APS360 GROUP 34 FINAL REPORT

Miguel Serra
miguel.serra@mail.utoronto.ca

Ahmet Utku Hamamcioglu
u.hamamcioglu@mail.utoronto.ca

Xiaxiao Zhao
seanxiaoxiao.zhao@mail.utoronto.ca

ABSTRACT

In classrooms, deciphering handwritten notes on the board can be challenging, diverting students' focus from lectures. To address this, our project aims to create a model that transcribes images of handwritten notes into text, easing the note-taking process. Leveraging machine learning for pattern recognition and text extraction, our model seeks to enhance the educational experience by making lecture content more accessible. Despite limitations such as dataset size and hardware constraints, our model shows promise in accurately approximating text, highlighting its potential to improve academic productivity and foster inclusive learning environments. —-Total Pages: 10

1 INTRODUCTION

Imagine you're in class, trying to listen to the professor, navigate the slides, and keep your focus. Suddenly, a new challenge appears: deciphering your professor's handwriting on the board. This scenario, far from uncommon, presents students with a dilemma: do they lose focus trying to decipher the notes, or do they ask their peers for help, potentially causing them to lose focus as well? It's within this context that our project finds its motivation.

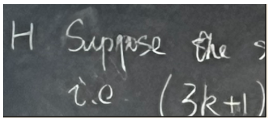

Figure 1: An image our group captured in class.

Project Goal

Our goal is to create a model that enables users to upload a picture of handwritten notes and accurately transcribe them into text. This tool aims to alleviate the challenge of interpreting difficult handwriting during lectures, making note-taking a smoother, more efficient process.

Significance

The significance of this project lies in its potential to enhance the educational experience for students. By streamlining the note-taking process and making lecture content more accessible, our tool aims to improve academic productivity and foster a more inclusive learning environment where all students can focus more on understanding the lecture material and less on deciphering handwriting.

Why Machine Learning?

Machine learning presents a compelling solution for this task due to its capabilities in pattern recognition and text extraction from varied handwriting styles. By leveraging advanced algorithms, our project seeks to develop a robust tool that adapts to different handwriting, ensuring accurate transcription and thus, enhancing the learning experience for students.

The link for the Google Collab can be found here: `https://drive.google.com/file/d/1hAZ-ZN1FUp0QrctB2Zwz4Lq3xuddrutA/view?usp=sharing`

2 ILLUSTRATION

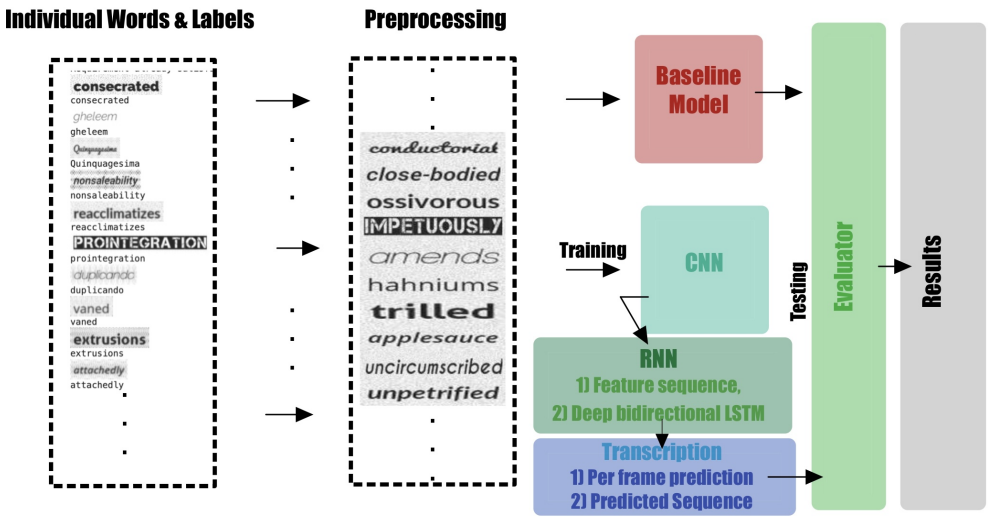

Figure 2: An illustration of our model that provides a general overview of the architecture

3 BACKGROUND & RELATED WORK

In 2015 after the resurgence of neural network approaches to machine learning problems, researchers published a paper outlining a viable architecture for image to word recognition. This model consisted of both convolutional and recurrent layers, and achieved accuracy scores of above 90% (Shi et al., 2015). It was soon after that this proposed model began to be implemented widely. For example, a GitHub user by the name of Meijieru built a functional OCR model on PyTorch, closely following the model architecture of Shi et al (Meijieru, 2023).

