# George Mason University

# DAEN 690

# Summer 2020

# **Accure Autoencoder for Denoising OCR**

**Project Report**

**Team Autoencoder**

**Gauthami Karavi**

**Jun Wang**

**Matt Machado**

**Stephen Schade**

**Yun Li**

**8/17/2020**

Table of Contents

[Abstract 3](#_Toc48596629)

[1 Introduction 4](#_Toc48596630)

[1.1 Background 4](#_Toc48596631)

[1.2 Problem Space 4](#_Toc48596632)

[1.3 Solution Space 4](#_Toc48596633)

[1.4 Project Objectives 5](#_Toc48596634)

[1.5 Research 5](#_Toc48596635)

[1.6 Primary User Story 5](#_Toc48596636)

[1.7 Product Vision 5](#_Toc48596637)

[Scenario #1 5](#_Toc48596638)

[Scenario #2 6](#_Toc48596639)

[1.8 Definition of Terms: 6](#_Toc48596640)

[2 Data Acquisition 7](#_Toc48596641)

[2.1 Overview: 7](#_Toc48596642)

[2.2 Field Descriptions: 7](#_Toc48596643)

[2.3 Data Context: 7](#_Toc48596644)

[2.4 Data Conditioning 7](#_Toc48596645)

[2.5 Data Quality Assessment: 8](#_Toc48596646)

[2.6 Other Data Sources 8](#_Toc48596647)

[3 Analytics and Algorithms 8](#_Toc48596648)

[3.1 Convolutional Neural Network 8](#_Toc48596649)

[3.2 Noisy Autoencoder 10](#_Toc48596650)

[3.3 Convolution 10](#_Toc48596651)

[4 Results 12](#_Toc48596652)

[5 Summary 17](#_Toc48596653)

[6 Future Work 17](#_Toc48596654)

[7 Lessons Learned 17](#_Toc48596655)

[8 Appendix 18](#_Toc48596656)

[8.1 Appendix - GitHub 18](#_Toc48596657)

[8.2 Appendix – Project Risk Matrix 18](#_Toc48596658)

[8.3 Appendix – Agile Workflow Summary 18](#_Toc48596659)

[9 References 19](#_Toc48596660)

# Abstract

Optical Character Recognition (OCR) is a process in which text from scanned images is converted to computer enabled characters to be used in further computational analysis. This process works best on documents that contain highly visible and clean images to reliably extract information. However, not all documents are ready made to be scanned into OCR and require additional pre-processing steps to enhance the image sufficiently. These ‘noisy’ documents contain stains, marks, crumples, and other items that reduce the effectiveness of OCR efforts. The objective of this project is to build an autoencoder capable of taking noisy images and removing these elements to improve OCR recognition. We constructed a Convolutional Neural Network (CNN) autoencoder to denoise image files and improve OCR results. To train our model, we leveraged the RVL-CDIP dataset, which includes thousands of publicly disclosed scanned internal documents from large tobacco companies. Our neural network learns how to remove less-relevant pixel information. We then subject our model-enhanced image to standard OCR processes, comparing the results to the same non-processed image. Our results show small incremental improvements in OCR, and we hope this effort will exist as a first-step in the refinement of a fully functional scanned-document autoencoder.

# Introduction

## Background

There are always obstacles when transitioning from a paper-based world into the new digital era with computers and technology. Specifically, there are vast amounts of old books, manuscripts, receipts, and papers currently sitting in boxes and waiting to be brought into in a digital world. The issue in translating these old documents into digital form is dealing with potential errors in the documents themselves. Google was the first to pioneer a methodology for translating problematic texts due to image noise into computer enabled text using a service called reCAPTCHA. The need for this service originated from Google’s project to digitalize books via the Google Books Project. The company discovered that approximately 20% of text from scanned physical books could not be properly transcribed by computers, and it needed a way to properly capture all the information present. The way reCAPTCHA works is by displaying errored words from Object Character Recognition (OCR) to users to verify if they are human or not. These words were displayed directly as an image. If enough humans wrote the same word, this word would be entered into the digitalized version of the item transcribed. This of course takes effort from human users to scale up to the needs of the sheer volume of physical items that need to be digitalized. As of September 2008, reCAPTCHA had transcribed 440 million problemed words from OCR with 99% accuracy. [1]

## Problem Space

OCR technology enables the extraction of text data from images. There are endless uses for this type of technology, from the digital processing of historical documents to the extraction of text in personal photos. It is designed to capture numbers, letters, and punctuation and transfer these characters into data that can be searched, processed, analyzed, and stored.

