Signature Verification Using a Convolutional Neural Network

Brittany Cozzens[1], Richard Huang[2], Maxwell Jay[3], Kyle Khembunjong[2], Sahan Paliskara[2], Felix Zhan[2], Mark Zhang[3], Shahab Tayeb[4]
[1]RET, [2]AEOP UNITE, [3]UNLV STEM, [4]UNLV
Las Vegas, Nevada

Abstract— Many studies have been conducted on Handwritten Signature Verification. Researchers have taken many different approaches to accurately identify valid signatures from skilled forgeries, which closely resemble the real signature. The purpose of this paper is to suggest a method for validating written signatures on bank checks. This model uses a convolutional neural network (CNN) to analyze pixels from a signature image to recognize abnormalities. We believe the feature extraction capabilities of a CNN can optimize processing time and feature analysis of signature verification. Unique characteristics from signatures can be accurately and rapidly analyzed with multiple layers of receptive fields and hidden layers. Our method was able to correctly detect the validity of the inputted signature approximately 83 percent of the time. We tested our method using the SIGCOMP 2011 dataset. The main contribution of this method is to detect and decrease fraud committed, especially in the banking industry. Future uses of signature verification could include legal documents and the justice system.

Keywords—convolutional neural network, handwriting, deep learning, signature authentication, signature verification, machine learning, image classifier

I. Introduction

This paper's purpose is to suggest a method of utilizing CNNs to analyze signatures on checks to recognize anomalous differences. The 2017 AFP Payments Fraud and Control Survey [1] results stated that 8 out of 10 business to business transactions are performed through checks. Seventy-four percent of the participant organizations, such as banks, experienced some form of payment fraud [1]. The recurrence of these check frauds has been increasing. By searching for abnormalities through pattern recognition, one can conclude that a signature was not written by the same person, thus indicating possible fraud.

A. Hypothesis

We hypothesize that CNNs used in signature authentication will be able to function with less error than the traditional deep neural networks typically used in such static object recognition [2] [3]. We expect the application of CNNs to validate written signatures will reliably match a signature to the correct signee. These expectations are motivated by the ability of CNNs to apply multiple layers of receptive fields and successive hidden layers to rapidly and accurately compare unique characteristics of pairs of given signatures, and determine if they come from the same signer.

Customarily, pattern recognition, at a simple scale, only requires elementary-level classification tools such as a Support Vector Machine (SVM) or logistic regression. However, as inputs for data in the neural network increase, other methods begin to significantly surpass these previously self-sufficient processes. For more sophisticated patterns with multi-layered neural networks, the number of possible patterns grows exponentially, requiring more nodes in each layer. This can lead smaller neural networks to become unable to sufficiently recognize the pattern, which negatively affects accuracy.

Utilizing a CNN allows for more accurate pattern recognition as it can efficiently identify spatial features to classify images independently. When such networks are applied to offline signature verification, one is able to better detect forgery. In a similar sense, our solution can also apply to the field of forensics in helping officials identify the writer of a document, possibly aiding legal cases. Unfortunately, recent attempts to classify forgeries have been relatively unsuccessful with a significant rate of error, even with larger datasets [4]. Similarly, nuances such as variation in characters and strokes found in a person's handwriting, which are traditionally difficult to detect, have resulted in a history of failure for recognition of handwriting using traditional neural networks [5]. However, the use of CNNs for handwriting analysis has proven more fruitful, with an accuracy rate of about 98 percent.

B. Theories Used in the Research

Images are stored in a computer as a series of pixel data. Image recognition involves the analysis of multiple images compared pixel by pixel to derive conclusions about their similarities. Utilizing CNNs in the recognition of signatures enables the proposed algorithm to quickly and accurately recognize signatures and find corresponding signatories. By using deep learning and image recognition, it is theoretically possible to compare a user signature to previous signatures already stored in the host database. Signatures can be identified using either template-based methods or learning-based systems. Template-based methods compare template images of the signature being searched for with the input document images. Learning-based systems use supervised learning to train signature models.

