AI211 - Course Project

ASL Character Recognition using CNNs

...

Atesam Abdullah
BAI, FCSE

BACKGROUND

This project uses **CNN and Tensorflow to translate ASL** and to **improve communication** between **hearing and deaf individuals**.

Computer vision has made significant progress in interpreting sign language through the use of CNN and Tensorflow, showing promise in converting ASL into written or spoken language.

The value of ASL lies in its ability to bridge communication gaps between the hearing and deaf communities. ASL is widely used among those who are deaf in the US.

Many with functional hearing struggle to understand ASL, hindering communication. Improving ASL translation technology promotes inclusivity.

LITERATURE REVIEW

We will go through 3 things:

- ASL datasets
- ASL Alphabet Recognition Techniques
- CNN Architectures for Image Recognition

ASL Datasets

Dataset Name	Number of Classes	Number of Samples	Image Size	Data Source
ASL Alphabet	29	87,000	200x200	Custom collection
MNIST-ASL	24	22,200	28x28	Kaggle
ASL Finger Spelling	24	67,000	100x100	Custom collection
Synthetic ASL Alphabet	27	27,000	512x512	Kaggle
ASL Sign Language Alphabet Pictures	26	8442	1920x1920	Kaggle

ASL Alphabet Recognition Techniques

Technique	Description
Traditional Computer Vision	Extracts features from ASL alphabet images using handcrafted algorithms and manual feature engineering.
Convolutional Neural Networks	Deep learning models that automatically learn discriminative features from raw pixel data for accurate ASL alphabet recognition.
Transfer Learning	Utilizes pre-trained models to leverage knowledge from related tasks, improving ASL alphabet recognition performance.
Ensemble Methods	Combines multiple models to enhance ASL alphabet recognition by aggregating their predictions.
Hybrid Approaches	Integrates multiple techniques, such as traditional computer vision and deep learning, for improved ASL alphabet recognition accuracy.

CNN Architecture for Image Recognition

Architecture	Description	Link
LeNet-5	A classic CNN architecture with stacked convolutional and pooling layers, followed by fully connected layers, widely used for ASL alphabet recognition.	LeNet-5
AlexNet	Introduced in 2012, AlexNet is a deep CNN architecture with multiple convolutional and fully connected layers, known for its significant impact on computer vision.	AlexNet
VGGNet	VGGNet is a popular CNN architecture with multiple stacked convolutional layers and small receptive fields, achieving good performance in ASL alphabet recognition.	VGGNet
ResNet	ResNet utilizes residual connections to enable the construction of very deep CNNs, addressing the vanishing gradient problem and achieving high accuracy in ASL recognition.	ResNet
Inception	The Inception architecture, introduced by GoogLeNet, features multi-path convolutional layers capturing information at different scales, suitable for ASL alphabet recognition.	Inception (GoogLeNet)

GAP ANALYSIS

Current State

- The current state of ASL alphabet recognition methods is characterized by various limitations and challenges.
- Cannot replace human translators.
- These limitations hinder the accuracy, robustness, and overall performance of ASL alphabet recognition systems.
- Can not accuratly adapt to various sign lanuages and dialects and variability in usage.

Ideal State

- Accurately recognize a wide range of signing styles and variations
- Key attributes of the ideal state include high-quality and diverse datasets, comprehensive class representation.
- Able to recognize ASL signs in real-time, and be able to accurately translate signed language into written or spoken language with high accuracy.

Gaps in Current Approaches

1. **Limited recognition of signing styles and variations:** Current technology struggles to accurately recognize the full range of signing styles and variations, making it difficult to interpret signs from individuals with unique signing patterns or physical differences.
2. **Real-time sign recognition:** There is a gap in the ability to recognize signs in real-time, hindering applications that require immediate interpretation or response.
3. **Accurate translation of signed language:** Existing technology faces challenges in accurately translating signed language into written or spoken language due to the complex grammar and syntax of ASL.
4. **Limited understanding of ASL grammar and syntax:** The technology lacks a deep understanding of the grammar and syntax of ASL, limiting its ability to generate accurate translations.
5. **Variability in ASL gestures:** ASL incorporates a wide range of gestures, facial expressions, and body movements, which pose challenges for accurate recognition and interpretation.