OCR technology is valuable in any instance where documents or images have text. This is especially useful as society moves away from physical documents and into the digital data space. Using OCR technology, entire document archives once relegated to file cabinets can instead be harvested for digital text. Doing so creates opportunities for non-digital text to be leveraged by other computer-enabled technologies.

This technology is not perfect, and it is difficult to translate character images into a digital text without some inaccuracies or errors. Image “noise,” which can occur if image resolution or coloring issues interfere with the characters, is one reason that OCR technology may have trouble reading character text.

## Solution Space

Building upon the efforts of reCAPTCHA, a better solution is needed to operate at sufficient scale utilizing new automated technologies. Google’s effort requires manual users across the world to use their service repeatedly to generate sufficient data to accurately transcribe a noisy word. Our Autoencoder is intended to allow for the programmatic denoising of a large data corpus, resulting in cleaner, computer-readable images that will foster higher OCR accuracy.

## Project Objectives

This project constructed a Convolutional Neural Network (CNN) autoencoder that attempts to reduce image noise during the OCR process. We evaluate the fidelity of our CNN by conducting OCR text extraction on a dataset before and after the implementation of our method. The project uses Python’s Tesseract library (i.e., PyTesseract), as well as Tensorflow, and Keras. We use Google Colab and AWS SageMaker computing resources. In addition to delivery of a prototype, the project team will evaluate our denoising model with respect to accuracy and performance. [2]; [3]; [4]; [5]; [6]; [7]; [8]

## Research

Research efforts for this project focused on the construction and employment of Convolutional Neural Network (CNN) autoencoders. The primary method of research was utilizing internet searches either from Google or George Mason University library services to view webpages, research papers, and textbooks.  
  
For Convolution Neural Networks, the primary resource was titled “Practical Convolutional Neural Networks” [9]. This gave an outline of CNNs and their applications either via supervised classification methods or unsupervised Autoencoders. Using this book, the team was able to identify an intuitive understand of how these networks derived value to apply to our Autoencoder. Additionally, the resource “Gentile Dive into Math Behind Convolutional Neural Networks” [10] provided insight into the mathematics of CNNs. This assisted the team in understanding the behind the scenes math that supported the code and the methodology applied in this model.

For Autoencoders, a critical aspect was reviewing other open data projects with similar project objectives. The main dataset in which CNN autoencoders are trained with is the MNIST, which contains 28x28 input images of various English-language characters. This MNIST dataset is the standard on which most open data project Autoencoders are tested and is outlined in the article “Convolutional Autoencoders for Image Noise Reduction” [11]. This resource helped formed a standard in which to compare our image denoising based upon a well-established dataset and methodology. The team utilized this resource as an initial template for autoencoder model construction.

## Primary User Story

Our team has developed the following user story to assist in the guidance of our project:

“As a user, we want to be able to process images into a something that a computer can easily digest for further use cases. For our users with an interest in taking older scanned documents and extracting data, they may run into several issues involving noise such as coffee stains, markings, or wrinkles that could impede OCR technology from performing optimally. Our user would need a product that could accurately and efficiently remove this noise and increase the ability of the OCR technology to capture the data into a computational form for further analysis. We imagine our product helping these users to enhance their OCR and data acquisition efforts.”

## Product Vision

### Scenario #1

One example of a use case for this product is to take old corporate SEC filings from the era in which typewriters were primarily the form of creating documents and transform this potential noisy text into useful text for computers. These documents may have markings that prevent a computer from being able to read into an OCR technology and valuable data could be lost in the translation. Unlike current OCR technology, our autoencoder is a pre-processing machine and ensures information is captured appropriately. Without an autoencoder, the process would have to entail a lot of manual reading and typing by a human which is costly in both man hours as well as invoking human error into the process. A trained autoencoder could provide confidence that historical data captured via OCR technology is useful for furthering the development of algorithms that determine the proper value of a security.

### Scenario #2

Another use care of this product is for a business that has medical records of patients before any Electronic Medical Record (EMR) was implemented. These patients’ records could have been typed in with a typewriter or could be old enough to sustain damage such as water stains and wrinkles. Doctors may want to use our product to scan these documents into computer text to assist them in understanding a patient’s medical history. These older medical records can then be integrated with current EMR technologies and used in variety of analysis such as medical research and enhancing diagnostics.

## Definition of Terms:

**OCR** (Optical Character Recognition): A technology that can recognize the printed and handwritten character from images in a real document.

