University of Duisburg-Essen
Chair of Dynamics and Control
Univ.-Prof. Dr.-Ing. Dirk Söffker

Project work

Image Classification - MNIST
Regression - CFRP NASA

Mohamed Ibrahim
Matrikelnummer: 315605

Advisor

Jonathan Liebeton, M.Sc.
Univ.-Prof. Dr.-Ing. Dirk Söffker

September 2023

Versicherung an Eides Statt

Ich versichere an Eides statt durch meine untenstehende Unterschrift,

- dass ich die vorliegende Arbeit - mit Ausnahme der Anleitung durch die Betreuer - selbstständig ohne fremde Hilfe angefertigt habe und
- dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus fremden Quellen entnommen sind, entsprechend als Zitate gekennzeichnet habe und
- dass ich ausschließlich die angegebenen Quellen (Literatur, Internetseiten, sonstige Hilfsmittel) verwendet habe und
- dass ich alle entsprechenden Angaben nach bestem Wissen und Gewissen vorgenommen habe, dass sie der Wahrheit entsprechen und dass ich nichts verschwiegen habe.

Mir ist bekannt, dass eine falsche Versicherung an Eides Statt nach §156 und nach §163 Abs. 1 des Strafgesetzbuches mit Freiheitsstrafe oder Geldstrafe bestraft wird.

Ort, Datum

Unterschrift

Urheberrechtsvereinbarung

Die vorliegende Arbeit entstand unter intensiver Mitwirkung der genannten, betreuenden Personen im Rahmen von Forschungsarbeiten im Lehrstuhl Steuerung, Regelung und Systemdynamik (SRS) der Universität Duisburg-Essen. Die in dieser Arbeit enthaltenen Lösungen und Ideen unterliegen daher nicht nur dem Urheberrecht der zu qualifizierenden Person, sondern dem genannten Personenkreis gemeinsam. Die persönliche, private sowie die Nutzung im Forschungskontext des Lehrstuhls SRS ist hiervon unbenommen.

Ort, Datum

Unterschrift

Contents

List of Figures

List of Tables

1 Einleitung

2 Image Classification - MNIST: CNN

2.1 Environment Setup

The code starts by clearing all variables, closing all figures, and clearing the console. A parallel pool is started for parallel computation if it doesn't already exist.

```
clear all;
close all;
clc;
if isempty(gcp('nocreate'))
    parpool;
end
```

2.2 Data Loading and Preprocessing

The MNIST dataset is loaded into variables, reshaped, and normalized to have pixel values between 0 and 1.

```
data = load('mnist.mat');
trainImages = reshape(trainImages, [28, 28, 1, size(
   trainImages, 3)]);
trainImages = double(trainImages) / 255;
```

2.3 CNN Architecture Definition

The Convolutional Neural Network (CNN) architecture is defined. It comprises Convolution-BatchNorm-ReLU blocks, MaxPooling layers, a fully connected layer, a softmax layer, and a classification layer.

```
lgraph = layerGraph([ ... ]);
```

2.4 Training Configuration

Training options are defined, which include the choice of optimizer (Adam), learning rate, and number of epochs among other settings.

```
options = trainingOptions('adam', ...);
```

2.5 Model Training and Evaluation

The CNN model is trained, and its performance is evaluated on the test data to compute the accuracy.

```
[net, info] = trainNetwork(trainImages, categorical(
   trainLabels), lgraph, options);
pred = classify(net, testImages);
accuracy = sum(pred == categorical(testLabels)) / numel(
   testLabels);
```

2.6 Metrics Calculation

Precision, recall, and F1-score are calculated for each class based on the confusion matrix.

```
% Calculate precision, recall, and F1-score
```

2.6.1 Network Design

- **Input Layer**: Accepts grayscale images of size $28 \times 28 \times 1$.
- **Convolution Layers**: Two layers with 3×3 filters. The first has 8 filters, and the second has 16.
- **Batch Normalization Layers**: Used for normalizing the activations.
- **ReLU Layers**: Used for introducing non-linearity.
- **Max-Pooling Layers**: Used for downsampling the feature maps.
- **Fully Connected Layer**: A dense layer with 10 output neurons for the 10 classes in MNIST.
- **Softmax and Classification Layers**: Used for multi-class classification.

3 Image Classification - MNIST: KNN

3.1 Environment Setup

The initialization segment of the code sets up the MATLAB environment. By executing commands ‘close all;‘, ‘clear all;‘, and ‘clc;‘, it ensures the workspace is clear from variables, closes all open figure windows, and cleans the Command Window. This is a standard practice to ensure that old data or figures do not interfere with new operations. Furthermore, a parallel pool is initiated to speed up certain computational tasks, especially when leveraging multi-core processors.

```
clear all;
close all;
clc;
if isempty(gcp('nocreate'))
    parpool;
end
```

3.2 Data Loading and Preprocessing

After setting up the environment, the code proceeds to load the MNIST dataset. The MNIST dataset is a collection of handwritten digits, commonly used in the field of machine learning and computer vision for training and testing algorithms related to image processing.

