PhotoLingo:

CNN-based Language Identification from Images

Progress Report

Team Members of Group 2:

Anais Corona Perez *(CoronaPerez@wisc.edu)* **UG**

Carmine Shorette *(CShorette@wisc.edu)* **UG**

Kentaro Takahashi *(Takahashi5@wisc.edu)* **UG**

Amelia Zanin *(AZanin@wisc.edu)* **UG**

December 14, 2023

1 Abstract

In the PhotoLingo project, our undergraduate team developed a machine learning model using Convolutional Neural Networks (CNN) to identify five languages: Arabic, Hindi, Japanese, Korean, and Latin-script in photographs. We experimented with two principal architectures: a foundational CNN model trained for 10 epochs, and a more advanced ResNet-18 model trained for 25 epochs, both utilizing Adam's optimizer and CrossEntropyLoss() function. Our basic CNN comprises three convolutional layers, each followed by max-pooling and a fully connected layer. Additionally, we introduced a VGG-based model, trained for 5 epochs with the SGD (Stochastic Gradient Descent) optimizer. Unique to the VGG model, we integrated a Squeeze-and-Excitation network as an attention mechanism, specifically focusing on enhancing the model's ability to identify crucial text areas within images. This integration aims to refine language detection accuracy. While this project is a preliminary step in its domain, it highlights the practical use of CNNs for language recognition in images and serves as a valuable learning experience for further explorations in the field. Code can be found in the project repository hosted on GitHub here.

2 Introduction

2.1 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.

2.2 Motivations

The 'PhotoLingo' project is driven by the goal of exploring the capabilities of Convolutional Neural Networks (CNNs) in the unique application of identifying languages from photographs. This project emerged from our interest in practical applications of machine learning in everyday situations, such as helping travelers or students quickly identify unfamiliar languages. By focusing on language recognition in images, we are developing PhotoLingo for educational purposes and personal curiosity, serving as a hands-on opportunity for us to deepen our understanding of CNNs and their potential uses in real-life scenarios.

2.3 Significance

The 'PhotoLingo' project holds significant value both academically and practically. Academically, it provides a practical application of CNNs, showcasing how theoretical knowledge from our coursework can be applied to solve real-world problems. This project also demonstrates the interdisciplinary nature of machine learning, bridging the gap between technology and linguistics.

2.4 Related Works

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.

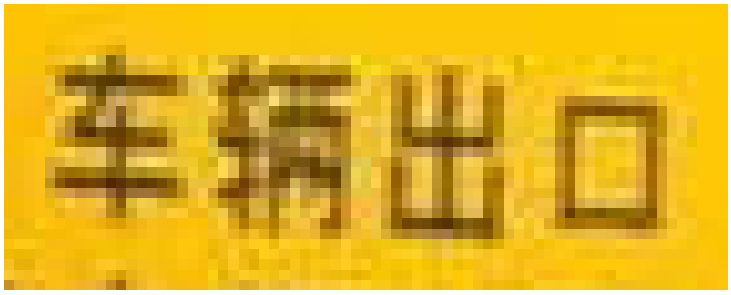
3 Data

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.

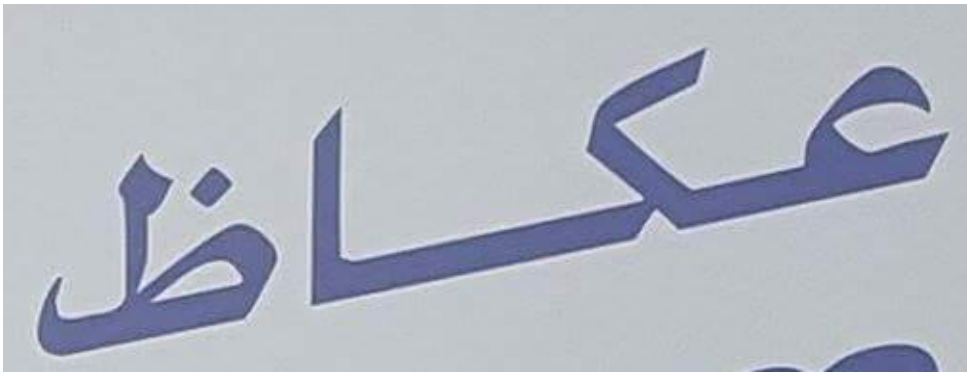
The preliminary inspection reveals the dataset is high-quality, with small-sized images featuring a single language. This specificity aligns with our project's goal of determining the language in a photo, streamlining the process by eliminating the complexity of multi-language images. However, we plan 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 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 in Table 1. Latin in this case means English or some other Latin-based language. To clean our data set, we removed Symbols, Bangla and Chinese to simplify our model. Due to the uneveness of the language distribution in our dataset, we settled for a stratified data partition, taking a 70/30 split for training and testing for each category. Our cleaned and partitioned dataset is broken down in Table 2



(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 data set.