II. Related Work

Image recognition of words, although largely ignored by the greater part of the deep learning community, has gained some traction in recent years, especially with the publication of studies conducted on signature recognition using deep learning. In 2012, Khalajzadeh et al. [6] published a study using CNN's to analyze a set of Persian signatures, reaching an accuracy rate of around 95 percent. Similarly, in 2015, Miah et al. [7] used an artificial neural network to analyze signatures and monetary values on a check. However, unlike Miah et al., Khalajzadeh et al.'s study did not account for signatures written on physical checks. Our method improves upon Miah et al.'s study and Khalajzadeh et al.'s study by using a CNN to analyze signatures on checks, thus bridging the gap between these two reports.

Although CNNs are widely used, we incorporated methods studied previously that focused on improving the efficiency and accuracy of neural networks. For instance, in 2010, Sulong et al. [8] presented a hybrid strategy for recognition of strings of characters, overcoming the individual weaknesses of explicit segmentation based and implicit segmentation based approaches for analytic cursive word recognition. This is a two-stage, dynamic programming based, lexicon driven approach. In response to Sulong et al.'s call for a feature set in handwriting, Wicht et al. [9] proposed a feature extraction system based on Convolutional Deep Belief Networks, using sets of sliding window features learned from handwriting or word images without human supervision. Additionally, Chherawala et al. [10] proposed a framework for feature set evaluations based on a collaborative setting. They used a weighted vote combination of Recurrent Neural Network classifiers, all trained with a particular feature set. Alternatively, Kumar [11] offered a review of various characteristics of the handwriting recognition.

Multiple types of deep learning networks exist; however, their efficacy has rarely been compared and studied. In 2000, He et al.'s [12] research proving deep neural networks are harder to train than residual learning frameworks. Data collected during their experiment using optimized residual learning frameworks proved that although they are less complex than deep neural networks, they learn and process more with fewer errors. Based on He's research, Donahue et al. [13] evaluated whether or not specialized deep CNN's could be adapted to generalized tasks. Semantics clustering data was gathered while the networks performed various tasks, and the efficacy of these networks operating at different levels while performing the same task was also compared. This information was collected and implemented inside DeCAF, which allows others to replicate their experiments.

In a later paper, Donahue et al. [14] proposed a class of recurrent convolutional architectures, Long-term Recurrent Convolutional Networks (LCRNs), well suited for large-scale visual understanding tasks, and trained these to learn temporal dynamics so as to more effectively analyze variable-sized parameters, such as video. These LRCNs were more effective at analyzing sequential input, such as dynamic temporal visuals, and marked the beginning of deep sequence modeling's centrality to visual representations and systems. Additionally, for constrained nonsmooth invex optimization, Li et al. [15] proposed a single-layer recurrent neural network. Alternatively, Liang and Hu [3] proposed doing object recognition with a recurrent CNN.

Digital verification can occur using a variety of methods, each with their own pros and cons. In 2016, Katiyar et al [5] recognized that signatures are the most common source of digital verification; however, signature forgeries have prompted the need for verification systems. Katiyar et al.'s proposed system would work online or offline based on input, and are normalized and scaled into a standard format using algorithms that extract global features (area, height, width, etc.) to find a match between the stored signature and test signature [16].

Signatures can also be verified using Nonnegative Matrix Factorization (NMF) [17], which is extremely useful for pattern recognition. Given a matrix of nonnegative elements (such as pixels), NMF tries to break it down as a product of two lower rank matrices. From this one can get a representation of the various parts of a pattern. In such a method, one can use the sum of the squared horizontal and vertical distances in order to achieve a higher point of accuracy in NMF. This allows NMF to take the distance between two elements into account which can allow for more accurate recognition. Araki et al. [18] also provided a sampling algorithm for Bayesian variable selection.

III. Methods

In our experiment, we constructed a CNN architecture using the Keras library on Python with a Tensor Flow backend. We utilized an image comparison system, based on an image classification system. Similar to an image classification system, each signature will be associated with an author via a label.

When a signature is entered into the program, it will be compared with the features of other signatures with the same label. Due to the nature by which signatures are written, computer vision systems will typically recognize a signature as a single object. As such, it is possible to compare the features of a signature, such as particular edges and spacing, with other signatures with the same label. Then, our image comparison system will output a probability that an inputted signature is valid. Each check will be adjusted to a predefined image size and flattened into a feature vector that will then be used as the input signature. *Fig. 1* illustrates the model of the neural network developed.