Subsequently, the data is separated into training and testing sets. Reshaping images into vectors is a common preprocessing step before feeding them into many machine learning algorithms.

```
data = load('mnist.mat');
training_set = data.training;
test_set = data.test;
train_Labels = gpuArray(training_set.labels);
test_Labels = gpuArray(test_set.labels);
```

To simplify the representation for the algorithms, each image is reshaped into a vector and the reshaped data is then transferred to GPU.

```
trainData = gpuArray(reshape(training_set.images, [], size(
   training_set.images, 3))');
```

```
testData = gpuArray(reshape(test_set.images, [], size(
   test_set.images, 3))');
```

3.3 Principal Component Analysis (PCA)

PCA is utilized to reduce data dimensionality. Here's a summary of its application to our dataset:

3.3.1 PCA on Training Data

Principal component vectors, representations, and variances are computed with:

```
[coeff, score, latent] = pca(gather(trainData));
```

3.3.2 Selecting Principal Components

To retain 95% of the variance:

```
explainedVariance = cumsum(latent) / sum(latent);
numComponents = find(explainedVariance > 0.95, 1);
```

3.3.3 Data Projection

Training data is mapped to the PCA space directly from the 'score'. Test data is centered using the training mean and then projected:

```
trainDataPCA = gpuArray(score(:, 1:numComponents));
meanTrain = mean(trainData, 1);
testDataPCA = gpuArray((gather(testData) - meanTrain)...
 * coeff(:, 1:numComponents));
```

These transformed datasets can be used for further tasks with reduced computational costs.

3.4 Data Visualization

Visualizing data is essential for understanding and interpreting results. In this code, two plots are constructed. The first plot shows the proportion of variance explained by each principal component, while the second illustrates the cumulative variance. These plots are valuable in comprehending the diminishing returns of including more components in the analysis. Two plots are generated for visualization:

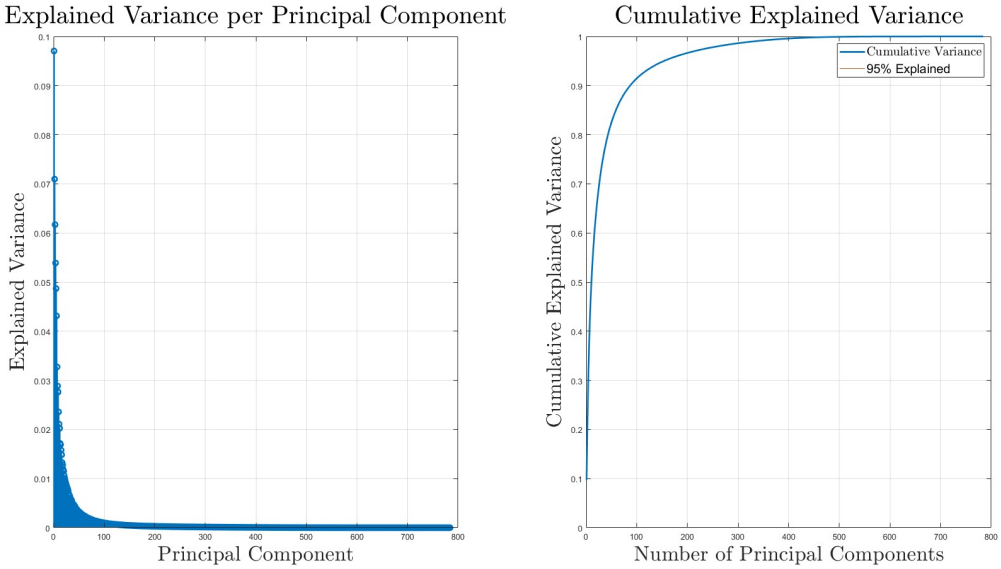

Figure 3.1: Explained Variance Plot.

- The first plot illustrates the proportion of variance explained by each principal component.
- The second plot displays the cumulative variance explained by the principal components.

3.5 k-Nearest Neighbors Classification

The k-NN algorithm is a non-parametric method used for classification tasks. Here, the k-NN model is trained using PCA-transformed training data. The number of neighbors (‘k‘) is set to 5, with distance measured using correlation and weighted by the squared inverse. After training, predictions are made for the test set.

3.6 Evaluation Metrics

Performance evaluation is crucial to understanding how well the classifier works. The confusion matrix, a table layout of actual versus predicted classifications, is presented. From this matrix, various metrics such as precision, recall, and F1-score for each class are calculated. These metrics provide a more comprehensive view of the classifier's performance beyond just accuracy.

3.7 Conclusions

The provided code offers a comprehensive workflow for classifying the MNIST dataset using k-NN after dimensionality reduction with PCA. The inclusion of visualization tools and performance metrics ensures a robust understanding of the process and outcomes.

4 ...

References

[HC04] Huang, A.C. ; Chen, Y.C.: Adaptive Sliding Control for Single-Link Flexible-Joint Robot with Mismatched Uncertainties. In: *IEEE Transactions on Control Systems Technology* 12 (2004), Nr. 5, S. 770–775

[Söf99] Söffker, D.: Observer-based measurement of contact forces of the nonlinear rail-wheel contact as a base for advanced traction control. In: Wallaschek, J. (Hrsg.) ; Lückel, J. (Hrsg.) ; Littmann, W. (Hrsg.): *Mechatronics and Advanced Motion Control* Bd. 49. HNI-Verlagsschriftenreihe, 1999, S. 305–320

[SV89] Spong, M.W. ; Vidyasagar, M.: *Robot Dynamics and Control*. New York : John Wiley and Sons, 1989