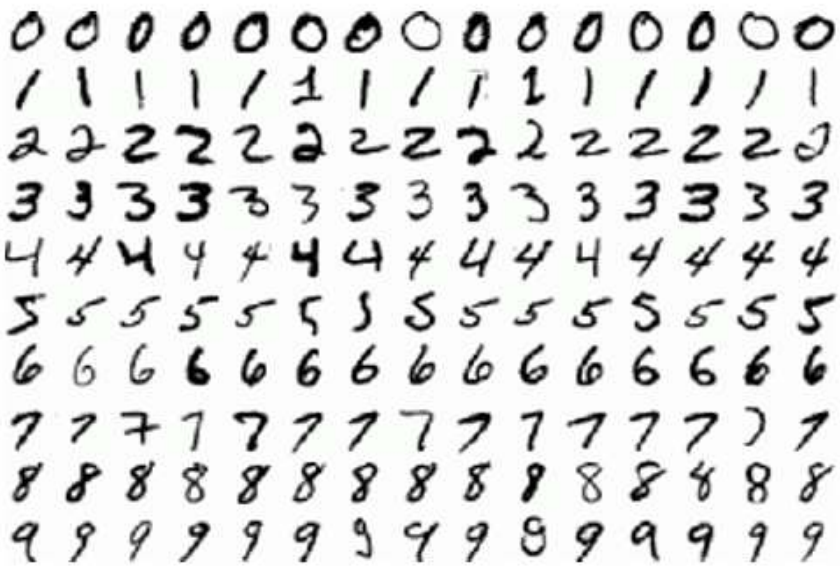
PROC. OF THE IEEE, NOVEMBER 1998

Gradient-Based Learning Applied to Document Recognition

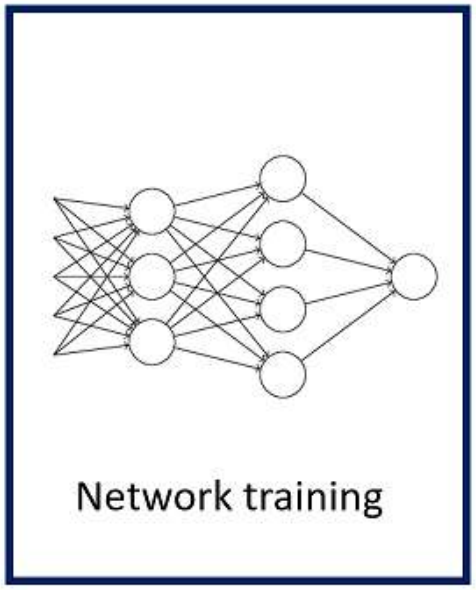
Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

Dataset: MNIST



Data & Labels



0
1
2
3
4
5
6
7
8
9

https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d

Dataset: MNIST

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)	Reference
Linear Classifiers			
linear classifier (1-layer NN)	none	12.0	LeCun et al. 1998
linear classifier (1-layer NN)	deskewing	8.4	LeCun et al. 1998
pairwise linear classifier	deskewing	7.6	LeCun et al. 1998
K-Nearest Neighbors			
K-nearest-neighbors, Euclidean (L2)	none	5.0	LeCun et al. 1998
K-nearest-neighbors, Euclidean (L2)	none	3.09	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	none	2.83	Kenneth Wilder, U. Chicago
K-nearest-neighbors, Euclidean (L2)	deskewing	2.4	LeCun et al. 1998
K-nearest-neighbors, Euclidean (L2)	deskewing, noise removal, blurring	1.80	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	deskewing, noise removal, blurring	1.73	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	deskewing, noise removal, blurring, 1 pixel shift	1.33	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	deskewing, noise removal, blurring, 2 pixel shift	1.22	Kenneth Wilder, U. Chicago

http://yann.lecun.com/exdb/mnist/

Dataset: MNIST

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

Boosted Stumps			
boosted stumps	none	7.7	Kegl et al., ICML 2009
products of boosted stumps (3 terms)	none	1.26	Kegl et al., ICML 2009
boosted trees (17 leaves)	none	1.53	Kegl et al., ICML 2009
stumps on Haar features	Haar features	1.02	Kegl et al., ICML 2009
product of stumps on Haar f.	Haar features	0.87	Kegl et al., ICML 2009
Non-Linear Classifiers			
40 PCA + quadratic classifier	none	3.3	LeCun et al. 1998
1000 RBF + linear classifier	none	3.6	LeCun et al. 1998
SVMs			
SVM, Gaussian Kernel	none	1.4	
SVM deg 4 polynomial	deskewing	1.1	LeCun et al. 1998
Reduced Set SVM deg 5 polynomial	deskewing	1.0	LeCun et al. 1998
Virtual SVM deg-9 poly [distortions]	none	0.8	LeCun et al. 1998
Virtual SVM, deg-9 poly, 1-pixel jittered	none	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 1-pixel jittered	deskewing	0.68	DeCoste and Scholkopf, MLJ 2002
Virtual SVM, deg-9 poly, 2-pixel jittered	deskewing	0.56	DeCoste and Scholkopf, MLJ 2002

http://yann.lecun.com/exdb/mnist/

Dataset: MNIST

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

Neural Nets			
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
2-layer NN, 300 HU, MSE, [distortions]	none	3.6	LeCun et al. 1998
2-layer NN, 300 HU	deskewing	1.6	LeCun et al. 1998
2-layer NN, 1000 hidden units	none	4.5	LeCun et al. 1998
2-layer NN, 1000 HU, [distortions]	none	3.8	LeCun et al. 1998
3-layer NN, 300+100 hidden units	none	3.05	LeCun et al. 1998
3-layer NN, 300+100 HU [distortions]	none	2.5	LeCun et al. 1998
3-layer NN, 500+150 hidden units	none	2.95	LeCun et al. 1998
3-layer NN, 500+150 HU [distortions]	none	2.45	LeCun et al. 1998
3-layer NN, 500+300 HU, softmax, cross entropy, weight decay	none	1.53	Hinton, unpublished, 2005
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et al., ICDAR 2003
2-layer NN, 800 HU, cross-entropy [affine distortions]	none	1.1	Simard et al., ICDAR 2003
2-layer NN, 800 HU, MSE [elastic distortions]	none	0.9	Simard et al., ICDAR 2003
2-layer NN, 800 HU, cross-entropy [elastic distortions]	none	0.7	Simard et al., ICDAR 2003
NN, 784-500-500-2000-30 + nearest neighbor, RBM + NCA training [no distortions]	none	1.0	Salakhutdinov and Hinton, AI-Stats 2007
6-layer NN 784-2500-2000-1500-1000-500-10 (on GPU) [elastic distortions]	none	0.35	Ciresan et al. Neural Computation 10, 2010 and arXiv 1003.0358, 2010
committee of 25 NN 784-800-10 [elastic distortions]	width normalization, deslanting	0.39	Meier et al. ICDAR 2011
deep convex net, unsup pre-training [no distortions]	none	0.83	Deng et al. Interspeech 2010

