Simple Neural Networks for Image Classification

Giovanni Ballarin and Stefanie Bertele

2018

Presentation

Table of Contents

Introduction

We base this Project on the paper "*Digit Recognition Using Single Layer Neural Network with Principal Component Analysis*" by Vineet Singh, and Sunil Pranit Lal (2015).

Their approach is to use Principal Component Analysis to "compress" the number of features of the MNIST dataset, and then feed them into a neural network with 1 hidden layer for classification. We follow their approach in part, but we also expand on it.

Motivation

Image classification tasks are the first class of visual recognition problems that one has to tackle in computer vision. While the set up in many cases is remarkably easy, lots of effort has been put into devising effective classification algorithms.

Even a basic dataset like MNIST can be complex to classify if it is not approached with the "correct" tools. It is thus important to study what constitutes the *minimal* set of techniques necessary to solve

Datasets

We consider the following datasets:

- MNIST
- notMNIST
- Fashion MNIST

Each consists of 28×28 pixels images that belong to 10 classes.

MNIST

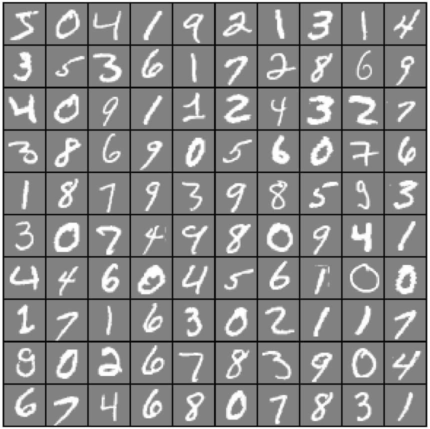


Figure 1: Example images from the MNIST dataset.

notMNIST

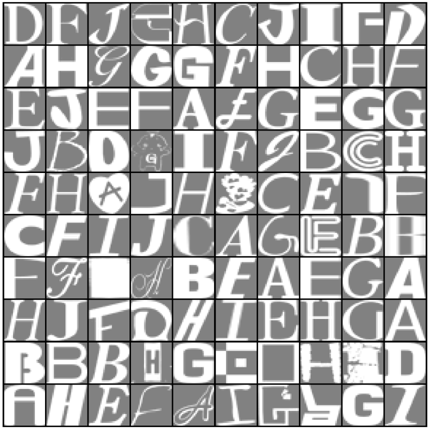


Figure 2: Example images from the notMNIST dataset.

Fashion MNIST

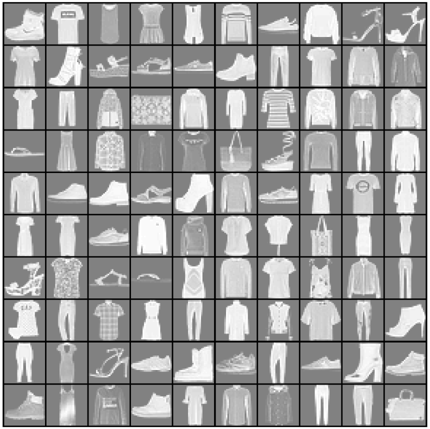


Figure 3: Example images from the Fashion MNIST dataset.

Table of Contents

Approach

The following techniques:

- **PCA + 1D Neural Network**
  Idea: Reduce the dimensionality of the 784×1 vector obtained from each image by projecting it into a smaller vector space, then classify it with a NN;

- **Convolutional Neural Networks**
  Idea: Account for the spatial nature of the image by using a 2D Neural Network architecture;

Principal Component Analysis - PCA

PCA is widely used as a projection technique to reduce the dimensionality of data.
The algorithm is straightforward:

1. Demean the data: $\overline{X} = X - m(X)$
2. Compute the covariance matrix:
$$C = \frac{1}{N}\,\overline{X}\,\overline{X}^{\top}$$
3. Find the eigenvectors (and eigenvalues) matrices, V and D:
$$V^{-1}\,C\,V = D$$
4. Order the eigenvectors by highest associated eigenvalue and project the data X onto the first K components:
$$Z = \overline{X}\,V(1:K)$$

PCA - An Example

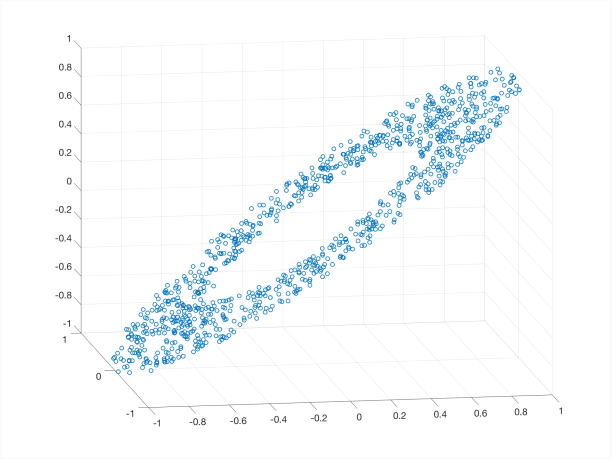


Figure 4: A "3D Ring" point cloud before PCA.