A. Process

In this solution, we train a CNN to classify images of signatures in order to determine whether or not they were forged. To account for the varying backgrounds of the images, a kernel is used to check where differences in pixel intensity are most prevalent. In other words, the filter alters the image

to give the signature a more pronounced outline. This acts to make the background arbitrary and make the CNN applicable to real banks where checks have a background. We tag each signature with a label identifying the signer. Signatures entered into the network are compared with features of other signatures with the same label (*See Table 1*).

Before inputting the image into the network, we first preprocess the image such that it fits into the architecture. Our CNN architecture expects inputs of dimensions 150 x 230, so we reshape each image using the OpenCV library into the appropriate dimensions. We then convert the image to grayscale, and apply a filter on the image that emphasizes differences in pixel intensities, causing the signature to stand out in the image and making its edges clearer. This preprocessing allows our network to more easily identify the signature. Additional preprocessing measures commonly used on images to be fed to a neural network include centering the images and boxing the image, but the dataset from which our signatures come from has already handled this.

Next, 20 percent of the data is partitioned off as the testing dataset described above. The CNN is then trained with a validation rate of 25 percent. We designed the CNN to separate the signatures into datasets. Then, we categorize them into classes determined by their origin and include a separate class for forgeries.

B. *Justification of Methods*

In a CNN for image recognition and specialized fields of recognition, a series of images is inputted, one at a time, into the network. Each of these images will be labelled as a specific class.
This input image is then acted on by successive layers, where each layer's output is the following layer's input, until the final layer which will output a vector of probabilities where the image fits one of the classes. A CNN uses convolutional layers to generate customized filters capable of detecting various features, such as lines and contours. With each successive layer, convolutional filters become more abstract and high-level, eventually resulting in the capability to recognize objects in their entirety. CNNs claim superiority over other neural networks due to a CNNs' minimal requirement for preprocessing. To elaborate, CNNs learn by generating their own filters, and gradually modify those filters to recognize features. This ability allows CNNs to be independent from prior knowledge, and will not require hand-engineered filters, reducing time and effort for constructing the network, compared to other neural networks.

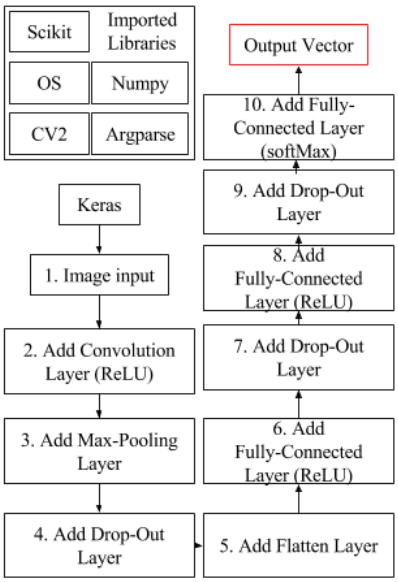

Fig. 1 Model of Neural Network

TABLE I. LAYER SUMMARY

Layer	Size	Parameters
Input	150 x 230	None
Convolutional Layer	128 x 5 x 5	Stride =1, pad = 4
Activation (ReLU)	128 x 5 x 5	None
MaxPool	128 x 5 x 5	Pool = (2, 2) Stride = 2
Dropout	128 x 5 x 5	Drop = 0.50
Flatten	4416000	None
Dense	96	None
Activation (ReLU)	96	None
Dropout	96	Drop = 0.25
Dense	54	None
Activation (ReLU)	54	None
Dropout	54	Drop = 0.25
Dense	M	None
Activation (Softmax)	M	None

C. *Network Architecture*

The architecture of the convolutional neural network used in this experiment uses the elements as follows*:*

C0: Input Layer: An image will be inputted into the CNN as a matrix of pixel data. Each image will be represented with a width and height of a specified number of pixels, and 3 channels, for red, green, and blue, as a 3-dimensional numeric python array. However, this will then be converted to grayscale. The labels will be read as well, and the input will be sent to the next layer, the convolution layer.
C1: Convolutional Layer*:* Convolutional layers are the signature layer of the CNN and perform a convolution on the input image using several filters, getting a number of output images equal to the number of filters. A grayscale image is typically represented as a matrix of integer values representing intensity, across a single black to white channel. The layer involves the usage of filters, or kernels, which are a matrix of values of previously specified dimensions that take each pixel, along with its surrounding region of pixel numbers as a matrix of the same dimensions, performs an element-wise multiplication of the matrices, and then sums

the results. The resulting sum is then outputted into an output image with the same pixel coordinates as the pixel that the kernel originally centered on. Because the kernel observes surrounding pixel values, a side effect of convolution is decreasing border size, which can be prevented by padding the border with numbers that are unlikely to affect results *(See Fig. 2).*

