**Handwritten Letter Recognition**

**Using**

**Deep Convolutional Neural Network**

**Axente Andrei**

**Technical University of Cluj-Napoca**

**Faculty of Electronics, Telecommunications, and Information Technology**

**3rd year**

Table of Contents

[I. Theme 2](#_Toc155478858)

[II. Theoretical Fundamentals 2](#_Toc155478859)

[III. Preprocessing 4](#_Toc155478860)

[IV. Labeling the data 6](#_Toc155478861)

[V. The Convolutional Neural Network 6](#_Toc155478862)

[VI. Improvements 8](#_Toc155478863)

[VII. Conclusion 8](#_Toc155478864)

[VIII. References 9](#_Toc155478865)

# **I. Theme**

The project implies research in the Machine Learning area, more specifically, into Deep Convolutional Neural Networks. A well-suited illustration for this project involves developing a deep convolutional neural network designed to recognize 32x32 greyscaled images of handwritten letters.

# **II. Theoretical Fundamentals**

**Neural Networks** are a subset of **machine learning**, and they are at the **heart of deep learning algorithms**. They are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

**Convolutional neural network** (**CNN**) is a [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)" \o "Regularization (mathematics)) type of [feed-forward neural network](https://en.wikipedia.org/wiki/Feed-forward_neural_network" \o "Feed-forward neural network) that learns [feature engineering](https://en.wikipedia.org/wiki/Feature_engineering" \o "Feature engineering) by itself via [filters](https://en.wikipedia.org/wiki/Filter_(signal_processing)" \o "Filter (signal processing)) (or kernel) optimization.

A diagram of a diagram of a variety of cubes

Description automatically generated

The fundamental principles driving Neural Networks involve the application of linear algebra, calculus, and concepts from probability and statistics.

**How is an image processed by a computer?**

A RGB image is processed by the computer using 3 separate matrices that indicate the intensity of the colors. Each element of the matrix represents a pixel of the image.A group of white rectangular boxes with numbers

Description automatically generated with medium confidence

Because each color is depicted by 8 bits, the elements inside the matrices take values between 0 (no intensity) and 255 (max intensity). An RGB image has three channels. A greyscale image is composed of a single channel, wherein the computer perceives it as a single matrix containing values ranging from 0 (black) to 255 (white).

A number with numbers and dots

Description automatically generated with medium confidence

# **III. Preprocessing**

**Why preprocessing the image?**

Consider an RGB image with dimensions of 3024x3024 that is stored in the computer. That means there are 3024\*3024\*3 = 27.433.728 independent values that make up the image. Scaling that to 5200 unique pictures (27.433.728 \* 5200 = 142.655.385.600) would be a very complicated task, computationally speaking. That's why there is a critical need to reduce the size of the images before feeding them to the CNN.

A blue letter on a white surface

Description automatically generated

3024x3024 (3channel) 1MB 32x32 (1channel) 499B

Most of the steps are the same as shown in the [EMNIST research paper](https://arxiv.org/pdf/1702.05373.pdf). The first step is to convert the BGR image into Gray. Then, the image is converted into a binary one (takes values of 0 or 255). A white mask is applied for the outer region, to get rid of unnecessary black pixels.

A graph of a grid and a mask

Description automatically generated

Following these procedures, the pixel values are inverted, resulting in the letter appearing in white against a black background. A gaussian blur is applied, with σ = 0.3, the image having values between 0 and 255. The last step is to use a ROI (region of interest) function to center and extract the essential features of a specific 32x32 image. It's important to acknowledge that additional steps were undertaken to attain the outcome, although they are not detailed in this paper.

A collage of images of a letter l

Description automatically generated

# **IV. Labeling the data**

To annotate the images, a [CSV](https://www.youtube.com/watch?v=q7ZuZ8ZOErE) file was employed. The file comprises two columns, namely 'file\_name' and 'label', ensuring that each picture is associated with a corresponding label.

**A screenshot of a computer

Description automatically generated**

An example of labeled images

A mapping dictionary was established to associate each picture (identified by 'file\_name') with its corresponding label.

A black background with numbers and letters

Description automatically generated

# **V. The Convolutional Neural Network**

The images were split as following: 70% for the training set, 15% for cross validation and 15% for the test set.

