Objective & Motivation

Objective:
Create a mobile application which recreates editable digital equivalents of physical documents using character recognition and machine learning.

Motivation:
P2D makes sharing and processing digital documents more secure, more efficient, and more convenient. It accepts camera photos of physical documents and outputs an editable, digital recreation of the original. This application enables users to send copies of their edited changes to various companies, even when the original copy is unavailable.

Definitions

Optical Character Recognition (OCR) - conversion of images of handwritten or typed text to machine-encoded (digital) text

Convolutional Neural Network (CNN) - in machine learning, an artificial neural network inspired by organization of the visual cortex, takes input and classifies based on features

Transfer Learning - with a pre-trained CNN, utilize the existing connections for feature recognition to recognize new classes with similar types of features (eg: CNN trained for numbers can be fine-tuned to recognize letters)

Background Research

In order to recognize handwritten and printed text, we decided to create OCR software by training a CNN to recognize characters.

- Tools used train:
  - Python
  - Keras with Theano/Tensorflow backend - Keras is a deep learning API that runs on top of Theano/Tensorflow, which are deep learning libraries

- Data used to train:
  - MNIST dataset - subset of the NIST dataset, contains ~80,000 handwritten digits
  - How we trained:
    - Baseline code to train a CNN on four different machines using Theano or Tensorflow
    - Hyperparameters were optimized
    - Comparison of speed and accuracy

- Results/Conclusions:
  - Theano is significantly faster (almost twice as fast)
  - Accuracy rates are comparable (see Figure 1 and Table 1)

- Implications:
  - Still need to train for alphabetical characters, symbols (punctuation), whitespace
  - Transfer learning and fine-tuning CNN to recognize handwritten/ typed text
  - Creating a dataset and programmatically augmenting images

Table 1: CNN Recognition Rates/Accuracy

<table>
<thead>
<tr>
<th></th>
<th>Train set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digits</td>
<td>99.9%</td>
<td>99.8%</td>
</tr>
<tr>
<td>Alphabet</td>
<td>99.5%</td>
<td>99.4%</td>
</tr>
<tr>
<td>Symbols</td>
<td>98.9%</td>
<td>98.8%</td>
</tr>
</tbody>
</table>

Challenges

- Training and Fine-tuning CNN to recognize handwritten/ typed text
- Creating a dataset and programmatically augmenting images
- Transfer Learning - Training the fully connected layer of the CNN
- Recognition accuracy of text
- Efficient Backend (Processing Time)
- Data Security
- Data persistence and scope between activities of application
- Google Cloud Vision API
- Text overlay/scaling of processed text on image
- Obtaining and formatting non-text elements of background
- Large image Bitmap Storage - Enable storing large image files without affecting the application’s performance

Tools

- Keras with Theano and Tensorflow Backends
- Google Cloud Vision API
- Android Studio
- AWS S3
- AWS Lambda

Acknowledgements

We would like to thank our advisor, Kristin Dana, for her guidance, and the Electrical and Computer Engineering Department, especially Undergraduate Director Hana Godrich, for the resources and support.

References