http://yann.lecun.com/exdb/mnist/

Dataset: MNIST

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

Convolutional nets			
Convolutional net LeNet-1	subsampling to 16x16 pixels	1.7	LeCun et al. 1998
Convolutional net LeNet-4	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with K-NN instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-4 with local learning instead of last layer	none	1.1	LeCun et al. 1998
Convolutional net LeNet-5, [no distortions]	none	0.95	LeCun et al. 1998
Convolutional net LeNet-5, [huge distortions]	none	0.85	LeCun et al. 1998
Convolutional net LeNet-5, [distortions]	none	0.8	LeCun et al. 1998
Convolutional net Boosted LeNet-4, [distortions]	none	0.7	LeCun et al. 1998
Trainable feature extractor + SVMs [no distortions]	none	0.83	Lauer et al., Pattern Recognition 40-6, 2007
Trainable feature extractor + SVMs [elastic distortions]	none	0.56	Lauer et al., Pattern Recognition 40-6, 2007
Trainable feature extractor + SVMs [affine distortions]	none	0.54	Lauer et al., Pattern Recognition 40-6, 2007
unsupervised sparse features + SVM, [no distortions]	none	0.59	Labusch et al., IEEE TNN 2008
Convolutional net, cross-entropy [affine distortions]	none	0.6	Simard et al., ICDAR 2003
Convolutional net, cross-entropy [elastic distortions]	none	0.4	Simard et al., ICDAR 2003
large conv. net, random features [no distortions]	none	0.89	Ranzato et al., CVPR 2007
large conv. net, unsup features [no distortions]	none	0.62	Ranzato et al., CVPR 2007
large conv. net, unsup pretraining [no distortions]	none	0.60	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [elastic distortions]	none	0.39	Ranzato et al., NIPS 2006
large conv. net, unsup pretraining [no distortions]	none	0.53	Jarrett et al., ICCV 2009
large/deep conv. net, 1-20-40-60-80-100-120-120-10 [elastic distortions]	none	0.35	Ciresan et al. IJCAI 2011
committee of 7 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.27 +-0.02	Ciresan et al. ICDAR 2011
committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions]	width normalization	0.23	Ciresan et al. CVPR 2012

http://yann.lecun.com/exdb/mnist/

Best result (error rate 0.23%)

CVPR '12
Providence, Rhode Island
June 16-21, 2012

Multi-column Deep Neural Networks for Image Classification

Dan Cireşan, Ueli Meier, Juergen Schmidhuber

Traditional methods of computer vision and machine learning cannot match human performance n tasks such as the recognition of handwritten digits or traffic signs. Our biologically plausible deep artificial neural network architectures can. Small (often minimal) receptive fields of convolutional winner-take-all neurons yield large network depth, resulting in roughly as many sparsely connected neural layers as found in mammals between retina and visual cortex. Only winner neurons are trained. Several deep neural columns become experts on inputs preprocessed in different ways; their predictions are averaged. Graphics cards allow for fast training. On the very competitive MNIST handwriting benchmark, our method is the first to achieve near-human performance. On a traffic sign recognition benchmark it outperforms humans by a factor of two. We also improve the state-of-the-art on a plethora of common image classification benchmarks.

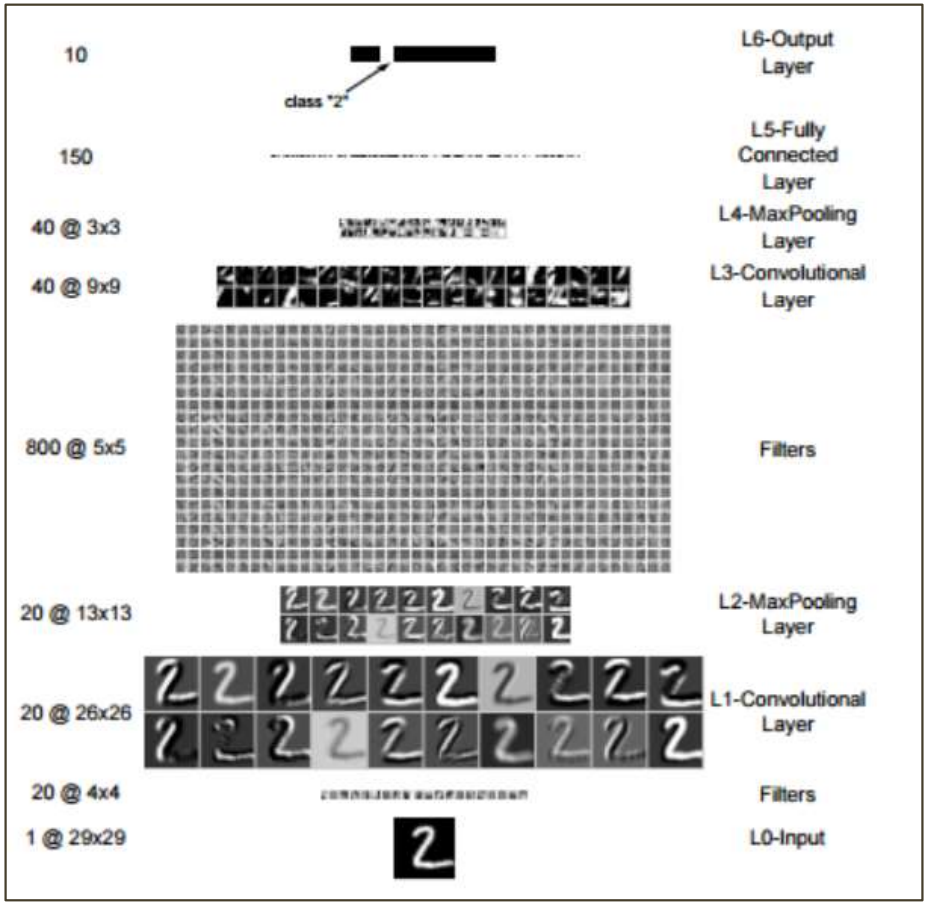
Cornell University Library
arXiv.org > cs > arXiv:1202.2745

pdf *https://arxiv.org/abs/1202.2745*

THE MNIST DATABASE

of handwritten digits

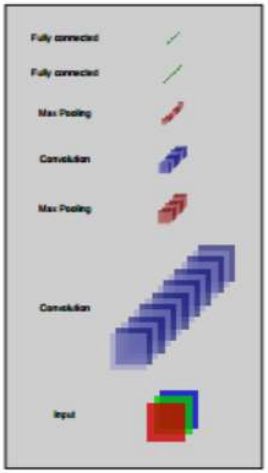
Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond



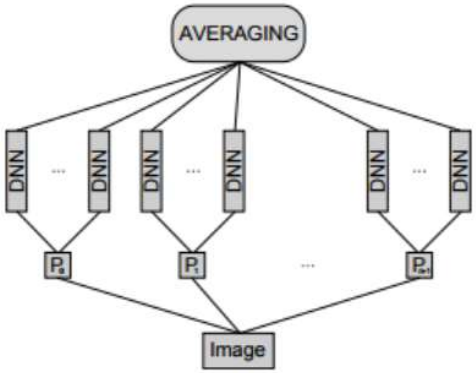
Best result (error rate 0.23%)

Table 1: Test error rate [%] of the 35 NNs trained on MNIST. Wxx - width of the character is normalized to xx pixels

Trial	W10	W12	W14	W16	W18	W20	ORIGINAL
1	0.49	0.39	0.40	0.40	0.39	0.36	0.52
2	0.48	0.45	0.45	0.39	0.50	0.41	0.44
3	0.59	0.51	0.41	0.41	0.38	0.43	0.40
4	0.55	0.44	0.42	0.43	0.39	0.50	0.53
5	0.51	0.39	0.48	0.40	0.36	0.29	0.46
avg.	0.52±0.05	0.44±0.05	0.43±0.03	0.40±0.02	0.40±0.06	0.39±0.08	0.47±0.05
	35-net average error: 0.44±0.06						
5 columns MCDNN	0.37	0.26	0.32	0.33	0.31	0.26	0.46
	35-net MCDNN: **0.23%**						



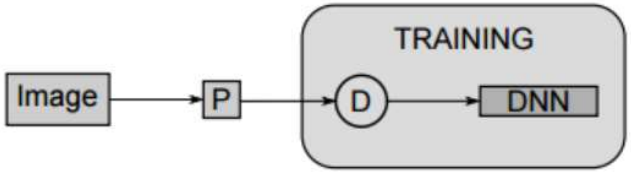
(a)



(b)

Table 2: Results on MNIST dataset.

Method	Paper	Error rate[%]
CNN	[35]	0.40
CNN	[28]	0.39
MLP	[5]	0.35
CNN committee	[6]	0.27
MCDNN	this	**0.23**



How are the MCDNN errors affected by the number of preprocessors? We train 5 DNNs on all 7 datasets. A MCDNN 'y out-of-7' (y from 1 to 7) averages $5y$ nets trained on y datasets. Table 3 shows that more preprocessing results in lower MCDNN error.

We also train 5 DNN for each odd normalization, i.e. W11, W13, W15, W17 and W19. The 60-net MCDNN performs (0.24%) similarly to the 35-net MCDNN, indicating that additional preprocessing does not further improve recognition.

We conclude that MCDNN outperform DNN trained on the same data, and that different preprocessors further decrease the error rate.

pdf *https://arxiv.org/abs/1202.2745*

Best result (error rate 0.21%)

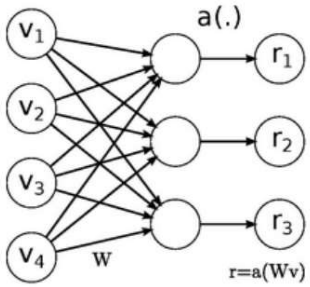
Regularization of Neural Networks using DropConnect

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, Rob Fergus

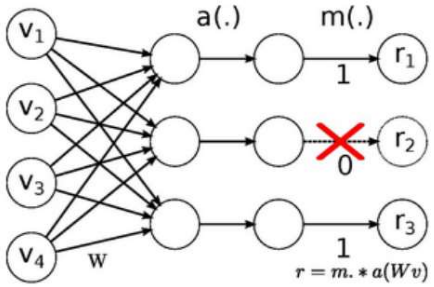
Dept. of Computer Science, Courant Institute of Mathematical Science, New York University

International Conference on Machine Learning 2013

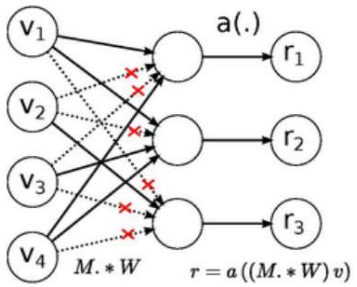
ICML | Atlanta
International Conference on Machine Learning
16-21 JUNE 2013 ATLANTA



No-Drop Network



DropOut Network



DropConnect Network

Image Classification Error(%) of DropConnect v.s. Dropout

DataSet	DropConnect	Dropout	Previous best result(2013)
MNIST	**0.21**	0.27	0.23
CIFAR-10	**9.32**	9.83	9.55
SVHN	**1.94**	1.96	2.80
NORB-full-2fold	3.23	**3.03**	3.36

pdf *http://cs.nyu.edu/~wanli/dropc/*

Dataset: MNIST-like fashion product database

https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/

https://github.com/zalandoresearch/fashion-mnist

MNIST-like fashion product database

Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms

Han Xiao, Kashif Rasul, Roland Vollgraf

(Submitted on 25 Aug 2017 (v1), last revised 15 Sep 2017 (this version, v2))

We present Fashion-MNIST, a new dataset comprising of 28x28 grayscale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images. Fashion-MNIST is intended to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms, as it shares the same image size, data format and the structure of training and testing splits. The dataset is freely available at this https URL

https://arxiv.org/abs/1708.07747

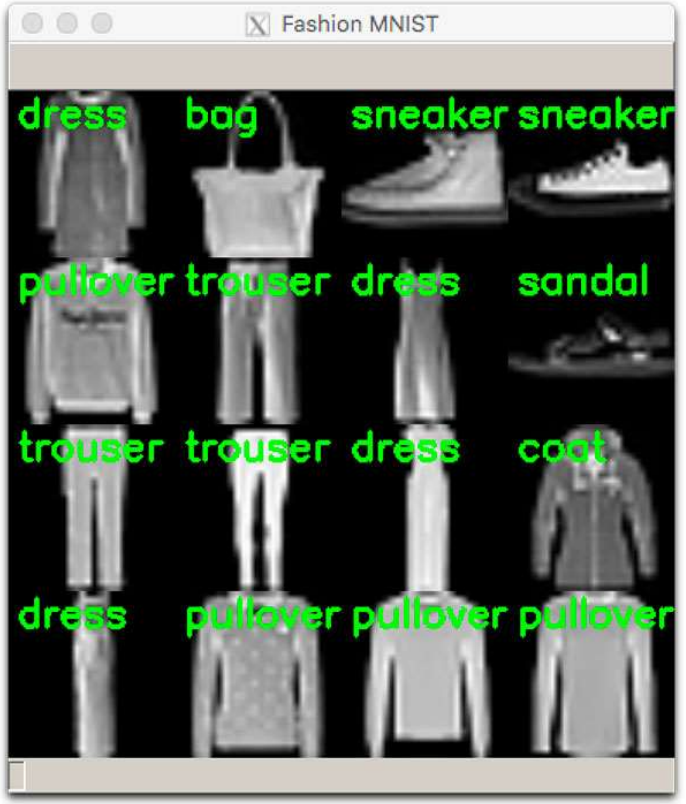
MNIST-like fashion product database

Similar to the MNIST digit dataset, the Fashion MNIST dataset includes:

- 60,000 training examples
- 10,000 testing examples
- 10 classes
- 28×28 grayscale/single channel images

The ten fashion class labels include:

1. T-shirt/top
2. Trouser/pants
3. Pullover shirt
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot



https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/

https://github.com/zalandoresearch/fashion-mnist

MNIST-like fashion product database

https://www.pyimagesearch.com/2019/02/11/fashion-mnist-with-keras-and-deep-learning/

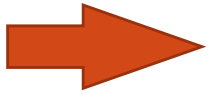
https://github.com/zalandoresearch/fashion-mnist

MNIST-like fashion product database

Loading data with other languages

As one of the Machine Learning community's most popular datasets, MNIST has inspired people to implement loaders in many different languages. You can use these loaders with the `Fashion-MNIST` dataset as well. (Note: may require decompressing first.) To date, we haven't yet tested all of these loaders with Fashion-MNIST.

- C
- C++
- Java
- Python and this and this
- Scala
- Go
- C#
- NodeJS and this
- Swift
- R and this
- Matlab
- Ruby



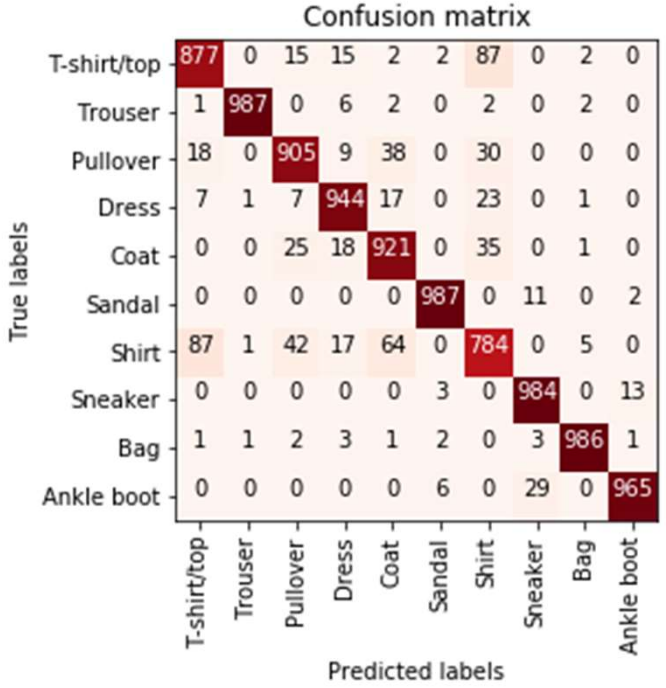
The image and label data is stored in a binary format described on the website. For your convenience, we have provided two MATLAB helper functions for extracting the data. These functions are available at http://ufldl.stanford.edu/wiki/resources/mnistHelper.zip.

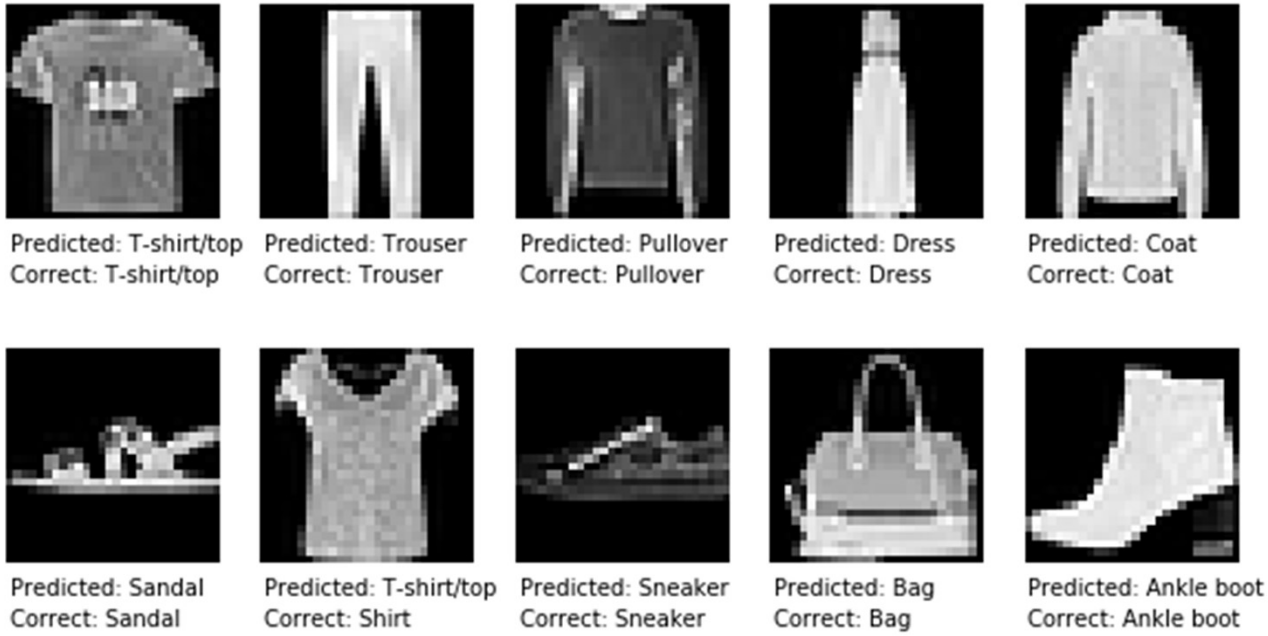
As an example of how to use these functions, you can check the images and labels using the following code:

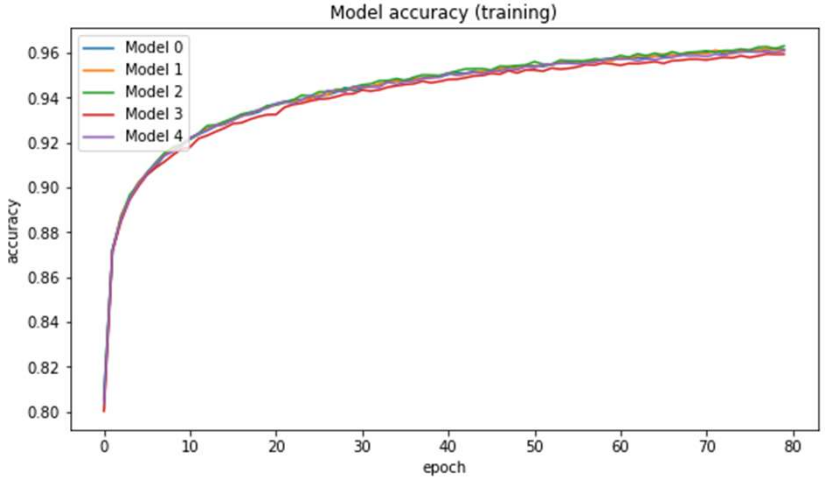
```
% Change the filenames if you've saved the files under different names
% On some platforms, the files might be saved as
% train-images.idx3-ubyte / train-labels.idx1-ubyte
images = loadMNISTImages('train-images-idx3-ubyte');
labels = loadMNISTLabels('train-labels-idx1-ubyte');

% We are using display_network from the autoencoder code
display_network(images(:,1:100)); % Show the first 100 images
disp(labels(1:10));
```

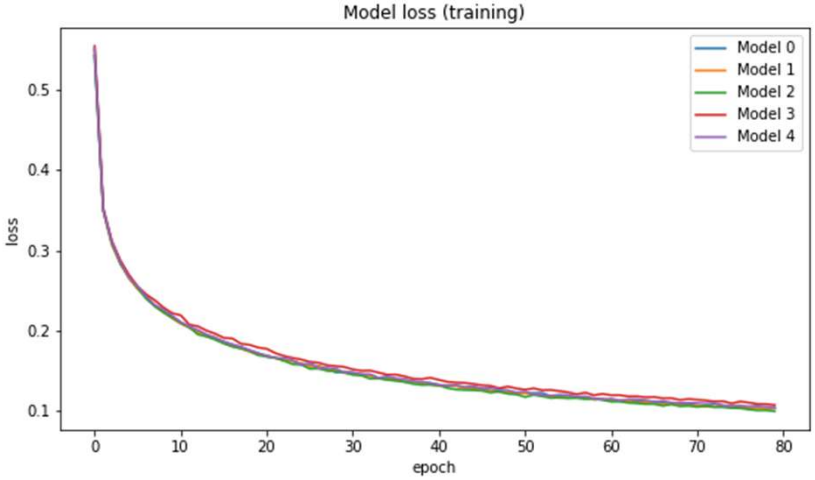

https://github.com/zalandoresearch/fashion-mnist/blob/master/README.md

Some results and codes









https://github.com/cmasch/zalando-fashion-mnist

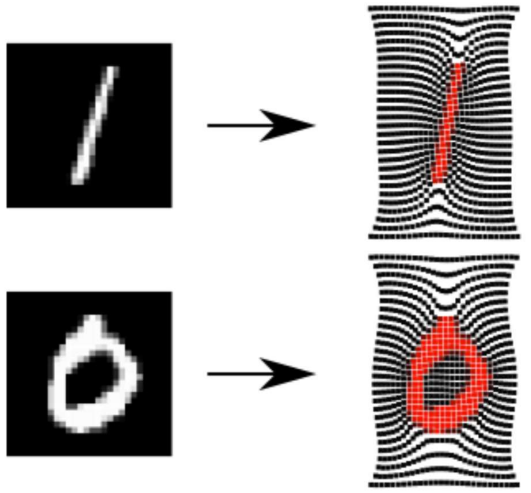
Other dataset: Mechanical-MNIST

Mechanical-MNIST is a benchmark dataset for mechanical meta-models

Each dataset in **the Mechanical MNIST** collection contains the results of 70,000 (60,000 training examples + 10,000 test examples) finite element simulation of a heterogeneous material subject to large deformation. Mechanical MNIST is generated by first converting the MNIST bitmap images (*http://www.pymvpa.org/datadb/mnist.html*) to 2D heterogeneous blocks of material.

Consistent with the MNIST bitmap, the material domain is a 28 x 28 unit square. All simulations are conducted with the FEniCS computing platform (*https://fenicsproject.org*). The code to reproduce these simulations is hosted on GitHub (*https://github.com/elejeune11/Mechanical-MNIST/tree/master/generate_dataset*).

The **paper** "Mechanical MNIST: A benchmark dataset for mechanical metamodels" can be found at *https://doi.org/10.1016/j.eml.2020.100659*.
All code necessary to reproduce the metamodels demonstrated in the manuscript is available on **GitHub** (*https://github.com/elejeune11/Mechanical-MNIST*). For questions, please contact Emma Lejeune (elejeune@bu.edu).



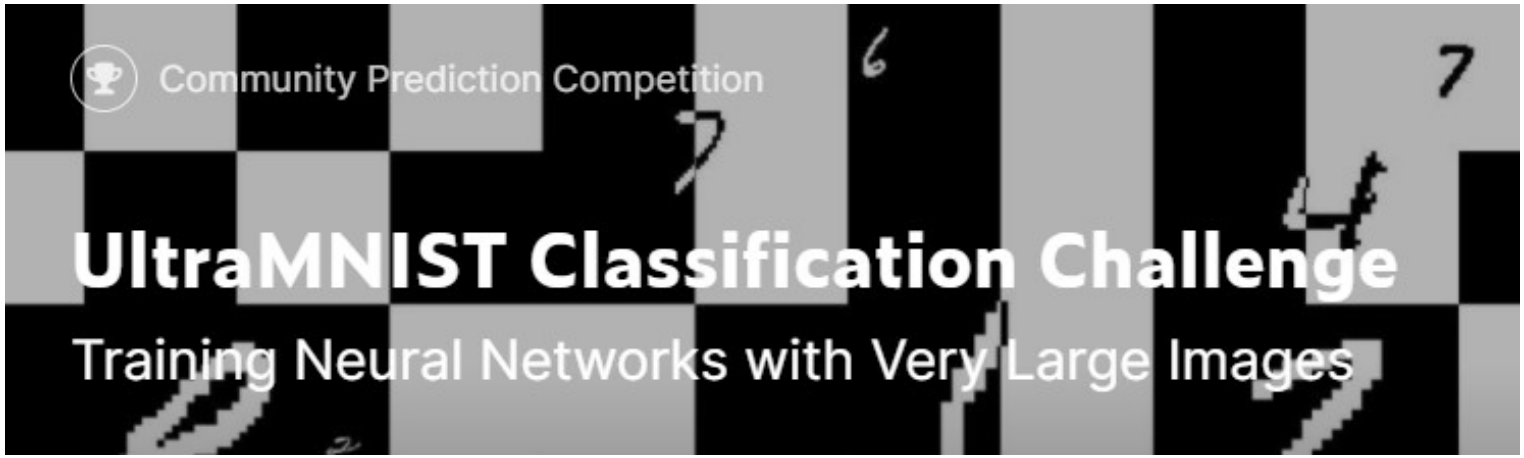
GitHub

https://open.bu.edu/handle/2144/39371

https://github.com/elejeune11/Mechanical-MNIST

Other dataset: UltraMNIST 128 x 128

kaggle



We pose the problem of the classification of UltraMNIST digits. UltraMNIST dataset comprises very large-scale images, each of 4000x4000 pixels with 3-5 digits per image. Each of these digits has been extracted from the original MNIST dataset. Your task is to predict the sum of the digits per image, and this number can be anything from 0 to 27.

https://www.kaggle.com/c/ultra-mnist

Example using Matlab

Deep Learning Toolbox

Deep Learning Toolbox

by Rasmus Berg Palm
24 Sep 2012 (Updated 02 Dec 2015)

Deep Belief Nets, Stacked Autoencoders, Convolutional Neural Nets and more. With examples.

