SMU Data Science Review

Volume 3
Number 3 *Fall 2020*

Article 1

2020

Reading PDFs Using Adversarially Trained Convolutional Neural Network Based Optical Character Recognition

Michael B. Brewer
Southern Methodist University, mbbrewer@smu.edu

Michael Catalano
Southern Methodist University, mcatalano@smu.edu

Yat Leung
Southern Methodist University, yleung@smu.edu

David Stroud
Troy University, jdstroud@troy.edu

Reading PDFs Using Adversarially Trained Convolutional Neural Network Based Optical Character Recognition

Blaine Brewer, Michael Catalano, Yat Leung, David Stroud
[1] Master of Science in Data Science, Southern Methodist University,
Dallas, TX 75275 USA
[2] Kimbell Royalty Partners, 777 Taylor Street, Suite PII-C,
Fort Worth, TX 76102 USA
mbbrewer@smu.edu, mcatalano@smu.edu, yleung@smu.edu, jdstroud@troy.edu

Abstract. A common problem that has plagued companies for years is digitizing documents and making use of the data contained within. Optical Character Recognition (OCR) technology has flooded the market, but companies still face challenges productionizing these solutions at scale. Although these technologies can identify and recognize the text on the page, they fail to classify the data to the appropriate datatype in an automated system that uses OCR technology as its data mining process. The research contained in this paper presents a novel framework for the identification of datapoints on check stub images by utilizing generative adversarial networks (GANs) to create stains that are superimposed onto images which are used to train a convolutional neural network (CNN). For this project, the MNIST dataset is used as a proxy for validating the effectiveness of our approach. A baseline CNN is used to recognize text from unperturbed images, and the results are validated with 97.38% accuracy. Once the perturbations are introduced to the baseline CNN, the accuracy dips to 94.7%. The results from the adversarial-trained data are favorable, with an accuracy of 97.3%, roughly a three-percentage increase in the ability to properly identify the character in an environment with perturbed images.

1 Introduction

Data entry is a task that anyone can be trained to do. Even the most digitally focused business today still relies on teams of individuals manually input data into their systems. Although this method is highly effective, it is usually very costly to employ or

contract the people necessary to manually extract the data from a physical document and input into a digital system. Even though Optical Character Recognition (OCR) has been available to the public for years (while many are highly performant and extremely accurate at identifying characters on the page) they fall short of being able to classify the data points on the page as a mechanism to associate with fields in a database. Once the analyst uses an OCR algorithm to identify the characters on the page, they are still tasked with the manual process of determining which fields those characters belong to and physically placing that data in the correct location. Additionally, a certain level of domain knowledge is required to ensure the data is going to the correct place. If the data the company is receiving is homogenous, meaning that the format never changes, then it would be a relatively simple task to build a rules-based system to capture the data and store it based on the text's location within the document. However, if the format of the data changes over time or the formats of the documents vary in any way, building a rules-based system would be virtually impossible.

The research team is stewarding a project of a publicly traded Oil and Gas company named Kimbell Royalty Partners (KRP) to build a proof of concept of a custom OCR process. KRP receives check stubs that contain essential business data from various operators for the wells in which KRP owns an interest. KRP's ultimate goal is to create an automated process to read the check stub data, identify which data points are necessary, properly categorize those data points to their respective fields, and format the data in such a way as to be easily imported into the KRP accounting system. Since the checks received are not always without defects, the system needs to be robust enough that perturbations in the check stub images will not negatively impact the automated process. Each check stub is not only a record of payment, but also contains crucial information associated with the wells associated with that payment. The check stubs contain pertinent well, production and revenue data, usually formatted to contain data from multiple wells. The well operators, that produce and send the check stubs, are only obligated to send the payment and pertinent data associated with the payment. There is no operational advantage to share well data with the royalty owner in a standard format, or in such a way that would make automating the data input an easy task. The data points that identify a check stub are operator name, payee name, check number and check date. The additional data points that are required on the check stubs are well names, production dates, unit prices for the production volumes, gross production volumes, net production volumes, gross revenues, net revenues, production tax, and revenue deductions.

Currently, KRP employs a large team of contractors with the sole purpose of hand entering data from the check stubs into a format that is easily imported into KRP's accounting system. Not only does this contract labor very costly, averaging $47,000 per month, but the contractors often make data entry errors which can have a direct impact on financial decisions for KRP. KRP would like to develop a system to replace the manual entry method, that automatically reads the data from the check stubs and formats this data for import into their accounting system. KRP currently receives checks from more than 1,600

different operators and the formats of the check stubs vary from operator to operator (Fig. 1 & 2). Even for an individual operator, the formats of the checks can change over time. An additional complication, because the check stubs are scanned images, is that the images are not always high-quality images that are free of defects. These defects, or perturbations, can range from creases created during scanning to coffee stains inadvertently placed on the check stubs prior to scanning. Although it may be easy for a human operator to make sense of the data disturbed by a crease or coffee stain, a machine lacking intuition will fail at the same task. For these reasons, a simple rules-based system that maps different operators with different formats and extracts data from specific areas of the page is not a sufficient replacement for the manual method. Based on the requirements set forth by KRP, a more intelligent system needs to be developed.

