PhotoLingo:

CNN-based Language Identification from Images

Team Members:
AnaisCT Corona Perez *(CoronaPerez@wisc.edu)*

Carmine Shorette *(CShorette@wisc.edu)*

Kentaro Takahashi *(Takahashi5@wisc.edu)*

Amelia Zanin *(azanin@wisc.edu)*

October 21, 2023

1 Overview

PhotoLingo is a software developed using convolutional neural networks (CNNs) to detect and identify languages from images, offering a new way to understand textual content in different languages within pictures.

2 Background

The intersection of machine learning and computer vision has made leaps in recent years, particularly in detecting and interpreting textual content within images. The OpenAI Detector stands as a benchmark, using neural networks to discern languages from text inputs, capable of identifying over 100 languages, even those that are rare or underrepresented. Its prowess lies not just in language identification but also in detecting multiple languages within a singular text input. [1]

On a similar note, Facebook's Rosetta employs machine learning to comprehend text within images and videos. It capitalizes on the Faster R-CNN method, an advanced object detection system, for text detection. Initially, it pinpoints potential text-containing regions, and subsequently, a CNN identifies and transcribes the text. Like the OpenAI Detector, it supports over 100 languages. [2]

An equally important development in this domain is Scene Text Detection. This involves the automated process of spotting and localizing text within real-world visuals, like photos or video frames. The state-of-the-art models in this area, including EAST, CTPN, and FOTS, can precisely label text in dynamic and intricate settings like billboards or vehicle license plates. These models leverage both CNNs and RNNs for their tasks. [3]

The current landscape is rich with sophisticated, accurate models: Detector and Rosetta both excel in language detection from textual content. Scene Text Detection models, on the other hand, specialize in the identification of text within intricate real-world images and are optimized enough to provide live translation of text given videos.

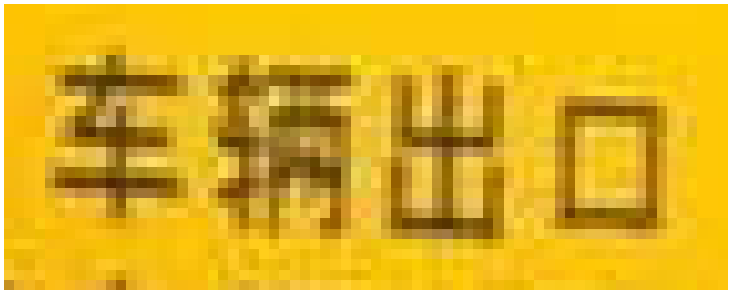
3 Statement of Work

3.1 Datasets

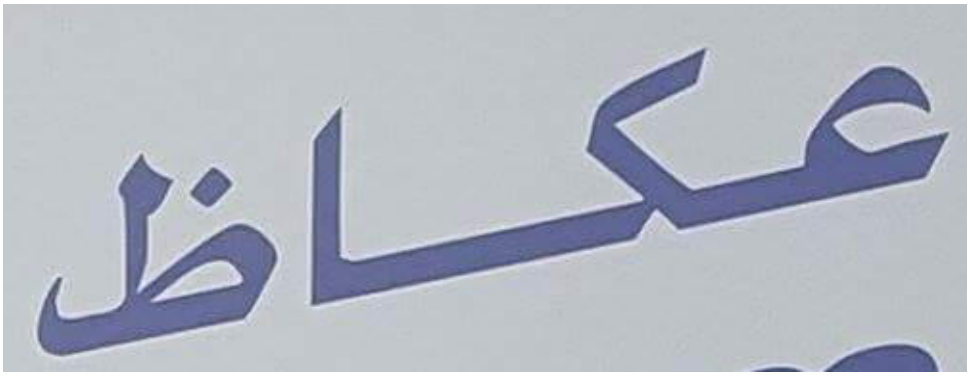
Our project utlizes the dataset from the ICDAR 2019 Robust Reading Challenge on Multilingual Scene Text Detection and Recognition, comprising 80,000 high resolution images of cropped text. The dataset is diverse, covering languages such as Latin, Japanese, Arabic, Korean and Hindi, with accompanying ground truth labels for each image. The dataset can be found here, but an account is required to view the data.

The preliminary inspection reveals the dataset is high-quality, with small-sized images featuring a single language. This specificity aligns with our project's goal of determining the language in a photo, streamlining the process by eliminating the complexity of multi-language images. However, we plan to refine the dataset further by excluding categories like "Symbols" that are less relevant to our objectives, ensuring a more focused and efficient training process for our model.

In Figure 1, two examples from the ICDAR 2019 dataset are shown. Figure 1a demonstrates a sample of Mandarin text, while Figure 1b showcases an example of Arabic



(a) An example of a Chinese text from the dataset.



(b) An example of an Arabic text from the dataset.