In 2020, Abhishek Anand published an educational article explaining his implementation of a word detection model using the Python Keras library, which followed a similar architecture. His model achieved letter accuracy of approximately 95% (Anand, 2020). In subsequent years, the number of educational material regarding CRNN models began to grow, with YouTube channels like Python Lessons which made a tutorial on how to construct a CRNN word detection model in both Tensorflow and PyTorch, which also closely follows the publication of Shi et al. (2015) (PythonLessons, 2023).

Finally, we reach the state of the art OCR model architecture from JaidedAI's EasyOCR. This model builds greatly on Shi et al's proposed architecture (more on this in Section 6), and is even able to accurately detect multiple words in one picture (JaidedAI, 2023).

4 DATA PROCESSING

For our project, we have implemented a robust data preparation pipeline using the trdg (Text Recognition Data Generator) (Belval, 2020) library to create a comprehensive dataset of synthetic images for training our deep learning model. The library generates images that simulate text in various conditions, which is ideal for training models to recognize text in real-world scenarios. We achieved the desired data following the steps: **Data Generation and Cleaning Process:** Installation and Library Imports: We began by installing trdg and importing necessary libraries such as GeneratorFromDict, GeneratorFromRandom, GeneratorFromWikipedia, and supporting libraries for display and file management.

Data Generation: We utilized GeneratorFromDict to create a diverse set of 10000 images. The generator was configured to simulate random skewing and blurring to reflect real-world imperfec-

tions in text presentation. Images were produced with English characters, varied backgrounds, and random perturbations to ensure a challenging dataset. Currently, our model is concentrating on recognizing individual words. The chosen data source proves advantageous, as it provides the flexibility to augment our dataset on demand.

Dataset Splitting: The dataset was split into training, validation, and test sets with proportions of 0.7, 0.12, and 0.18 respectively. This division ensures a sufficient amount of data for the model to learn from while also providing a separate, unbiased dataset for validation and final testing.

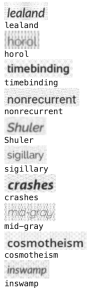

Figure 3: An example of synthetic data generated with TRDG, showcasing simulated real-world text variations for model training.

Image Saving: We created directories within Google Drive to store the generated images, ensuring durability and easy access. A script was employed to populate these directories, carefully labeling each image with its corresponding text for later reference.

Preprocessing: For preprocessing, we implemented a function to resize images to a uniform size of 170x32 pixels and converted them to grayscale.

Challenges Faced: We encountered several challenges, such as ensuring the synthetic data closely mimics the complexities of real-world text and maintaining a balance between realistic text perturbations and readability. Additionally, finding the optimal preprocessing techniques to aid in model training without oversimplifying the data proved to be a nuanced task. By following the outlined steps and overcoming these challenges, we've ensured that our data is well-prepared for the next stages of model training and evaluation.

5 Model Architecture

Since the task at hand is optical character recognition at the word level, we cannot simply use a CNN's and fully connected layers to accomplish this task. This is because instead of just 26 letters in the alphabet + 10 numerical digits as the classes we seek to identify, we now want to be able to reconstruct up to 250 000 English words (and other latin languages). Training over 250 000 classes is not a realistic approach for word recognition. This means the team also needed to resort to recurrent layers, which can accurately train its parameters with the sequential nature of words in mind. A diagram of our approach is seen in Figure 4.

The architecture of our model followed Shi et al. (2015)'s end to end CRNN architecture, consisting of seven convolutional layers, to two bidirectional LSTM recurrent layers, complete with two fully connected layers that output a probability distribution over all of our classes which are the letters and numbers we seek to identify in our strings. Between each layer ReLU activation was used to introduce non-linearity's to the training process. Max pooling was also used between each layer, along with occurrences of batch normalization, followed by a logarithmic softmax output.

Outside of the neural network architecture, it is crucial that the team implement encoding and decoding functionality for the logits outputted by the model. Given that CRNN models, like ours, used for word identification are trained with Connectionist Temporal Classification (CTC) loss, we can implement a CTC decoder function that uses a greedy approach to return the most probable string given a logarithmic softmax input. Although there exists slightly better options for CTC decoding, the team prioritised ease of implementation to ensure we met deadlines.

