Comparing the Efficacy of Machine Learning Methods in Classifying the ASL Alphabet

Aalto University

CS-C3240 Machine Learning

Stage 2

October 2024

1. Introduction

Sign language serves as a critical component of communication for the hard-of-hearing community. However, in everyday life, automatic recognition and interpretation of sign language are still underdeveloped, limiting the use of sign language communication with machines or systems. Developing machine learning models that can accurately classify images could help with various real-life scenarios, including enhancing accessibility in communication and integrating with human-computer interaction systems. The application domain for the project is enhancing a greatly overlooked part of everyday life for some of us, that being accessibility.

The ultimate goal of the model will be to categorize images of the signs in the alphabet of American Sign Language (ASL). We will be testing two very different methods for this, those being Logistic Regression (LR) and Convolutional Neural Networks (CNNs). The former is a simpler method, while the latter is more complicated and power-intensive to run.

This report is structured as follows: section 2 discusses the dataset used, which consists of images of ASL letters. Section 3 outlines the methods used, including data preprocessing (reducing image size and color depth), feature learning (using Principal Component Analysis), and the two models employed: LR and CNNs. This section also details the loss functions used for each model. Section 4 presents the results, comparing the accuracy and loss of the two models, showing CNNs outperformed Logistic Regression. Section 5 concludes the report by summarizing the findings and suggesting future improvements.

2. Problem Formulation

The ASL Alphabet training dataset [1] used consists of 26 classes with each class representing a single alphabet character in ASL. This dataset has been sourced from the publicly accessible online data science platform Kaggle. Each dataset instance therefore represents multiple feature vectors, where each element represents a single pixel from an image. Each pixel's value is represented by an 8-bit integer, ranging from 0 (black) to 255 (white). We will be testing different amounts of dimensions for the input, which range from 20 to 2500.

The objective is to build a model that accurately classifies images based on their corresponding labels. This task is perfect for supervised machine learning, where a model is developed using labeled data from the ASL Alphabet dataset. These labels act as the ground truth, allowing for an assessment of the model's accuracy by comparing its predictions and measuring the resulting loss. While there are other similar applications of machine learning utilized for sign language classification, most have only attempted to use a neural network [2]. This study will also employ a Logistic Regression approach. While less common than neural networks, this method has been explored in some studies [3]. For a comparison, a Convolutional Neural Network will also be used. The main goal of the study is to see how close LR with its simplistic approach can get to a neural network.

3. Methods

3.1. Dataset

The original categorical ASL Alphabet training dataset contains 87,000 200 × 200 RGB images, classified into 29 different classes. The classes consist of the 26 English alphabet letters A-Z, a space character, a delete character, and images with no hand signs. For the training dataset, only the letters from A through Z will be needed, leaving us with 78,000 images. The original dataset also contains 29 test images which will not be required since the dataset will be split with the common ratio of 80% for training, 10% for validation, and 10% for testing. This results in the following data split:

Training Set, 80% of the dataset (62,400 images). Used for training the model. The rationale behind such a large dataset is to allow the model to learn effectively.

Validation Set, 10% of the dataset (7,800 images). Used for hyperparameter tuning and monitoring model performance during training. It is also essential for model selection and to avoid overfitting the training data.

Test Set, 10% of the dataset (7,800 images). Held out until the end to evaluate the final model's performance on previously unseen data. This provides an unbiased estimate of the model's generalization performance. Using a greater test set than the 29 test images provided will result in greater accuracy in tracking the performance of the model.

The splitting and processing of the data are done with *cuML* [4]. Do note that in this study we will be using the *RAPIDS* [5] suite of APIs (like the aforementioned *cuML*) for their integration of NVIDIA's CUDA so that we can accelerate the processing by utilizing a GPU. The *RAPIDS* APIs provide similar functionality to familiar Python APIs like *scikit-learn* [6] and *pandas* [7] but offload the processing to the GPU.

3.2. Data Preprocessing

Each instance in the dataset [1] consists of 3 channels to represent colored images. To reduce the feature vector size, the three color channels are combined into a single grayscale channel. This is accomplished using the Python Image Library fork *Pillow* [8]. Each image will also be scaled down to 50 × 50 pixels, greatly reducing the complexity of the feature vectors. This specific resolution is chosen so that each size is downscaled by a factor of 4, avoiding scaling artifacts. After preprocessing, each image is flattened to a singular vector with 2500 features.

Additionally, Label Encoding is used to convert categorical class labels (e.g. 'A', 'B', 'C') into a numerical format. The models we use require numeric inputs for the target labels to perform optimizations and calculate probabilities for the classification task.

3.3. Feature Learning

After preprocessing, each image is flattened to a singular vector with 2500 features. While this is significantly less than the ~120,000 features in the original dataset, we initially considered further dimensionality reduction using Principal Component Analysis (PCA). However, due to the efficiency of the Logistic Regression model with the full 2500 features as can be seen in Figure 1, PCA was deemed unnecessary for this study. The runtime cost incurred (that being around a minute to fit all the features) wasn't that great to justify a further detail reduction with PCA.

