

Creating a Robust Deep Learning Model to Recognize Handwritten Text

Archishma Goli

Wheeler High School

Advanced Research: Final Research Paper

Dr. Ashley Deason

December 3, 2021

Abstract

Over the past few decades, advances in computing have instilled in society a drive to streamline daily tasks, specifically through artificial intelligence and machine learning (IBM Cloud Education, 2020). One popular field facing an influx of machine learning solutions is handwriting recognition, which is a subset of image recognition. Handwriting recognition is the process of converting text from handwritten documents to digital text (Aneja et al., 2021; Balci et al., 2017). Current models used to recognize handwritten text use a deep learning approach, mainly through Convolutional Neural Network (CNN) architectures (Shivanna et al., 2020). However, despite constant advances in these models, many still encounter challenges in two domains: generalization and robustness. With generalization, handwriting recognition models struggle to classify characters in different handwriting styles (Cheng et al., 2018), and the architecture of these models makes them less robust because they are vulnerable to adversarial attacks (Deoras, 2020; Jain, 2019; Steenbrugge, 2018). Literature has indicated that character segmentation and adversarial training are effective at making deep learning models more generalizable and resistant to attack (Aneja et al., 2021; Balci et al., 2017). However, research combining these respective solutions is relatively limited. This study aimed to help solve this problem and contribute to existing research in handwriting recognition. A deep learning model for handwriting recognition was constructed with a CNN-based architecture and showed a final accuracy of 93.20%. The Character Error Rate (CER) and Word Error Rate (WER) measures supplemented this strong accuracy, with final percentages falling well below the established thresholds of 10% and 40%, respectively. After the model was developed, it was attacked and adversarially trained against the Fast Gradient Sign Method (FGSM). While the model's pre-training accuracy ranged from 14.11% to 62.30%, its post-training accuracy consistently improved by more than 30%. Overall, the study's results support the efficacy of creating a deep learning model, and the model can be further improved upon and defended in order to be applied to real-world applications like reading medical forms or digitizing written notes.

Key Words: Handwriting recognition, deep learning model, Convolutional Neural Network (CNN), Fast Gradient Sign Method (FGSM)

Table of Contents

Creating a Robust Deep Learning Model to Recognize Handwritten Text	4
History of Handwriting Recognition	4
Current Deep Learning Models	5
Rationale	9
Statement of Focus and Sub Problems	11
Literature Review	13
Synthesizing Neural Network Layers	13
Evaluating a Handwriting Recognition Model	14
Increasing Model Robustness	16
Summary and Implications	17
Research Map	18
Scope of Study	19
Delimitation & Limitations	19
Assumptions	19
Definition of Terms	20
Methodology and Design	22
Model Development and Evaluation	22
Adversarial Example Generation	24
Results	26
Developing the Model	26
Adversarial Training	33
Conclusions	38
Implications	39
Limitations	39
Future Work	40
References	41

Creating a Robust Deep Learning Model to Recognize Handwritten Text

As the world becomes increasingly dependent on technology, new systems and applications are being developed to increase convenience and improve quality of life, a trend that is prevalent in machine learning and artificial intelligence. Unprecedented advances in object classification, text-to-speech translation, and voice recognition can be seen in everyday devices like the Google Home assistant and Tesla's self-driving car (Cheng et al., 2018), and the field of handwriting recognition is no different. Recognizing handwritten text is a pioneering area of study, where deep learning models are used to classify postal addresses, validate signatures, and read handwritten forms (Balci et al., 2017). However, these models experience much difficulty in trying to account for variations in handwriting styles and classify perturbed images correctly. This study focuses on the process of creating a robust handwriting recognition model using TensorFlow, which is an open source library in Python that is often used to construct effective deep learning models (Ramasubramanian & Singh, 2019).

History of Handwriting Recognition

The history of handwriting recognition begins with Optical Character Recognition (OCR). One of the earliest physical OCR systems was developed in the late 1940s, and with improvements in technology over time, OCR models are able to recognize both printed and handwritten characters (Memon et al., 2020). Until the 1970s, researchers focused on improving the time efficiency of these physical systems; beginning in the 1980s, however, new software OCR systems were developed and distributed in educational institutions (Memon et al., 2020).

In recent years, handwriting recognition has taken a deep learning approach as researchers work to develop strategies to digitize handwritten text, and improvements in deep

learning models and supporting Graphics Processing Unit (GPU) architectures have aided in the shift to neural networks for recognition problems (LeCun et al., 2015).

Current Deep Learning Models

The most popular learning architectures currently used for image handwriting recognition are Convolutional Neural Networks (CNNs). These networks are created using some combination of convolutional, pooling, and dense neural network layers. The main benefits of CNNs include automatic feature extraction and computational efficiency with complex pooling functions (Dertat, 2017). In addition, convolutional networks have fewer parameters in comparison to traditional neural networks which makes them easier to train and prevents them from overfitting by memorizing the patterns seen in training data (Krizhevsky et al., 2017). Due to these benefits, established convolutional networks like ResNet-18 and ResNet-50 have been proven to achieve high accuracy in image recognition, resulting in their use as transfer learning models to classify a variety of handwritten character datasets (Balci et al., 2017). In addition, recent advances in Recurrent Neural Networks (RNNs), which are mainly used to analyze sequential data, have also allowed for their success in handwriting recognition, especially with datasets including images of handwritten sentences and paragraphs where the surrounding context is important (Pham et al., 2014). This can be attributed to the introduction of Long Short-Term Memory (LSTM) cells, which store node outputs from numerous previous iterations and feed them into each successive time series input.

Most recently, however, deep-belief neural networks (DBNs), which employ fully-connected layers and restricted Boltzmann machines (RBMs), have entered the image recognition scene. They have been proven to be as effective as CNNs and less time-consuming in

classifying handwritten characters accurately, specifically in the MNIST handwritten digit dataset (Cheng et al., 2018).

Areas of Improvement

Despite the advances made in recent years, handwriting recognition models are still faced with numerous challenges such as dealing with various handwriting styles, recognizing punctuation in sentences, or examining writing at various slants and angles (Shivanna et al., 2020). Optical Character Recognition (OCR) technologies, which are the embodiment of current recognition models, usually require handwriting to be spaced evenly and in block capital letters; this poses limitations to the applicability of handwriting recognition in the real world, where people write in numerous block and cursive handwriting styles (ThinkAutomation, 2020).

Although models using ensemble and part-based classification methods were developed to recognize cursive and slanted handwriting, they're not considered reliable due to their low accuracy in comparison to the conventional CNN (Shivanna et al., 2020). As a result, researchers have been developing new models based on convolutional network architectures. For example, the layered training mechanisms of deep-belief networks (DBNs) seem to greatly reduce training difficulty and time taken at roughly the same accuracy (Cheng et al., 2018). These networks were proven effective in recognition of individual handwritten numerical digits, but they haven't been proven effective on image datasets including handwritten words with alphabetical characters.

Vulnerability to Adversarial Attacks

In addition to incomplete recognition capabilities, deep learning algorithms are vulnerable to adversarial attacks, particularly through image manipulation. Different lighting orientations, small image perturbations, and "patches" are used to create adversarial images that

are then fed into the network (Jain, 2019). Although these image changes are not drastic enough to catch the human eye, they can very easily cause deep learning models to incorrectly classify images with a concerning high confidence level (Steenbrugge, 2018).

This is particularly concerning in handwriting recognition models that involve biometric data such as signatures. For example, incorrect classification could lead to signatures being mapped to the wrong customer, resulting in a breach of access to sensitive data. Hafemann, Sabourin, & Oliveira (2019) tested this phenomenon in their study, implementing two different types of image attacks in order to determine how easily signature recognition models would be fooled into misclassifying signatures. In the Type-I attack, noise was added to genuine signatures in order to fool the neural networks to reject them. In the Type-II attack, noise was added to forged signatures in order to pass them as genuine. Type-I attacks could lead to a denial of service in identity verification, and Type-II attacks can result in unauthorized access and identity vulnerabilities. This further exacerbates the need to address the vulnerability of handwriting recognition models to adversarial attacks.