PROBLEM STATEMENT

Current Challenges in ASL Alphabet Recognition:

1. **Accuracy Gap:** Insufficient accuracy hampers effective communication.
2. **Real-Time Performance:** Limited responsiveness restricts usability.
3. **Diversity in Signing Styles:** Difficulty in interpreting diverse signing styles.

Our Objectives:

1. **Enable seamless communication:** Improve accuracy, real-time performance, and interpretation of signing styles.
2. **Support effective interpretation:** Enhance grammar understanding for clear communication.
3. **Foster accessibility:** Advance gesture recognition for equal participation and understanding.

PROPOSED SOLUTION

To address the challenges in ASL alphabet recognition, we propose the following solution:

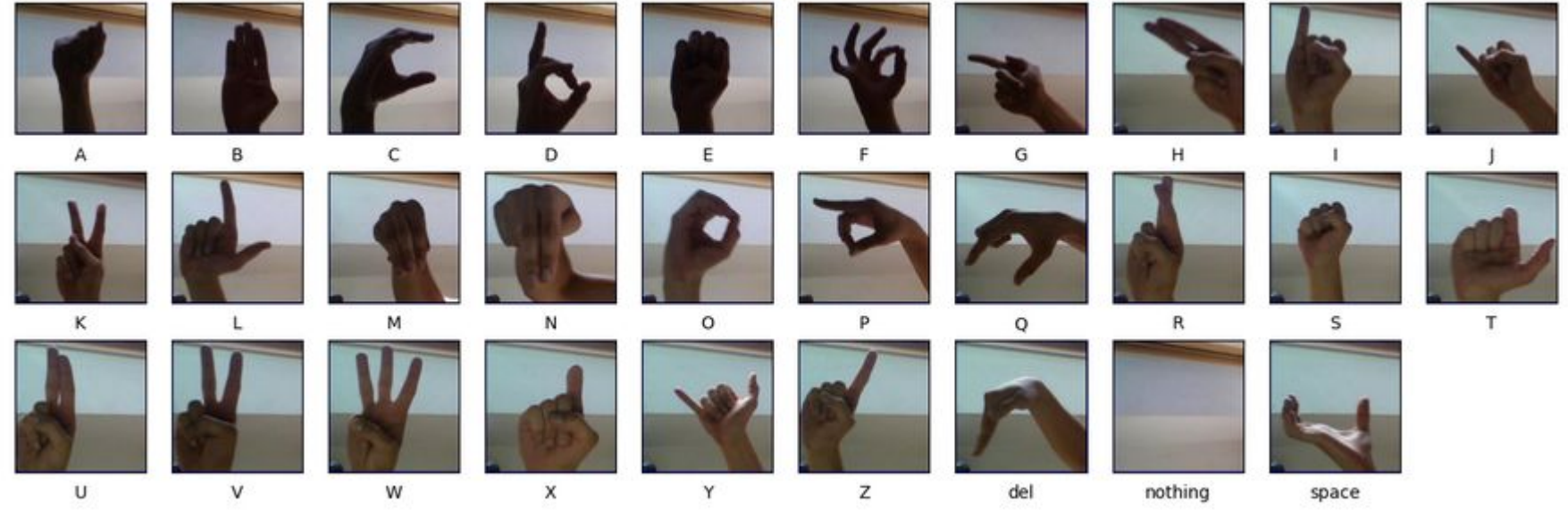
- **CNN Architecture:** Design a specialized CNN model for ASL alphabet recognition, leveraging its effectiveness in image recognition tasks.
- **Diverse Dataset:** Utilize a comprehensive dataset containing a wide range of ASL alphabet images, capturing various signing styles, hand positions, and orientations.
- **Preprocessing Techniques:** Apply image resizing, normalization, and augmentation to enhance dataset quality and consistency.
- **Hyperparameter Optimization:** Conduct extensive experiments to fine-tune hyperparameters, such as learning rate, batch size, and epochs, for optimal model performance.
- **Evaluation Metrics:** Define appropriate metrics (accuracy, precision, recall, F1 score) to evaluate and refine the model's performance.