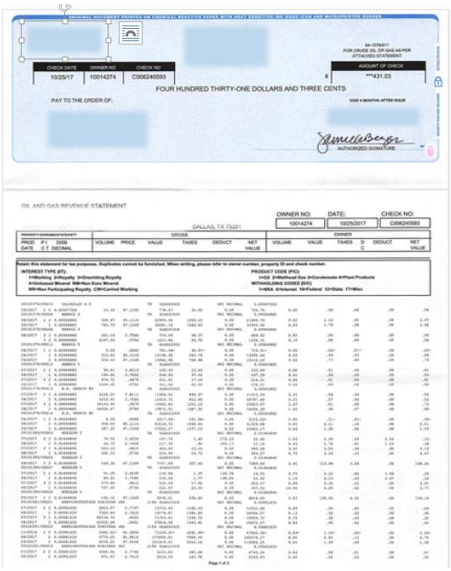

Fig. 1. Sample Check from one of KRP's Operators

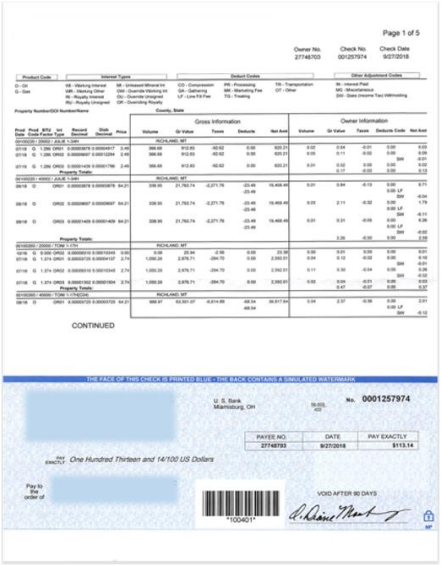

Fig. 2. Sample Check from Different KRP Operator

2 Related Work

2.1 Optical Character Recognition (OCR)

The origins of Optical Character Recognition can be dated back to the eve of World War I when physicist Emanuel Goldberg invents a machine that can convert text to telegraph. Emanuel Goldberg would continue his development and eventually receive the patent for the “Statistical Machine" in 1931, which was an electronic document retrieval system that performed search and retrieval of microfilm by using a photoelectric cell to perform pattern recognition [1]. Even with the advancement of technology over the coming decades, early OCR processes could only perform character recognition on single fonts at a time. Therefore, the characters would need to be an exact match as the templates provided by the source font. It wasn’t until the 1970’s that an “omni-font OCR," which could read text from virtually any font, would be invented [1]. Although more robust, “omni-font OCR" still relied on matrix matching, which decomposes a character into a matrix of values and tries to match that matrix to a catalog of stored matrices. Although still limited, the “omni-font" advancement would eventually lead to the commerciality of OCR technology in the late 1970’s and early 1980’s, when Kurzweil Computer Products developed the

Kurzweil Reading Machine and subsequently making the technology commercially available [24]. OCR technology is now freely available on the internet, offered by companies like Google and Amazon, and accessible via API's in many programming languages.

2.2 MNIST Database

The 60,000 images of the modified National Institute of Standards and Technology (MNIST) data set have become the standard data set for machine learning algorithms whose goal is text and character recognition. Each of the 60,000 characters is contained within a 28x28 pixel image for a total of 784 pixels, making the data easy to access and leverage for text recognition problems [4] [15]. LeCun & Cortes (2010) have categorized different machine learning techniques into six groups with the intention of identifying the best models for text recognition. Of all the techniques tested, which includes linear regression, k-nearest neighbors (KNN), boosted stumps, non-linear classifiers, support vector machines (SVM), and convolutional neural networks, the latter was the most performant with a test accuracy of 99.77% [15].

2.3 OCR and Feature Extraction

Current OCR technology is based on feature extraction, where machine learning models can learn character patterns by decomposing each letter into a series of pixels. From this pixel decomposition they can detect edges, loops, and lines within the matrices being processed [24]. One key improvement of OCR technology, using feature extraction, is that now OCR processes can identify handwritten text within documents. Unrestricted by font templates, text in documents can be recognized with single implementations, no matter if the text is a hybrid of written and typed text.

Using text data, a comparison study was performed by Palka & Palka (2011), where they reported a significant misclassification rate using a neural network (NN), a 10% error during k-fold cross validation and a 50% error rate on the test data. Palka & Palka (2011) propose that input pattern displacement, scaling, and rotation contributed to the difference between train-test classification success. Meanwhile, a convolutional neural network (CNN) model was able to decrease the test data error rate by 11%. The performance gain came at a cost, as the new model required a 25-fold increase in processing time [18]. Since KRP's primary objective is a highly accurate model and model processing time is a secondary concern, a CNN approach seems more appropriate in this use case.

2.4 OCR and Neural Networks

Figuring out the machine learning algorithm to perform the OCR function is only half the battle. Once the technique is selected, exhaustive training must be performed to determine the model structure and hyperparameters that best suit the problem at hand. In one notable case using CNN's to perform an OCR process, Palka & Palka (2011) used a deep learning system, with a tiered approach, for the classification of characters. The first tier determined the detection probabilities of the characters from the training images. The following layer applied Long Short-Term Memory (LSTM) to increase the decoding accuracy from learning a data driven vocabulary. Four CNNs using different identification processes: Simple Nearest Neighbor (SNN), Spatial Transform Net (STN), AlexNet, and AutoML were used to determine the characters. Finally, three different approaches were used at the LSTM layer to obtain the highest accuracy of 99.84%, which greatly outperforms non-optimized CNNs. In another case, a CNN based OCR was utilized and refined by Mahpod & Kelle (2020). The researchers noted the intricacies and issues required to overcome differences in sample scripts. Whether this was due to multiple print houses, damage to pages, fold lines and skewed text. The resulting model which included a stacked network architecture which included a CNN for visual inference, a LSTM to learn the dialect, and a final CNN trained on the specific book being researched in order to increase OCR accuracy. The resulting model achieved an accuracy of 99.8%. To see these levels of accuracy, conventional thinking dictates that the environment needs to be closed and relatively pristine. Additionally, exhaustive investigation needs to be performed to identify the structure and parameters that best suit the data. Although a final production model will require this level of accuracy, the proof of concept just needs to validate that the workflow is possible.