Prevention Methodologies

Fortunately, there are numerous preventive strategies currently in development to ensure greater protection for deep learning models. While some popular approaches include defensive distillation and ensemble learning, a simple but effective methodology is to train deep learning models using pre-constructed adversarial examples (Deoras, 2020). In handwriting recognition, recent work has been conducted on the MNIST handwritten digit dataset to create and use adversarial datasets to train recognition models (Aneja et al., 2021). More specifically, the fast-gradient sign attack and other manipulation techniques like luminance variation were

implemented to create three adversarial image datasets to be fed into the established ResNet recognition models. This strategy is effective because it addresses adversarial attacks during model construction, preventing the problem of misclassification before it occurs.

Rationale

Handwriting recognition models have a variety of real-world applications. While hospitals around the world are implementing recognition software to digitize written health records and combat misinformation by transforming illegible script, students benefit from handwriting recognition technologies when taking notes on digital devices (ITBE, 2021). Despite recent advances, research has shown that this software still faces difficulties with recognition across handwriting styles and cannot effectively classify adversarial images (Deoras, 2020; UKEssays, 2021). This presents the need to create more generalizable and robust models.

Although there are various methodologies to create a handwriting recognition model, image pre-processing, segmentation, and feature extraction should be conducted to ensure baseline performance (Balci et al., 2017; Shivanna et al., 2020). Using a convolutional neural network seems most reliable for image recognition due to high validation accuracy in previous models (Shivanna et al., 2020). Additionally, it is important to implement preventive strategies against adversarial attacks during the model design process. Specifically, using adversarial image datasets to train the recognition model, a process called adversarial training, has proven to be an effective methodology (Aneja et al., 2021).

These considerations were important to the scope of this study which centered on creating a robust handwriting recognition model using the TensorFlow library in Python. The model focused on converting handwritten images to digital text and was trained on manipulated image datasets to increase its reliability against adversarial attacks. The dataset utilized in this study contained images of handwritten names in a variety of writing styles. The research in the

study was conducted while interning under Dr. Selena He, an assistant professor who is part of the Department of Computer Science at Kennesaw State University.

Statement of Focus and Sub Problems

Research has indicated that convolutional neural networks are most effective in handwriting recognition models (Shivanna et al., 2020). In addition, character segmentation has been proven to be more reliable in recognition compared to word classification (Balci et al., 2017). The vulnerability of convolutional neural networks for classification to adversarial attacks has also been widely documented (Deoras, 2020; Jain, 2019; Steenbrugge, 2018). These attacks have been applied to and analyzed mainly on handwritten signatures and not individual characters (Hafemann et al., 2019). Due to this phenomenon as well as the open research problem of effectively classifying handwritten text across writing styles, the following focus statement was generated:

- Develop a robust deep learning model that recognizes handwritten text across a variety of handwriting styles.

In order to better understand how to develop and evaluate this deep learning model, the following sub-problems were addressed:

- Basic Sub-Problem: How can different types of neural network layers be synthesized to create an effective handwritten recognition model?
- Basic Sub-Problem: How can a handwriting recognition model be evaluated?
- Basic Sub-Problem: How can the handwriting recognition model be made more robust during training in order to protect against adversarial attacks?

In addition, the following design sub-problems were created to answer the sub-problems above:

- Design: Develop and train a deep learning model that accurately recognizes characters in images of handwritten names.

Measure of Success: The deep learning model reaches an accuracy level of at least 85% when recognizing handwritten names from the testing dataset.

- Design: Train the deep learning model to prevent adversarial attacks.

Measure of Success: When attacked, the trained deep learning model has a greater accuracy of at least 10% compared to the untrained model. Modifications are made to the code until this is achieved.

Literature Review

This study focuses on developing a deep learning model for handwriting recognition. It contributes to existing literature in the field and attempts to address the problem of classification over a variety of handwriting styles. The recognition model built in the study recognizes the characters in images of handwritten names, and the dataset used contains over 400,000 image examples which are further divided into training, validation, and testing datasets (Saini, 2020). After construction, the recognition model was trained on adversarial image examples to increase its robustness. This literature review provides information on how to best construct and evaluate a handwriting recognition model as well as information on how to generate adversarial image examples for training.

Synthesizing Neural Network Layers

The main goal of a handwriting recognition model is to accurately classify words in input images. Two common approaches to achieving this goal are word-level classification and character-level classification. Word-level classification categorizes words holistically, referencing a given vocabulary list (Gumilang & Purwarianti, 2018). However, this is ineffective because the English vocabulary has tens of thousands of individual words that may not have enough classification examples in the average handwriting dataset (Balci et al., 2017). On the other hand, character-level classification relies on character segmentation methodologies; word images are split into separate character images, which are categorized and then strung together to generate the final prediction (Balci et al., 2017). The method has proven to be more effective on larger datasets (Gumilang & Purwarianti, 2018) which is especially important as the dataset used for training this study's recognition model contains thousands of images.

Both convolutional and deep belief networks have shown high accuracy in character-level image classification (Balci et al., 2017; Cheng et al., 2018). Convolutional neural networks (CNNs) tune learnable parameters such as weights and biases to identify unique features in images. By using a filter, the CNN reduces the size of the image while still capturing its complexities in order to make image classification easier (Saha, 2018). Deep belief networks (DBNs) are generative models that use a stack of Restricted Boltzmann Machine (RBM) layers that can communicate with each other while adjusting the network weights. DBNs were initially created to deal with the challenges of traditional neural networks, such as slow learning and requiring a lot of data (Canuma, 2020). However, one disadvantage of DBNs is that they don't always account for the 2D structure of an image, which may impact its classification capability (Voulodimos et al., 2018). Overall, using a CNN as the basis of this handwriting recognition model is the best way forward because it is equally as effective as a DBN but more well-documented for implementation using the TensorFlow library.

In addition to the traditional convolutional and dense layers of a CNN, the Keras interface in the TensorFlow library includes various other types of layers in its documentation. One notably helpful layer is a dropout layer, which is a method of regularization that prevents neural networks from overfitting the training data (Keras Team, n.d.). Keras also offers the ability to create custom layers for specific model use cases. Using both of these features to add layers in the handwriting recognition model would increase its robustness and accuracy.

Evaluating a Handwriting Recognition Model

After a deep learning model is constructed, there are various metrics used to determine its effectiveness in image classification. In addition to general measures such as training accuracy,

specific evaluation metrics for handwriting recognition include Character Error Rate (CER) and Word Error Rate (WER).

General Measures of Evaluation

Traditional neural networks use a few key evaluation metrics to rate the accuracy and efficacy of all constructed models. Before the data is fed into the model, it is first split into three datasets: training, validation, and testing (Shah, 2017). Each of these datasets then has an accuracy value associated with them. The first of these metrics is training accuracy; this represents the percentage of correct guesses made during the training phase when the model is learning to be more accurate and tuning its parameters. As the model goes through the training dataset recursively, it gradually increases its training accuracy. This measure is a good indicator of the effectiveness of a model's construction; if it is low, then the model isn't capturing all of the complexity of the data and needs to have its parameters or layers reconstructed (Apple Developer, n.d.).

The second measure is validation accuracy; this represents the percentage of correct guesses made by the model after the initial training phase to provide an unbiased estimate of the model fit on non-training data (Brownlee, 2017). Often, the validation dataset is incorporated back into the training dataset to increase the model's robustness. Lastly, one can use the testing accuracy to evaluate deep learning models. Testing accuracy refers to the model accuracy on the testing dataset (Brownlee, 2017). It is a final evaluation of the model fit and is calculated after the validation data is fed into the training dataset (Firmin, 2021).