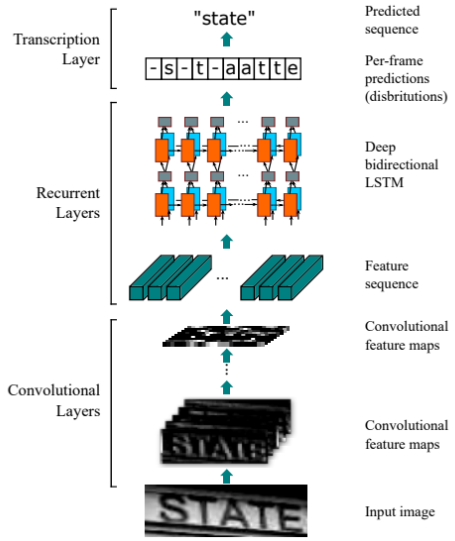

Figure 4: The general approach to our model's architecture as specified by (Shi et al., 2015)

6 Baseline Model

As mentioned in the team's progress report, the complicated nature of optical character recognition calls for a more sophisticated baseline model. This is because "simple" models which do not use deep learning cannot be made to have coherent results for OCR. Given this, we have decided to take an upper bound approach to the usage of the baseline model, where instead of implementing a simple machine learning model and using its results as a lower bound, we take a more sophisticated model and see how our model performs relative to the state-of-the-art. The model we will be using as our baseline is JadedAI's EasyOCR. An outline of EasyOCR's framework is shown in Figure 5

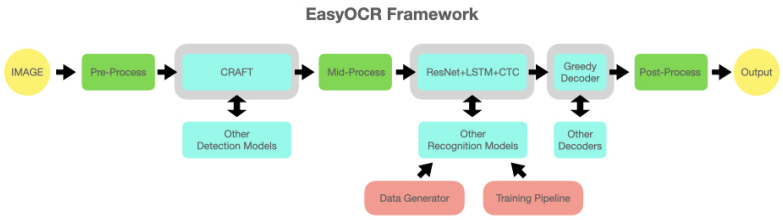

Figure 5: EasyOCR's Framework (JaidedAI, 2023).

6.1 EasyOCR Background

EasyOCR is a pre-trained deep learning OCR model, which is capable of taking in an input image that contains text anywhere in the image, and outputs the borders where text was found, predicted labels, and confidence level. EasyOCR's neural network architecture primarily consists of residual networks, LSTM RNN's, and Connectionist Temporal Classification (CTC) loss. However, EasyOCR also relies on external models like CRAFT for text-detection, and other recognition models as well.

6.2 Quantitative Results

EasyOCR will be tested on the teams testing dataset. The metrics we will be comparing are percentage of predictions that are exactly correct, average Levenshtein distance for predicted label vs. true label, and percentage of total failures (model returned nothing).

For percentage predictions that are exactly correct, EasyOCR showed moderate to poor performance, with a result of 50.83%. However, percent of exactly correct predictions may not be representative of the true performance of the model.

The Levenshtein distance of two strings is described as the number of changes it takes to convert one string into another (Nam, 2019). The results showed that on average, EasyOCR had a Levenshtein distance of 3.44 when comparing predictions to truth labels, which furthers the argument that the performance of EasyOCR over our dataset is moderate.

The percentage total failure of EasyOCR was very low, only failing to return a result 0.55% of the time. All results are summarized in Figure 6.

```
print("EasyOCR Performance Over Test Data:")
print(f"Percent of predictions that were exactly correct: {percentCorrect}")
print(f"Average prediction confidence level: {confidenceAvg}")
print(f"Averege Levenshtein Distance for predicted label vs. true label: {levAvg}")
print(f"Percent failures (where model returned nothing): {percentFails}")
```

```
EasyOCR Performance Over Test Data:
Percent of predictions that were exactly correct: 50.83333333333333
Average prediction confidence level: 62.4648554833262
Averege Levenshtein Distance for predicted label vs. true label: 3.4444444444444446
Percent failures (where model returned nothing): 0.5555555555555556
```

Figure 6: EasyOCR performance over test data.

6.3 Qualitative Results

EasyOCR does not perform well against input images which contain moderate to the high levels of skewing. As mentioned in section 3.1, TRDG distributes these skewing techniques throughout our training, test and validation data. An example of performance against stretched, blurry image vs. a clear, legible image can be viewed in figures 7 and 8.