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

Language	Training Set	Testing Set
Latin	41,196	17,655
Arabic	3,290	1,410
Japanese	4,158	1,782
Korean	4,751	2,035
Hindi	2,752	1,179

Table 2: Language Distribution in Training and Testing Sets

4 Tasks Performed

4.1 Baseline Implementation Review

Our baseline implementation involved the construction and training of a basic Convolutional Neural Network (CNN). This model, designed as a starting point, consisted of three convolutional layers with subsequent max-pooling, followed by a fully connected layer. The primary task of this CNN was to process input images and classify them into one of the five language categories: Arabic, Hindi, Japanese, Korean, or Latin-script. CNNs are a class of deep neural networks, primarily used in analyzing visual imagery. They are characterized by their convolutional layers, which apply a number of filters to the input to capture spatial hierarchies of features (like edges in the lower layers, complex shapes in the deeper layers). The implementation served as a foundational approach to understand the basic dynamics of image-based language recognition using CNNs.

4.2 Experimentation and Enhancements

1. **ResNet-18 (Residual Network) CNN Architecture:** Moving beyond our baseline, we incorporated the ResNet-18 architecture. Known for its deeper layers and residual connections. It is composed of 18 layers, including convolutional layers, batch normalization, ReLU activations, and a fully connected layer at the end. ResNet's key feature is its residual connections that help in addressing the vanishing gradient problem in deep networks. This feature allowed us to tackle more complex features in the images, improving accuracy over our basic CNN model.

2. **VGG Model with Squeeze-and-Excitation Network:** VGG is known for its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. The SE network, an attention mechanism, focuses on enhancing the representational capacity of the network by learning to use global information to selectively emphasize informative features and suppress less useful ones. This integration was specifically aimed at improving the model's focus on text-critical regions within the images, a feature not present in our baseline model.

3. **Optimizer and Epoch Variations:** We explored various optimizers and training durations to optimize model performance. We employed Adam's optimizer, known for its adaptive learning rate, for both our baseline CNN and ResNet-18 models, whereas we chose Stochastic Gradient Descent (SGD) for the VGG model due to its stable convergence properties. The training epochs were set differently for each model to assess their learning capabilities: 10 epochs for the basic CNN, 25 for ResNet-18, and 5 for the VGG. These choices helped us understand how different optimizer and epoch configurations affect language recognition efficacy in CNN models.

5 Results

In this section, we present the outcomes of our experiments focusing on the language recognition task. For a comprehensive evaluation, we report on two sets of accuracy measurements: one from our internal testing dataset, constituting 30% of our data, and

the other from an external ICDAR testing dataset. The external dataset, while not providing labels for direct validation, allowed us to submit our model's predictions for evaluation. The external evaluation program also provides us with a confusion matrix and we will include that as well.

Other metrics that we will include specifically on our test set is precision, recall, specificity and F1 score (which is especially useful in cases of uneven class distribution).

5.1 Baseline CNN Model

GT/Detection	Arabic	Hindi	Japanese	Korean	Latin
Arabic	1285	1	11	10	103
Hindi	0	1114	6	8	52
Japanese	8	7	1422	66	279
Korean	9	4	63	1731	229
Latin	88	35	239	161	17133

Table 3: Confusion Matrix for Base CNN Language Detection (Local Results)

GT/Detection	Arabic	Hindi	Japanese	Korean	Latin
Arabic	3834	20	106	132	1050
Hindi	9	3679	100	47	389
Japanese	228	131	2939	1027	3832
Korean	330	222	1505	6425	4510
Latin	914	395	2160	1593	55575

Table 4: Confusion Matrix for Language Detection for Base CNN (ICDAR Results)

The confusion matrices above lead to an accuracy of .943 and .795, respectively.

Language	**Precision**	**Recall**	**F1 Score**	**Specificity**
Arabic	0.924	0.911	0.918	.995
Hindi	0.960	0.944	0.952	.998
Japanese	0.817	0.798	0.807	.986
Korean	0.876	0.850	0.863	.989
Latin	0.963	0.970	0.967	.897

Table 5: Precision, Recall, F1 Score and Specificity per Language for Base CNN

5.2 ResNet-18 Model

The confusion matrix present in Table 6 referring to local testing results led to an accuracy of .938. The ICDAR confusion matrix present in Table 7 shows an accuracy of .863. Other metrics such as specificity, recall, F1 score are described in Table 8

GT/Detection	Arabic	Hindi	Japanese	Korean	Latin
Arabic	1205	1	22	13	169
Hindi	2	1079	16	8	75
Japanese	12	4	1347	54	365
Korean	5	2	83	1663	283
Latin	50	18	203	98	17287

Table 6: Confusion Matrix for Language Detection using ResNet-18 (Local Results)

GT/Detection	Arabic	Hindi	Japanese	Korean	Latin
Arabic	4167	7	140	40	788
Hindi	5	3938	39	15	227
Japanese	64	46	3975	576	3496
Korean	74	44	823	8299	3752
Latin	359	118	1064	827	58269

Table 7: Confusion Matrix for Language Detection using ResNet-18 (ICDAR Results)

Language	**Precision**	**Recall**	**F1 Score**	**Specificity**
Arabic	0.946	0.855	0.898	0.997
Hindi	0.977	0.914	0.945	0.999
Japanese	0.806	0.756	0.780	0.985
Korean	0.906	0.817	0.859	0.992
Latin	0.951	0.979	0.965	0.861

Table 8: Precision, Recall, F1 Score and Specificity per Language using ResNet-18

5.3 VGG with Squeeze-and-Excite attention mechanism

We do not show any results for this model. It was by far the most inaccurate model with an accuracy of .79. This was our most complex model, but was also by far the slowest to train and provided predictions much slower. Even though it was trained on 5 epochs, it is possible over fitting occurred or some other hard-to-identify bug in the model since it mostly predicted Latin $\geq 90\%$ of the time. It is also possible this model just suffered from an uneven dataset that is majorly Latin to begin with.

