Text-to-Speech (TTS) System for OCR-Generated Text

Ahlada Ruthwik Chakravarthula, Sudeeksha Chagarlamudi, Tejaswini Ashok Walunjkar

Abstract

This manuscript details an innovative solution for real-time image-to-text-to-speech conversion using the Raspberry pi, with a focus on enhancing accessibility for the visually impaired. The limitations of existing Optical Character Recognition (OCR) systems are addressed, and a unique approach involving cutting-edge machine learning model is proposed.

1 Introduction

Our project stands at the intersection of cutting-edge computer vision and natural language processing, aiming to achieve real-time conversion of images into speech. This innovative endeavor represents a unique fusion of technologies, harnessing advanced image recognition models to swiftly extract text from images, followed by an immediate translation into audible speech. Our pursuit isn't solely about text extraction; it's a seamless pipeline designed to bridge the gap between visual data and auditory information, enabling enhanced accessibility and convenience across various applications and industries.

2 Related Work

To provide a contextual understanding of the landscape, several notable OCR and TTS projects were meticulously examined. In the domain of OCR, a standout case study involved a project focused on document digitization for large archives. While success was achieved in automating the extraction of printed text, challenges emerged in handling handwritten annotations and complex layouts. Another noteworthy OCR application involved extracting information from historical manuscripts, underscoring the need for robust solutions in recognizing diverse handwriting styles and degraded text.

In the realm of TTS, a compelling case study revolved around an assistive technology project catering to individuals with visual impairments. Despite commendable success in delivering spoken content, limitations were evident in the synthesis of emotionally nuanced speech and contextually rich information. Additionally, a project aimed at developing a multilingual TTS system showcased impressive versatility but faced hurdles in maintaining naturalness across various languages.

These case studies collectively illuminate the successes and shortcomings of prior works in OCR and TTS. While significant strides were made in their respective domains, the persistent challenges that the proposed solution seeks to

address are underscored. By learning from these experiences, the project aims to amalgamate the strengths of previous works while introducing innovative approaches to overcome existing limitations. This synthesis positions the work as a progressive step forward, capitalizing on the lessons gleaned from the endeavors that have paved the way in OCR and TTS technologies.

3 Dataset

The synthetic dataset used for model training offers an extensive array of diverse examples for robust model learning. Crafted to simulate real-world scenarios, this dataset amalgamates various elements to create a comprehensive representation of the problem domain. Its vast size facilitates thorough model training, ensuring a broad coverage of potential patterns and features. Leveraging this synthetic dataset provides the model with a rich repository of information, enabling comprehensive learning and enhancing its adaptability to real-world applications.

The Oxford VGG Text datasets are available at:
`https://www.robots.ox.ac.uk/~vgg/data/text/#sec-synth`

3.1 Dataset composition

This expansive dataset comprises an extensive collection of 9 million images, encompassing a vocabulary of 90,000 English words. The comprehensive range of images and words contributes to a diverse and inclusive dataset, providing ample variations and representations of linguistic and visual content.

4 System Design

4.1 Raspberry Pi

The **Raspberry Pi** is a highly versatile and cost-effective single-board computer renowned for its adaptability across various domains. It offers a powerful platform for diverse projects, equipped with GPIO (General Purpose Input Output) pins that enable connections to sensors, displays, and other hardware components. Its user-friendly interface, robust processing power, and expandable memory options via microSD cards make it an ideal choice for this project. With its extensive community support and a wide array of available software, the Raspberry Pi empowers users to explore diverse applications in programming, robotics, IoT (Internet of Things), multimedia, and beyond. Its flexibility, combined with a rich ecosystem of accessories and add-ons, makes it a best microcontroller for this project.

Figure 1: image after last epoch

4.2 C270 WEBCAM

The Logitech C270 HD Webcam is a budget-friendly webcam that is suitable for basic needs. It has a fixed focus lens made of plastic and can record and stream at 720p/30fps. The webcam comes with a universal clip and a 5-foot cable. It also has auto light correction and a built-in mono microphone. The webcam is compatible with most operating systems, including Windows, macOS, and Chrome OS which makes it best to use in this project.