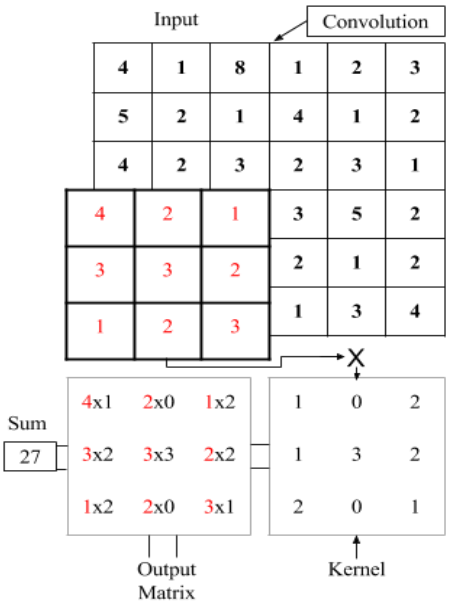

Fig. 2 Model of Convolutional Layer

C2: Rectifier Layer: The Rectifier Linear Unit (ReLU) layer performs an activation function on the outputs of the convolution layer. For each node, the layer inputs each pixel value into function

$$\boldsymbol{f(x) = max(0, x)}.$$

This sets all negative pixel values from the convolution to zero, while leaving each non negative pixel value unchanged (*See Fig. 3).*

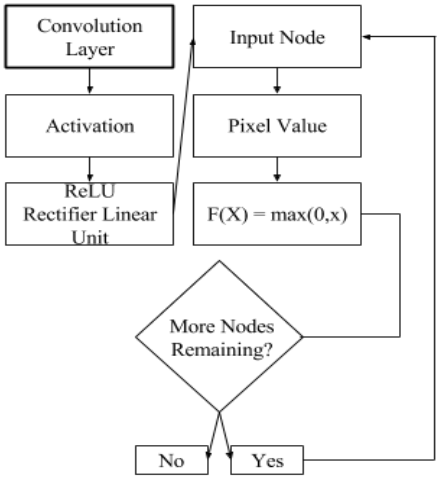

Fig. 3 Model of Rectifier Layer

C3: Max Pooling Layer: A max pooling layer divides the input image matrix into several non-negative regions of pixels which are separated both vertically and horizontally by a stride of N. The layer then takes the maximum value of each of the matrices and outputs them to a smaller output image, which will then be inputted to the next layer. The max pool requires that the dimensions of the input image are divisible by the corresponding dimensions of the size of the regions *(See Fig. 4).*

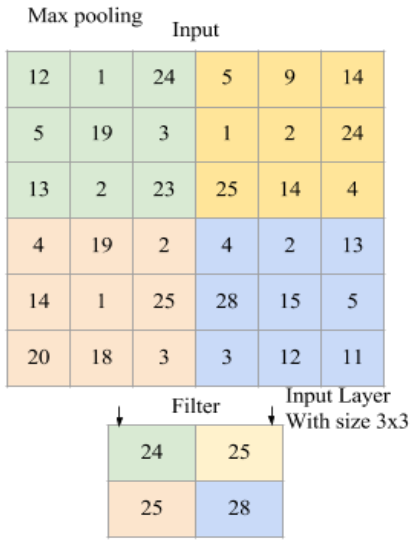

Fig. 4 Model of Max Pooling

C4: Fully Connected Layer: The max pooling layer output image is then connected to a series of fully connected layers, in which the nodes of the fully connected layer, the output nodes, are connected to each node of the previous layer, the input nodes. In this case, the fully-connected layers have a specified number of nodes. Each output node is associated with a matrix of weights. The matrix representing each node will then be multiplied by the matrix of the weights. The result of this will be summed across all the input nodes and placed into the appropriate output nodes (*See Fig. 5).*