Specific Measures for Handwriting Recognition

Two specific measures for handwriting recognition include Character Error Rate and Word Error Rate. First, the Character Error Rate (CER) is a popular and important performance metric for handwriting models. It represents the minimum number of insertions, deletions, and substitutions needed to be made in order to match the predicted and actual word outputs. This is expressed as a percentage of the total number of characters being recognized (READ-COOP SCE, 2021). Second, the Word Error Rate (WER) is typically used in both handwriting and speech recognition systems. Like the CER, the WER measures word substitution, deletion, and insertion errors (IGI Global, n.d.). This study's handwriting model uses input consisting of one word per image, so the WER would be analogous to the classification error (Pham et al., 2014). Both the WER and CER should be minimized as much as possible since these values further reflect the model's success.

Increasing Model Robustness

Adversarial examples are specialized training inputs that fool a neural network, resulting in misclassification of the input (TensorFlow, n.d.), but generating and using simple adversarial examples to train a model is an effective method to increase the robustness of recognition models (Deoras, 2020). This process, known as adversarial training, is a data-centric approach to deterring adversarial attacks and has been proven to significantly increase the accuracy of handwriting classification models (Aneja et al., 2021). As a result, common generative techniques can be used to create adversarial examples and improve this study's recognition model.

Generative Techniques

There are various techniques used to generate these adversarial examples, including the Fast Gradient Sign Method (FGSM) and non-local denoising method. The Fast Gradient Sign Method (FGSM) is a popular technique used to create adversarial images. In this method, the gradients of the neural network with respect to each image are taken and manipulated to create new adversarial examples that maximize the loss, or prediction error, of each image. This method assumes that the attacker has complete access to the entire construction of the model, making it a white-box attack (Chakraborty et al., 2018). One can generate FGSM adversarial examples using the GradientTape() function available in the TensorFlow library (TensorFlow, n.d.).

Aneja et al. (2020) describes using the non-local denoising method to generate adversarial images from the MNIST & CIFAR-10 datasets. This method manipulates images based on three different image luminance or color component values, creating three separate adversarial datasets. Feeding both these images and the original unperturbed images into the classification model increased its accuracy up to 9.3% and 13% on the MNIST and CIFAR-10 datasets, respectively. So, this method could be used in addition to FGSM in order to create adversarial images and feed them into the model as training data to further improve its accuracy.

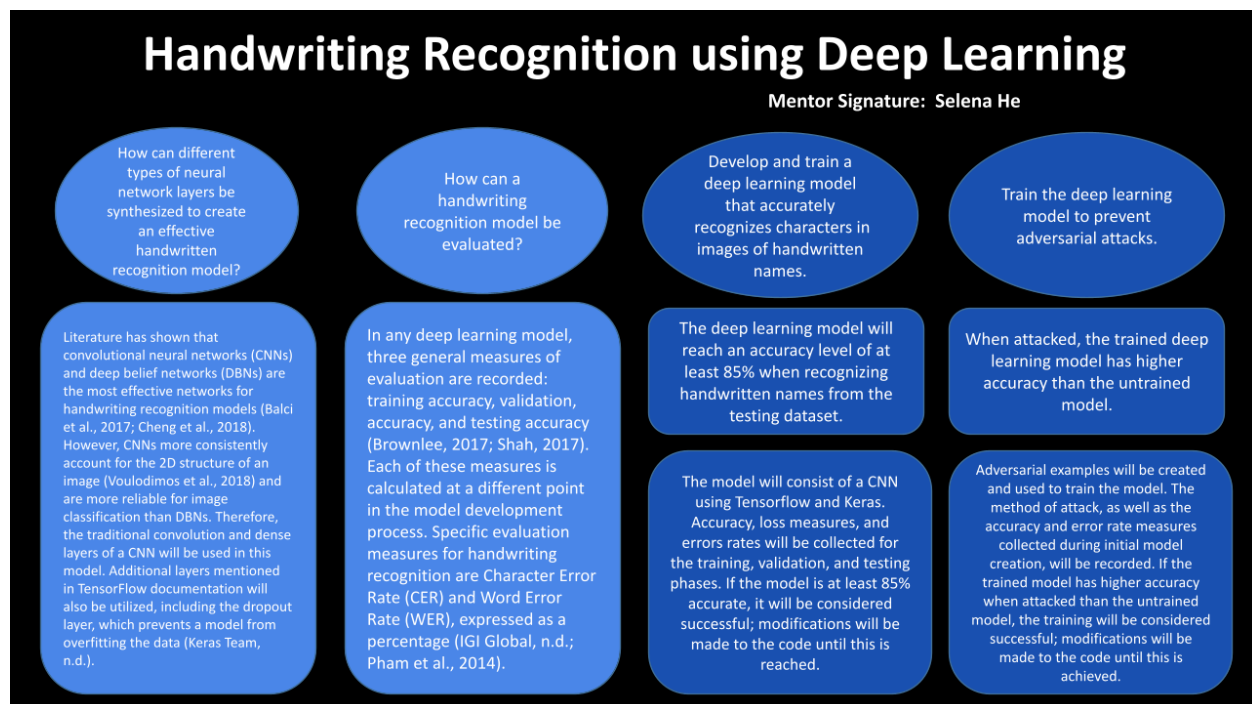
Summary and Implications

Literature shows that using CNNs to construct a deep learning model for handwriting recognition is effective in comparison with other models (Shivanna et al., 2020). Options to use customized layers to further improve the model have also been delineated through the Keras interface documentation. Moreover, there are numerous general and specific measures of evaluation for a handwriting recognition model, including dataset accuracy and CER (Balci et

al., 2017). Lastly, increasing model robustness through adversarial training has been proven effective (Deoras, 2020), with numerous methodologies used to generate adversarial examples (Aneja et al., 2021; TensorFlow, n.d.).

All of these findings were important in the construction of the handwriting recognition model. After using the TensorFlow library to develop an efficient model, the model was trained on original and adversarial examples to improve its robustness. Lastly, evaluation metrics such as testing accuracy and CER were used to determine the success of the model. Ultimately, the goal of the study was to create a model that can recognize characters in a variety of handwriting styles, as this type of model may be useful in real-world applications like reading postal addresses, verifying bank check amounts, or analyzing handwritten forms.

Research Map



Scope of Study

Delimitation & Limitations

- This study focuses on offline handwriting recognition where inputs are static images.
- A successful model is developed and trained to classify handwritten text from the chosen dataset over a span of 9 weeks.
- The deep learning model developed in this study is only trained on images with characters from A-Z, spaces, and dashes.
- This study only defends the model against the Fast Gradient Sign Method (FGSM).
- This study is limited to classifying handwritten text.
- The model's training and evaluation is bounded by the processing capability of the Google Colaboratory Platform.

Assumptions

- The dataset used in this study contains a diverse group of writing styles.
- Adversarial attacks could sufficiently fool the deep learning model.
- The images in the chosen dataset are prone to an adversarial attack specifically from the Fast Gradient Sign Method (FGSM).
- The adversarial attack is carried out in a white-box fashion; the parameters of the model are accessed when generating the adversarial examples.
- TensorFlow is the most efficient deep learning library for the model's implementation and adversarial training.