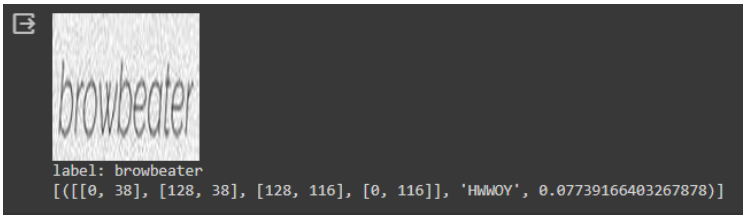

Figure 7: EasyOCR failure to predict text on a blurry and vertically stretched image.

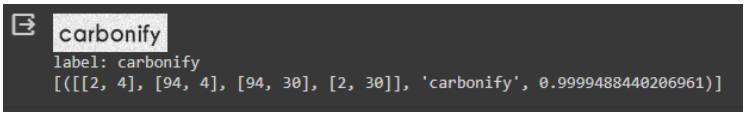

Figure 8: EasyOCR success in predicting text on a image with minimal skewing.

7 Primary Model

The team initially trained the model on a smaller dataset with 5,000 images. The training results indicated signs of over-fitting. The team then trained the model on a larger dataset with 10,000

images and obtained reasonable results. A detailed description of the metrics that we used for evaluation can be found in table 1.

Metrics	**Description**
Training and Validation Loss	The discrepancy between model predictions and actual targets in the training / validation sets
Exact Match Accuracy	The percentage of model outputs that exactly matched the labels
Average Levenshtein Distance	The average number of single-character edits required to change outputs into correct labels
Failure Rate	The percentage of model outputs that were empty

Table 1: Description of evaluation metrics

7.1 Quantitative Results

The training and validation results of the model on the 5,000 images dataset are shown below. In Figure 9, we can see that the validation loss shifted up and down and remained at relatively high values while the training loss stably approached zero. In Figure 10, we can also see that there's a large gap between training and validation exact match accuracy. These are clear signs of model overfitting. The team proceeded to apply several regularization techniques such as dropout and early stopping but they all resulted in minor improvements. We then concluded that due to the complex architecture, the model is overfitting because of the small dataset.

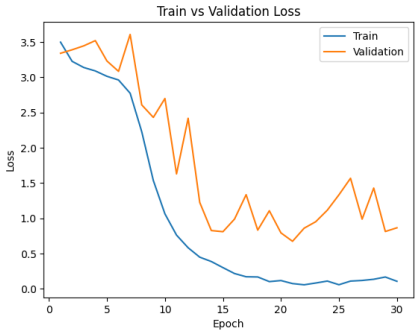

Figure 9: Training and validation loss v.s. number of epochs.

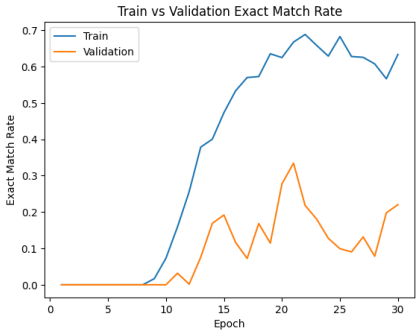

Figure 10: Training and validation exact match rate v.s. number of epochs.

The team preceded to train the model on a larger dataset, containing ten thousand images. In Figure 11, we observe a more reasonable exact match accuracy as the validation rate caught up with the training rate to some extent. In Figure 12, we can also see that the model eventually achieved a relatively low average Levenshtein distance, meaning that each output deviates from the ground truth with an average of 2 to 3 characters on the validation dataset.

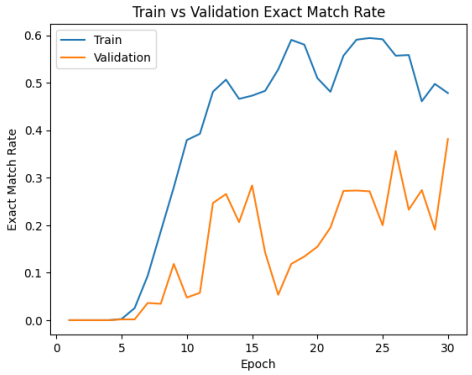

Figure 11: Training and validation exact match rate v.s. number of epochs.

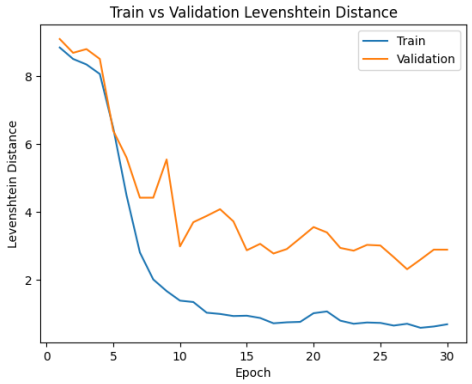

Figure 12: Training and validation average Levenshtein distance v.s. number of epochs.