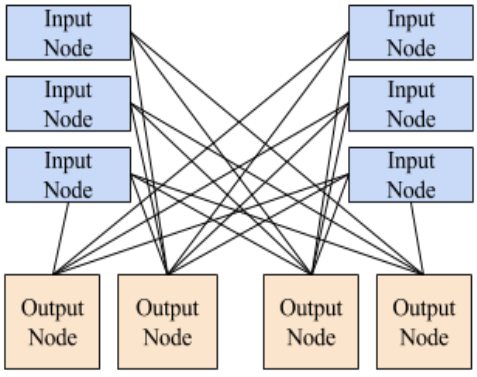

Fig. 5 Model of Fully Connected Layers

C5: Softmax Function: In the final layer, the softmax function is used to extract probabilities for each of the classes. The equation

$$\sigma(\mathbf{z})_j = e^{zj} / (\sum_{k=1}^{k} e^{zk})$$

for all values j = 1, k describes the function. It is meant to turn a k-dimensional vector of z values to another k-dimensional vector of probability values. We used a softmax function because a softmax function can emphasize higher probabilities and de-emphasize lower probabilities *(See Fig. 6)*

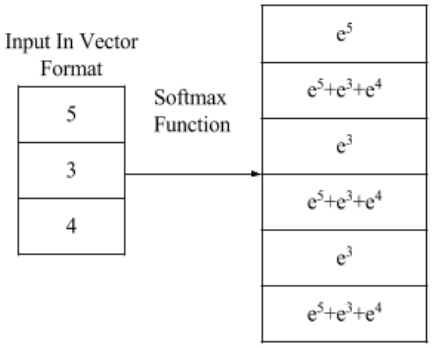

Fig. 6 Model of Softmax Layer

C6: Categorical Cross-entropy: To quantify the quality of the neural network's predictions, the cross-entropy of the probability distribution of the output vector is compared against the entropy. The function

$$H(p) = -\sum_{x} p_x \log(p_x)$$

(where p is the probability for some class x) represents the entropy quantitatively. The cross-entropy is calculated similarly, except the parameter inputted into the logarithm function is the corresponding probability predicted by the neural network. *(See Fig. 7)*

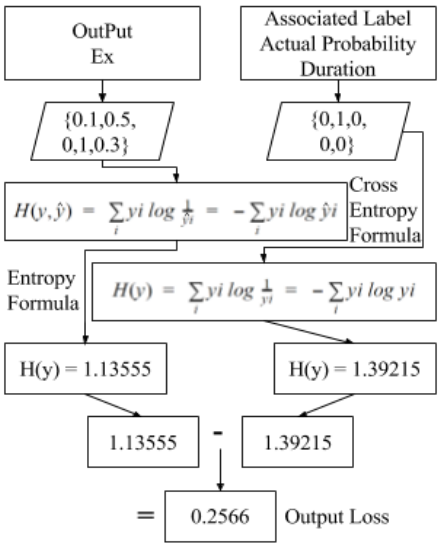

Fig. 7 Model of Categorical Cross-Entropy

C7: Stochastic Gradient Descent: The neural network is optimized between epochs using a stochastic gradient descent. We specify a learning rate, and we optimize the values of weights according to the equation

$$\omega = \omega_i - \mu Q_i(\omega)$$

In this equation, ω is the value of the weight, μ is the learning rate as a decimal, and $Q_i(\omega)$ is an approximation of the gradient. (*See Fig. 8)*

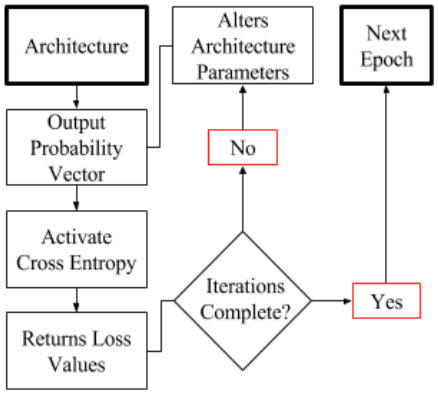

Fig. 8 Model of Stochastic Gradient Descent

D. *Experimental Protocol*

We used Keras, a neural network library, and a Tensorflow backend to analyze signatures. Keras was used instead of other application program interfaces (APIs) because it supports both convolutional and recurrent networks, has a relatively fast runtime, and is more user- and program-friendly than similar APIs. This allows rapid creation and testing of CNNs to be relatively efficient.