**PyTesseract**: A wrapper for leveraging OCR. It can process different types of images, such as jpeg, png, gif, tif, etc.

**CNN** (Convolutional Neural Network): A deep learning algorithm which consists of one or more convolutional layers and one or more fully connected layers, which is similar with a standard multilayer neural network. CNN can detect and classify images.

**Encoder**: A network that accepts inputs and outputs feature vectors and can keep the feature and information of inputs.

**Decoder**: A network that takes the feature vectors from the encoder and outputs a result that is closest to the expected output.

**Kernel Convolution**: The process of applying a digital matrix to a part of an image and transforming it according to the values in the matrix.

**Rectified Linear Activation Function (ReL)**: A activation function introduces the nonlinear factor into the neural network, through which the neural network can fit various curves.

**Sparse Categorical Cross Entropy Loss:** a loss function for machine learning models that is the log loss, or probability between 0 and 1. Loss decreases as the predicted probability converges with the actual result. This specific cross entropy function considers the presence of only one category.

**Mean Squared Error (MSE) Loss**: A regression-based loss function for machine learning models. MSE is the sum of squared distances between our target variable and predicted values

# Data Acquisition

## Overview:

This project uses the Ryerson Vision Lab Complex Document Information Processing (RVL-CDIP) dataset [12]. This dataset is the product of academic research by Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. Their 2015 paper, “Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval,” researched the ability of convolutional neural networks to conduct state-of-the-art document image classification and retrieval [13]. RVL-CDIP is a subset of the IIT-CDIP Test Collection, which was used for a 2006 conference paper titled “Building a test collection for complex document information processing” [14]. The IIT-CDIP dataset is itself a subset of the Legacy Tobacco Document Library (TLDL), an archive of 14 million documents related to tobacco industry advertising, manufacturing, marketing, scientific research, and political activities. The full dataset is hosted by the University of California, San Francisco university library [15].

## Field Descriptions:

The data consist of 400,000 grayscale images and 16 classes with 25,000 images per class. All images are unique. Data is classified using the following labels:

|  |  |
| --- | --- |
| 0 – letter | 8 – file folder |
| 1 – form | 9 – news article |
| 2 – email | 10 - budget |
| 3 - handwritten | 11 - invoice |
| 4 - advertisement | 12 - presentation |
| 5 – scientific report | 13 – questionnaire |
| 6 – scientific publication | 14 - resume |
| 7 - specification | 15 - memo |

## Data Context:

This dataset was used by researchers at Carnegie Mellon University in 2015 to build a convolutional neural network focused on document image, classification, and retrieval.

## Data Conditioning

The data is stored as a downloadable file on Google Drive using a link provided on the dataset website. The data is zipped into a .tar file with two separate data sources, one for the images and one that provides information on labeling.

## Data Quality Assessment:

The RVL-CDIP dataset is an academic creation that is designed to lend itself to computer vision and machine learning efforts. For each image, the largest dimension is not to exceed 1000 pixels. All are image formats and are classified by type of document as well as a predefined train, test, and validation set.

The raw dataset has several outstanding issues that need to be mitigated before final analysis can occur. The first issue is that each image in the dataset is separated into individual file folders making it difficult to extract the image into a tidy corpus for analysis. The second issue is that the data is currently in .TIF format (Tagged Image File) and may be insufficient for analysis in further steps. The third issue presented is the sheer size of the dataset corresponding with the preferred data storage on Google Drive. Google Drive uploads the information slowly and either the data needs to be scaled down for analysis only or another tool needs to be research for proper uploading. Finally, the dataset contains a few “bad” images for our analysis. These images contain little or no text and almost all noise and need to be eliminated from our purposes.

* Completeness: The RVL-CDIP dataset includes 400,000 grayscale images in 16 classes, with 25,000 images per class. Also, it is split into 320,000 training images, 40,000 validation images, and 40,000 test images.
* Uniqueness: The dataset consists 16 different classes, and each image in each class is unique.
* Consistency – Images are sized so not exceed 1000 pixels for largest dimension.
* Integrity – All images are .tif (Tagged Image File), which may be challengeable for analysis in further steps.
* Conformity – All images are stored in a standard format.
* Accuracy – Data appears to be accurately managed given the quality of the source.
* Overall Quality: The quality of the dataset is high.

## Other Data Sources

We identified several additional datasets that may also be used for our project. These datasets vary in quality, volume of data, and usefulness toward our defined project scope.