2.5 Challenges with OCR

The primary challenge that KRP faces and the research team aimed to solve was the effect of check stub perturbations. These perturbations; whether stains, creases, blurring or other forms of defects are a valid concern for a generic OCR system. An antiquated system using a font matching algorithm would be unable to identify letters with a stain or crease that spans the text. Many people might incorrectly assume that a machine learning based OCR system would be capable enough to learn what are stains and what are characters and find a way to circumvent any negative effects from the stains. In the paper *Intriguing Properties of Neural Networks*, the authors explored and substantiated a property of neural networks that contradicted the conventional understanding of deep neural networks at the time. A fatal flaw in neural networks is that perturbed samples, termed "adversarial examples" by the authors, can be specifically engineered to trick the network into misclassifying or failing the classify altogether [28]. Samples that were previously classified correctly could have slight modification made before being passed back to the model for validation and completely fool the network. The machine learning community, prior the research

conducted by Szegedy et al (2013), believed that both the structure of neural networks and the amount of data used to train them would provide all the defense needed and small changes to samples would have little to no effect. Their work provided ample evidence that perturbed samples, regardless of hyper-parameter settings or network structure, were incorrectly classified after small, calculated perturbations were made. Moreover, the authors performed the testing with various deep neural network models on several different image datasets, suggesting that neural networks are generally susceptible to "adversarial examples" and the problem was not attributed to a specific type of model or dataset [28].

2.6 OCR Using Adversarial Training

In a similar work that this paper inspired, the researchers developed a method of adjusting an image that would result in a 97% misclassification rate. What is even more impressive about this study is that the average adjustment required to produce a 97% misclassification rate was only a 4% image modification [19]. Goodfellow et al. (2014) coined the termed "adversarial attack," which describes the process of incorporating minor perturbations into the input samples with the goal of causing a misclassification or failure to classify by the neural network model. Huang et al. (2017) examined two types of black-box adversarial attacks: transferability across policies and transferability across algorithms. The real concern with the transferability property of adversarial attacks is that this points out that attacks do not need to be specifically designed to bring down a network. In short, the research team believes, that any unexpected perturbation or perturbations not included in training can have a negative impact on the results of the network.

Papernot, et al (2015) identified two different methods of defending against adversarial attacks: improving the training phase of the target network or identifying the adversarial samples as part of the output phase of the network. Other research by Chen et al. (2017) suggested the idea of null labeling where the adversarial examples were discarded without attempting to classify them into their original labels. The model would train to separate clean and perturbed samples, however only the clean samples were able to get classified back into their original labels. Unfortunately, KRP's algorithm would need to leverage the data contained within the perturbed images. The more viable approach for the KRP algorithm would be training the CNN model with adversarial examples to make the algorithm more robust to potential perturbations. With adversarial training, the goal is to develop a resistance to adversarial examples which decreases the effects of adversarial examples. Goodfellow et al. (2014) proved this theory by decreasing the error rate in a deep neural network from 89.4% to 17.9% using adversarial training in a controlled experiment.

3 Fundamental Concepts

Although the tested version of the OCR pipeline was a stripped down proof of concept, the productionized version should have the following structure: A GAN model should generate stains from random samples of all the common possibilities of defects that could be found on check stub images in the real world. GAN generated images will be superimposed on a randomly selected subset of training images. The altered images will be randomly introduced with unaltered images for the custom OCR process based on a CNN model to prepare the CNN model for perturbations in a production setting. After the CNN model correctly identifies and recognizes text, the text will be correctly classified into the correct data type by a deep neural network model using both the recognized text and positional metadata stored during the OCR process. A physical representation of the model can be found in figure 3 and more detail on the individual parts of the pipeline can be found in sections 3.1 through 3.3.

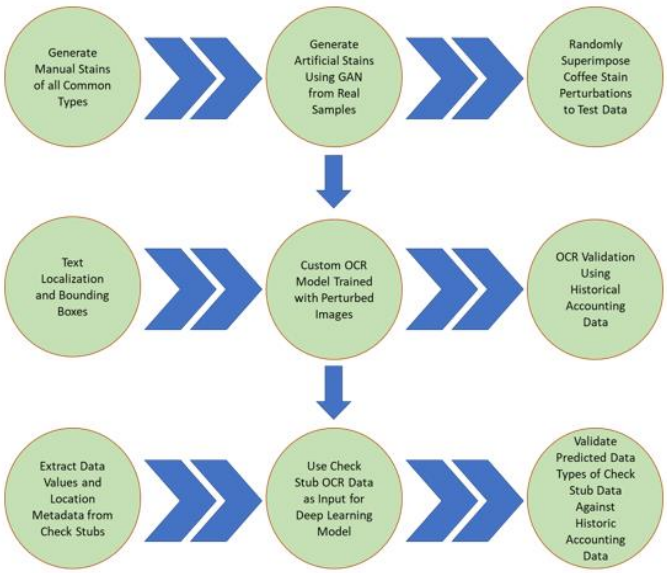

Fig. 3. KRP's Hypothetical OCR Pipeline

3.1 Generative Adversarial Networks (GANs)

In adversarial training, the traditional GAN pits two neural networks (a generator and discriminator) against each other. The generator produces fake images using a noise component with the goal of producing an output indistinguishable from real data. The discriminator network makes the determination if the image is real or fake. In the context of this research, the generator network produces an artificial coffee stain by pairing the knowledge of the known distribution of the real samples with the adjusted weights and biases calculated from the reduction of the loss function. Randomly the algorithm passes real and fake stains to the discriminator network for a decision and iteratively learns to trick the discriminator network (Fig. 4). Once the generator network learns to consistently trick the discriminator network, the coffee stains produced from the generator network are even convincing to the human eye.