Definition of Terms

Term	Definition
Offline handwriting recognition	Also known as optical character recognition (OCR); refers to the process of “converting handwritten text into digital form” (Tappert & Cha, 2007)
Deep learning model	A machine learning model that is built using neural networks “with three or more layers.” Attempts to simulate the human brain by “learning from large amounts of data” (IBM Cloud Education, 2020)
Convolutional Neural Network (CNN)	A neural network created using some combination of convolutional, pooling, and dense neural network layers (Dertat, 2017)
Convolutional layer	The building block of a CNN; where the majority of computation occurs within the model. Uses a kernel/filter to move across the fields of an image to detect patterns (IBM Cloud Education, 2020).
Max pooling layer	Used for downsizing the number of weighted parameters the network remembers. Selects pixels with maximum values to send to the output array, thus reducing complexity and limiting overfitting (IBM Cloud Education, 2020).
Training epoch	The number of times a machine learning model has worked through the entire training dataset (Gaillard, 2020).
Character Error Rate (CER)	The “minimum number of insertions, deletions, and substitutions” needed to be made in order to match the predicted and actual word outputs; expressed as a

	percentage (READ-COOP SCE, 2021)
Word Error Rate (WER)	Measures word substitution, deletion, and insertion errors; expressed as a percentage (IGI Global, n.d.)
Adversarial attack	An intentional manipulative action that “aims to undermine machine learning performance, cause model misbehavior, or acquire protected information” (Dickson, 2020)
Adversarial example	An “input to machine learning models” that is used in an adversarial attack (Goodfellow, 2020)
Fast Gradient Sign Method (FGSM)	A method to generate adversarial examples by manipulating the gradients of the neural network with respect to each image (Chakraborty et al., 2018)
Robustness	Refers to adversarial robustness, which measures a network’s resilience against adversarial inputs (Dahal, 2019)

Methodology and Design

The focus of this study is to develop and train a deep learning model that accurately recognizes characters in handwriting images. The dataset that was used was from the Kaggle platform, and it included images of 400,000 handwritten names in a variety of writing styles (Saini, 2020). The images were grouped into training, validation, and testing subsets; each subset was used at various stages in the model development process. The deep learning model was a convolutional neural network (CNN), which is regarded to be one of the most effective network topologies for image classification (Cheng et al., 2018; Shivanna et al., 2020). In addition, the model was constructed using the TensorFlow library in Python.

Model Development and Evaluation

The images in the dataset were first preprocessed and normalized to remove any noise that may preemptively reduce the accuracy of the neural network in classification; this includes cropping and resizing images in each dataset to identical dimensions and using the `cv2.fastNlMeansDenoisingMethod()` function to remove stray marks or dots from each image. Literature has indicated that character segmentation and subsequent classification is effective in classification of both printed and handwritten text because the comprehensive character vocabulary the network needs to work with, which includes A-Z, is much smaller than a comprehensive word vocabulary (Balci et al., 2017; Khare et al., 2019). As a result, a segmentation function was constructed to separate the characters in each handwritten image. Then, the CNN was developed to classify the characters. The images in the training subset were then separated into component characters and fed into the network so the model could be trained.

After the training phase is complete, the validation data was utilized to test the accuracy of the model. If the model achieved at least 85%, which is an optimal accuracy rate for gradient descent-based deep neural networks like the CNN (Wilson et al., 2019), then it would be considered reliable enough to proceed to the testing phase. The construction of the model was modified until this accuracy was reached. This was done by adding or removing convolutional and dense layers in the model as well as by tuning hyperparameters like the kernel size, the batch size, and the activation function used (Amidi & Amidi, 2018). The final accuracy of the deep learning model was assessed by classifying the characters in the images from the testing dataset, and it was considered successful when the accuracy of this recognition reached at least 85%.

Character and Word Error Rate Analysis

In addition to the overall accuracy of the neural network, the Character Error Rate (CER) and Word Error Rate (WER) was also collected during the validation and testing phases to evaluate how well the model recognized names in the images of the dataset. For this, a function was developed to record both of these values and determine the final word prediction for each handwritten image as the data is fed into the model for assessment. Although traditional recognition models are considered satisfactory when they present a CER of 5% or less while recognizing printed text (Alvermann, 2019), the dataset used in this study was more complex, involving much more varied combinations of letters in the form of handwritten names. Others agree that models recognizing handwritten text containing similar information are satisfactory while exhibiting a CER of over 20% (Tomoiağa, 2019; Leung, 2021), so based on this information, a reasonable upper limit for the CER based on classifications from the model in this study was established at 10%. On the other hand, the absolute value of the WER is 3 to 4 times

higher than the CER because the rate operates at the word level (Alvermann, 2019; Leung, 2021). Specifically, if a single character is classified incorrectly in the final prediction, the entire word is marked wrong, thus leading to a more broad measure of accuracy (Leung, 2021). Thus, an upper limit of 40% was established for the WER. Both the CER and WER were used in addition to the overall accuracy in order to improve the model.

Validity and Reliability

This proposed model development process was valid because it was approved by the mentor and was based on conclusions from credible literature on handwriting recognition (Balci et al., 2017; Cheng et al., 2018; Khare et al., 2019; Wilson et al., 2019; Shivanna et al., 2020). It was also valid because the model was constructed based on architectures of well-known and thoroughly tested convolutional networks for image classification, including ResNet50 and VGG16 (Karim, 2019). Lastly, reliability was ensured because the model was consistently improved based on results from the validation dataset until its accuracy reached at least 85%. The CER and WER were used as supplementary measures of evaluation for the model.

Adversarial Example Generation

Despite monumental advances in the accuracy and computational power of convolutional neural networks, literature has shown that these deep learning models are still very vulnerable to adversarial attacks (Gomes, 2018). One remedy to this problem is adversarial training (Jain, 2019) which is the focus of the second design problem in the study; specifically, the goal was to train the deep learning model for handwriting recognition using adversarial examples. The well-documented Fast Gradient Sign Method (FGSM) was implemented using TensorFlow to generate these adversarial examples for training and testing (TensorFlow, n.d.).

First, the original images in the testing dataset underwent a segmentation process, and then the FGSM was used to introduce noise in the segmented character images to generate adversarial examples. An initial accuracy test was performed using these examples to determine how well the untrained model can classify the noise-filled characters; this accuracy percentage was considered the base value for the model to improve upon. A portion of the images in the training dataset then underwent the segmentation and noise generation process and were fed into the convolutional neural network for training; additionally, adjustments were made to the code during this process so the model is thoroughly trained. The training process was considered a success when the model resulted in an accuracy improvement of at least 10% when classifying adversarial examples after adversarial training compared to the initial accuracy test (Ishibashi, 2017; Ren, 2020). The validity of this training method could be justified with the thorough documentation of the adversarial attack method in the TensorFlow library. The method was also reliable because it utilizes the same dataset from the Kaggle platform and records a baseline accuracy value for comparison.

Results

The purpose of this section is to discuss the process for developing the handwriting model and analyze the impact of adversarial training to increase the model's robustness. Literature indicates that current deep learning models struggle with generalization across writing styles and resistance against adversarial attacks (Balci et al., 2017; Deoras, 2020; Jain, 2019; Steenbrugge, 2018). So, the model developed in this study addresses these problems by implementing character segmentation to classify handwritten text, a method which has been previously proven effective (Balci et al., 2017). Adversarial training (Aneja et al., 2021) has also been implemented on the model to increase its robustness against adversarial attacks, specifically against the Fast Gradient Sign Method (FGSM). The dataset used in this study is a Kaggle dataset with images of handwritten names (Saini, 2020).

Developing the Model

To answer the first sub-problem of how different types of neural network layers can be synthesized in order to create an effective handwriting recognition model, an iterative model development and documentation process was implemented. Literature shows that CNNs work most effectively in image classification scenarios (Cheng et al., 2018; Shivanna et al., 2020), so the model was based on a CNN architecture. Literature also indicates that the Character Error Rate (CER) and Word Error Rate (WER) measures should be calculated in addition to the accuracy of the deep learning model (IGI Global, n.d.; READ-COOP SCE, 2021), so these were used as additional measures of success for the study.

Preprocessing and Segmentation

Before the model could be developed, the images in each dataset were first preprocessed and segmented. Due to fluctuations in processing limitations on Google Colaboratory (Google Colaboratory, n.d.) and after discussion with the mentor, it was decided that the number of images used to develop and train the model in each dataset would be reduced. A random subset of images from each dataset was generated to create smaller training, testing, and validation subsets (Table 1).