# Analytics and Algorithms

## Convolutional Neural Network

This autoencoder model will be built using a CNN with unsupervised learning to reduce noise in various text images. A regular neural network contains three layers: an input layer, a hidden layer, and an output layer. The input layer is simply a vector of data that is passed through a series of hidden layers to produce a result in the output layer. A traditional neural network is show below:

A close up of a logo

Description automatically generated

Figure 1: A Neural Network

Source: Adapted from [9]

A CNN has a similar structure but differs by having three distinct layers: the convolutional layer, the pooling layer, and the fully connected layer. CNNs also almost always include images so spatial information and their relation to each other at the pixel level is important. This is the key aspect of the convolutional layer which reduces the image to smaller pixel vectors while retaining the relationship of those pixels to each other. This process is called filtering and the more filters used in the model the longer the model takes to produce a result. The next layer is called the pooling layer, which takes information from the convolutional layer and reduces the pixels even further. The process by which it reduces the image is by using max pooling by looking for which pixels have the highest value between them. The higher the value, the more important the pixel is to the image. The convolution and pooling layers continue until finally being flattened into one vector called the fully connected layer [9]. An example of a basic CNN is below:

A picture containing clock, drawing

Description automatically generated

Figure 2: A Convolutional Neural Network (Layers)

Source: Adapted from [9]

## Noisy Autoencoder

The autoencoder will use CNN to train and reduce image noise by reconstructing the image as an improved version. This means there are two steps in the training process. The first step is shrinking the images down into flattened vectors with the most important features remaining. This step is the encoder step. The second step, or decoding step, is to take the flattened vectors from the encoding step as an input. These inputs are then reshaped and ran through the convolutional and pooling layers to recreate an improved image. The pooling layers in the decoder instead of reducing pixels by max pooling increase the size of the image using Up sampling. This process produces an output of the improved original image. An example of the Autoencoder process is below:

A close up of text on a white background

Description automatically generated

Figure 3: Autoencoder Overview

Source: Adapted from [11]

## Convolution

Pixels are the building blocks of digital imagery. In digital graphics, the pixel is the basic and smallest unit. In simplest terms, pixels are colored squares. Each square’s color is usually based on the intensity of basic or primary colors.

Pixels have no defined size. Instead, the quantity of pixels in a digital image is defined by its resolution, or how many pixels are found in a certain area. For example, a five-megapixel image (or consisting of five million pixels) may occupy the same area as a 10-megapixel image, but the 10-megapixel image would have twice the resolution as the five-megapixel image. Convolution attempts to take entire sections of colored pixels and translate it into numbers that can be used by a computer. [16]



Figure 4: Digital Image Data Structure [10]

Kernel Convolution is the process of applying a matrix of numbers to portions of the image, transforming it as you go based on the values in the matrix. This creates a feature map of the image by using the below formula (see Figure 5). The input image (f) and the matrix, known as a kernel (h), are applied to the pixel vector space rows and columns (m and n). As the function steps through the pre-defined row-column space each time, the pixel values are matrix multiplied with the kernel value and the resulting number is added to the now-reduced feature map (see Figure 6). This process is considered one convolution. [10]

Figure 5: Convolution equation

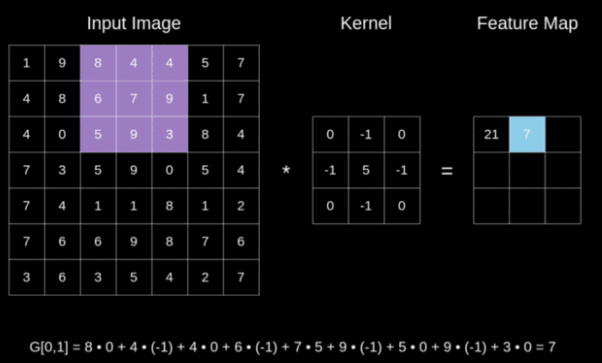


Figure 6: Digital Image Data Structure [10]

Next, in order to make use of the new feature map, we need to apply an activation function. The role of the activation function is to turn the summed feature map into an output for that node. There are several activation functions used in CNNs. We chose to use the rectified linear activation function (ReL), which is a nonlinear function that uses stochastic gradient descent with backpropagation. ReL is popular because of its built-in gradient descent/backpropagation learning methods, but also because it acts both linear and nonlinear in nature. For values greater than 0, the function performs linearly, making it easier to understand and optimize. However, it also acts nonlinear for values less than 0 by creating a resulting output of 0. The simple statement, “If input > 0, Return input; else Return 0” sums up the idea behind ReL. A convolutional node that uses the ReL function is known as a Rectified Linear Unit (ReLU). Neural networks that leverage RELU are often called Rectified Networks. [17]