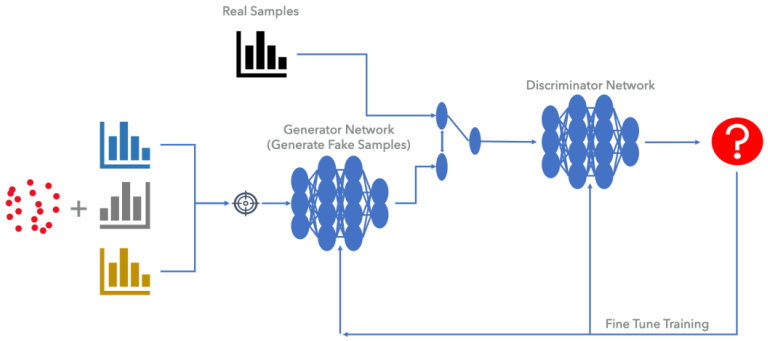

Fig. 4. GAN Structure [20]

3.2 Convolutional Neural Networks (CNNs)

Convolutional neural networks have been shown to be effective in image classification and image recognition tasks. Like traditional neural networks, CNNs consist of neurons with weights and biases that are learned during training. Each neuron receives several inputs, takes a weighted sum, passes through an activation function, and responds with an output. CNNs start to deviate from vanilla neural networks with the inclusion of a convolutional layers, the main building block of a CNN (Fig. 5).

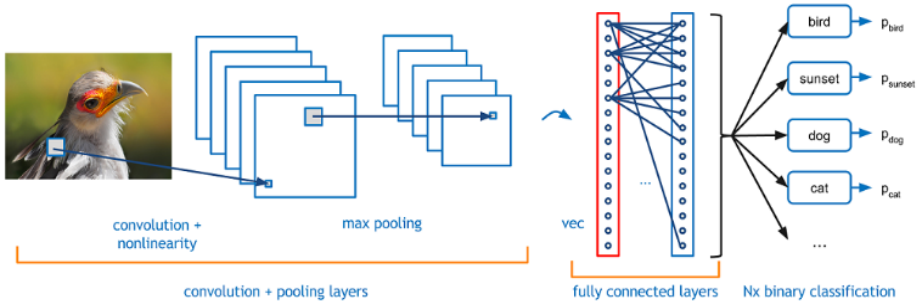

Fig. 5. Sample Convolution Layer of CNN [4]

In the convolution layer, a matrix is convolved with a filter, or kernel, using a sliding window technique, the product of the input matrix and filter matrix produces a feature matrix. This is also known as element-wise multiplication [5]. The resulting feature matrix becomes smaller as more information is discarded and aggregation is performed on the previous layers. Additional steps concerning the ReLU layer and pooling stages of the model are ways of enhancing the CNN's ability to both increase the non-linearity and spatial variance of the image. Pooling also serves to reduce overfitting of the model since it minimizes the size of the image. In the final phases of the convolutional layers, flattening occurs, creating a long vector of input data that is passed to a neural network for classification.

3.3 Optical Character Recognition (OCR)

OCR can be simply explained as using an application to digitize a document and breaking down that document into the individual characters that makeup the entirety of the document. These individual characters can be made up of either hand-written or printed text. The harder of the two to identify is hand-written text. This is because the characters in hand-written are not formed as a composite of templates of individual characters. A hand-written "a" will be different every time no matter how hard the creator tries for them to be the same. Other issues that OCR must overcome are imperfections added after the document was created. These imperfections include such things as folds, stains (coffee, ink, dirt, etc.), smudges, blurring, and physical deteriorations.

OCR is broken down to six major phases: image acquisition, pre-processing, character segmentation, feature extraction, classification, and post-processing. The description and different approaches can be found in Table 1.

Table 1. Major Phases of OCR System [12]

Phase	Description	Approaches
Acquisition	The process of acquiring image	Digitization, binarization, compression
Pre-processing	To enhance quality of image	Noise removal, skew removal, thinning, morphological operations
Segmentation	To separate image into its constituent characters	Implicit Vs Explicit Segmentation
Feature Extraction	To extract features from image	Geometrical feature such as loops, corner points Statistical features such as moments
Classification	To categorize a character into its class	Neural Network, Bayesian, Nearest Neighborhood
Post-processing	To improve accuracy of OCR results	Contextual approaches, multiple classifiers, dictionary-based approaches

4 Methods & Results

In order to measure the negative impact of potential defects on a CNN model not trained to handle perturbations, the research team developed a baseline CNN that included a 3x3 convolutional layer with a rectified linear activation function (ReLU), max pooling to a 2x2, a flattening layer, a hidden layer with 100 nodes that employed ReLU activation, and a 10-node output layer which corresponds to the 10 possible digits for output. Using the baseline CNN and unaltered MNIST data the research team was able to build a model with a mean 5-fold cross validation score of 98.67%. Using the predetermined 10,000 image test set from the MNIST database, the baseline CNN model achieves a baseline test validation performance of 97.38%. With the baseline model in place the next step was to introduce perturbations to the model. The entire validation pipeline is shown in figure 6.

4.1 Process Pipeline

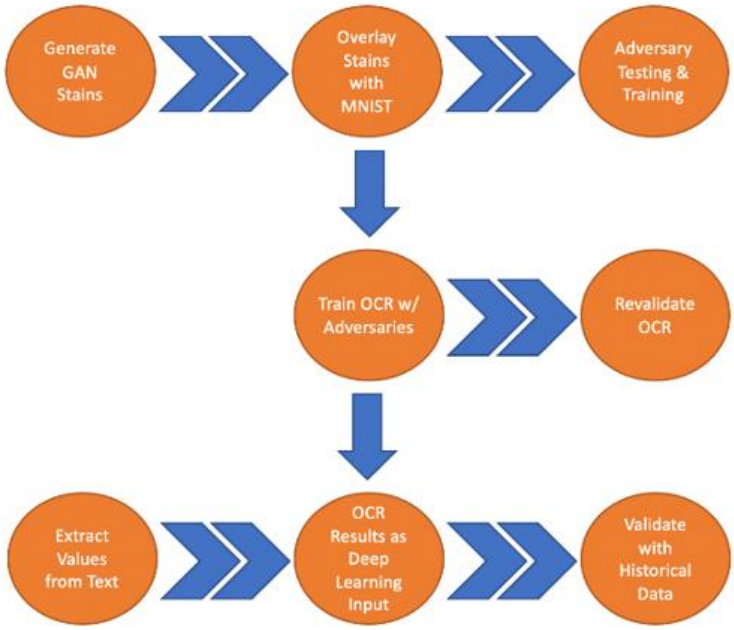

Fig. 6. Adversarial OCR Validation Pipeline

4.2 GAN Image Generation

The GAN used to generate artificial coffee stains was trained on 179 images of manually generated stains (Fig. 7). These stains were created by team members taking various circular objects and placing them in coffee and then marking a piece of paper that was later scanned. The coffee stains ranged slightly in size and varied in placement in order to create a realistic distribution of stains which would ultimately be learned by the GAN model.