Table 1

Subset Distributions for Datasets

Dataset Name	Total Number of Images	Number of Images in Subset
Training	330,961	25,000
Validation	41,292	7,500
Testing	41,370	20,000

After the training subset was generated, a list of intended steps was created to prepare the component images for model classification:

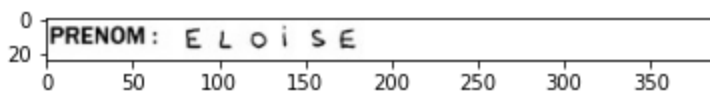
1. Load an image from the dataset.
2. Remove extraneous noise from the image.
3. Segment the image into component character images.
4. Save each component image, along with its character identity, to a separate dataset.

Each intended action was then thoroughly researched to find the code implementation in Python, and the OpenCV and Pandas Python libraries were utilized for loading images from and saving images to a dataframe for future reference (OpenCV, n.d.; Pandas Development Team, n.d.).

Then, to remove noise from each image in the dataset, both generalized and customized methods were implemented. First, the *fastnLMeansDenoising()* method was initially used to smooth out each image; for each pixel in the image, surrounding windows of pixels were searched and averaged to replace the original pixel. This generalized method effectively removed stray dots or marks found in each of the images by blurring them; however, noise specific images in this dataset included extraneous text in printed form (Figure 1).

Figure 1

Sample Image in Training Dataset with Extraneous Noise



Specifically, it was observed that the noise was clustered near the edges of the images, like the extraneous text “PRENOM” seen in Figure 1; this meant that using cropping techniques would sufficiently remove this noise from the image and isolate the text in interest (Khandelwal, 2020). So, the key was to make a cropping algorithm generalizable enough so a representative character dataset could be generated for training, validation, and testing after segmentation. While this cropping algorithm was created, the segmentation function was simultaneously developed. Contour detection properties were utilized from the OpenCV Python library to create the function; since the handwritten text in each image was much darker than the overall background, using contours to separate the boundaries between each letter was effective in generating the segmented character images. The function used binary and adaptive Otsu thresholding to bolster contour detection probabilities in the image.

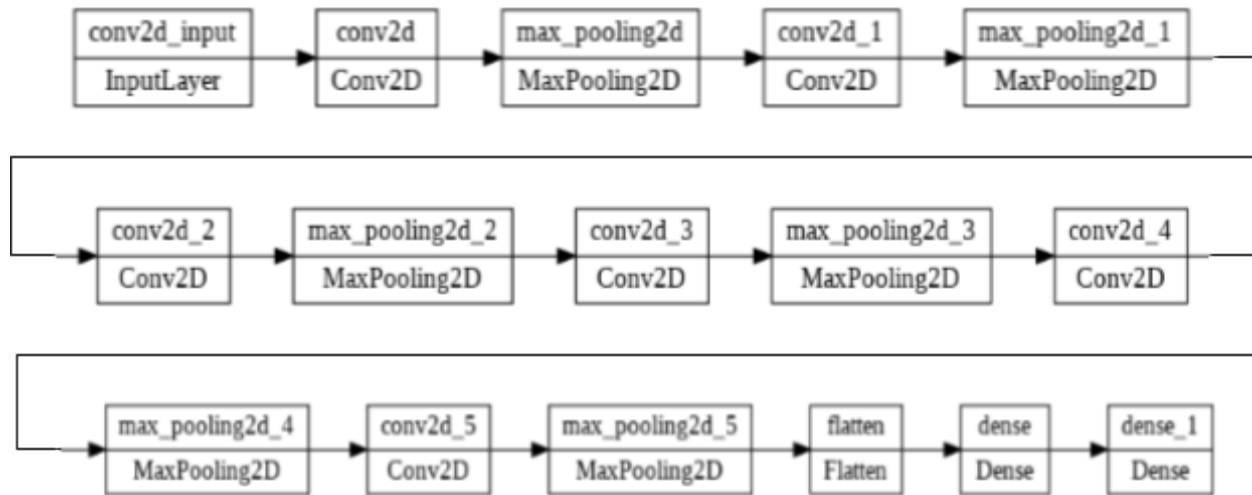
Next, to assess the effectiveness of the cropping algorithm in noise removal, the images in the dataset were passed to the segmentation function after being cropped. The image files that weren't segmented properly after each iteration of the cropping algorithm were analyzed to determine the pixel cutoffs for the following iteration (Table 2). After the 7th iteration, 125,985 characters were successfully segmented, which constituted approximately 80% of the characters in the training subset; from this and discussion with the mentor, it was determined that this generated a sufficiently representative character sample for training. Finally, following the cropping and segmentation processes, the preprocessing phase was concluded by separating the character images into batches for training the deep learning model and reshaping each character into a 256-by-256 pixel image.

Table 2*Iterations and Segmentation Results for Cropping Algorithm*

Iteration #	Description of Iteration	Number of Successfully Segmented Characters
1	No cropping conditionals implemented.	66,057
2	If image_height > 50 pixels: crop top 25 pixels.	87,874
3	If image_height > 45 pixels: crop top 25 pixels. If image_width > 250 pixels: crop left 75 pixels.	106,391
4	If image_height > 25 pixels and < 38 pixels <i>after</i> modification in iteration #3: crop bottom 5 pixels.	110,231
5	If extraneous text is present after iteration #4: crop top 5 pixels.	113,901
6	If image_width > 250 pixels: crop left 78 pixels.	122,209
7	If image_width > 300 pixels: crop left 78 pixels and top 5 pixels.	125,982

Creating Model Architecture

Before the deep learning model was created, three previous models were examined. The first two models were ResNet50 and VGG16, which are deep CNN models presenting a test accuracy rate of over 90% when considering the top 5 predictions for each image (Hassan, 2021; MathWorks, n.d.); from this, it was observed that an alternating combination of convolutional and pooling network layers, along with numerous dense layers near the output of the model, was the most effective model construction for classification. This observation was further supported in the third model, which was a previously developed handwriting recognition model for a separate character dataset (Dave, 2009). After examining these models, the first prototype was developed with a total of sixteen layers; the construction most closely resembled the third model but still included characteristics of the first two networks. Iterations were made to this initial prototype to customize the number of training epochs as well as the batch, kernel, and input sizes to the dataset used in the study (Amidi & Amidi, 2018). The final model prototype also included sixteen layers with the first twelve layers after the input layer consisting of alternating convolution and pooling layers and the last three layers consisting of flattened and dense layers. Figure 2 shows a final model construction in graphical form, visualized using the `plot_model` utility from the Keras interface.

Figure 2*Construction of the Handwriting Recognition Model*

Following each model iteration, the training accuracy was collected. The model was trained over one epoch, which was determined to be the optimal number for the model to prevent overfitting due to the large size of the dataset (Komatsuzaki, 2019). The images were split into a total of 3,938 batches over this epoch and fed into the model, and after training, the final model yielded a maximum training accuracy of 91.60%. This was a good sign that the model was sufficiently trained but didn't overfit and memorize the patterns seen in the training data (Komatsuzaki, 2019; Shah, 2020; Wilson et al., 2019).

Validation and Testing

After the final training accuracy was calculated, the validation subset was generated and preprocessed using the previously developed cropping and segmentation algorithms. These algorithms again proved to be generalizable in this phase, as they successfully cropped and segmented 80% of the characters in the subset. When the successfully segmented characters in

the validation subset were evaluated by the final prototype, the resulting accuracy and loss were 92.92% and 0.3143 respectively (Table 3). This accuracy was higher than the 85% threshold established as the measure of success for the model.

To further establish the credibility of this measure, the Character Error Rate (CER) and Word Error Rate (WER) were also collected. Each of these values was calculated once by developing a Python algorithm and again using Google Sheets formulas to ensure reliability. In the end, both of these methods generated identical error rates for each measure. Out of 38,182 successfully segmented characters in the validation subset, 2,703 were incorrectly predicted, resulting in a CER of approximately 7.08%. This amounted to a total of 6,015 names, of which 1,660 were incorrectly predicted, resulting in a WER of approximately 27.86%. Both of these percentages fell under the 10% CER and 40% WER thresholds respectively, which solidified the accuracy of the model (Table 3).

