Optical Character Recognition

A Udacity Machine Learning Nanodegree Capstone Project

Domain Background

In the modern world, we maintain order through digital organization. Be it expense forecasting reimbursement requests, or discovering exactly where our personal budget went wrong, an electronic record of monetary transactions is becoming a vital part of everyday finance. Unfortunately, to sustain a comprehensive database of all relevant dealings one must manually record each entry or limit the purchases to one credit/debit card account.

The proposal being presented here is to offer an alternative solution, the creation of a program that will convert an image of a receipt to a digital replication with an accurate depiction of both the characters location and ASCII values. The technology to recognize numbers and letters has been pioneered by Yann LeCun and others in the MNIST (LeCun, Cortes, & Burges) and EMNIST datasets (Cohen, Afshar, Tapson, & Schaik). Meanwhile, the ability to locate, rotate, and crop a document from an image is widely popular in applications such as OneNote, Evernote, and Google Drive (O'Neill, 2016). By uniting these two principal functions with a blend of Decision Tree and Tensor Flow algorithms we can hope to provide a backbone to future applications such as monetary recordkeeping, collective expensive allocation, and automatic document verification.

Problem & Solution Statements and Benchmarks

Problem Statement

The problem is to convert a standard cell phone receipt image into a digital reproduction including a display of correctly located characters derived from their respective ASCII values. The project objective is to create a functioning pipeline for optical character recognition with better than random guessing accuracy. In an effort to keep the scope at a manageable level the second dataset and testing will be limited to multiple images of one receipt. This receipt contains both upper and lowercase letters, fading ink, and human-made pen marks drawn in the vertical direction. In summary, an OCR nightmare.

Solution Statement

This could potentially be solved by locating the receipt using the Canny Edge Detection function and then dividing the image into lines of text and finally character blocks. These blocks would be manipulated using techniques suggested by Yann LeCun in an effort to make them match the EMNIST dataset as closely as possible. The new set of character images would then be analyzed along with a normalized representation of their original size and location in a Decision Tree matrix. The algorithm would separate images into real, recognizable characters and dead space or unrecognizable figures before passing the first category on to a Tensor Flow algorithm. This algorithm classifies inputs as letters A to Z or digits 0 to 9. Finally, the letters or numbers would be placed on a reproduction of the original image in the same location as the block from which they were derived and displayed to the user.

Benchmarks

Current commercial Optical Character Recognition programs achieve character by character accuracy rates between 81% - 99% (Holley, 2009) on standardized printed newspapers. This means that when the network is trained on dataset of character images it correctly classifies 81% - 99% of the validation set from that same database.

Within the OCR process, the program must locate groups of text that are of interest. This task could be completed in several ways, one of which provides a great level of accuracy is human labeling. Apps such as Receiptmate ask the user to physically highlight areas of text (O'Neill, 2016). This process is essentially a binary labeling procedure, the area in question is text or is not text. Andrej Karpathy analyzed his own binary labeling and found human accuracy to be around 94% (Karpathy, 2014). Therefore, it can be inferred that a good benchmark for locating groups of text that are of interest is 94%.

Datasets and Inputs

Datasets

There will be two datasets used for this project. First is the EMNIST or Extended MNIST is derived from the Special 19 Database and uses the same conversion paradigm as its more well-known counterpart, MNIST. EMNIST is a collection of datasets containing handwritten digits, uppercase and lowercase letters. It is also divided into 6 different architectures, for this project we will use the By_Merge dataset. This dataset is intended to address the similarity between some uppercase and lowercase letters. If the two forms of the letter are similar enough to create confusion, their labels were merged. This results in an architecture consisting of 47 classes.