5.4 Ensemble Method

Our best performing model based on the confusion matrices present in Table 9 and 10. This leads to a .961 accuracy on the local dataset and 0.902 accuracy on the ICDAR dataset. Other metrics are summarized in Table 11.

GT/Detection	Arabic	Hindi	Japanese	Korean	Latin
Arabic	1205	1	22	13	169
Hindi	2	1079	16	8	75
Japanese	12	4	1347	54	365
Korean	5	2	83	1663	283
Latin	50	18	203	98	17287

Table 9: Confusion Matrix for Language Detection using Ensemble (Local Results)

GT/Detection	Arabic	Hindi	Japanese	Korean	Latin
Arabic	4344	5	76	52	765
Hindi	5	3981	18	16	204
Japanese	69	34	3875	532	647
Korean	106	67	752	8269	3798
Latin	220	69	647	567	59134

Table 10: Confusion Matrix for Language Detection using Ensemble (ICDAR Results)

Language	**Precision**	**Recall**	**F1 Score**	**Specificity**
Arabic	0.98	0.90	0.94	0.98
Hindi	0.99	0.95	0.97	0.99
Japanese	0.91	0.82	0.86	0.91
Korean	0.95	0.86	0.91	0.95
Latin	0.96	0.99	0.98	0.96

Table 11: Precision, Recall, F1 Score and Specificity per Language using Ensemble

5.5 Summary

Our summary results for the evaluation of the different models is provided in Table 12. According to these results, our Base CNN was suprisingly accurate. The ResNet-18

model performed worse on the local dataset, but better on the ICDAR dataset, indicating it is better at generalizing other languages since it is more diverse compared to the local one.

Our optimal strategy involved integrating the two models. We calculated the likelihood of the image representing each language using both models. Then, we averaged these probabilities for each language. Finally, we selected the language that had the highest average probability.

Model	**Local**	**ICDAR**
Basic CNN	0.943	0.795
ResNet-18	0.938	0.863
Ensemble	0.961	0.902
VGG	∼0.70	∼0.70

Table 12: Summary Performance on Local and ICDAR Testing per Model

6 Discussion

As discussed in class, ensemble methods usually do produce better results, and this is the case here. However, there are several further steps we can take to refine our models and potentially enhance their performance.

A more systematic approach to hyperparameter tuning should be of primary concern. We have consistently used a batch size of 32 and used 62 once, but exploring a range of batch sizes could help us identify the optimal number for model training. Similarly, the number of epochs was chosen somewhat arbitrarily. Implementing a more rigorous method such as cross-validation, coupled with early stopping, could prevent over fitting and ensure that the models are trained just the right amount.

Regularization techniques are another area that could help avoid over fitting. Our initial efforts did not focus on preventing over fitting. Future work could include the application of dropout, L1/L2 regularization, or batch normalization to promote model robustness.

The choice of optimizer also plays a crucial role in the convergence and performance of deep learning models. While we used the Adam optimizer, experimenting with Stochastic Gradient Descent (SGD) on other models besides the VGG could offer benefits such as potentially better generalization at the cost of requiring more careful tuning of learning rates and momentum.

Lastly, Squeeze-and-Excitation (SE) networks hold promise by allowing the network to perform dynamic channel-wise feature recalibration. Applying SE networks could help our models focus on the most informative features, thus improving their accuracy and reliability in language detection tasks. The building blocks for these networks are already in our code (we used them for VGG), but we have not applied them to our more successful models. [4]

We have plenty of ways on the path towards improvement, but we are also very happy with the results where they stand. We originally came into the project with the goal of building a fast model that had an accuracy $\geq 60\%$. We should of set our aims higher since the ICDAR competition was a bad metric to attempt to compare our own performance.

Given the results from just the Base CNN model, it is clear that we luckily removed the most challenging languages from the original ICDAR dataset. This makes sense, Bangli is very similar to Hindi. The symbols category theoretically would of been difficult, since symbols share almost no characteristics amongst each other.

But achieving a 93.15% accuracy (weighted between Local and ICDAR) is a significant accomplishment. This marks a 6.2% improvement over our base model. Considering the unexpected effectiveness of the base model right out of the gate, this improvement is not just incremental, but a considerable stride forward in our project.

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] Jie Hu. *Squeeze-and-Excitation Networks*. 2019. arXiv: 1709.01507 [cs.CV].