Figure 1: Examples of cropped text data from the ICDAR 2019 dataset.

text. These examples highlight the nature of the dataset and its potential challenges in text detection and recognition. For instance, the dataset includes images varying significantly in size; some are large and easily readable, whereas others are quite small and tend to pixelate when enlarged, contributing to the challenge. This variability presents a unique set of challenges, particularly in predicting how a CNN will process and interpret these diverse image dimensions.

We have also presented a breakdown of the category of languages in this dataset. Latin in this case means English or some other Latin-based language

Language	Count
Latin	57,830
Arabic	4,700
Symbols	1,750
Chinese	3,112
Japanese	5,891
Korean	6,671
Bangla	3,766
Hindi	3,884

Table 1: Distribution of languages in the dataset

3.2 Method

Our approach for language detection from images will primarily employ Convolutional Neural Networks (CNNs), inspired by their proven effectiveness in image-based tasks and similar works in text detection and recognition. The dataset provided by the ICDAR 2019 Robust Reading Challenge serves as our foundational resource, originally utilized in a competition where various teams applied diverse techniques for multilingual text detection and recognition.

One of the top-performing teams in this competition was from Tencent, achieving an accuracy of 94.03%. Their methodology involved recognizing text lines and their character-level language types using an ensemble of recognition models. These models were based on a combination of CTC/Seq2Seq and CNN, further enhanced with self-attention mechanisms and RNNs. Their success demonstrates the potential efficacy of using advanced CNN architectures and ensemble methods in this domain. [4]

While their success showcases the potential of complex architectures, we aim to focus on implementing and optimizing a basic CNN model to ensure simplicity and understandability, especially given the constraints of our undergraduate class team.

The primary goal of our project is to create a functional model that can accurately detect the language in a given image, providing a foundation for future work and exploration in this domain. By keeping our approach straightforward and focusing on the basics, we aim to achieve a balance between performance and simplicity, ensuring that the project remains manageable and educational.

3.3 Outcome and Performance Evaluation

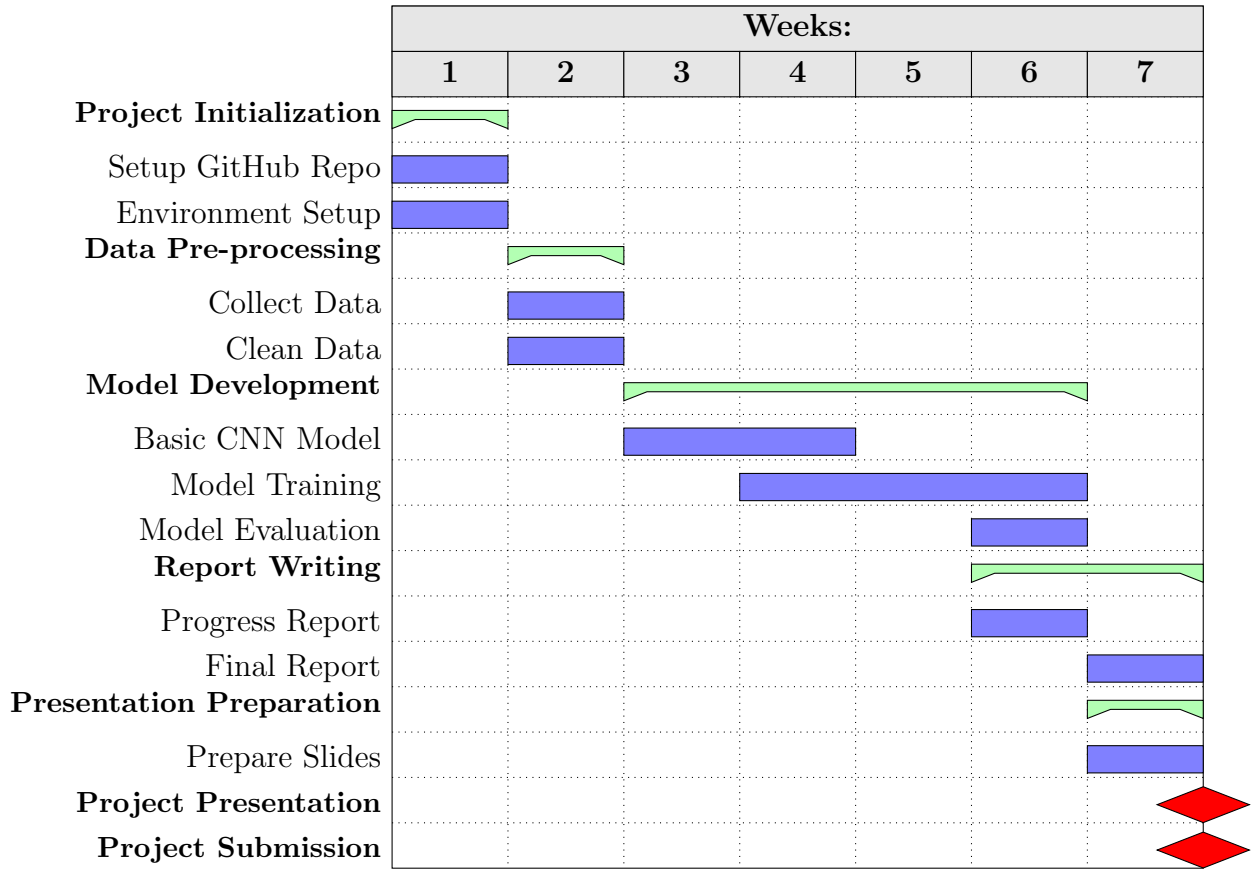
In undertaking this project, we anticipate developing a functional Convolutional Neural Network (CNN) model capable of accurately identifying the language of text in a given image.