Watch this File

4.4 | 38 ratings
Rate this file

712 Downloads (last 30 days)
File Size: 16 MB
File ID: #38310
Version: 1.2

File Information

Description

DeepLearnToolbox
A Matlab toolbox for Deep Learning.
Deep Learning is a new subfield of machine learning that focuses on learning deep hierarchical models of data. It is inspired by the human brain's apparent deep (layered, hierarchical) architecture. A good overview of the theory of Deep Learning theory is Learning Deep Architectures for AI

For a more informal introduction, see the following videos by Geoffrey Hinton and Andrew Ng.

The Next Generation of Neural Networks (Hinton, 2007)
Recent Developments in Deep Learning (Hinton, 2010)
Unsupervised Feature Learning and Deep Learning (Ng, 2011)
If you use this toolbox in your research please cite:

Prediction as a candidate for learning deep hierarchical models of data (Palm, 2012)

Directories included in the toolbox
NN/ - A library for Feedforward Backpropagation Neural Networks

CNN/ - A library for Convolutional Neural Networks

DBN/ - A library for Deep Belief Networks

SAE/ - A library for Stacked Auto-Encoders

CAE/ - A library for Convolutional Auto-Encoders

util/ - Utility functions used by the libraries

data/ - Data used by the examples

tests/ - unit tests to verify toolbox is working

https://github.com/rasmusbergpalm/DeepLearnToolbox

https://github.com/skaae/DeepLearnToolbox

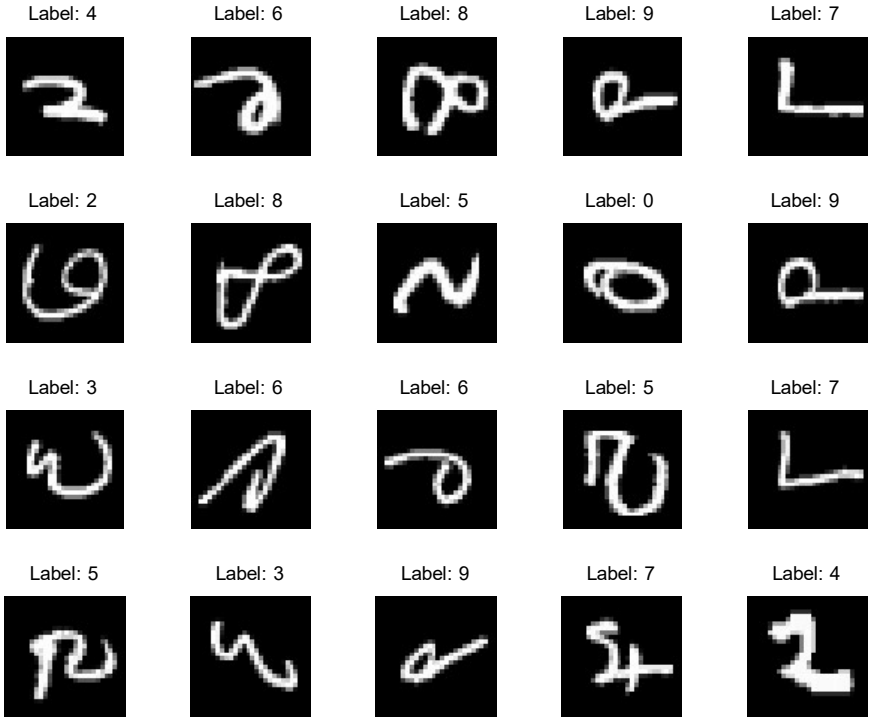
DeepLearnToolbox-GPU

http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox

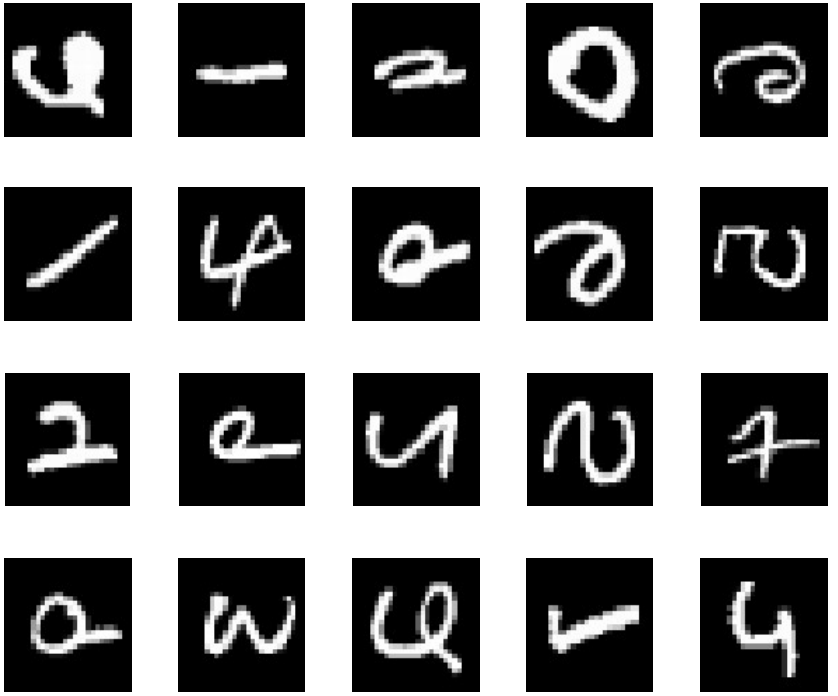
Run the code

```
>> show_sample_digit
```

training samples - INPUTS



test samples - INPUTS



Run the code (modified)

MNIST database of handwritten digits in MATLAB format
https://github.com/sunsided/mnist-matlab

Modify the path to access the files

```
addpath('d:\teste dl Matlab\rasmusbergpalm-DeepLearnToolbox-5df2801\rasmusbergpalm-DeepLearnToolbox-5df2801\data')
addpath('d:\teste dl Matlab\rasmusbergpalm-DeepLearnToolbox-5df2801\rasmusbergpalm-DeepLearnToolbox-5df2801\CNN')
addpath('d:\teste dl Matlab\rasmusbergpalm-DeepLearnToolbox-5df2801\rasmusbergpalm-DeepLearnToolbox-5df2801\util')
```