PCA - An Example

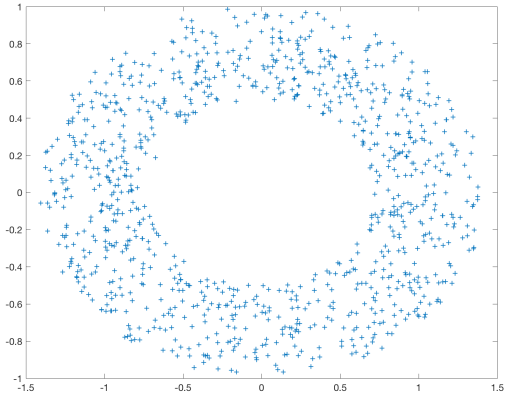


Figure 5: PCA projects the "3D Ring" point cloud to 2D by removing the planar dimension.

Architecture

PCA sizes

- 10
- 25
- 50
- 100

Fully connected NN architectures

- 10
- 25
- 100
- $25 \longmapsto 15$
- $25 \longmapsto 20 \longmapsto 15$
- $100 \longmapsto 50 \longmapsto 20$

Convolutional Neural Networks - CNNs

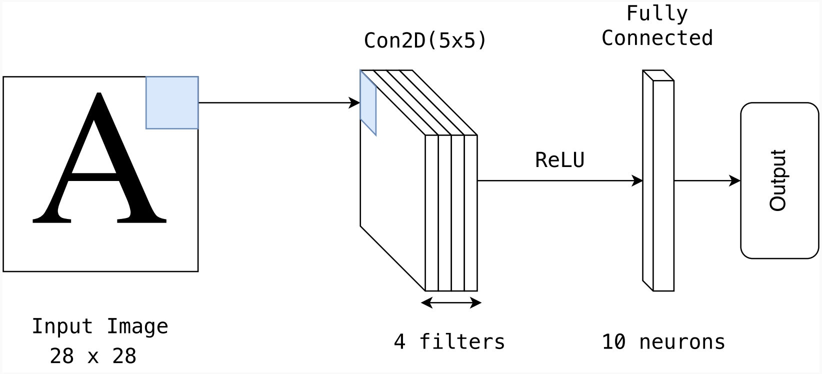
CNNs are a variation of multilayer perceptrons that were designed to mimic the visual processing of living organisms.

Each CNN layer has a number of **filters**: each filter is used to make a convolution with the image, producing a filter bank. The output of the convolutional layer is simply the collection of filter banks.

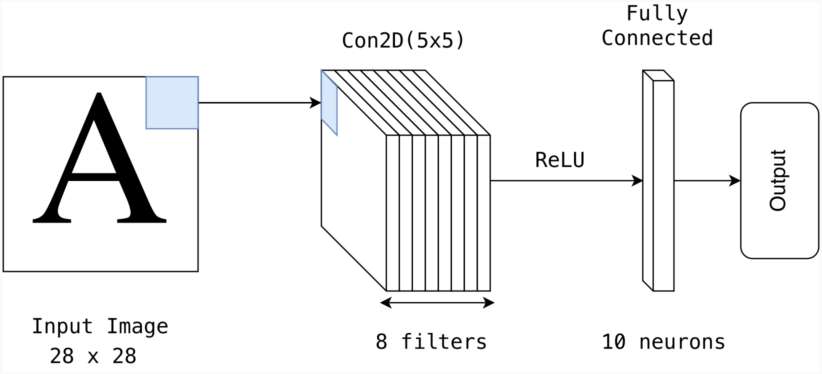
CNNs have notable advantages compared to fully-connected NN:

- Filters are usually small in size (e.g. 5×5), thus they have considerably less parameters to train;
- CNNs preserve the spatial nature of 2D signals like images or video, thus allowing for more effective feature recognition.