Fig. 7. Manually Generated Coffee Stains

Preparing the GAN model involved setting up an environment in Google Colab, importing the 179 training images, enumerating them, resizing the images to a 64x64 pixel format, and converting the images to a numeric array. The full GAN model uses the array representation of the digital images as an input. The generator network uses the input arrays to learn the distribution of the real samples and create images that are similar to the real images. Once the generator network has created artificial images, these images are randomly combined with the real samples and introduced to the discriminator network for a binary prediction of real (class 1) or fake (class 0).

Generator networks typically use several decomposition layers to create more representative images from lower quality array representations. Upsampling, as it is called, is a simple way of scaling up an image to make it appear smoother. Technically, it takes a point in the latent space (Gaussian distribution with = 0 and = 1) and outputs a single 64x64 color image by decoding the representation to the size of the output image [29]. In our Keras model, the UpSampling2D framework is combined with Conv2DTranspose with the goal of reconstructing the coffee stain images into their original sizes without losing too much detail. The Conv2DTranspose layers use the LeakyReLu activation function and a hyperbolic tanh function in the final output layer. The Adam optimizer is used with an adaptive learning rate of 0.00015 and exponential decay rate (momentum) of 0.5.

At first the discriminator network is very good at determining whether the images are real or fake, but the generator model iteratively adjusts based on the discriminator's loss function and starts to produce images that are more realistic by learning how to trick the discriminator network. After several thousand epochs, the artificial images produced by the generator model are indecipherable from the real images. The architecture used for the GAN model is comprised by a Keras sequential framework that consists of an input layer, five convolutional 2d layers, a 25% dropout layer, and an output layer with sigmoid activation.

The final component combines the discriminator and generator networks into one holistic model, which is used to train the weights in the generator and calculate the error rates using the discriminator. The overarching goal in this step is for the generator to generate a new

image and feed it to the discriminator, which outputs a real or fake class. Below are some sample images generated from the generator network while training at five, 75, 500, and 2,500 epochs (Fig. 8). Images generated at the 2,500-epoch mark were fed into the superimposition component of the project.

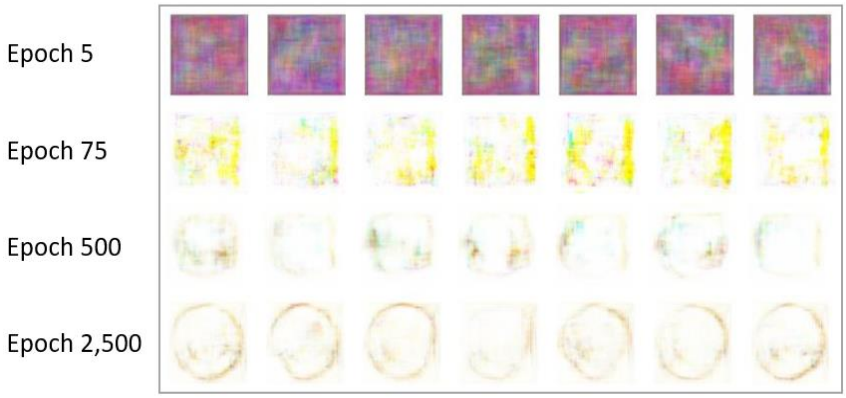

Fig. 8. GAN Generated Coffee Stains at 5, 75, 500 and 2,500 Epochs

4.3 Perturbed Image Superimposition with MNIST Data

Although coffee stains do not present the only problem faced with an OCR task, coffee stains were chosen as a defect test because coffee stains are relatively easy to produce, their shape is relatively uniform, and they do present a real problem for KRP's theoretical OCR process. It was estimated that several thousand coffee stains would need be generated to have enough samples to sufficiently perturb the MNIST data and to produce enough samples in the future to adversarially train KRP's production model. To avoid manually generating several thousand coffee stains, the team decided to use GANs to create artificial coffee stains from real samples.

In order to validate the effect of the perturbations on the baseline CNN model and to use perturbed images for training, MNIST images (Fig. 9) need to be blended with GAN generated stains. This was done by randomly selecting from a pool of artificial stains then randomly selecting a contiguous 28x28 section within the artificial image (Fig. 10). Using partial stains on the images would replicate a real-world problem where a coffee stain might spread across multiple characters on a page and the individual characters would only be affected by a portion of the coffee stains. The artificial stains were then converted to a grayscale format (Fig. 10) and a binary inversion was performed to match the MNIST data format (Fig. 11). Now both the MNIST data and stains were lighter where part of the image

was present and darker where it was not. Since the selection was random, there were many cases were only a small portion of a stain was selected and other cases where the partial stain was positionally irrelevant after superimposition on the MNIST character (Fig. 12). These cases would prove to be inconsequential to the CNN. After iterating through this process several times and looking at the results of the process, the team decided to use a 40% random sampling of the training and test data to blend with the artificial stains. This 40% random selection provided enough perturbed samples to validate the effect of introducing the samples to the baseline CNN model and enough to adversarially train a new CNN model that was otherwise identical to the baseline model.