The team then evaluated this current model on our testing dataset, the results are as shown in figure 13 From the low exact match rate and fairly good Levenshtein distance, we can tell the model is doing better in getting most of the characters right, rather than producing a perfect match. Comparing the result to the performance of EasyOCR on the same testing dataset (Figure 6), our model is not quite as good as EasyOCR in making 100% accurate predictions, but are better at making mismatched outputs closer to the ground truth. It's worth noting that there seems to be a trade-off between word-level and character-level accuracy.

```
Test Loss: 0.8519
Test Exact Match Accuracy: 21.6557%
Test Average Levenshtein Distance: 2.1195
Failure Rate (Empty Outputs): 0.11%
```

Figure 13: Primary model performance over test data.

7.2 Qualitative Results

To put things into perspective, some sample outputs of the primary model are shown in Figure 14. They are chosen randomly from the testing dataset; the original images and the model outputs are put side-by-side for comparison. Although only one out of six predictions ('TOUCHWOOD') exactly matched the ground truth, most of them closely resemble the original image, with 2 or 3 mismatched characters. We can see that the sample outputs are an accurate demonstration of the model performance: low exact match rate but with fairly good Levenshtein distance.

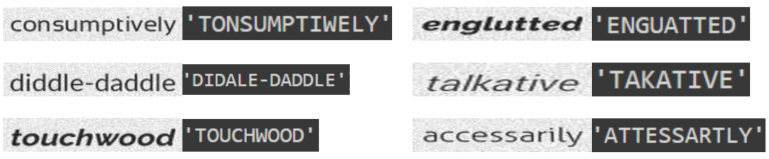

Figure 14: Sample outputs of the primary model over test data.

8 Evaluate Model on New Data

To validate the functionality of our model and to see whether or not it solves the problem, the team used a blackboard image captured in UofT lecture hall to test model performance. Several

text images are then cropped out of the blackboard image as input for the model (Figure 15). The predictions on these new data are shown in Figure 16. As we can see, the model is not doing well on either word or character level matching. The team suspect that this is potentially caused by the lack of real hand-written word images in the dataset. For comparison, the team generated a few more non-handwritten images (we used team members' names to ensure the model have never seen the data before) and the test results are shown in Figure 17. We now see that the model demonstrate a performance similar to the results in section 7.2 as expected.

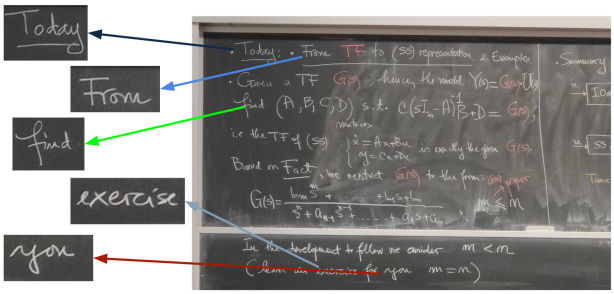

Figure 15: Extracting new data from the blackboard image.

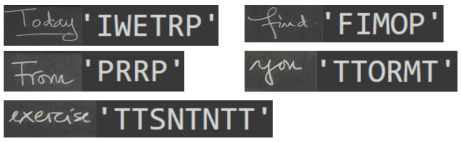

Figure 16: Primary model performance on new handwritten data.

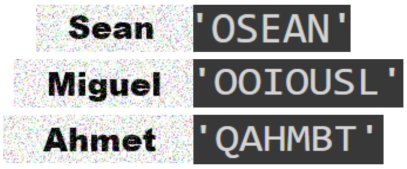

Figure 17: Primary model performance on new non-handwritten data.

9 Discussion

We can see in sections 7.1 through 8 that the model's performance on new data was inadequate, but as expected. It is considered inadequate because the model rarely achieves a 100% correct output, particularly with collected handwritten data. However, it is worth noting that the model often has minimal error in regards to the amount of characters in a word it needs to detect, meaning that the training of the model was working somewhat.

This is not surprising, since given the time constrains, the team was only able to train, validate and test over 10,000 images, which is a very small dataset in respect with the large complexity of our model. It is seen in the publication of Shi et al. that in order for a CRNN model of a complexity like ours, training over a much larger dataset (on the scale of hundreds of thousands, and even millions of images), is a large factor when it comes to high levels of accuracy during inference (Shi et al. (2015)). Unfortunately, due to the project time constraints and hardware limitations the team was not able to train the model enough to achieve satisfactory results.