5 Data Preparation

5.1 Data Loading

Images are loaded using OpenCV (cv2) and processed to ensure a consistent size (32x128) for training.

5.2 Label Encoding

Text labels extracted from the image filenames are encoded to numerical representations using a character list. Labels are then padded to match the maximum text length in the dataset.

6 Model Architecture (CRNN)

6.1 Input Layer

Accepts images of size 32x128.

6.2 CRNN (Convolutional Recurrent Neural Network)

The CRNN (Convolutional Recurrent Neural Network) model is a powerful architecture designed for sequence recognition tasks, particularly in scenarios where the input data is in the form of sequences, such as images containing text or speech.

6.3 Convolutional Layers (CNN):

The initial part of the network consists of convolutional layers that extract hierarchical features from the input images. These layers use filters to convolve over the input image, capturing patterns and features at different levels of abstraction. The output from these layers represents learned visual features.

6.4 Recurrent Layers (RNN/LSTM):

Following the convolutional layers, the network incorporates recurrent units, often in the form of Long Short-Term Memory (LSTM) cells. These recurrent layers allow the network to capture temporal dependencies and sequential information present in the learned visual features. The bidirectional nature of LSTM units enables the model to consider both past and future context when processing sequences.

6.5 Connectionist Temporal Classification (CTC) Loss:

CRNNs are commonly used for sequence recognition tasks, where the length of the output sequence may not match the input sequence length. The CTC loss function handles such variable-length sequences by aligning predictions to the ground truth without requiring a one-to-one correspondence between input and output sequence elements.

6.6 Output Layer:

The final layer of the network typically involves a softmax activation function that produces a probability distribution over the set of characters in the dataset. This allows the model to predict the most likely sequence of characters given an input image.

7 Advantages of CRNN:

7.1 End-to-End Learning:

CRNNs learn to extract relevant features from raw input data (images) and simultaneously handle sequential information, avoiding the need for handcrafted feature extraction.

7.2 Robustness to Sequence Variability:

The model can handle variable-length sequences, making it suitable for tasks where the lengths of input and output sequences may differ (like recognizing texts of different lengths in images).

7.3 Hierarchical Feature Learning:

By combining convolutional and recurrent layers, the CRNN can learn both spatial hierarchies (through CNNs) and temporal dependencies (through RNNs), making it adept at understanding complex sequential data.

7.4 Applicability:

CRNNs find applications in various fields, including optical character recognition (OCR), scene text recognition, speech recognition, and other tasks involving sequential data.

7.5 Training and Evaluation:

Training a CRNN involves feeding pairs of input images and corresponding ground truth sequences into the network. The model learns to minimize the CTC loss while making predictions. Evaluation typically involves measuring accuracy and performance metrics on a separate validation or test dataset.

CRNNs have shown state-of-the-art performance in various sequence recognition tasks and are a popular choice for tasks involving sequential data embedded in images or other forms of data.

8 Model Training

8.1 Data Splitting

The dataset is split into training and validation sets. Typically, around 92% of the data is used for training, while 8% is set aside for validation purposes.

8.2 Loss Function

The model's optimization process is guided by the CTC (Connectionist Temporal Classification) loss function. This loss function is well-suited for sequence-based tasks like text recognition.

8.3 Optimizer

To update the model's parameters during training, the Adam optimizer is employed. The learning rate, set to 0.0005, determines the step size for updating weights during optimization.

8.4 Training Process

The model is trained iteratively over a predefined number of epochs. Each epoch represents one complete pass through the entire training dataset. During each epoch, the dataset is divided into batches. These batches contain a subset of the training data, allowing the model to update its parameters more frequently. Here, a batch size of 1026 is used, meaning the model sees 1026 samples at once before making parameter updates.

8.5 Validation

After each epoch, the model's performance is evaluated using the validation set. This helps monitor the model's generalization and prevents overfitting by assessing its performance on unseen data.