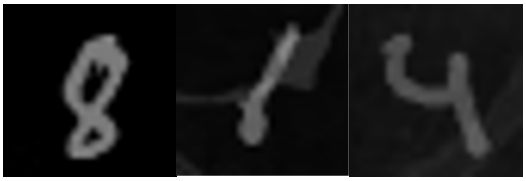

Fig. 9. Sample MNIST Images without Modification

Fig. 10. 28x28 Slices of GAN Generated Coffee Stains in Grayscale

Fig. 11. Binary Inversion of GAN Generated Stains

Fig. 12. MNIST Data with Superimposed GAN Generated Coffee Stains

4.4 Determining the Effect of the Perturbed Images on Baseline CNN Model

Once the images were altered to replicate what the characters would look like if a coffee stain was placed over them, the baseline CNN model was re-tested using the MNIST test data with the 40% altered data replacement. The test validation accuracy dipped drastically from 97.38% to 94.72% (Fig. 13). Obviously, the perturbed images had a drastic effect on the classification accuracy of the CNN model. In a production environment, this misclassification could have a detrimental impact on KRP's business.

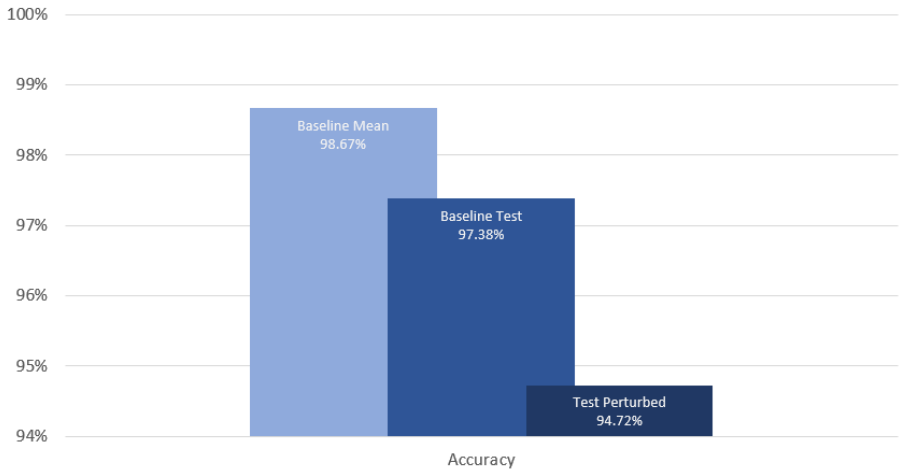

Fig. 13. Accuracy of the Baseline CNN Model: Mean 5-fold, Test Accuracy and Test Accuracy Against Perturbed Images

4.5 CNN Training with Perturbations

A new CNN model with the same structure as the baseline model was trained but this time the adversarial training data was included in the training dataset. During training, the resulting CNN model's 5-fold cross validation accuracy increased for every fold and the mean 5-fold accuracy increased from 98.67% on the baseline model (the model trained and validated with unperturbed images) to 99.35% on the adversarially trained model (Fig. 14). This result was quite surprising to the research team. The expectation was that using the perturbed images to train would slightly decrease the 5-fold accuracy, but the model would be slightly more accuracy against the test dataset.

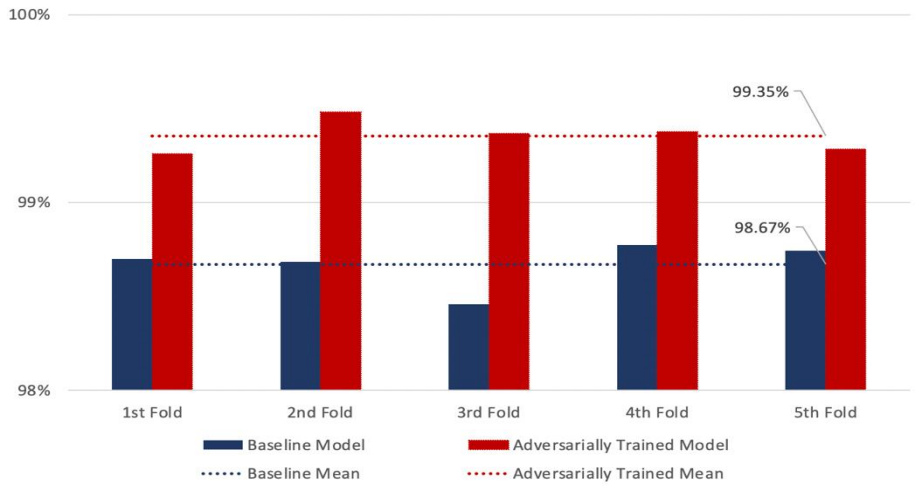

Fig. 14. 5-fold and Mean Accuracy of Baseline (blue) Versus Adversarially Trained (red)

Using the adversarially trained model, the perturbed test images were also validated. When testing the perturbed images in the baseline model, this resulted in a 2.66% dip in accuracy. The adversarially trained model's test accuracy, on the other hand, dipped only slightly in comparison. With the tested accuracy on the unperturbed dataset at 98.09% dropping to 97.32% when validating against the perturbed test set; a 0.77% drop (Fig. 15). Not only does adversarially training the CNN model make the model more resistant to image defects, but the model's accuracy increases overall.

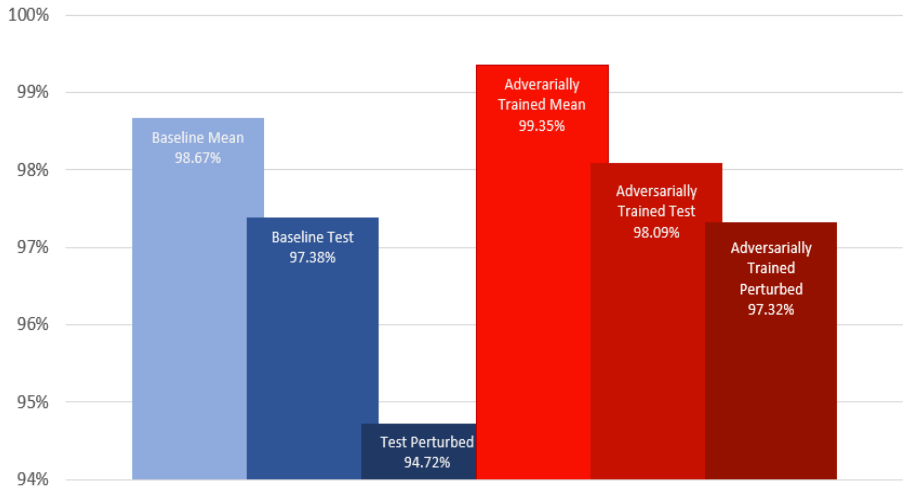

Fig. 15. CNN Model Comparison Baseline (blue) Versus Adversarially Trained (red)