By implementing these solutions, we aim to improve ASL alphabet recognition accuracy, accommodate diverse signing styles, and optimize the overall performance of the system.

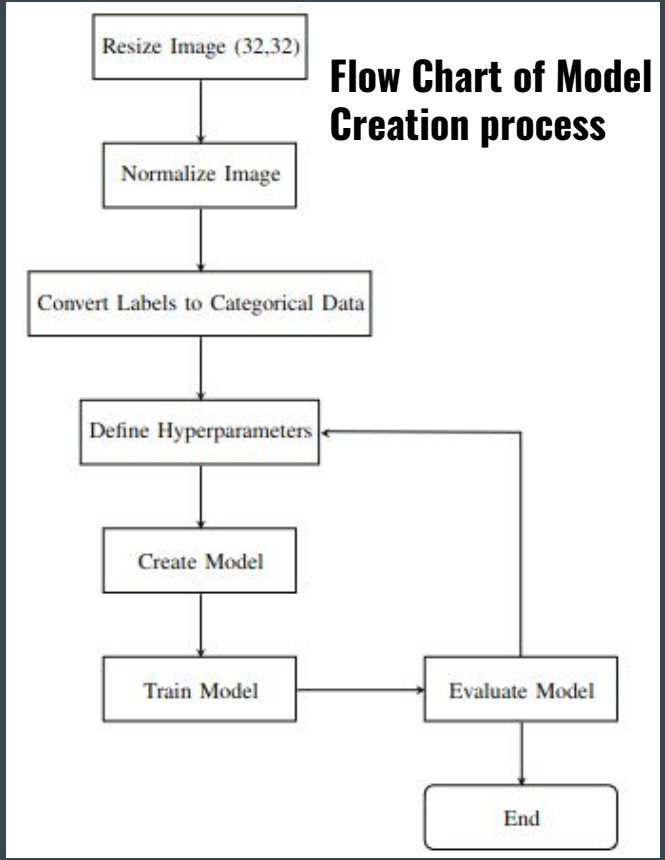
DATASET USED

The ASL alphabet recognition dataset has **87,000 images** at **200x200 pixels**, with **29 classes** including **A-Z.** There are three classes left: **SPACE, DELETE, and NOTHING**. They enable deleting a letter and adding a space, useful for real-time applications. Model can use "NOTHING" to detect no gestures being performed.

Dataset is from Kaggle : https://www.kaggle.com/datasets/grassknoted/asl-alphabet?resource=download



METHODOLOGY



Flow Chart of Model Creation process

Table of CNN configuration

Layer (type)	Output Shape	Param
Conv2D	(None, 32, 32, 64)	640
MaxPooling2D	(None, 16, 16, 64)	0
BatchNormalization	(None, 16, 16, 64)	256
Conv2D-1	(None, 16, 16, 128)	73856
MaxPooling2D-1	(None, 8, 8, 128)	0
BatchNormalization-1	(None, 8, 8, 128)	512
Dropout	(None, 8, 8, 128)	0
Conv2D-2	(None, 8, 8, 256)	295168
MaxPooling2D-2	(None, 4, 4, 256)	0
BatchNormalization-2	(None, 4, 4, 256)	1024
Flatten	(None, 4096)	0
Dropout-1	(None, 4096)	0
Dense	(None, 1024)	4195328
Dense-1	(None, 29)	29725