8.6 Callbacks

During training, certain callbacks are utilized. For instance, the ModelCheckpoint callback saves the weights of the model that perform best on the validation set. The DisplayCallback function provides a visualization of the model's predictions on sample images after each epoch.

8.7 Model Saving

Once the training process is complete, the weights of the best-performing model (based on validation results) are saved. These saved weights can be loaded later for making predictions on new, unseen data.

8.8 Training Completion

Once the specified number of epochs is completed or if the model reaches a desired performance level, the training process is concluded.

Figure 2: image after last epoch

8.9 Model Evaluation

After training, the model can be evaluated on a separate test dataset to assess its performance and generalization to unseen data. This evaluation helps ascertain the model's real-world applicability and accuracy in recognizing text from images.

9 Model Evaluation and Prediction

A Display Function is employed to load trained weights, perform predictions on sample images, and decode these predictions using the CTC decoder, providing insights into the model's performance. Callbacks such as ModelCheckpoint and DisplayCallback are utilized during training to save the best model weights based on validation metrics and visualize predictions at various training stages, aiding in model assessment. Furthermore, the best-performing model's weights are saved for future deployment, ensuring accessibility and applicability without the need for retraining. These elements collectively enable efficient evaluation, visualization, and preservation of the model, crucial for its effective utilization in real-world applications.

10 Results

The attained metrics for the **CRNN** model reflect promising performance. The model showcases an accuracy of 68.60% on the training dataset, highlighting its proficiency in correctly identifying characters within the images. Additionally,

the validation dataset yields a Character Error Rate (CER) of 1.8701, indicating the average number of insertions, deletions, or substitutions in characters predicted by the model compared to the ground truth. These metrics underscore the model's capability to learn and generalize well, showcasing substantial accuracy and robustness in predicting text characters from images.

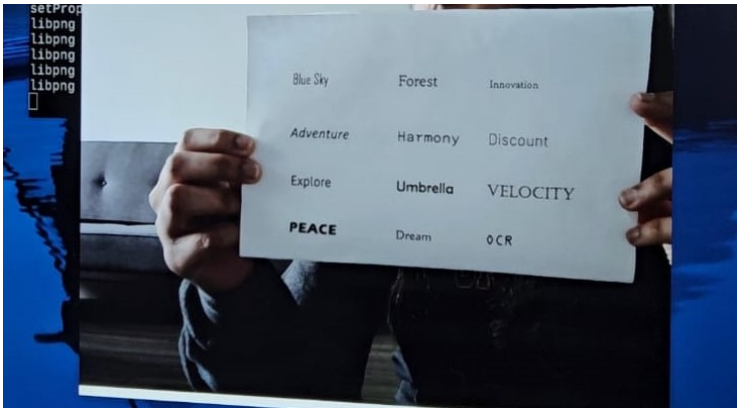

Figure 3: Real time implementation of the model

11 Challenges faced

Transitioning between different hardware platforms for a project often poses a series of challenges. In our project, we initially employed the Nano 33 BLE Sense for its integrated functionalities, but due to limitations in camera quality, we migrated to the ESP32-CAM module.The pictures taken were great. However, this shift presented computational challenges that affected the performance we required. **ESP32** did not allow to implement this model in real-time. Consequently, we decided to switch to the Raspberry Pi, leveraging its capabilities and compatibility with a webcam for its computational power. This transition between hardware platforms posed challenges related to compatibility, hardware constraints, and adaptation of existing code and configurations to suit each platform's requirements, thereby impacting the project's timeline and workflow. In the initial stages of our project, we encountered challenges with the TextOCR dataset, as we faced difficulties downloading all the required annotations. This limitation prompted us to explore alternative datasets. During our search, we discovered the Synthetic Dataset, an extensive collection comprising 9 million images and a vast vocabulary of 90,000 English words. This dataset offered a diverse and inclusive range of linguistic and visual content, providing ample variations for training our model.