Our primary objective is to utilize Keras to scan through a dataset of checks to determine which one was written by a select individual, and to recognize the others as false. The signature dataset we used consists of multiple signatures of the same name, with slight variations among each one; multiple sets of different names are used. Our current experimental stage relies on placing signatures over a placeholder check background in order to train the program. However, we plan to place these signatures over the checks of major banking institutions instead of using a blank background; this allows the program to get used to identifying signatures with those backgrounds, which would most likely be used by large banks.

The initial training set is intended to allow a convolutional neural network to acquire functional, writer-independent weights. In other words, weights for the general classification of signatures. From the training set, we train two sets of weights using our network, one for genuine signatures, and another for forged signatures. After training writer-independent weights, we load the weights of all but the last layer and input the images from the reference set into the network with the weights loaded. We freeze all but the last layer, preventing them from being further trained. This serves a dual purpose to test whether weights from the training set generalize to test set, and prevents overfitting on the new data.

The final layer will then be trained for the reference set, looking for and adapting to writer-specific features in forgeries and genuine signatures of the reference set.

After modifying the weights for the reference set, we put each signature through the network, first using the genuine signature weights, then the forged signature weights. If the maximum probabilities of the output vectors from each output don't indicate the same label, we compare the probability label to the real label, and track the accuracy of each classifier. If the classifiers match, we put the maximum probabilities in a ratio that gives the percentage that a signature is forged. If the match for a forgery in this ratio is greater than forty percent, the signature is classified as forged, favoring classifying signatures as genuine.

IV. Results

The SigComp 2011 dataset was sub-setted into a training set and a test set. The training set was sub-setted into genuine signatures and forged signatures, while the testing set was sub-setted into reference signatures and questioned signatures, with genuine and forged signatures. We saved approximately 20% of the original training sets to be used as validation sets after training. The CNN was then trained on the remaining 235 images with the intention of acquiring writer-independent features that could generalize to another set of signatures. After training, the CNN was, at its best, able to correctly classify 98.8% of the training set, although this dropped to 95% on the validation set. After training the forged weights, the forgery classifier had an accuracy rate of 85%, which also dropped to 82.09% for the validation set. Afterwards, all but the last layer was frozen, and the final layer was trained on the reference set to analyze writer-dependent features. The accuracy of for classification on this dataset was 85.43%. Each reference signature was labelled as genuine, so the reference set for forgeries has not been included in Table 2. Then, the final trained genuine and forged weights were used for the questioned set, and the accuracy of genuine and forgery identification was tracked, resulting in accuracies of 84.74% and 83.12%, respectively. The code for the main classifier is located in reference [24].

The highest accuracy rates for genuine and forged signatures were both present in the initial training set. This is understandable, as the network was initially trained based on those images. The accuracy then declined on the validation and testing sets, indicating minor levels of overfitting on the training data. The datasets used for the forged signatures had a consistently lower accuracy rate than the datasets for genuine signatures, which is attributable to the fact that different people may try to forge the same signature, without consistent writer-specific features for the network to analyze. It is also worth noting that the results of our tests yielded accuracy rates considerably lower than those of other researchers who conducted similar experiments. However, even though this may seem to represent failure, it served to prove that a neural network was able to achieve a measurable amount of success with analyzing signatures, which could be further enhanced with more resources at our disposal.

TABLE II. Results

Data Set From SigComp 2011	Accuracy	Image Count	Time per Epoch (s)	Number of Signatories
Genuine Set				
Training Set	98.8%	235	28	16
Validation Set (Sectioned From Training)	95%	60	6.73	16
Reference Set	85.43%	648	127	54
Questioned Set Correct Genuine Identification	84.74%	648	128	54
Forged Set				
Training Set	85%	92	42.8	10
Validation Set (Sectioned From Training)	82.09%	31	12.9	10
Questioned Set Correct Genuine Identification	83.12%	648	142	54

V. Discussion

As the merits of CNNs would suggest, the network was able to accurately distinguish between different signatures. The ability of CNNs to efficiently create filters to detect an image while avoiding preprocessing allowed for a relatively high degree of accuracy for the case of verification. This indicates that CNNs have the potential to effectively be applied to the verification of signatures on bank checks. Initial testing on a training set of 235 signatures yielded an accuracy rate of 98.8 percent, though subsequent tests suggested that the network was over fitted on the training set, as future results were less accurate.