After the model was proven effective on the validation subset, the final assessment using the test data was conducted. The cropping and segmentation algorithms effectively segmented about 80% of the characters in the dataset, and the final test accuracy of the characters was 93.20%, which is also above the 85% success threshold established. Moreover, out of 99,870 characters, the model incorrectly classified 6,796 characters, resulting in a CER of approximately 6.80%; further, the total number of names made up by the component characters was 15,867, and out of this total, 4,164 names were incorrectly predicted, leading to a WER of approximately 26.24%. Overall, the CER and WER percentages also fell under the maximum thresholds established previously and proved the reliability of the model's construction (Table 3).

Table 3*Error and Accuracy during Validation and Testing*

Dataset	Accuracy	Character Error Rate	Total Number of Names (Words)	Word Error Rate
Validation	92.92%	7.08%	6,015	27.86%
Testing	93.20%	6.80%	15,867	26.24%

Adversarial Training

Literature has indicated that deep learning models, especially those with CNN-based architectures, are vulnerable to adversarial attacks (Gomes, 2018; Steenbrugge, 2018), where the attacker attempts to manipulate the input images in order to generate a misclassified prediction for the image. However, literature has also indicated that adversarial training has been proven effective in countering these adversarial attacks (Aneja et al., 2021; Jain, 2019). So, in order to defend the developed deep learning model against such adversarial attacks, the Fast Gradient Sign Method (FGSM) was used to attack the model; in addition, adversarial training was then implemented to defend the model against the attack.

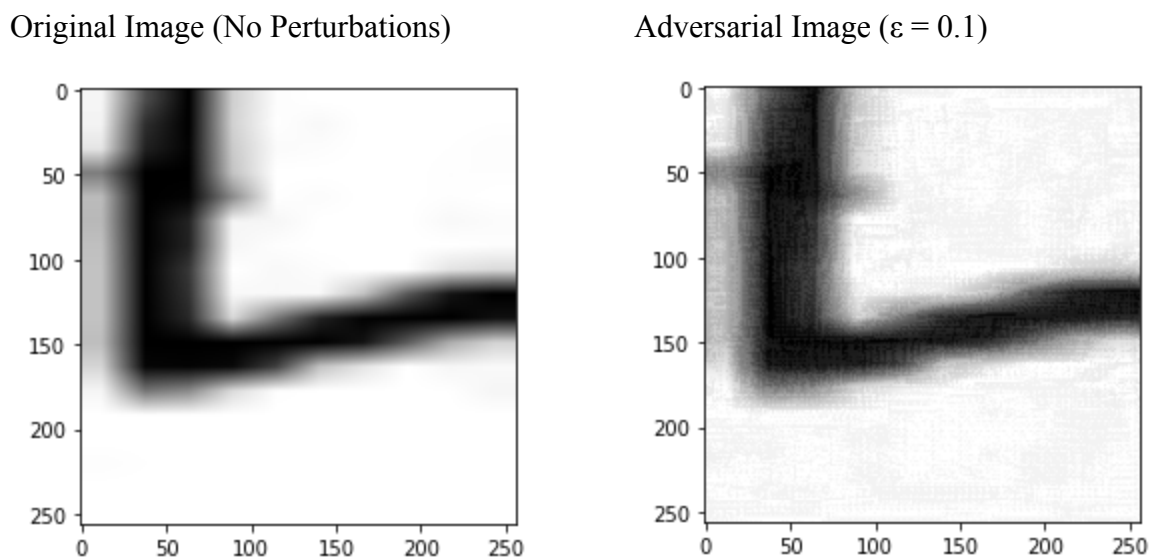
Generating the Adversarial Examples

Before generating the adversarial examples, a unique subset of images was generated from the data in the initial Kaggle dataset. This subset included 1,000 images and was segmented into approximately 5,000 characters. To create the adversarial examples from these characters using FGSM, the official TensorFlow documentation was referenced (TensorFlow, n.d.). However, because the official documentation was specific to the MobileNetV2 deep learning model, the methods had to be customized to the dataset used in this study. In the end, three

functions were created: one to convert each image into a Tensor, one to generate the perturbations of each image using the Tensor image data, and one to add these perturbations to the original image to create the final adversarial image. After a few test cases, an epsilon (ϵ) value of $\epsilon = 0.1$ was decided for the adversarial dataset, as this value ensured the noise generated wasn't obvious but still caused more misclassifications in the data. Figure 3 shows a sample image and its adversarial counterpart generated using FGSM.

Figure 3

Original and Adversarial Examples from the Adversarial Subset



After generation, the images were saved to a new dataset for future reference, and the developed handwriting model was called to predict the adversarial characters. Literature has shown that adversarial attacks usually generate an accuracy of around 60-70% (Jain, 2019, Aneja et al., 2021), so this was the expected threshold for the attack. When the handwriting model in this study was attacked, it presented a pre-training accuracy of 62.30% on the adversarial subset. This was a significant drop compared to the clean images in validation and testing, and the

model was sufficiently fooled by the attack. Because of this, the adversarial subset generated with the original epsilon value of $\epsilon = 0.1$ was utilized and defended against in the adversarial training process.

Training and Testing the Model

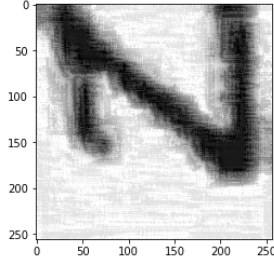
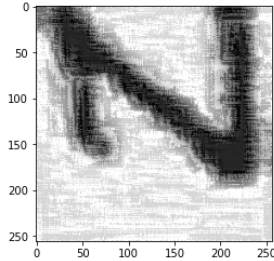
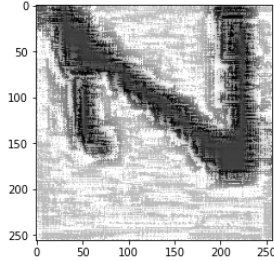
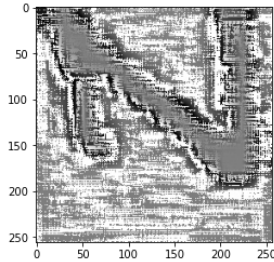
To train the model against the Fast Gradient Sign Method, a separate adversarial training dataset was created with 5,000 training images and a resulting total of 24,495 characters; this dataset was also generated with an epsilon value of $\epsilon = 0.1$. After generation, the model was trained on the adversarial images with the same training process utilized on the initial unperturbed training dataset during the model development phase. The final training accuracy achieved during this stage was 94.81%. The model was then re-evaluated on the adversarial testing examples generated previously; the post-training assessment yielded an accuracy of 95.72%. The resulting increase in accuracy of the model was 33.42%, which is above the 10% increase threshold established. Because adversarial training was successful, the model is now more robust and can defend itself against the FGSM attack at $\epsilon = 0.1$.

Further Evaluation of Robustness

During adversarial training, the model was defended against the FGSM attack using perturbed images generated at $\epsilon = 0.1$; the high post-training assessment accuracy showed that the model was sufficiently robust at this value. To determine the extent of the model's robustness against additional examples, further pre-training and post-training assessments were conducted on the model using noise-generated images with higher epsilon values (Table 4).

Table 4

Model Accuracy Against FGSM Attack

Epsilon Value	Image Example	Accuracy Before Adversarial Training	Accuracy After Adversarial Training
$\varepsilon = 0.1$		62.30%	95.72%
$\varepsilon = 0.15$		57.48%	95.80%
$\varepsilon = 0.25$		41.50%	94.90%
$\varepsilon = 0.5$		14.11%	72.39%

Note: Because the generated images at $\varepsilon = 0.5$ were so perturbed, they wouldn't be part of a standard adversarial attack. However, the adversarially trained model was still robust against images in this more severe dataset, as indicated by the high increase in post-training accuracy.