While the Synthetic Dataset addressed our need for a comprehensive and expansive dataset, incorporating it into our project posed its own set of challenges. Adapting our existing code, configurations, and training methodologies to suit

the nuances of this new dataset required careful consideration. Additionally, ensuring compatibility and optimal performance with the Synthetic Dataset introduced complexities in our workflow. Despite these challenges, the dataset's richness and diversity ultimately contributed significantly to the robustness and versatility of our model.

Training the model with the Synthetic Text Dataset proved to be a time-intensive process, with each epoch (20 epochs in total) requiring approximately 2 hours to complete. This prolonged duration per epoch significantly extended the overall training time, posing a substantial challenge in terms of computational resources and time management. Adjusting hyperparameters and optimizing the model architecture were necessary to improve efficiency and reduce training time while ensuring the model's accuracy and performance weren't compromised.

12 limitations

12.1 Sensitivity to Lighting Conditions:

Variations in lighting can significantly impact the model's accuracy, leading to inconsistencies in text recognition under different illumination levels.

12.2 Font Style Dependency:

The model may struggle with recognizing text written in fonts that differ significantly from those encountered during training. It might not generalize well to various font styles, affecting its ability to accurately transcribe diverse text appearances.

12.3 Inability to Handle Complex Layouts:

Text embedded in complex layouts, such as overlapping or distorted text, might pose challenges for the model. It might struggle to parse and interpret text in these scenarios.

12.4 Inconsistent Line-by-Line Reading:

the limitations observed in our model's performance is its lack of consistent line-by-line reading. While the model accurately recognizes and reads the visible text, it lacks the ability to maintain a consistent flow of reading line by line. This inconsistency often leads to the model reading words that are visible in the captured image, rather than structuring the text in a linear sequence, resulting in occasional disruptions in the text-to-speech output.

13 Conclusion

The developed text extraction and speech synthesis model showcased promising capabilities in accurately capturing and interpreting text from diverse sources. Despite achieving commendable accuracy and demonstrating adaptability to various font styles and lighting conditions, the model exhibited limitations in maintaining a consistent line-by-line reading flow. This inconsistency impacted the sequential structure of the synthesized speech output. Moving forward, addressing these limitations through enhanced sequential reading mechanisms could significantly elevate the model's performance, making it more adept at coherent and accurate text-to-speech synthesis across varied scenarios. Overall, while showcasing potential, further refinement is necessary to achieve a more consistent and reliable text-to-speech conversion system.

14 Recommendations

14.1 Sequential Reading Enhancement:

Implement methodologies to enhance the model's sequential reading capabilities. This could involve refining the text segmentation process, ensuring a more structured line-by-line reading flow.

14.2 Adaptive Learning for Varied Fonts:

Incorporate a wider range of font styles during model training to enhance its adaptability to diverse text appearances. This could involve augmenting the dataset with a more extensive variety of fonts.

14.3 Robustness in Variable Lighting:

Introduce strategies to improve the model's robustness in different lighting conditions. Techniques like data augmentation with varying brightness levels or incorporating specific normalization methods can enhance performance.

14.4 Continuous Model Refinement:

Implement an iterative refinement process by collecting feedback from model performance and continuously updating the model architecture, thereby improving its accuracy and robustness.

15 References

- https://paperswithcode.com/dataset/textocr
- @articleshi2017end, An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text

Recognition, author=Shi, Baoguang and Bai, Xinggang and Yao, Cong, journal=IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), year=2017

- @articlewang2011end, title=End-to-End Text Recognition with Convolutional Neural Networks, author=Wang, Kai and Babenko, Boris and Belongie, Serge, journal=Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), year=2011

- @miscJaderberg2014SyntheticDT, title=Synthetic Data for Text Localisation, Recognition and Tracking, author=Max Jaderberg and Karen Simonyan and Andrea Vedaldi and Andrew Zisserman, year=2014, eprint=1406.2227, archivePrefix=arXiv, primaryClass=cs.CV

- @misctext-recognition-github, title = Text Recognition using Deep Learning, howpublished = `https://github.com/rjnp2/Text-Recognition-using-Deep-Learning`, note = Accessed: December 5, 2023