During a post-experiment analysis of the results, we noticed the disparity in accuracy between the datasets, and upon further investigation discovered a variety of reasons as to why this occurred. A primary reason that because we ran our CNN on a CPU instead of a graphics processing unit (GPU), thus limiting its processing power and forcing it to cut corners. This combined with the limited amount of layers in our CNN resulted in overfitting', which is when the network becomes too accustomed to certain signatures and regards all others as false. This caused a massive number of accurate signatures to become classified as false, thus accounting for the low accuracy results of the last three signature sets.

A. Recommendations for Further Research

The most important direction that should be taken from this research should be the continuation of this research with a much larger dataset with a more efficient method of

processing, particularly using GPUs. Doing so would increase the accuracy due to more signatures being scanned into the program, and with a higher accuracy, it could actually be applied to the real world. Similarly, the implementation of a multitasking convolutional neural network would have a similar effect.

It is also possible to study the further implications of these methods to signature verification in commercial methods by conducting the research on a more realistic dataset. For example, one could also account for the various security features on checks such as the MP icon.

One should also attempt to explore the other implications of signature verification in applications in forensics and medicine where one could create a system for recognizing signatures for those whose signatures change due to the onset of progressive movement disorders such as Parkinson's disease.

B. Implications of the Research on the Field of Study

The solution we achieved shows potential in applications regarding individuals, businesses, and financial institutions that may require fraud detection, especially on checks. Beyond economics and banking, however, it could be used in the legal system by crosschecking words and signatures that may be used as evidence instead of a handwriting expert. The network can also be used this way in nearly all fields involving important documents, as signatures can be checked to prevent any type of criminal activity that might involve faking or altering critical documents, like contracts. Conversation with Previous Research

We support Khalajzadeh et al's [14] and Miah et al.'s [15] studies, as they were a large source of inspiration for our own proposed research. Although we maintain that using a CNN, such as Khalajzadeh's team did is much more efficient due to its minimum preprocessing requirements, we realize that Miah et al's usage of signatures on checks provides the neural network with a more accurate representation of how signatures might appear in a real life scenario. Other research conducted on improving neural networks by changing the layers and using different recognition methods, such as Katiyar et al's, greatly improved our network's accuracy and efficiency by allowing us to test multiple variants of our network until we determined the most efficient version. We improve upon this knowledge by properly the linking the use of CNNs to signature verification in these applications. Instead of verifying signatures in a vacuum, this experiment conveys that it is possible to implement CNNs as a relatively accurate form of signature verification in commercial applications.

C. Limitations

Our experiment faced various limitations such as the use of a CPU instead of a GPU, causing the experiment's system to run computations more slowly. Prior experiments indicate that a larger dataset and better hardware would allow higher accuracies. In particular, the size of each layer in the network architecture along with the number of layers we could use were both limited by the computation power of out experiment's system. Were it not for this limitation, additional convolutional layers and more nodes could have been used, allowing for more abstract representations of objects that would promote more accurate general recognition of signatures.

In addition, our team lacked the experience implement Keras' functional API, which allows multi-task network models with multiple inputs and outputs that would have likely provided better results than the current sequential model.

As stated previously, the drop in accuracy from the third dataset onwards occurred due to a variety of reasons. As we ran our CNN on a CPU instead of a GPU, which is specially designed to do matrix multiplication faster, our network's processing power was limited. This, combined with our time constraints, forced us to use a smaller dataset instead of larger ones. However, this caused the CNN to begin to over fit on the smaller datasets, causing the accuracy issues. A larger dataset would have allowed the neural network to create a more general solution to classify the signatures. Overall, our results indicate that with a larger dataset and better hardware the results would be more accurate, as highlighted by prior experiments.

D. Anomalous Data

During our initial testing of the CNN, as shown previously with our data, we noticed that the accuracy considerably dropped during and after scanning the testing signature dataset. This was due to a relatively shallow network that prevented significantly higher-level analysis. Should a revision to the network be made, and by running it on a GPU to prevent a lack of processing power from hindering it, we would likely be able to achieve higher accuracy results consistent with the initial training sets.