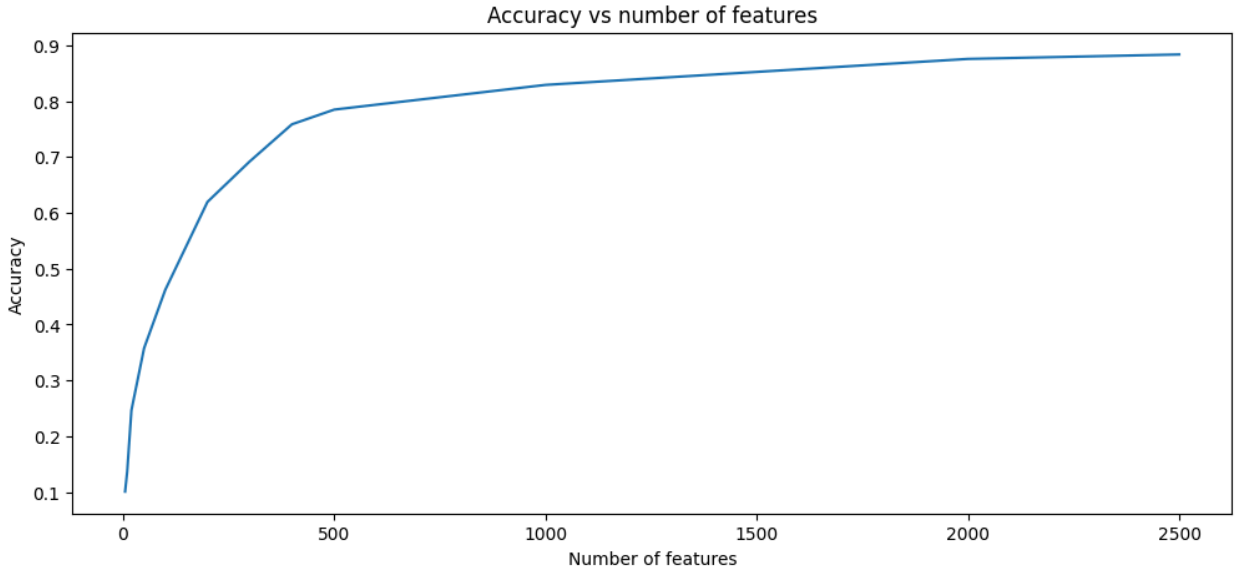

Figure 1: Accuracy of Logistic Regression compared to the number of features

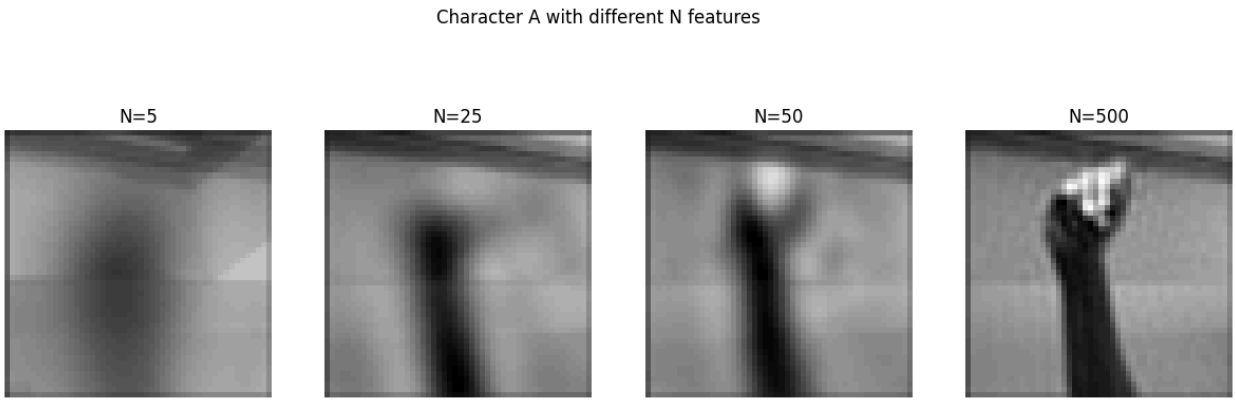

Figure 2: The detail reduction cost of PCA. The figure clearly shows how a greater amount of features is advantageous for recognizability.

3.4. Model

3.4.1. Logistic Regression

For this multi-class classification problem, a Logistic Regression model has been selected. This model is suitable due to its simplicity and efficiency, especially when dealing with a relatively large number of features, which is the case even after the PCA dimensionality reduction. Logistic Regression models the probability of each class given the input features using a linear function followed by a sigmoid transformation. This allows for direct interpretation of the model's output as probabilities, facilitating the selection of the most likely class.

3.4.2. Convolutional Neural Networks

Additionally, a Convolutional Neural Network (CNN) has been developed with *TensorFlow* [9] to compare the former model. CNNs are particularly effective for tasks involving image data, due to their ability to automatically learn spatial hierarchies of features through convolutional layers. The model first resizes the grayscale images to (50, 50, 1), representing a single channel image (a floating point number between 0 and 1).