The ICDAR 2019 challenge, which provided our dataset, is an invite-only competition that is available to graduate-students and working professionals. Therefore, we will consider it a success if our model can beat any team associated with that event. This sets our goal at $\geq 54.74\%$

In addition to accuracy, the speed at which our model processes images and returns results is crucial, especially for potential real-time applications. We will measure the average time taken per image and seek to optimize our model to minimize this metric, ensuring a balance between speed and accuracy. We will not set a specific goal, but will make sure to include the results in our final presentation.

4 Project Plan

The project will be conducted using a GitHub project, accessible via this link. The instructor and the TA will be provided with read access to monitor the progress and review the materials.



1. **Project Initialization (Week 1)**:
   (a) *Setup GitHub Repo*: Initialize a dedicated repository on GitHub named "PhotoLinngo". Set branch protections, add collaborators, and ensure that the main branch is protected against direct pushes.
   (b) *PyTorch Environment Setup*: Create a virtual environment using 'conda' or 'pipenv'. Install PyTorch, torchvision, and other dependencies like PIL for image processing.
2. **Data Pre-processing (Week 2)**:
   (a) *Dataset Retrieval*: Fetch the dataset from the ICDAR 2019 Robust Reading Challenge.
   (b) *Dataset Cleaning*: Filter out irrelevant categories, such as "Symbols". Utilize PyTorch's DataLoader to handle batch processing and transformations.
3. **Model Development (Weeks 3-6)**:
   (a) *Base CNN Model*: Set up a basic CNN model using PyTorch as a prototype to establish a baseline accuracy.
   (b) *Transfer Learning*: Opt for pretrained models like ResNet or VGG from torchvision's models module. Replace the final layer to suit our classification needs and freeze the earlier layers to retain pretrained weights.
   (c) *Model Training*: Employ PyTorch's optimization libraries, such as Adam or SGD, for training. Utilize a cyclical learning rate schedule and employ early stopping based on validation loss.

(d) *Performance Metrics*: Track metrics such as accuracy, precision, recall, and the F1 score using PyTorch's functionalities and tools like TensorBoard.

4. **Report Writing (Weeks 6-7)**:

   (a) *Progress Report*: Detail the journey of the project, highlighting challenges like potential overfitting, model selection, and optimization issues. Discuss interim solutions and results.

   (b) *Final Report Compilation*: Elaborate on the methodologies, especially the merits of using transfer learning and PyTorch's utilities. Incorporate visual aids like loss curves, accuracy plots, and confusion matrices.

5. **Presentation Preparation (Week 7)**:

   (a) *Slide Creation*: Design a slide deck with sections dedicated to each major phase: data preprocessing, modeling approach (emphasizing transfer learning), results, and conclusions.

   (b) *Dry Run and Practice*: Conduct a team dry run to gauge time management and to polish the delivery. Ensure clear communication of technical details without overwhelming the audience.

   (c) *Project Artifacts Submission*: Organize the GitHub repository, ensuring clean, well-commented code, the final report, and presentation slides. Adhere to the submission guidelines and grant read permissions to the instructor and TA.

Throughout the project, regular commits and updates should be made to the GitHub repository to ensure all changes are tracked and the repository stays up to date.

References

[1] Torry Mastery. "Exploring OpenAI's State-of-the-Art Language Detector". In: *DotCom Magazine* (2023). URL: https://dotcommagazine.com/2023/04/exploring-openais-state-of-the-art-language-detector-openai-detector/.

[2] Viswanath Sivakumar. *Rosetta: Understanding text in images and videos with machine learning*. 2018. URL: https://engineering.fb.com/2018/09/11/ai-research/rosetta-understanding-text-in-images-and-videos-with-machine-learning/.

[3] "Scene Text Detection". In: *Papers with Code* (2023). URL: https://paperswithcode.com/task/scene-text-detection.

[4] *Results - ICDAR 2019 Robust Reading Challenge on Multi-lingual scene text detection and recognition*. 2019. URL: https://rrc.cvc.uab.es/?ch=15&com=evaluation&task=2.