ACKNOWLEDGMENT

This material is based upon work supported in part by the Department of Defense under Army Educational Outreach Program (AEOP) and the National Science Foundation under Grant No. 1710716. The authors thank the UNLV writing center for helping with revising the manuscript.

REFERENCES

[1] 2017 AFP Payments Fraud and Control Survey Report of Survey Results. (2017, March). Retrieved from https://commercial.jpmorganchase.com/jpmpdf/1320732417358.pdf

[2] Araki, T., Ikeda, K., & Akaho, S. (2015). An efficient sampling algorithm with adaptations for Bayesian variable selection. Neural Networks, 61, 22-31.

[3] Chherawala, Y., Roy, P. P., & Cheriet, M. (2016). Feature set evaluation for offline handwriting recognition systems: application to the recurrent neural network model. IEEE transactions on cybernetics, 46(12), 2825-2836.

[4] Contreras, S., & De La Rosa, F. (2016, October). Using Deep Learning for Exploration and Recognition of Objects Based on Images. In Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016 XIII Latin American (pp. 1-6). IEEE.

[5] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2625-2634).

[6] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014, January). Decaf: A deep convolutional activation feature for generic visual recognition. In International conference on machine learning (pp. 647-655).

[7] Hafemann, L. G., Sabourin, R., & Oliveira, L. S. (2017, May 15). Learning features for offline handwritten signature verification using deep convolutional neural networks. Retrieved June 26, 2017, from http://www.sciencedirect.com/science/article/pii/S0031320317302017

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[9] Huang, G., Huang, G. B., Song, S., & You, K. (2015). Trends in extreme learning machines: A review. Neural Networks, 61, 32-48.

[10] Katiyar, S., Agarwal, S., Kaushal, S., & Vats, H. (2016). Signature Recognition and Verification System via Neural Network. Signature, 5(5).

[11] Kumar, V. Online Handwriting Recognition Problem: Issues and Techniques.

[12] Li, G., Yan, Z., & Wang, J. (2014). A one-layer recurrent neural network for constrained nonsmooth invex optimization. Neural Networks, 50, 79-89.

[13] Li, N., & Cao, J. (2015). New synchronization criteria for memristor-based networks: adaptive control and feedback control schemes. Neural Networks, 61, 1-9.

[14] Khalajzadeh, H., Mansouri, M., & Teshnehlab, M. (2012). Persian signature verification using convolutional neural networks. *International Journal of Engineering Research and Technology*, *1*.

[15] Miah, M. B. A., Yousuf, M. A., Mia, M. S., & Miya, M. P. (2015). Handwritten Courtesy Amount and Signature Recognition on Bank Cheque using Neural Network. *International Journal of Computer Applications*, *118*(5).

[16] Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3367-3375).

[17] Patil, P., & Patil, A. (2013, January). Offline Signature Recognition Using Global Features. Retrieved from http://www.ijetae.com/files/Volume3Issue1/IJETAE_0113_65.pdf

[18] Sulong, G., Saba, T., & Rehman, A. (2010, March). Dynamic programming based hybrid strategy for offline cursive script recognition. In Computer Engineering and Applications (ICCEA), 2010 Second International Conference on (Vol. 2, pp. 580-584). IEEE.

[19] Wang, J. J. Y., & Gao, X. (2015). Max–min distance nonnegative matrix factorization. Neural Networks, 61, 75-84.

[20] Wang, Z. R., & Du, J. (2016, October). Writer Code Based Adaptation of Deep Neural Network for Offline Handwritten Chinese Text Recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on* (pp. 548-553). IEEE.

[21] Wicht, B., Fischer, A., & Hennebert, J. (2016, December). Deep learning features for handwritten keyword spotting. In Pattern Recognition (ICPR), 2016 23rd International Conference on (pp. 3434-3439). IEEE.

[22] Zhang, G., Shen, Y., Yin, Q., & Sun, J. (2015). Passivity analysis for memristor-based recurrent neural networks with discrete and distributed delays. Neural Networks, 61, 49-58.

[23] Zhang, S., Xia, Y., & Zheng, W. (2015). A complex-valued neural dynamical optimization approach and its stability analysis. Neural Networks, 61, 59-67.

[24] https://github.com/Rich1720/SigClass_Verification