Each convolution in a neural network follows the same process, but each additional convolution does not have to use the same rules. These tunable features, or hyperparameters, are used to refine and direct the convolutional process based on need. One such parameter, Step length, or Stride, sets the rules for how many pixels the kernel will shift for each calculation. This will effectively set the spatial dimensions of the feature map for that convolution. The filter size, or Kernel size, is also considered a hyperparameter. [10]

# Results

We constructed several different models with a diverse set of parameters in the beginning, eventually settling on three model variants: the ad200, ad3.5K, and ad10K models. Each model was constructed using a combination of parameters that we believed most useful in our initial research. Figure 7 shows the basic parameters of each model, along with the size of our training set and the average improvement along three performance areas: visual quality, loss, and character recognition.

|  |  |  |  |
| --- | --- | --- | --- |
| PARAMETER | ad200 | ad3.5k | ad10k |
| training iterations | 200 | 3500 | 10000 |
| Encode Convolutional Layers | 3 | 4 | 4 |
| Encode Pooling Layers | 3 | 4 | 4 |
| Decode Convolutional Layers | 4 | 5 | 5 |
| Decode Upsampling Layers | 3 | 4 | 4 |
| Dense Layers | 1 | 1 | 1 |
| Latend Dimensions | 196 | 512 | 196 |
| Image size | 216x216x3 | 1000x504x3 | 1000x504x3 |
| training images | 19000 | 2000 | 2000 |
| validation images | 2500 | 600 | 600 |
| loss function | sparse categorical cross entropy | mean squared error | mean squared error |
| activation function | ReLU | ReLU | ReLU |
| optimizer | adadelta | adadelta | adadelta |
| Visual Performance Indicator | Blurry | Clear (larger font) | Clear (larger font) |
| Loss Performance Indicator | Start:18 / end:0.1 | Start:0.09 / end:0.0114 | Start:0.09 / end:0.0109 |
| Max char recognition over original | -null | +2 words | +4 words |

Figure 7: Model construction efforts chart

Our smallest ad200 model was trained using a large amount of training images with reduced. We leveraged local compute resources and a learning model consisting of 3 convolutional encoding layers and 4 convolutional decoding layers. Initial results were promising, and Figure 8 shows our initial loss improvement over the 200 training iterations. Although loss improved, the resulting images proved blurry when compared to the original and lacked any ability to properly extract text via OCR processing.

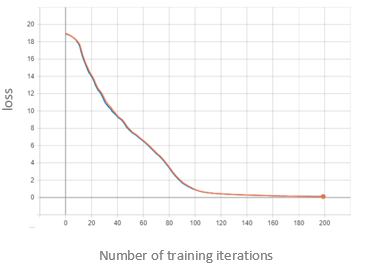


Figure 8: Original training loss – ad200 model

Our experience with the ad200 model reinforced our belief that we needed larger computing resources and the ability to process an image without distorting its shape. We secured Amazon SageMaker machine learning-optimized computing resources and increased the resolution of our training image set to near its original image size. We added layers to our model and increased kernel size (see Figure 7). The result was our ad3.5K and ad10K models. These models are mostly similar with two distinguishing characteristics, the number of latent dimensions and size of applied compute resources. Figure 9, next page, shows a layer-by-layer summary of our ad3.5k model.