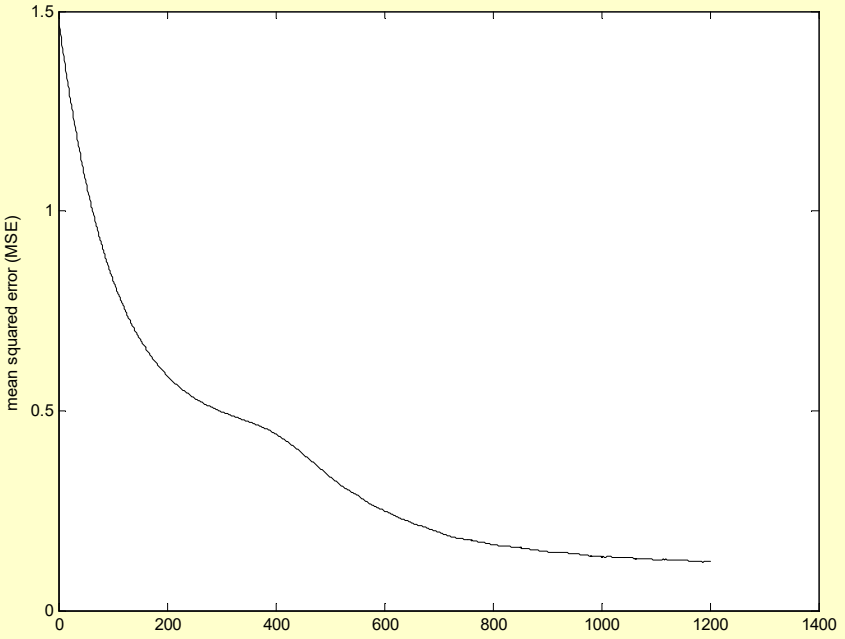
>> test_example_CNN_v2

Convolutional neural network

Case study: MNIST database of handwritten digit
Dataset given by http://yann.lecun.com/exdb/mnist/
Training with 60000 samples of size 28x28
Test with 10000 samples of size 28x28

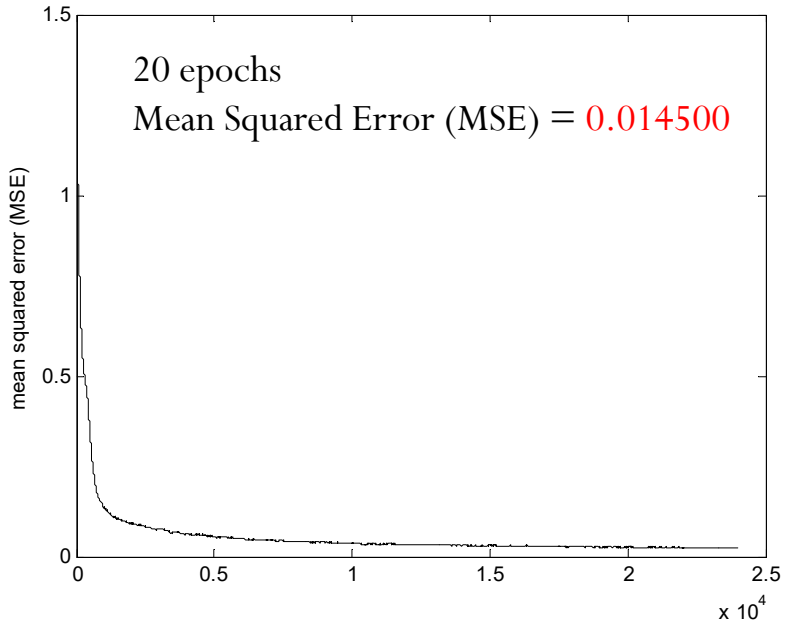
Setup (1-original, 2- proposed): 1

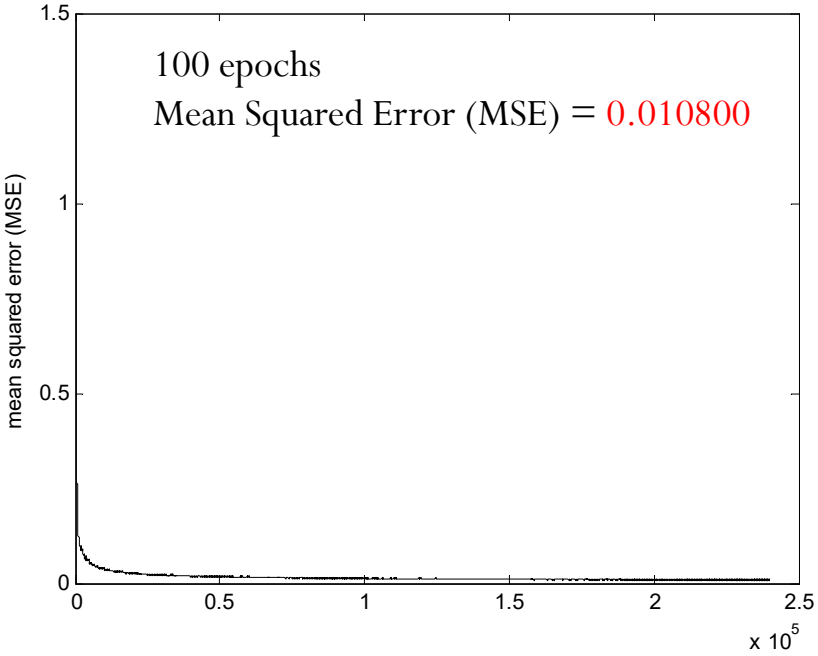
epoch 1/1
Elapsed time is 137.633908 seconds.
Mean Squared Error (MSE) = **0.111300**

