Image Processing Documents

Shefali Chadha, Rupam Jogal, Sejal Khinvasara

Northeastern University

A close up of a logo

Description automatically generated

Table of Contents

[Abstract 3](#_Toc26800922)

[Image Processing for Handwritten Scanned Documents 4](#_Toc26800923)

[CNN Model Implementation 4](#_Toc26800924)

Evaluate ...............................................................................................................................6

Optical Character Recognition ............................................................................................7

Conclusion ......................................................................................................................... 9

[References 8](#_Toc26800925)

Abstract

The project goal is to use the Plural point INC scanned images which consisted of 175 images of 5 classes to build an image processing machine learning algorithm which would successfully help retrieve the optical character information from the images and provide output which is readable to human eye and the information is machine accessible. The project was conducted in two parts, the first part involved building the image classification algorithm which was performed using the CNN. This required us to use the keras package which requires TensorFlow background. This CNN models are performed in 4 steps involving convolution, pooling, flattening and full connection. The second part of the project included performing the optical character recognition (ORC) for which we used the tesseract package. Let’s dive deep into the methods used to understand the working and implementation of the of the algorithms.

Image Processing Documents

The project began with firstly setting up the environment to run a machine learning algorithm. To setup the environment initially we ensured the keras package was present on the system. Post installation of Keras package functions like sequential, Conv2D, Flatten, Dense, ImageDataGenerator were imported. Once the environment was set the used data was loaded to be used as the training and testing data.

A screenshot of a cell phone

Description automatically generated

Image 1.1

# CNN Model Implementation

A convolutional neural network (CNN, or ConvNet) is a class of deep neural networks commonly applied to analyzing visual imagery. It is regularized version of multilayer perceptron. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. The model was trained on 120 images of 5 classes. To train and validate the data CNN is performed in four steps.

1. **Convolution**: it is performed on the input image using either filters or kernels. For better performance of the convolution the image is scanned from top left to right and covering the width of screen and iterating the process till the entire image is scanned.
2. **Pooling**: A common approach to address the image is the down sampling process. Post the convolution layer is formed effectively applied, the nonlinear ReLU, the pooling layer is added to feature the mapped output. It involves selecting the pooling factor and reducing the image by this factor.
3. **Flattening**: In this step all the polled features are flattened into a column for further processing. This is required to make the data ready for being feed into the neural network.
4. **Full Connection**: the full connection layer takes the inputs from the feature analysis and applies weights to predict the correct label associated and finally the output layer of the full connection gives the prediction of each variable.

A screenshot of a social media post

Description automatically generated

The other parameters specific for the use of algorithm include:

* **training\_set**: the set used to train the model
* **steps\_per\_epoch:** the number of batch iterations before a training epoch is considered finished
* **epochs:** Number of training epochs
* **validation\_data:** Learning the data
* **validation\_steps:** the functionality is similar to steps\_per\_epoch, it works on the validation dataset which here is the test dataset.

A screenshot of a social media post

Description automatically generated

Image 1.2

A screenshot of a cell phone

Description automatically generated

Image 1.3

The accuracy function is defined to check the accuracy of the model by passing the test and training data. **Accuracy of the model is 94.11%**

**Evaluate**

The images are evaluated through the network. The network outputs class probabilities and typically, one selects the class with the maximum probability as the final class output.

*A screenshot of a cell phone

Description automatically generated*

Image 1.4

**Optical Character Recognition**

The optical character recognition is the electronic conversion of images of all the types, handwritten or printed text into machine encoded text. It is commonly used in research for pattern recognition, artificial intelligence and computer vision.

We took the advantage of Amazon Textract to automatically extract text and data from scanned documents without any machine learning (ML) experience. While AWS takes care of building, training, and deploying advanced ML models in a highly available and scalable environment, you take advantage of these models with simple-to-use API actions.

Amazon Textract goes beyond simple optical character recognition (OCR) to also identify the contents of fields in forms and information stored in tables. This allows you to use Amazon Textract to instantly “read” virtually any type of document and accurately extract text and data without the need for any manual effort or custom code. In addition to the detected content, Amazon Textract provides additional information, like confidence scores.

Steps involved:

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

Traditional OCR solutions read left to right, do not detect multiple columns, and end up generating incorrect reading order for multi-column documents. In addition to detecting text, Amazon Textract provides additional geometry information that can be used to detect multiple columns and print the text in reading order.

Boto is the Amazon Web Services (AWS) SDK for Python. It enables Python developers to create, configure, and manage AWS services, such as EC2 and S3. Boto provides an easy to use, object-oriented API, as well as low-level access to AWS services.

Output:

A screenshot of a cell phone

Description automatically generated

**Conclusion**

The project really helped us learn a new area that is image processing. On implementing our CNN model, we obtained of around 94.1% accuracy and in the OCR part we received accuracy around 95%.

References

# “Tesseract OCR best practices”, na. nd

<https://ai-facets.org/tesseract-ocr-best-practices/>

“Image Processing”, na.nd

<https://keras.io/preprocessing/image/>

Allibhai Eijas, Oct 2018 “Building a Convolutional Neural Network (CNN) in Keras”

<https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5>