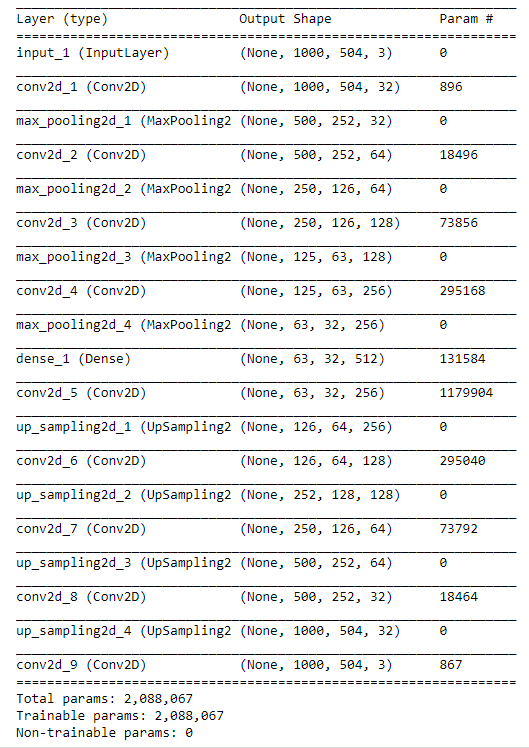


Figure 9: Model Summary – ad3.5k model

Both the ad3.5K and ad10k models performed better than the ad200 model. Images were clearer and both achieved a loss function of near .011 before training completed (see Figure 7 for specifics). Importantly, both models also allowed for more accurate OCR results when compared to the un-processed reference image. The examples below each show our model’s evolution. From left to right, we see the reference (original) image, the ad200 result, ad3.5k result, and ad10k result.

The example in Figure 10 shows a result of +1 accurately extracted words when compared to the reference image. This was achieved by our longest-trained model variant, ad10k. It also shows that our other two models faired poorly when compared to the reference image. It was often hard to visually distinguish between the ad3.5k and ad10k image renderings, however the ad10k model performed better despite visual similarity. We found this an interesting point in our research.

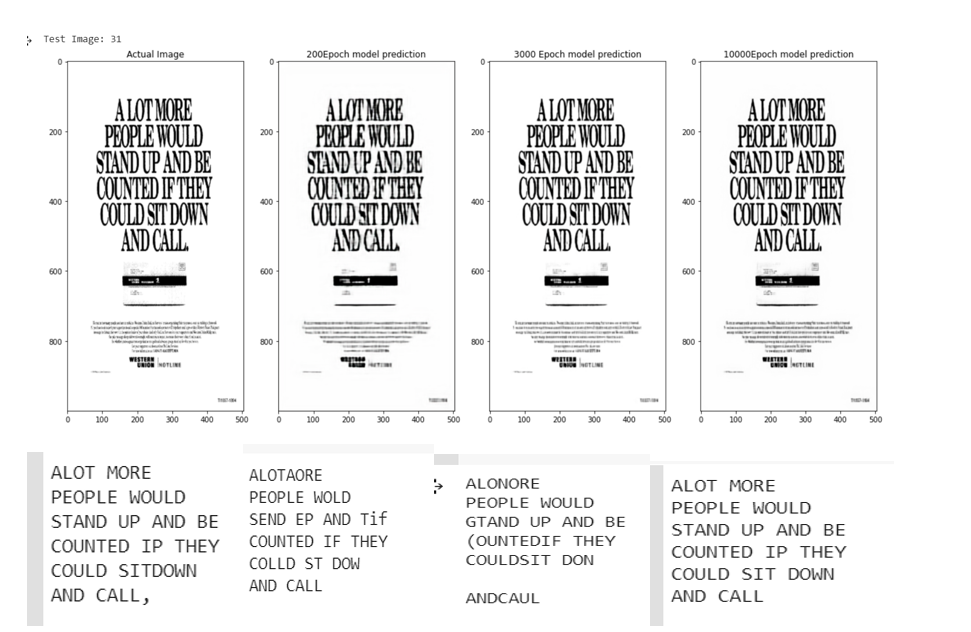


Figure 10: Example Comparison 1

We also found that our models all failed to accurately identify text when font size was small. In many of the advertisements we used to assess our models, larger text was easier to identify in both the reference image and processed images. We believe that we can ultimately achieve accurate identification at smaller fonts with more training effort, including the resizing and segmented training of our dataset – something we did not attempt as part of this effort.

Our second example resulted in +1 words identified with our ad3.5k model and +2 words identified with our ad10k model when compared to our reference image. Example 3, seen in Figure 12, provided a +4 words identified by the ad10k model when compared to the reference image.