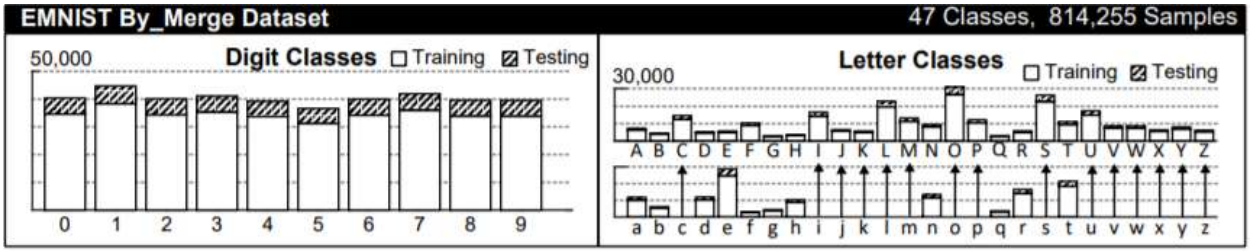


Figure 1 - Dataset Architecture

The second dataset, called the receipt database, has been hand-made for this project. Ten smartphone pictures were taken of the selected receipt and divided into character-blocks using the same process as the final image processing program. These blocks were then labeled by the human-eye resulting in 10,666 entries and 1,558 characters. The blocks and labels were saved into a *.pkl* format along with the respective height, width, horizontal location, and vertical location all normalized to the image.

The idea behind the labeling was to retain only the blocks that were legible and classifiable after preprocessing. If the engineer could not identify the character or it was not contained in one of the 47 classes it was given a '32' or dead-space identification. This logic aims at training the Decision Tree algorithm to give its TensorFlow complement the best chance possible at correctly classifying the image.

Before training, the labels of the blocks were temporarily converted to 0's and 1's indicating if it was a character or dead-space and feed into the DT algorithm. Likewise, before fitting the TensorFlow network the labels were transformed from the ASCII value to its 0 - 46 class designation.

Inputs

The inputs for the project are images of the selected restaurant receipt. They will be feed into the program as JPEGs because that is the camera-phone default. However, other formats such as BIP and PNG would also be accepted.

In testing scenarios, the same images, but in a labeled format, will be selected from the second dataset and feed into the algorithms.

The inputs for the Decision Tree algorithm will be the block image along with the normalized dimensions and locations.

The TensorFlow network requires only the block image as an input.

Evaluation Metrics

The vision of this project is to obtain an accuracy similar to existing benchmarks but on a poorly printed and photographed restaurant receipt. Unfortunately, state-of-the-art programs can require teams of engineers and millions of dollars in resources. To keep things in perspective we will break the evaluation down into three parts. First, the Decision Tree's binary evaluation of whether or not an image contains recognizable text. Second, the TensorFlow network's ability to correctly classify images into one of the 47 categories. Third, the overall ability of the program to convert blocks of pixels from the image to their corresponding character or blank space.

The Decision Tree's ability to select areas of interest and blank spaces is critical to the successful reproduction of the image. Because other options exist for identifying areas of text we will require the algorithm to compete with its most fearsome competitor, a human annotator. Objective studying of binary labeling found human accuracy to be around 94% (Karpathy, 2014). Therefore, 94% accuracy will be the threshold used to determine if the algorithm is performing successfully or not.

The TensorFlow network is expected to be held to the standards of current character by character recognition systems. This means that when evaluated on the test portion of the same dataset it was trained on, EMNIST, it should perform with an accuracy between 81% -99%.

The overall ability of the program to convert blocks of pixels to corresponding classifications is obviously the most interesting metric of the project and will be optimized as much as possible. For evaluation purposes, we will create a system that produces a result that is, at a minimum, better than random guessing. The 10 digits (0-9) and 52 letters (26 uppercase and 26 lowercase) will be categorized into 47 classes. When evaluating the system, one more category will be considered to accommodate the dead space. Therefore, overall classification accuracy is expected to be greater than 1/48 or 2.08%.

To calculate this number the DT algorithm and TF network accuracies will be multiplied together follow the formula and examples seen below.

Formula:

$$Accuracy_{DT} \times Accuracy_{TF} = Accuracy_{Overall}$$

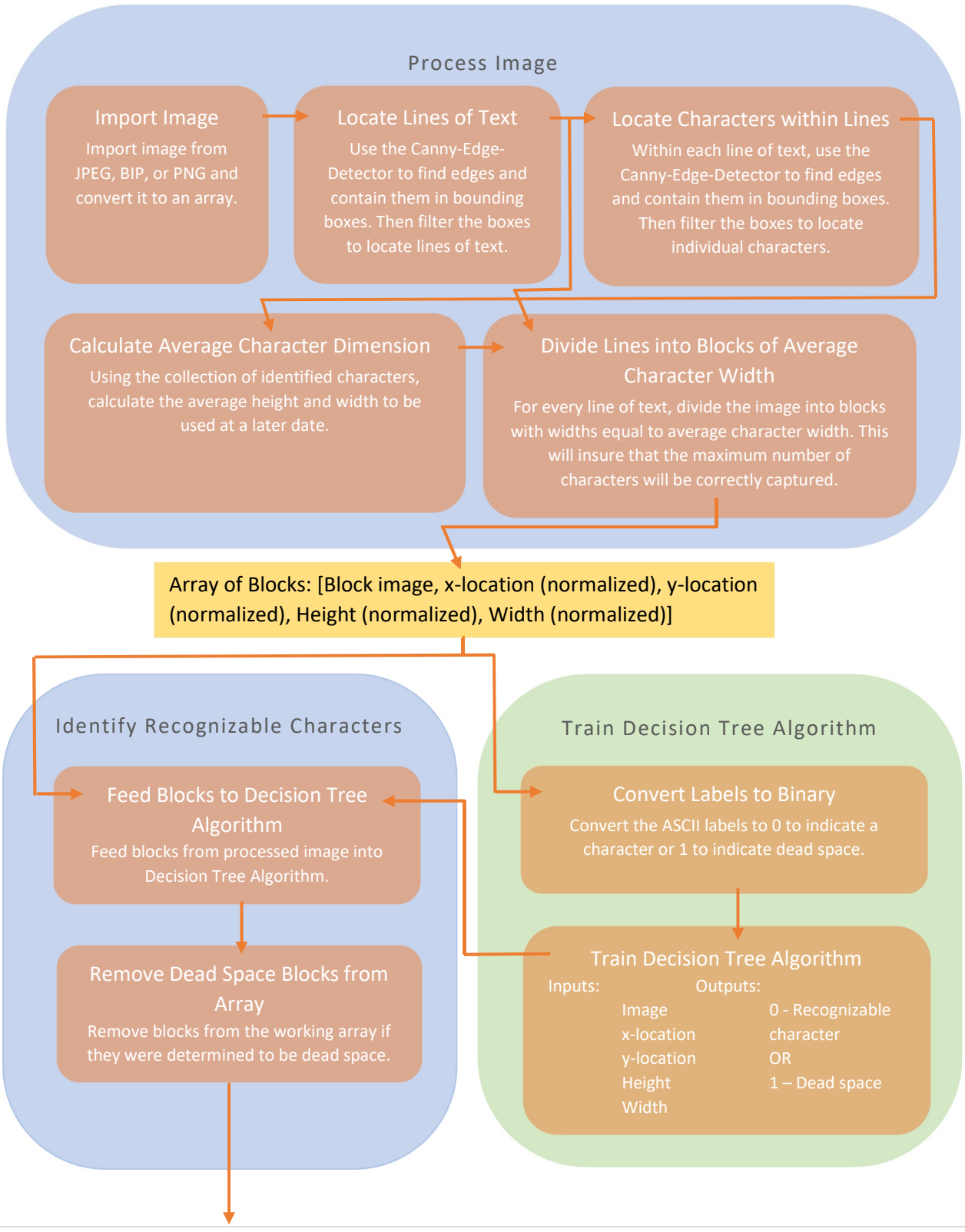
Example:

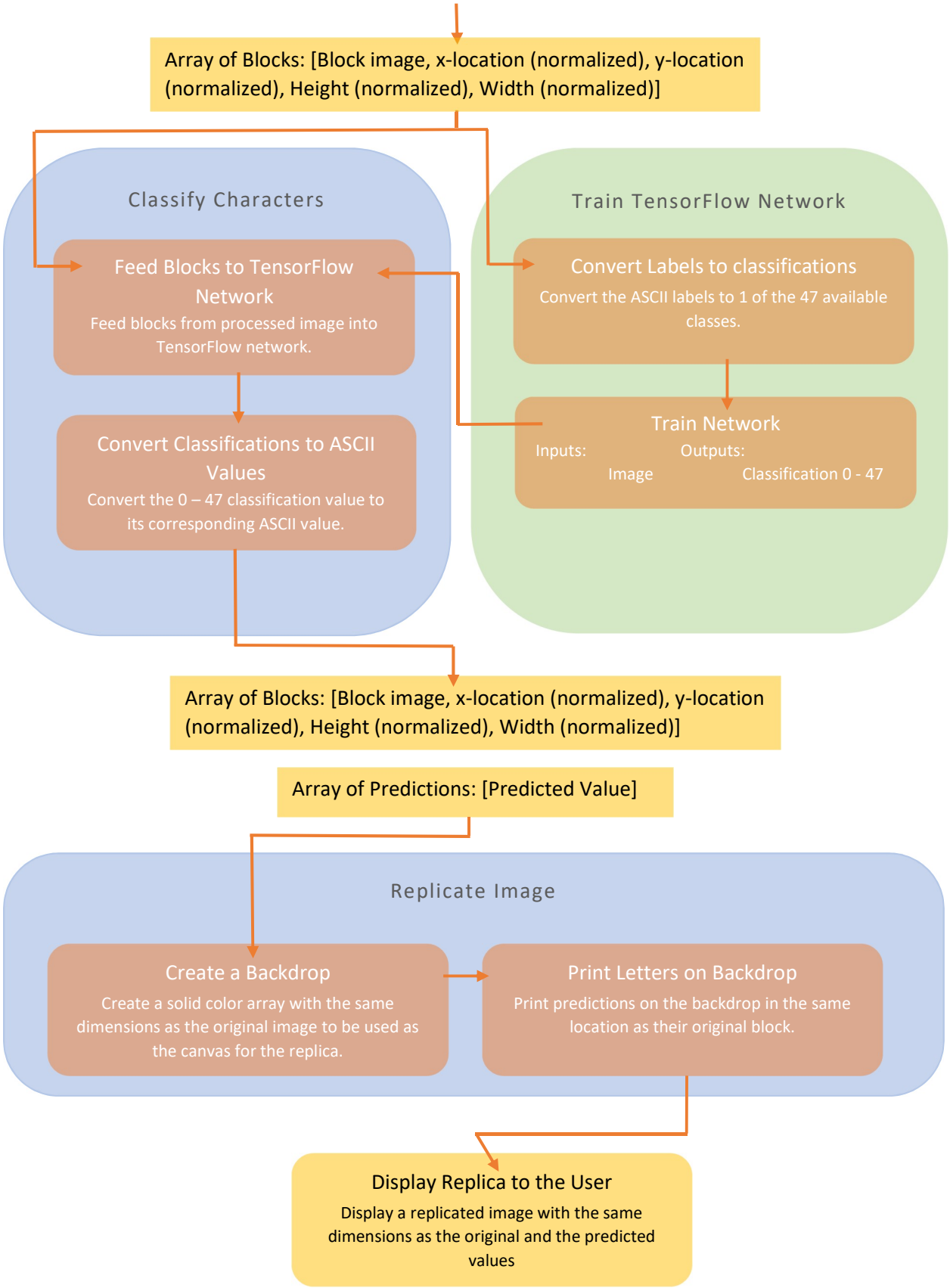
$$\frac{50\ blocks\ correctly\ identified\ by\ DT}{100\ total\ blocks} \times \frac{25\ blocks\ correctly\ classified\ by\ TF}{50\ total\ blocks}$$

$$50\%_{DT} \times 50\%_{TF} = 25\%_{Overall}$$

Project Design

The design for this project will be a single direction pipeline that processes the raw image and outputs a digital replica.




Array of Blocks: [Block image, x-location (normalized), y-location (normalized), Height (normalized), Width (normalized)]
Classify Characters
Feed Blocks to TensorFlow Network
Feed blocks from processed image into TensorFlow network.
Convert Classifications to ASCII Values
Convert the 0 – 47 classification value to its corresponding ASCII value.
Train TensorFlow Network
Convert Labels to classifications
Convert the ASCII labels to 1 of the 47 available classes.
Train Network
Inputs: Image
Outputs: Classification 0 - 47
Array of Blocks: [Block image, x-location (normalized), y-location (normalized), Height (normalized), Width (normalized)]
Array of Predictions: [Predicted Value]
Replicate Image
Create a Backdrop
Create a solid color array with the same dimensions as the original image to be used as the canvas for the replica.
Print Letters on Backdrop
Print predictions on the backdrop in the same location as their original block.
Display Replica to the User
Display a replicated image with the same dimensions as the original and the predicted values

Summary

In all likelihood, the outline of this program will grow and morph throughout the creation process. It is possible that Decision Trees will not be the best machine learning algorithm or that more/less features will be needed. Another area of uncertainty is the structure of the TensorFlow network. It is suspected that convolutional networks will be well suited but this will need to be verified during the construction of the program.

Altogether the Optical Character Recognition project is expected to be an intense, time-consuming undertaking. However, skills learned and challenges overcome should be directly transferable to state-of-the-art technologies being developed in our very near future.