Based on the accuracy percentages collected before and after adversarial training, the constructed model could defend itself against the FGSM attack across a broader range of perturbations than expected, even though it was only trained on noise-generated images at $\epsilon = 0.1$. This further supported the strong efficacy of the adversarial training process previously documented in literature (Aneja et al., 2021; Chakraborty et al., 2018; Ishibashi, 2017; Ren, 2020).

Conclusions

This study aimed to address basic sub problems regarding how an effective and robust deep neural network for handwriting recognition can be built, evaluated, and defended against adversarial attacks. As a result, two design sub problems were established. First, a deep learning model would be developed to classify handwritten text and generate word predictions. Literature indicated that character segmentation was an efficient method of classification (Balci et al., 2017), so an initial segmentation function was also developed to split images into component characters. Second, the deep learning model would be trained to defend against the Fast Gradient Sign Method (FGSM) adversarial attack. Current literature has shown that adversarial training is an effective way to defend against adversarial attacks (Aneja et al., 2021), so this method was employed to defend the deep learning model.

The validation and testing data assessments conducted indicate that the model was successfully developed, as the model's classification accuracy on both the validation and testing character datasets was nearly 10% higher than the threshold measure of success. In addition, the final Character Error Rate (CER) and Word Error Rate (WER), which were supplementary measures to the accuracy percentage, were under their established limits for both datasets and further supported the success of the model. The adversarial training conducted as part of the second design sub problem was also a success; while the pre-training accuracy of the model on the adversarial dataset was 62.30%, the post-training accuracy on the same dataset was 95.72%. In addition, the model displayed similar increases in accuracy for higher epsilon values and more severe perturbations. This indicated that the model was better trained to defend itself against the FGSM across numerous variations of the attack.

Implications

The goal of this study was to contribute to existing literature on handwriting recognition and create a model that was more generalizable when classifying characters. However, one unexpected characteristic of the images in the dataset used in this study was the presence of form-field names, such as printed characters. While a cropping function was used to remove these form fields from each component image, utilizing the function, along with the segmentation method, to iterate through thousands of images was time-consuming. Perhaps a new algorithm could be developed to automatically detect and ignore common form field names, thus reducing the effort required for pre-processing each image. In addition, the process of segmentation and subsequent character identification explored in this study could be implemented in other convolutional neural networks as a potential method to improve recognition accuracy. Recent research also shows the expansion of handwriting recognition using deep learning to other languages like Arabic (Ahmed et al., 2020). Along the same vein, the model can be trained on more styles and languages of handwriting to improve generalizability.

Limitations

There are numerous limitations to the model developed and trained in this study. First, the dataset used only included black and white images of handwritten names. Therefore, the model can only be generalized to datasets with matching characteristics. In addition, the model was adversarially trained on only one method of attack, the Fast Gradient Sign Method (FGSM). The successful defense of the model can thus only be generalized to other convolutional neural networks and similar white-box methods of attack.

Future Work

Although the model was generally successful in predicting the handwritten names in each sample image, there is still room for growth. For example, the segmentation function's generalizability could be improved through further iteration by examining more test cases and modifying the algorithm's parameters. This would allow for the function to recognize and the model to classify more characters in each of the datasets. Also, the adversarial training process only defended the model against the FGSM, the attack used in this particular study. In reality, attackers could use several other strategies to exploit other vulnerabilities in the deep learning model constructed. For future iterations, the model can be trained against other common white-box and black-box attacks against CNNs, including the Jacobian Saliency Map Attack and the One-Pixel Adversarial Attack (Jain, 2019; Deoras, 2020). Other methods like defensive distillation, using autoencoders, or creating a custom defense algorithm can also be explored (Dahal, 2019). These strategies could give way to more generalizable defenses to better protect the developed model so the chances of misclassification are reduced while keeping the original weights of the model unchanged (Jain, 2019).

References

- Ahmed, R., Dashtipour, K., Gogate, M., Raza, A., Zhang, R., Huang, K., Hawalah, A., Adeel, A., & Hussain, A. (2020). Offline Arabic handwriting recognition using deep machine learning: A review of recent advances. *Advances in Brain Inspired Cognitive Systems*, 457–468. https://doi.org/10.1007/978-3-030-39431-8_44
- Alvermann, D. (2019, October 28). *Word Error Rate and Character Error Rate: How to evaluate a model*. Jurisprudence in the Baltic Sea Region. Retrieved October 11, 2021, from <https://rechtsprechung-im-ostseeraum.archiv.uni-greifswald.de/word-error-rate-character-error-rate-how-to-evaluate-a-model/>
- Amidi, A., & Amidi, S. (2018, November 25). *Convolutional Neural Networks cheatsheet*. CS 230 - Convolutional Neural Networks Cheatsheet. Retrieved October 22, 2021, from <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>
- Aneja, S., Aneja, N., Abas, P. E., Naim, A. (2021). *Defense against adversarial attacks on deep convolutional networks through local de-noising: A data-centric approach* [Manuscript submitted for publication]. Faculty of Science, Universiti Brunei Darussalam. <https://bit.ly/2XdDxXI>
- Apple Developer (n.d.). *Improving your model's accuracy*. https://developer.apple.com/documentation/createml/improving_your_model_s_accuracy
- Balci, B., Saadati, D., & Shiferaw, D. (2017). *Handwritten text recognition using deep learning* [Unpublished manuscript]. Department of Computer Science, Stanford University. <http://cs231n.stanford.edu/reports/2017/pdfs/810.pdf>

Brownlee, J. (2017, July 14). *What is the difference between test and validation datasets?*

Machine Learning Mastery. Retrieved September 21, 2021 from

<https://machinelearningmastery.com/difference-test-validation-datasets/>

Canuma, P. (2020, December 23). *What are RBMs, deep belief networks and why are they important to deep learning?* Medium. Retrieved September 22, 2021, from

<https://medium.com/swlh/what-are-rbms-deep-belief-networks-and-why-are-they-important-to-deep-learning-491c7de8937a>

Cheng, F., Zhang, H., Fan, W., & Harris, B. (2018). Image recognition technology based on deep learning. *Wireless Personal Communications*, 102, 1917-1933.

<https://doi.org/10.1007/s11277-018-5246-z>

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., & Mukhopadhyay, D. (2018).

Adversarial attacks and defences: A survey. ArXiv, Cornell University.

<https://arxiv.org/pdf/1810.00069.pdf>

Dahal, P. (2019, July 30). *Adversarial attacks - breaking and defending neural networks.*

DeepNotes. Retrieved November 16, 2021, from

<https://deepnotes.io/adversarial-attack#analyzing-effects-of-adversarial-inputs>

Dave, D. (2009). *English Handwritten Characters* [Data set]. Kaggle.

<https://www.kaggle.com/dhruvildave/english-handwritten-characters-dataset>

Deoras, S. (2020, August 15). *How to deter adversarial attacks in computer vision models.*

Analytics India Magazine.

<https://analyticsindiamag.com/how-to-deter-adversarial-attacks-in-computer-vision-models/>

Dertat, A. (2017, November 13). *Applied deep learning - Part 4: Convolutional neural networks*. Medium.

<https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>.