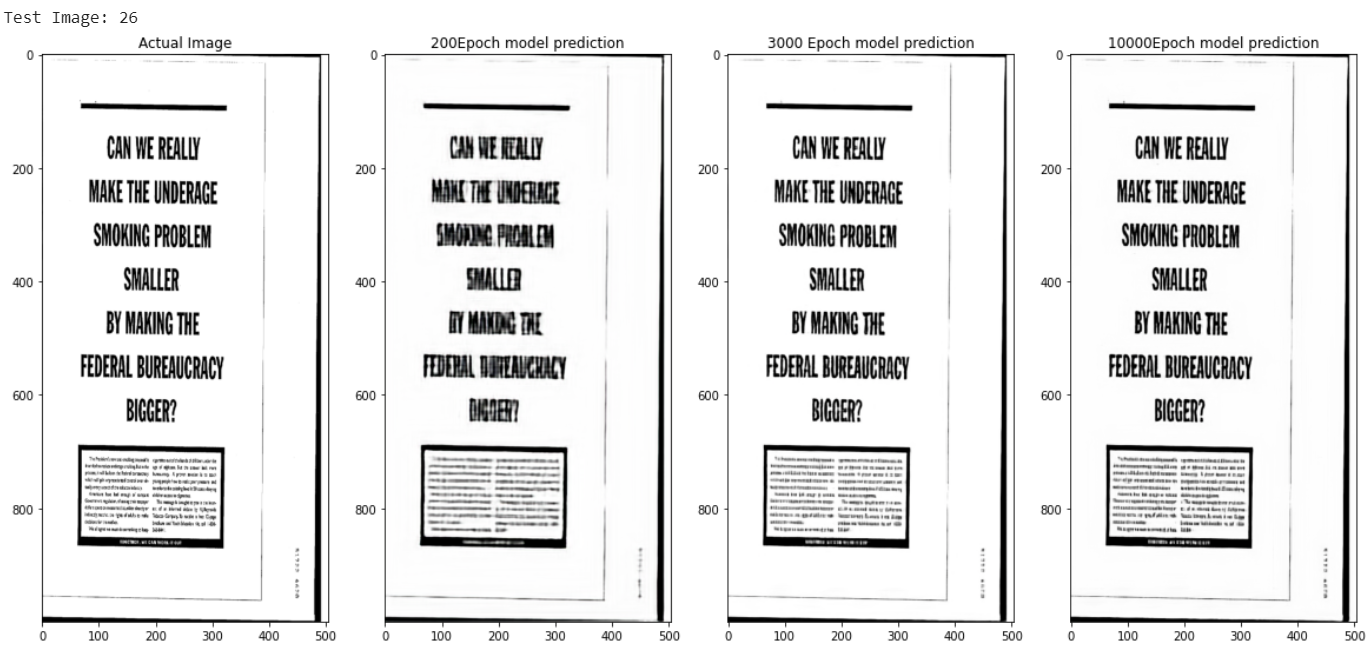


Figure 11: Example Comparison 2



Figure 12: Example Comparison 3

# Summary

This project aimed to construct an autoencoder capable of improving the identification of English-language text in scanned documents. With some effort, we were able to show incremental improvement over the original image by applying open-source OCR identification tools. We also identified several opportunities for improvements to our model, including additional training time, larger training datasets, and focused training on resized image segments. We also leave the door open to refinements made to kernel size and activation function.

# Future Work

The team recommends three main items to build upon the findings of this paper for future work. The first item is to improve the results of the Autoencoder by splitting the images into smaller pieces and running a CNN on those smaller images. This not only will improve the speed at which the images are processed but also has the potential to improve the denoising process. The second item is to continue adjustment of the parameters to optimize and improve the neural network. There are a multitude of parameters to adjust such as filter size, latent dimension space, number of training images, number of layers and optimizers. Experimentation is critical to building a successful Autoencoder and continued efforts to improve and optimize could result in a more sophisticated model to achieve the desired results. Finally, in our dataset there were 16 different document types and any future model could investigate and build models for each document type. These document types have different use cases and could add additional value to the end user. There is also a reasonable chance that certain document types could have worked better using the existing model and further experimentation would help determine a better solution.

# Lessons Learned

There are two primary lessons that we learned during the project that would help future projects achieve greater success. The first lesson is that running experiments on parameters on a smaller subset of the data is crucial to tuning the final model. A lot of time was spent training and running the algorithm on the full dataset only to realize that something was not correct or optimized appropriately to derive a useful result. Given the amount of training time, particularly before utilizing AWS, this created bottlenecks that took a large amount of time to correct. Our group started achieving success when smaller sets of data were used to train the model and produce results. The best aspects of these small experiments were added to the main model which produced a better version of the results.

The second lesson is to understand how difficult and unique the problem is and set expectations accordingly. At the beginning of the project, the group was optimistic that we could take examples of previous attempts at Autoencoders, apply those methods to our project, and build the perfect solution. However, we quickly realized that the only real examples of Autoencoders involved the MNIST dataset, which was too clean of a dataset to apply to our messy group of documents. Given this, there also was not a significant amount of solutions to errors in the code available on such websites as Stack Overflow or Google. A lot of the errors were discovered but no solutions were derived which created significant bottlenecks to our result. Our team determined that potentially the reason why there is a lack of real-world examples is because the resources to properly run something on a dataset like ours is too expensive for public experimentation. Our hope is that future teams can use this project and its results as an example for future autoencoders.