5 Discussion

5.1 Future Work & Application

Although the research team did start developing a custom OCR process to prove up the final piece of the OCR pipeline, the work was not finished at the time this paper was written. It will take many months, even years, to do the development required to build out the entire automation but the work the research team performed will be passed-on to the data science team at KRP where the development will be continued. A couple of steps that were not included in our analysis was the text localization phase where the texts position on the page will be found and text will be differentiated from other data found on the check stubs. Seeing that the checks can be convoluted with symbols, handwritten notes, or other random images, and the likely possibility that the text is skewed or rotated on axis, the text localization process should treat the text being recognized as the only item on the page. Thus, the method that needs to be employed should figuratively wrap the text with bounding boxes and treat everything outside the bounding boxes as inconsequential. A very rudimentary method to extract data from the page would be by segmenting the document into columns and lines and using regex parsing to extract the individual words or phrases. Prior to the deep learning age, text detection and localization algorithms either leveraged Connected Components Analysis or Sliding Window based classification [16]. Connected Components Analysis uses a graph-based approach where the image is

traversed pixel by pixel and connectivity between the pixels is determined according to region membership [7]. Sliding Window based classification takes a similar approach to Connected Components Analysis, however, in the Sliding Window approach, windows of fixed sizes pass over the image and regional membership is determined according to the type of object. Although the Connected Components Analysis approach could be quite effective on most of the data, there are certain instances where overlapping stamps or handwritten text would impede on the process. Therefore, both algorithms would fail to assign neighboring text as distinct entities in the way necessary to compartmentalize the strings as individual events. Additionally, the resulting regions would lack the coordinate metadata necessary to construct the deep learning model required to classify the text as specific types of data: well names, production dates, unit prices for the production volumes, gross production volumes, net production volumes, gross revenues, net revenues, production tax, and revenue deductions. These heuristic methods would also fail to properly localize the text in cases where the images were blurred or stained [16].

A more appropriate method to isolate the text would be to treat the strings or characters as unique entities, abstracting any association from neighboring text, while also gathering coordinate metadata from text bounding boxes. Once the pertinent text had been detected and recognized with the coordinate metadata, those features will be used as inputs for a deep neural network where the goal of the model is to classify the data point as one of ten categories (Fig. 16). Once the data has be recognized and classified the data can easily be pushed to KRP's accounting system or formatted as a flat-file input that can be easily imported using the accounting software.

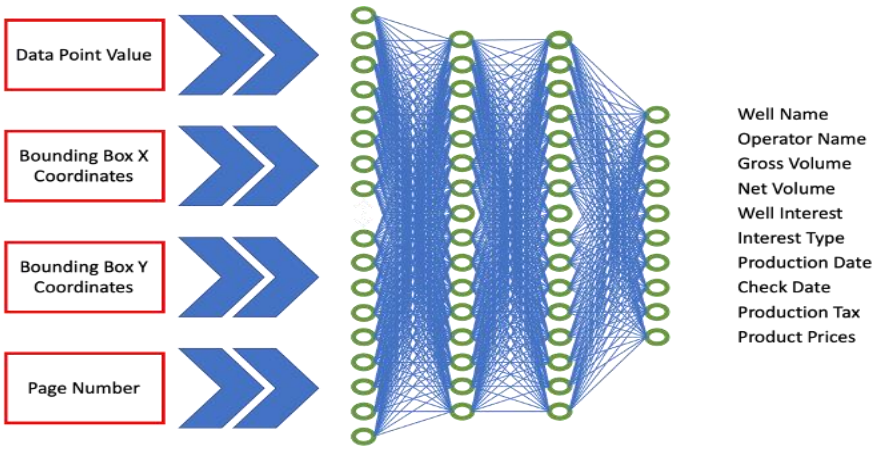

Fig. 16. Deep Learning Data Point Classification Model Structure

5.2 Ethics

The ability of GANs to generate data with relatively little training data, as shown in this research, make them a candidate for misuse. Real data should always be used for training when the data is available. While GANs have the ability to learn the distributions of a training dataset and generate realistic artificial samples, this is no replacement to the variability found in a real dataset. Another concern that could directly affect KRP and its investors was highlighted in the paper *Intriguing Properties of Neural Networks*, where the researchers showed that samples could be specifically engineered to confuse a deep learning model [28]. In KRP's hypothetical pipeline, it is possible that an adversarial attack, using engineered or even natural occurring adversarial samples, could go undetected without security protocols in place. Furthermore, as Szegedy et al (2013) demonstrated, and others did in subsequent research, adversarial attacks could render a deep learning model useless [3][8][11][19][28]. Without mechanisms in place to ensure an operational network, a lame network could have a detrimental impact on KRP's bottom line.

6 Conclusion

The research team has demonstrated increased accuracy in text recognition and resistance to perturbations for KRP's check stub OCR application by using GANs as a vehicle for creating training data for an adversarially trained OCR process. With KRP's ultimate goal in mind, the research has shown that an OCR solution can be deployed to remedy data entry errors and reduce contract labor costs without worry of the effects of document defects on the process. The value of adversarial training a CNN model in an OCR pipeline is evidenced by the increase in 5-fold training accuracy from 98.67% on the baseline CNN model to 99.35% on the adversarially trained model and the 2.6% increase in accuracy on the perturbed validation test set. Not only is the model more resistant to defects, but the model showed higher accuracy with non-perturbed data in both the training validation and testing validation phases. The research has also shown how GANs can be used to create data when data is not accessible or, in this case, where there is too little data to effectively train a machine learning model. With only 173 manually created images, the team was able to generate thousands of realistic looking coffee stains to be used in the adversarial training process. Manually creating those coffee stains would have required hundreds of hours of work. Although the team did not develop the entire OCR solution, the research was able to show KRP the strength of adversarial training in their production environment which, the evidence shows, will prevent the costly errors from being introduced into the process. KRP will continue the development of the custom OCR algorithm which will save the company more than $500,000 annually in contract labor and omit a system that introduces data entry errors.