Dickson, B. (2020, June 17). *Adversarial attacks against Machine Learning Systems – everything you need to know*. The Daily Swig | Cybersecurity news and views. Retrieved November 3, 2021, from

<https://portswigger.net/daily-swig/adversarial-attacks-against-machine-learning-systems-everything-you-need-to-know>

Firmin, S. (2021, March 4). *Holdouts and CROSS Validation: Why the data used to evaluate your model matters*. Alteryx Community. Retrieved September 26, 2021, from <https://community.alteryx.com/t5/Data-Science/Holdouts-and-Cross-Validation-Why-the-Data-Used-to-Evaluate-your/ba-p/448982>

Gaillard, F. (2020). *Epoch (machine learning): Radiology reference article*. Radiopaedia Blog RSS. Retrieved November 16, 2021, from <https://bit.ly/3oxDX5e>

Gomes, J. (2018, January 17). *Adversarial attacks and defences for Convolutional Neural Networks*. Medium. Retrieved October 12, 2021, from <https://medium.com/onfido-tech/adversarial-attacks-and-defences-for-convolutional-neural-networks-66915ece52e7>

Goodfellow, I. (2020, October 5). *Attacking machine learning with adversarial examples*. OpenAI. Retrieved November 3, 2021, from <https://openai.com/blog/adversarial-example-research/>

- Google Colaboratory. (n.d.). *Colaboratory: frequently asked questions*. Google Colab. Retrieved November 9, 2021, from <https://research.google.com/colaboratory/faq.html>
- Gumilang, M., & Purwarianti, A. (2018). *Experiments on character and word level features for text classification using deep neural network* [Abstract]. 2018 Third International Conference on Informatics and Computing (ICIC), Palembang, Indonesia.
<https://doi.org/10.1109/IAC.2018.8780509>
- Hafemann, L. G., Sabourin, R., & Oliveira, L. S. (2019). Characterizing and evaluating adversarial examples for offline handwritten signature verification. *IEEE Transactions on Information Forensics and Security*, 14(8), 2153-2166.
<https://doi.org/10.1109/TIFS.2019.2894031>
- Hassan, M. U. (2021, February 24). *VGG16 - convolutional network for classification and detection*. Neurohive Retrieved November 9, 2021, from
<https://neurohive.io/en/popular-networks/vgg16/>
- IBM Cloud Education. (2020, May 1). *What is deep learning?* IBM. Retrieved November 2, 2021, from <https://www.ibm.com/cloud/learn/deep-learning>
- IGI Global. (n.d.). *What is word error rate?* IGI Global. Retrieved September 22, 2021, from
<https://www.igi-global.com/dictionary/word-error-rate/32709>.
- Ishibashi, N. (2017). *Understanding adversarial training: Improve image recognition accuracy of Convolutional Neural Network*. [Master's thesis, CUNY City College]. CUNY Academic Works.
- ITBE. (2021, March 11). *Five industries benefiting from handwriting recognition technology*. IT Business Edge. Retrieved October 22, 2021, from

<https://www.itbusinessedge.com/mobile/five-industries-benefitting-from-handwriting-recognition-technology/>

Jain, A. (2019, February 9). *Breaking neural networks with adversarial attacks*. Towards Data Science.

<https://towardsdatascience.com/breaking-neural-networks-with-adversarial-attacks-f4290a9a45aa>

Karim, R. (2019, July 29). *Illustrated: 10 CNN architectures*. Towards Data Science. Retrieved October 11, 2021, from

<https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>

Keras Team. (n.d.). *Keras documentation: Keras layers API*. Keras. Retrieved September 22, 2021, from <https://keras.io/api/layers/>

Khandelwal, R. (2020, June 1). *Common image processing techniques in Python*. Medium. Retrieved November 9, 2021, from

<https://towardsdatascience.com/common-image-processing-techniques-in-python-e768d32813a8>

Khare, V., Shivakumara, P., Chan, C. S., Lu, T., Meng, L. K., Woon, H. H., & Blumenstein, M. (2019). A novel character segmentation-reconstruction approach for license plate recognition. *Expert Systems with Applications*, 131, 219–239.

<https://doi.org/10.1016/j.eswa.2019.04.030>.

Komatsuzaki, A. (2019). *One epoch is all you need*. Department of Mathematics, Georgia Institute of Technology. <https://arxiv.org/pdf/1906.06669.pdf>

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84-90.

<https://doi.org/10.1145/3065386>

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. *Nature*, 521, 436-444.

<https://doi.org/10.1038/nature14539>

Leung, K. (2021, September 23). *Evaluate OCR output quality with character error rate (CER) and word error rate (WER)*. Medium. Retrieved October 12, 2021, from

<https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510>

MathWorks. (n.d.). *ResNet-50 Convolutional Neural Network*. ResNet-50 convolutional neural network - MATLAB. Retrieved November 9, 2021, from

<https://www.mathworks.com/help/deeplearning/ref/resnet50.html#:~:text=ResNet%2D50%20is%20a%20convolutional,%2C%20pencil%2C%20and%20many%20animals>

Memon, J., Sami, M., Khan, R. A., & Uddin, M. (2020). Handwritten Optical Character Recognition: A comprehensive Systematic Literature Review (SLR). *IEEE Access*, 8, 142642-142688. <https://doi.org/10.1109/ACCESS.2020.3012542>

OpenCV. (n.d.). *Image file reading and writing*. OpenCV. Retrieved November 9, 2021, from

https://docs.opencv.org/4.5.3/d4/da8/group__imgcodecs.html

Pandas Development Team. (n.d.). *Pandas.DataFrame – pandas 1.3.4 documentation*. Retrieved November 9, 2021, from

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

- Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). *Dropout improves recurrent neural networks for handwriting recognition*. 2014 14th International Conference on Frontiers in Handwriting Recognition, Hersonissos, Greece.
<https://doi.org/10.1109/ICFHR.2014.55>
- Ramasubramanian K., Singh A. (2019). Deep learning using Keras and TensorFlow. *Machine learning using R*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4215-5_11
- READ-COOP SCE. (2021, May 27). *Character error rate (CER)*. READ-COOP. Retrieved September 22, 2021, from <https://readcoop.eu/glossary/character-error-rate-cer/>
- Ren, K., Zheng, T., Qin, Z., & Liu, X. (2020). Adversarial attacks and defenses in Deep Learning. *Engineering*, 6(3), 346–360. <https://doi.org/10.1016/j.eng.2019.12.012>
- Saha, S. (2018, December 15). *A comprehensive guide to convolutional neural networks – the ELI5 way*. Towards Data Science.
<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- Saini, V. (2020). *Handwriting Recognition* [Data set]. Kaggle.
<https://www.kaggle.com/landlord/handwriting-recognition>
- Shah, T. (2017, December 6). *About train, validation, and test sets in machine learning*. Towards Data Science.
<https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>
- Shivanna, P., Afrid, I. M., Karthik Hebbar, P., & Nishchay, S. K. (2020). Machine learning for handwriting recognition. *International Journal of Computer*, 38(1), 93-101.
<https://core.ac.uk/download/pdf/327266589.pdf>

Steenbrugge, X. [Arxiv Insights]. (2018, January 11). “How neural networks learn” – part II:

Adversarial examples [Video]. YouTube.

<https://www.youtube.com/watch?v=4rFOkpI0Lcg>

Tappert, C. C., & Cha, S.-H. (2007, September 28). *English language handwriting recognition interfaces*. Text Entry Systems. Retrieved November 2, 2021, from

<https://www.sciencedirect.com/science/article/pii/B9780123735911500061>

TensorFlow. (n.d.). *Adversarial example using FGSM: TensorFlow Core*. Retrieved September

21, 2021, from https://www.tensorflow.org/tutorials/generative/adversarial_fgsm.

ThinkAutomation. (2020, December 2). *Why is handwriting recognition so difficult for AI?*

<https://www.thinkautomation.com/bots-and-ai/why-is-handwriting-recognition-so-difficult-for-ai/>

Tomoiaga, C., Feng, P., Salzmann, M., & Jayet, P. (2019). *Field typing for improved recognition on heterogeneous handwritten forms*. 2019 International Conference on Document

Analysis and Recognition, Sydney, Australia. <https://doi.org/10.1109/ICDAR.2019.00084>

UKEssays. (November 2018). *Handwriting recognition software*. Retrieved October 22, 2021,

from <https://bit.ly/30utpew>

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018,

1–13. <https://doi.org/10.1155/2018/7068349>

Wilson, R.C., Shenhav, A., Straccia, M., & Cohen, J. D. (2019, November 5). The Eighty Five Percent Rule for optimal learning. *Nature Communications*, 10.

<https://doi.org/10.1038/s41467-019-12552-4>