# Appendix

## Appendix - GitHub

Project documents, code and research can be obtained at the following GitHub repository:

<https://github.com/Mmachad/autoencoder>

## Appendix – Project Risk Matrix

A screenshot of a cell phone

Description automatically generated

## Appendix – Agile Workflow Summary

This Project was tracked and managed using an Agile development workflow application called YouTrack. The summary of the project sprints, dates, and task accomplished is below:



# References

|  |  |
| --- | --- |
| [1] | L. v. Ahn, B. Maurer, C. McMillen, D. Abraham and M. Blue, "reCAPTCHA: Human-Based Character Recognition via Web Security Measures," *Science Magazine,* 12 September 2008. |
| [2] | C. G. K. S. I. Wiraatmaja, "The Application of Deep Convolutional Denoising Autoencoder for Optical Character Recognition Preprocessing," in *International Conference on Soft Computing, Intelligent System and Information Technology*, 2017. |
| [3] | L. H. X. Zheng, "Character Segmentation for License Plate Recognition by K-Means ALgorithm," *G. Maino and G.L. Foresti (Eds.): ICIAP,* vol. II, no. LNCS 6979, pp. 444-453, 2011. |
| [4] | J. L. Wei, "Autoencoders: Neural Networks for Unsupervised Learning," Intuitive Deep Learning : Medium, 18 Febuary 2019. [Online]. Available: https://medium.com/intuitive-deep-learning/autoencoders-neural-networks-for-unsupervised-learning-83af5f092f0b. [Accessed 6 June 2020]. |
| [5] | A. Rosebrock, "Using Tesseract OCR with Python," Pyimagesearch.com, 10 July 2017. [Online]. Available: https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/. [Accessed 6 June 2020]. |
| [6] | J. P. S. G. S. G. H. S. D. Shriansh Srivastava, "Optical Character Recognition on Bank Cheques Using 2D Convolution Neural Network," *Applications of Artificial Intelligence Techniques in Engineering. Advances in Intelligent Systems and Computing,* vol. 697, pp. 589-596, 2018. |
| [7] | A. I. A. Muna Ahmed Awel, "REVIEW ON OPTICAL CHARACTER RECOGNITION," *International Research Journal of Engineering and Technology,* vol. 6, no. 6, p. June, 2019. |
| [8] | M. M. N. M. Roland Graef, "A Novel Hybrid Optical Character Recognition Approach for Digitizing Text in Forms," *Lecture Notes in Computer Science,* vol. 11491, 2019. |
| [9] | R. a. P. P. Karim, Practical Convolutional Neural Networks, Packt Publishing, 2018. |
| [10] | P. Skalaski, "Gentile Dive into Math Behind Convolutional Neural Networks," Towards Data Science, 12 April 2019. [Online]. Available: https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9. [Accessed 11 July 2020]. |
| [11] | D. Dataman, "Towards Data Science: Convolutional Autoencoders for Image Noise Reduction," 20 November 2019. [Online]. Available: https://towardsdatascience.com/convolutional-autoencoders-for-image-noise-reduction-32fce9fc1763. [Accessed 6 July 2020]. |
| [12] | A. U. a. K. G. D. Adam W. Harley, "The RVL-CDIP Dataset," Carneigie Mellon University Department of Computer Science, 2020. [Online]. Available: https://www.cs.cmu.edu/~aharley/rvl-cdip/. [Accessed 21 June 2020]. |
| [13] | A. U. a. K. G. D. Adam W. Harley, "Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval," *ICDAR,* 2015. |
| [14] | G. A. S. A. O. F. D. G. a. J. H. D. Lewis, "Building a test collection for complex document information processing," in *Proc. 29th Annual Int. ACM SIGIR Conference*, 2006. |
| [15] | University of California, San Francisco, "The Legacy Tobacco Document Library (LTDL)," 2007. [Online]. Available: http://legacy.library.ucsf.edu/.. [Accessed 21 June 2020]. |
| [16] | A. Mireles, "How to Understand Pixels, Resolution, and Resize Your Images in Photoshop Correctly," Digital Photography School, [Online]. Available: https://digital-photography-school.com/understand-pixels-resolution-resize-photoshop/. |
| [17] | J. Brownlee, "A Gentle Introduction to the Rectified Linear Unit (ReLU)," Machine Learning Mastery, 9 January 2019. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/. [Accessed 11 July 2020]. |