In tackling this task, a Neural Network architecture was employed, comprising four convolutional layers, four max-pooling layers, and two dense layers. To address potential overfitting, two dropout layers were incorporated for regularization.

The model is trained with 32 images per batch (mini-batch) (normal batch = 64).

During forward propagation, the **Rectified Linear Unit (ReLU)** activation function (R(x) = max(0, x)) was applied to facilitate non-linearity. For the final layer, **softmax** activation was utilized to convert logits/numerical outputs into probabilities.

A graph of a function

Description automatically generated

The softmax function resembles the sigmoid function, with the sigmoid being a specific case of softmax designed for scenarios where there are only two possible outputs. In the context of backpropagation, the Adam (similar to Gradient Descent but more powerful because Adam dynamically adjusts the learning rates for each parameter) optimizer was implemented with a learning rate set at 0.0013.

A screenshot of a computer program

Description automatically generated

Using hyperparameter tuning, the model successfully reached a 97%+ accuracy for the training and cross validation sets.

A graph of a line

Description automatically generated with medium confidence

Note: The percentages will change each time the code is executed.

In this run, the test set achieved a surprisingly 99.49% accuracy <=> 4 out of 780 images were misclassified

A collage of images of letters

Description automatically generated

An impartial approach to evaluate the performance of the Deep Convolutional Neural Network involved designing a test that was administered to both the CNN and human participants. The evaluation comprised ten randomly preprocessed images, which achieved a 100% success rate when assessed by the neural network. The identical set of images underwent assessment by a sample of 23 individuals from my friends' population. The outcome resulted in a sample mean accuracy of 91.73%, providing a statistical measure of the performance evaluation. The model's consistent success in outperforming the human brain clearly demonstrates its capabilities in handling the given task. It’s essential to notice that the model achieves these performances only when using the dataset of my own handwriting. Generalization across different datasets was not intended for this project.

# **VI. Improvements**

First of all, there can be massive improvements in the preprocessing phase. The dots are excluded from the final preprocessed image, and consequently, characters like 'i' or 'j' aren’t displayed accurately.

Additionally, to generate more synthetic image samples, an augmentation function can be utilized to randomly rotate the images, thereby expanding the dataset, and preventing overfitting.

A Confusion matrix can be implemented to easily visualize the misclassified images.

Significant enhancement could be achieved by expanding the project to encompass **word recognition**. In such a scenario, the CNN would benefit from support provided by Long Short-Term Memory (**LSTM**) Neural Networks.

# **VII. Conclusion**

The CNN created performs as intended, classifying my own handwritten letters accordingly. However, it's essential to note that this is a basic demonstration, and CNNs, in general, possess increased capabilities for handling more complex classifications.

# **VIII. References**

<https://arxiv.org/pdf/1702.05373.pdf>

<https://www.khanacademy.org/math/statistics-probability>  
<https://www.khanacademy.org/math/multivariable-calculus/integrating-multivariable-functions>

<https://www.coursera.org/learn/advanced-learning-algorithms/lecture/couAA/neurons-and-the-brain>

<https://nanonets.com/blog/handwritten-character-recognition/>

http://www.bel.utcluj.ro/dce/didactic/eai/

<https://www.ibm.com/topics/convolutional-neural-networks>

https://www.coursera.org/learn/machine-learning/

<https://www.youtube.com/watch?v=tHzJMATAL0U> – processing images in python

<https://www.youtube.com/watch?v=Hs5RhjpQVRo> – processing images theoretical fundamentals

<https://www.youtube.com/watch?v=aircAruvnKk> – what is a neural network

<https://www.youtube.com/watch?v=w8yWXqWQYmU&list=WL&index=1> – Building a NN using numpy

<https://www.youtube.com/watch?v=HaHsqDjWMLU> – chain rule

<https://www.youtube.com/watch?v=Ilg3gGewQ5U> – backpropagation

<https://www.youtube.com/watch?v=WhRC31SlXzA> – handwriting recognition tutorial

<https://www.youtube.com/watch?v=q7ZuZ8ZOErE> – dataset tutorial / .csv files