The architecture consists of three convolutional layers, each followed by a ReLU activation function to introduce non-linearity and enable the model to learn more complex patterns. After the convolutional layers, the MaxPooling layers reduce the dimensionality, preserving only the most important information, while fully connected Dense layers finalize the classification task. This approach enables the model to make accurate predictions based on intricate image patterns.

3.5. Loss Functions

3.5.1 Categorical Logistic Loss

The Categorical Logistic Loss function (also known as the cross-entropy loss function) will be used as the loss function during training of the logistic regression model. It's the natural choice for logistic regression models as it directly measures the discrepancy between the predicted probabilities and the true labels. Minimizing this loss encourages the model to learn parameters that produce accurate probability estimates for each sign.

The combination of Logistic Regression and Logistic Loss provides a robust and interpretable framework for classifying the signs. The linear nature of the model allows for more efficient training and inference, while the probabilistic output aids in understanding the model's confidence in its predictions.

3.5.2. Sparse Categorical Cross-entropy

The Sparse Categorical Cross-entropy loss function is employed during the training of the CNN model. This loss is specifically designed for multi-class classification problems where labels are integers rather than one-hot encoded vectors. It calculates the difference between the true label and the predicted class probabilities, encouraging the network to increase the probability assigned to the correct class. The model adjusts its weights to improve classification accuracy by minimizing this loss and learning to better recognize patterns in the input images. This choice of loss function is well-suited to the structure of the CNN and the categorical nature of the classification task.

4. Results

Convolutional Neural Networks and Logistic Regression demonstrated their strengths and weaknesses.

4.1. Comparison

The LR model achieved a test set accuracy of approximately 88%. The reduced feature space and the GPU's faster processing time allowed the model to train efficiently on the large dataset. Regardless, the linear nature of the LR model led to some limitations in capturing more complex patterns in the image data.

The CNN outperformed LR with its ability to learn spatial hierarchies of features through convolutional layers. By leveraging the GPU for processing, the CNN achieved a high test set accuracy of approximately 99.8%, with a runtime comparable to the LR model

Method	**Accuracy of method on test data**	**Average loss (error)**
Logistic Regression	88.71%	0.4944
Convolutional Neural Network	99.81%	0.0058

Table 1: Accuracy and error of both methods

The Categorical Logistic Loss for LR resulted in a higher loss value, reflecting its limitations in modeling the more intricate relationships within the data. In contrast, the Sparse Categorical Cross-entropy for CNN decreased steadily during training, leading to a better fit and more accurate classification.

Taking these factors into account, a CNN is objectively the better model when it comes to the image classification done for this study.

5. Conclusion

In this project, two distinct approaches were evaluated for their efficacy in classifying the ASL alphabet. The results demonstrated that while the Logistic Regression's accuracy was surprisingly high for the simplicity of its model, it still struggled to capture the complexity of image data in this classification task. Convolutional Neural Networks, on the other hand, showed superior performance achieving highly more accurate classification suitable for a real-world application. The report also discussed how features extracted from images affect the accuracy of the result, demonstrating how the CNN's ability to learn complex features contributed to its superior performance compared to Logistic Regression.

The models were primarily designed to classify static images. However, for real-world applications, particularly in human-computer interaction, achieving a real-time classification and extending the model to recognize dynamic sign language movements (e.g. gestures and phrases) could be possible by improving the CNN model built so far. In the future, the accuracy of the model could be improved even further by adding more varied images of signs in different lighting conditions, of different people, and of different situations. The model could also be improved by adding a preprocessing step to remove the background of an image to remove a variable between different images.

References / Bibliography

[1] A. Nagaraj, “ASL Alphabet” (2018). [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/29550

[2] M. Ugale and O. R. S. Shinde and K. Desle and S. Yadav, “A Review on Sign Language Recognition Using CNN” (2023). https://doi.org/10.2991/978-94-6463-136-4_23

[3] D. Bala, M. A. Hossain, M. A. Islam, M. Mynuddin, M. S. Hossain and M. I. Abdullah, “Effective Recognition System of American Sign Language Alphabets using Machine Learning Classifiers, ANN and CNN” (2022), https://doi.org/10.1109/IATMSI56455.2022.10119336

[4] train_test_split — cuml 24.08.00 documentation [Online]. [Accessed 7.10.2024]. https://docs.rapids.ai/api/cuml/stable/api/#model-selection-and-data-splitting

[5] RAPIDS Development Team — “RAPIDS: Libraries for End to End GPU Data Science” (2023) https://rapids.ai

[6] Pedregosa, F., et al. — "Scikit-learn: Machine Learning in Python," in Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[7] The pandas development team (2020). Pandas. Zenodo. https://doi.org/10.5281/zenodo.3509134

[8] Image Module - Pillow (PIL Fork) 10.4.0 documentation [Online]. [Accessed 20.9.2024]. https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.Image.convert

[9] TensorFlow Developers. (2024). TensorFlow (v2.18.0-rc1). Zenodo. https://doi.org/10.5281/zenodo.13901223

Appendix