Another observation the team has made regarding training is that although the data generated by TRDG is diverse with various levels of skewing, the data is not robust enough for the model to perform well on real handwriting samples during inference. To combat this, the team should have mixed different datasets together, including datasets with real life samples of handwriting in academic settings.

Given our results and this experience, the team has learned that for future models, we should train over a dataset with an amount of samples proportional to the complexity of the model (number of layers and types of layers used). We also now know to create a more diverse, custom dataset to achieve better results across multiple types of input images.

10 ETHICAL CONSIDERATIONS

In the development of our model for transcribing handwritten notes into text, we've attentively navigated several ethical dimensions.

Privacy and Data Protection: We are committed to maintaining the privacy and security of all uploaded notes, ensuring the confidentiality of user data.

Bias and Fairness: Our model strives to accurately interpret a diverse array of handwriting styles and languages, promoting fairness. We've designed our technology to be inclusive, ensuring usability for individuals with varying abilities. However, currently, our model only works on words using the Latin alphabet, highlighting a limitation in accessibility that we aim to address in future iterations.

Impact on Learning: We've considered the potential impact of our tool on conventional note-taking practices and overall learning. Our goal is to bolster educational equity by aiding those who encounter obstacles in note-taking without introducing new disparities.

In refining our model, we've incorporated these ethical considerations into its design. It's also important to note that many of these considerations hinge on user engagement and behavior, underscoring the collaborative nature of ethical technology development.

11 PROJECT DIFFICULTY/QUALITY

Our project embarked on the challenging task of developing a model capable of optical character recognition at the word level, which is inherently more complex than recognizing individual characters due to the vast variety of word combinations and morphologies. The ambitious nature of this task was compounded by the diverse and skew-prone input images, further elevating the project's difficulty.

Our model, while not surpassing the highly sophisticated EasyOCR in perfect match accuracy, demonstrated an unexpected proficiency in closely approximating the ground truth, evidenced by a fair Levenshtein distance metric. This is particularly noteworthy considering the limitations we faced, including the modest size of our training dataset and hardware constraints. Additionally, our model showed promise in maintaining the correct character count, indicating a direction for potential refinement and optimization.

The journey of this project has been a testament to our team's ability to adapt and learn beyond the foundational knowledge acquired from our coursework. We've navigated through the intricacies of advanced CRNN architectures, grappled with overfitting by employing various regularization techniques, and meticulously fine-tuned our model parameters. This process was further enriched by the hands-on experience of confronting real-world data challenges and the pragmatics of dataset assembly and augmentation.

Moreover, our team's journey was marked by introspection and recalibration of our expectations and capacities. We initially overestimated our abilities and the scope of commitment required, which led us to downsize our project aims multiple times. This iterative process of scaling our objectives to align with realistic timelines and resource availability was an invaluable lesson in project management and team coordination. It has ingrained in us a deeper understanding of the delicate balance between ambition and feasibility, which will undoubtedly serve as a guiding principle in our future endeavors.

In conclusion, despite the project's inherent difficulty and the hurdles we faced, our model performed commendably, exceeding our expectations given the constraints. The experience has equipped us with a robust set of skills and insights that extend well beyond the project's immediate objectives, laying a solid foundation for tackling complex problems in our future machine learning projects.

REFERENCES

Abhishek Anand. Image text recognition: Using deep learning and deploying the model in cloud. Available at https://medium.com/analytics-vidhya/image-text-recognition-738a368368f5, 2020.

Edouard Belval. Textrecognitiondatagenerator, 2020. URL https://github.com/Belval/TextRecognitionDataGenerator.

JaidedAI. Easyocr github repository. Available at https://github.com/JaidedAI/EasyOCR?tab=readme-ov-file, 2023.

Meijieru. Easyocr github repository. Available at https://github.com/JaidedAI/EasyOCR?tab=readme-ov-file, 2023.

Ethan Nam. Understanding levenshtein distance equation for beginners. Available at https://medium.com/@ethannam/understanding-thelevenshtein-distance-equation-for-beginners-c4285a5604f0, 2019.

PythonLessons. Step-by-step handwriting words recognition with pytorch. Available at https://www.youtube.com/watch?v=N-AuM3_8IrA&t=1160s, 2023.

Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. Available at https://arxiv.org/pdf/1507.05717.pdf, 2015.