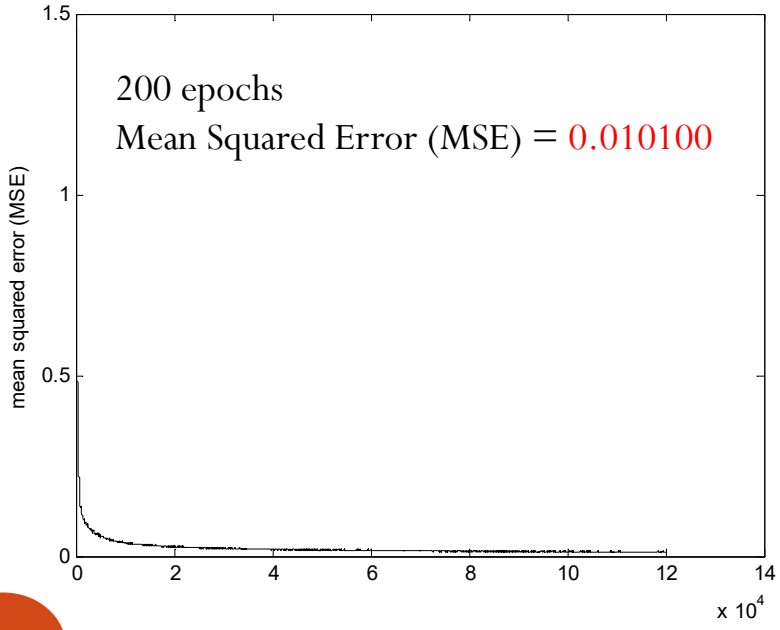


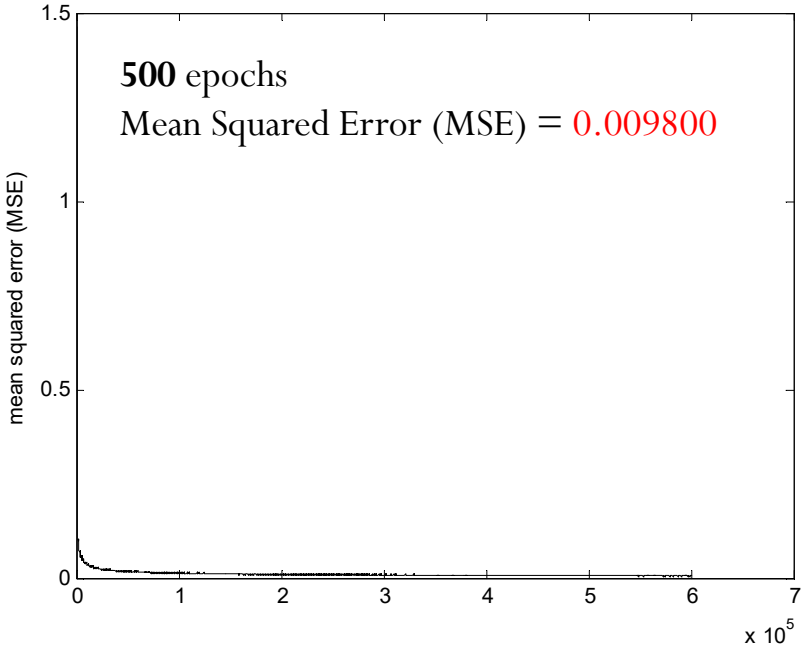
file_name = Res_SetupCNN_6_5_12_5_Epochs_1_MSE_0p1113_Date_20170516T125637.mat

Training with more epochs









Additional material

plotconfusion

Plot classification confusion matrix

Syntax

```
plotconfusion(targets,outputs)
plotconfusion(targets,outputs,name)
plotconfusion(targets1,outputs1,name1,targets2,outputs2,name2,...,targetsn,outputsn,namen)
```

Description

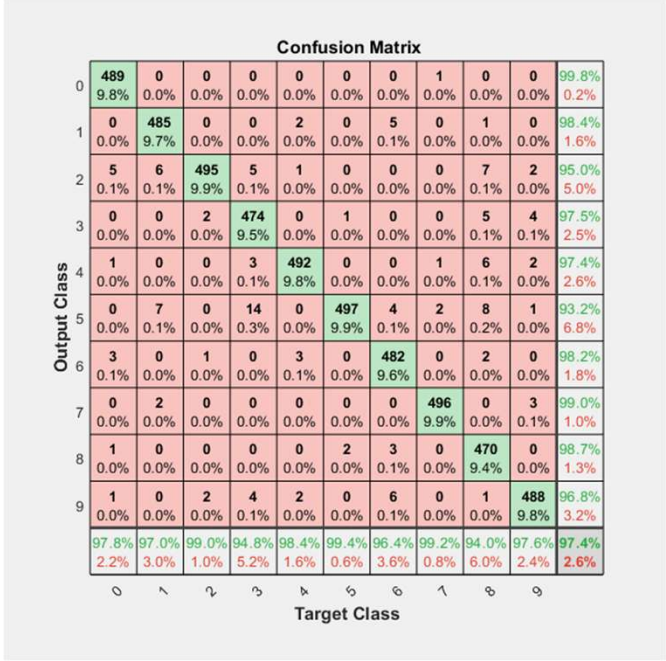
example

`plotconfusion(targets,outputs)` plots a confusion matrix for the true labels `targets` and predicted labels `outputs`. Specify the labels as categorical vectors, or in one-of-N (one-hot) form.

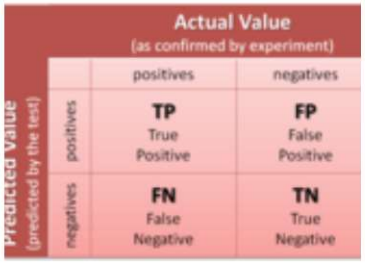
On the confusion matrix plot, the rows correspond to the predicted class (Output Class) and the columns correspond to the true class (Target Class). The diagonal cells correspond to observations that are correctly classified. The off-diagonal cells correspond to incorrectly classified observations. Both the number of observations and the percentage of the total number of observations are shown in each cell.

The column on the far right of the plot shows the percentages of all the examples predicted to belong to each class that are correctly and incorrectly classified. These metrics are often called the precision (or positive predictive value) and false discovery rate, respectively. The row at the bottom of the plot shows the percentages of all the examples belonging to each class that are correctly and incorrectly classified. These metrics are often called the recall (or true positive rate) and false negative rate, respectively. The cell in the bottom right of the plot shows the overall accuracy.

`plotconfusion(targets,outputs,name)` plots a confusion matrix and adds `name` to the beginning of the plot title.



https://mathworks.com/help/deeplearning/ref/plotconfusion.html;jsessionid=9423ac2ec7a85f2db1cf709bb0c2



Multi Class Confusion Matrix

version 2.1.0.0 (7.76 KB) by Abbas Manthiri S

Calclating kappa, acuuracy,error,Sensitivity ,Specificity , Precision,False positive rate etc.

Overview | Functions | Examples

This code is designed for two or more classes instance confusion matrix formation and Calclating
1acuuracy
2.error
3.Sensitivity (Recall or True positive rate)
4.Specificity
5.Precision
6.FPR-False positive rate
7.F_score
8.MCC-Matthews correlation coefficient
9.kappa-Cohen's kappa
Run demo.m for proof and demo

https://mathworks.com/matlabcentral/fileexchange/60900-multi-class-confusion-matrix

Quote

The greatest weapon against stress is our ability to choose one thought over another.

William James (1842-1910)

American philosopher and psychologist, a leader of the philosophical movement of pragmatism and a founder of the psychological movement of functionalism.