References

1. Britton, S. (2019, March 19). A Brief History of OCR. Retrieved from https://www.cloud-trade.com/2019/03/19/a-brief-history-of-ocr/.

2. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. Retrieved from https://arxiv.org/abs/1606.03657

3. Chen, Y., Hosseini, H., Kannan,S., Poovendran,R., B.Zhang (2017). Blocking Transferability of Adversarial Examples in Black-Box Learning Systems. Retrieved from https://arxiv.org/pdf/1703.04318.pdf

4. Deng, L. (2012), "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," in IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141-142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.

5. Deshpande, A. (2016). A Beginner's Guide to Understanding Convolutional Neural Networks Part 1. Retrieved from https://www.kdnuggets.com/2016/09/beginners-guide-understanding-convolutional-neural-networks-part-1.html.

6. Dutta, S., Sankaran, N., Sankar, P., & Jawahar, C. (2012). Robust Recognition of Degraded Documents Using Character N-Grams. Proceedings of 10th IAPR International Workshop on Document Analysis Systems (DAS), pages 130-134. Retrieved from http://web2py.iiit.ac.in/research_centres/publications/download/inproceedings.pdf.ac53d37078337bef.4e617665656e32303132526f627573742e706466.pdf

7. Fisher, R., Perkins, S., Walker, A. & Wolfart, E. (2003). Connected Component Labeling. Retrieved from http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm

8. Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and Harnessing Adversarial Examples. Retrieved from https://arxiv.org/pdf/1412.6572.pdf

9. He, X. Zhang, S. Ren & J. Sun. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

10. Hosangadi, R., Adiga, D., & Vyeth, V. (2019). OCR-Friendly Image Synthesis Using Generative Adversarial Networks. In Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition (pp. 226-234). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3373509.3373580

11. Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P. (2017). Adversarial Attacks on Neural Network Policies. Retrieved from https://arxiv.org/pdf/1702.02284.pdf

12. Islam, N., Islam, Z., & Noor, N. (2016). A Survey on Optical Character Recognition System. Journal of Information & Communication Technology-JICT Vol. 10 Issue. 2, December 2016.

13. Karimi, M., Veni, G., & Yu, Y.-Y. (2019). Illegible Text to Readable Text: An Image-to-Image Transformation using Conditional Sliced Wasserstein Adversarial Networks. ArXiv, abs/1910.05425. Retrieved from https://arxiv.org/pdf/1910.05425.pdf

14. Krizhevsky, A., Sutskever, I., & Hinton, G. (2011). ImageNet Classification with Deep Convolutional Neural Networks. Retrieved from https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

15. LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database. Retrieved from http://yann.lecun.com/exdb/mnist/

16. Long, S., He, X., & Yao, C. (2018). Scene Text Detection and Recognition: The Deep Learning Era.

17. Mahpod, S. & Kelle, Y. (2020). Auto-ML Deep Learning for Rashi Scripts OCR. CoRR, abs/1811.01290. Retrieved from https://arxiv.org/abs/1811.01290

18. Palka, J., & Palka, J. (2011). OCR systems based on neural networks. Annals of DAAAM & Proceedings, 555+. Retrieved from https://link-gale-com.proxy.libraries.smu.edu/apps/doc/A349222691/AONE?u=txshracd2548&sid=AONE&xid=6a212c15

19. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, B., & Swami, A. (2015). The Limitations of Deep Learning in Adversarial Settings. Retrieved from https://arxiv.org/abs/1511.07528

20. Patel, Poorvi. (2014). The different image processing techniques to text from natural images Priyanka Muchhadiya. JOURNAL OF OPERATING SYSTEMS DEVELOPMENT & TRENDS. 1. 1-7.

21. Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2019). Do ImageNet Classifiers Generalize to ImageNet? Retrieved from https://arxiv.org/abs/1902.10811

22. Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2018). Do CIFAR-10 Classifiers Generalize to CIFAR-10? Retrieved from https://arxiv.org/abs/1806.00451

23. Riley, Sean. (2017, October25). Generative Adversarial Networks (GANs) - Computerphile [Video file]. Retrieved from https://www.youtube.com/watch?v=Sw9r8CL98N0&feature=youtu.be.

24. Schantz, H. (1982). The history of OCR, Optical character recognition. Recognition Technologies Users Association. ISBN 9780943072012

25. Sharma, M., Verma, A., & Vig, L. (2019). Learning to Clean: A GAN Perspective. CoRR, abs/1901.11382. Retrieved from http://arxiv.org/abs/1901.11382

26. Simonyan, K. & Zisserman A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. https://arxiv.org/abs/1409.1556

27. Sutherland, D. J., Tung, H., Strathmann, H., De, S., Ramdas, A., Smola, A., & Gretton, A. (2016). Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy. Retrieved from https://arxiv.org/abs/1611.04488

28. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks.

29. Versloot, C. (2019). UpSampling2D: How to use upsampling with Keras. Retrieved from https://www.machinecurve.com/index.php/2019/12/11/upsampling2d-how-to-use-upsampling-with-keras/#what-is-upsampling

30. Yadiv, C. & Battou, L. (2019). Cold Case: The lost MNIST Digits. Retrieved from https://arxiv.org/abs/1905.10498

31. Zeiler, M.D. & Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. Retrieved from https://arxiv.org/abs/1311.2901