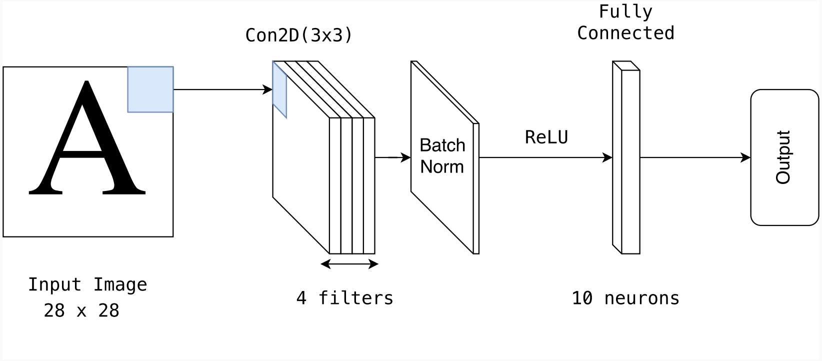
CNN - Architecture 1



CNN - Architecture 2



CNN - Architecture 3



CNN - Architecture 4

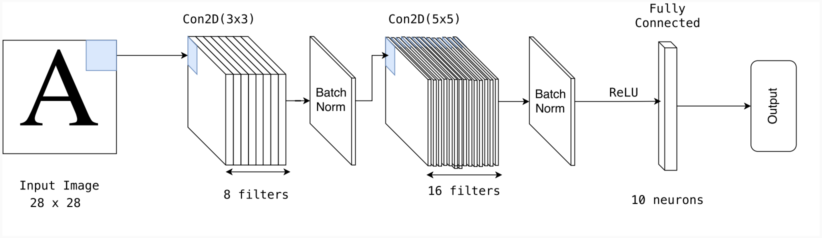


Table of Contents

Results

PCA + 1D NN

Results - PCA + 1D Neural Network - MNIST

% Var	PCs	NN Architecture	Training Error	Test Error
50	11	25	0.168	0.698
50	11	25-20-15	0.119	0.838
75	33	25	0.088	0.729
75	33	100	0.037	0.817
85	58	10	0.084	0.705
85	58	25-20-15	0.066	0.807
95	150	25	0.054	0.688
95	150	100-50-20	0.049	0.852

Table 1: Best and worst results for different PCA sizes depending on explanatory power of the variance. Training size: 9000 images. Test size: 1000 images.

Results - PCA + 1D Neural Network - notMNIST

% Var	PCs	NN Architecture	Training Error	Test Error
50	5	100-50-20	0.207	0.612
50	5	25-20-15	0.235	0.639
75	19	100	0.089	0.555
75	19	25-20-15	0.104	0.642
85	40	25	0.101	0.559
85	40	100-50-20	0.094	0.62
95	126	25-20-15	0.089	0.568
95	126	25-15	0.081	0.587

Table 2: Best and worst results for different PCA sizes depending on explanatory power of the variance. Training size: 9000 images. Test size: 1000 images.

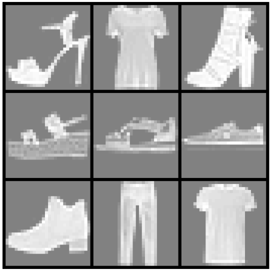
A Trick - 2D Fourier Transform

For the Fashion MNIST dataset, directly applying PCA and a fully-connected 1D neural network produces *very bad results*. The final test error for all combinations is around **70%**.

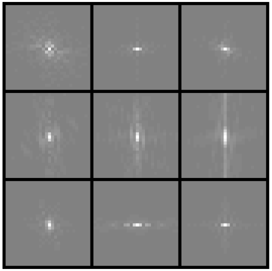
A simple trick allows to reduce greatly the error: pre-treat the dataset with a 2D Fourier Transform on the images to move to the frequency domain.

We loose the spatial information, but we reduce the test error by up to ***30*** *percentage points!*

A Trick - 2D Fourier Transform



(a)



(b)

Figure 6: Some images from the Fashion MNIST dataset (a), and their respective frequency decomposition through the 2D Fourier Transform (b).

Results - PCA + 1D Neural Network - Fashion MNIST

% Var	PCs	NN Architecture	Training Error	Test Error
50	2	25-15	0.498	0.516
50	2	10	0.52256	0.554
75	3	100	0.35622	0.398
75	3	25	0.40367	0.422
85	8	100-50-20	0.22089	0.257
85	8	10	0.28756	0.327
95	44	25-15	0.17978	0.399
95	44	100	0.151	0.444

Table 3: Best and worst results for different PCA sizes depending on explanatory power of the variance. The dataset is preprocessed by doing 2D Fourier transform of each image. Training size: 9000 images. Test size: 1000 images.

Results

CNN

Results - CNN - MNIST

#	CNN Architecture	Training Error	Test Error
1	2D(5x5,4)	0.025625	0.055
2	2D(5x5,8)	0.020375	0.049
3	2D(3x3,4)-BN	0.002125	0.045
4	2D(3x3,8)-BN-2D(5x5,16)-BN	0.001625	0.031

Table 4: Training size: 9000 images (10% for validation). Training options: Stochastic Gradient Descent with Momentum, maximum 20 epochs with validation patience of 10 epochs, Mini Batch size of 64, L2 regularization factor of 0.0005.

Results - CNN - notMNIST

#	CNN Architecture	Training Error	Test Error
1	2D(5x5,4)	0.08275	0.112
2	2D(5x5,8)	0.08125	0.108
3	2D(3x3,4)-BN	0.044875	0.099
4	2D(3x3,8)-BN-2D(5x5,16)-BN	0.0415	0.088

Table 5: Training size: 9000 images (10% for validation). Training options: Stochastic Gradient Descent with Momentum, maximum 20 epochs with validation patience of 10 epochs, Mini Batch size of 64, L2 regularization factor of 0.0005.

Results - CNN - Fashion MNIST

#	CNN Architecture	Training Error	Test Error
1	2D(5x5,4)	0.16287	0.179
2	2D(5x5,8)	0.16225	0.168
3	2D(3x3,4)-BN	0.1035	0.148
4	2D(3x3,8)-BN-2D(5x5,16)-BN	0.10475	0.144

Table 6: Training size: 9000 images (10% for validation). Training options: Stochastic Gradient Descent with Momentum, maximum 20 epochs with validation patience of 10 epochs, Mini Batch size of 64, L2 regularization factor of 0.0005.

Thank You