EXPERIMENTAL SETUP

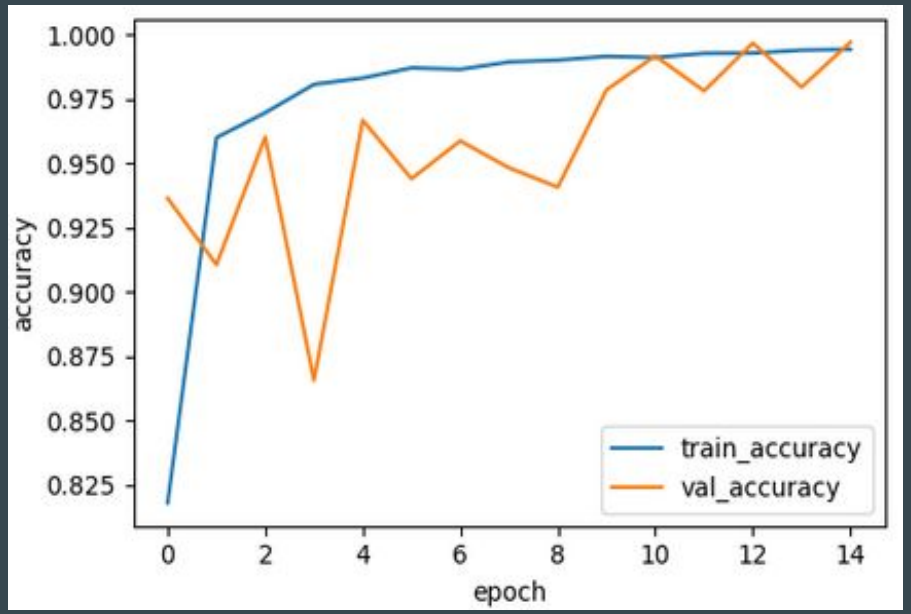
1. **Import Dataset:** Fetch (87000) images from directory and store as numpy array.Train/Test split by 90:10 making Train:Test size 78300:8700.
2. **Data Preprocessing:** Resize each image to 32x32 and normalize the dataset
3. **Model Architecture:** Create a model using the configuration provided in last slide. Activation function for all layers in ‘ReLU’ except for ‘softmax’ in last layer for classification.
4. **Training Process:** The model is trained using the ‘Adam optimizer’ with a ’learning rate of 0.001’. The training is performed over ‘15 epochs’ with a ‘mini-batch size of 32’.
5. **Hyperparameter Tuning:** The hyperparameters of the model, including the learning rate, number of epochs, and mini-batch size, are carefully tuned to achieve optimal performance.
6. **Evaluation Metrics:** Evaluate the model using ‘categorical_crossentropy’ and measure metrics accuracy and loss by epoch for both validation and test data. Plot the ‘confusion matrix’ of results to find problem inducing characters
7. **Document:** Make the Jupyter Notebook used is easy to read and understand. Also use version control through Git to make reversion easy.

RESULTS

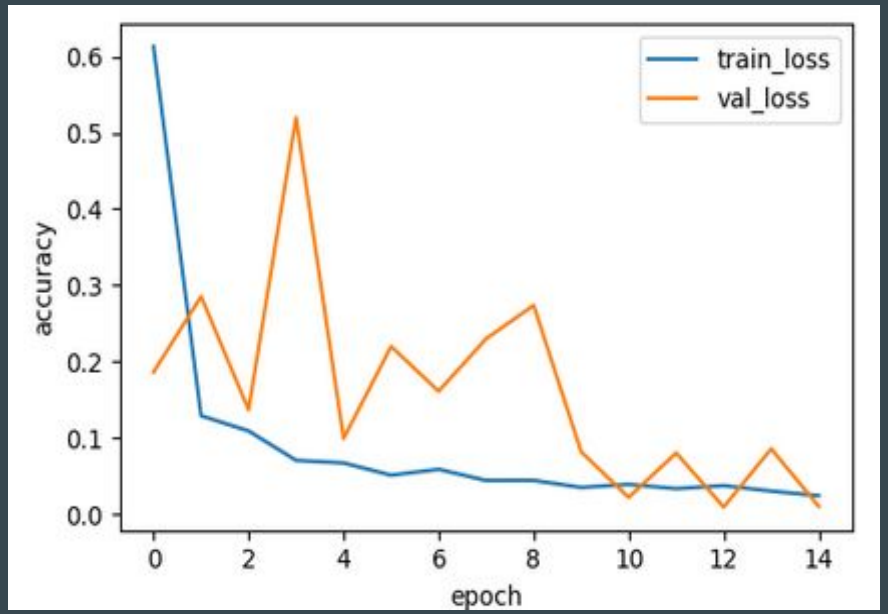
Test accuracy: 99.7%

Mean validation accuracy: 95.48%

Plot of Accuracy/Epoch



Plot of Loss/Epoch



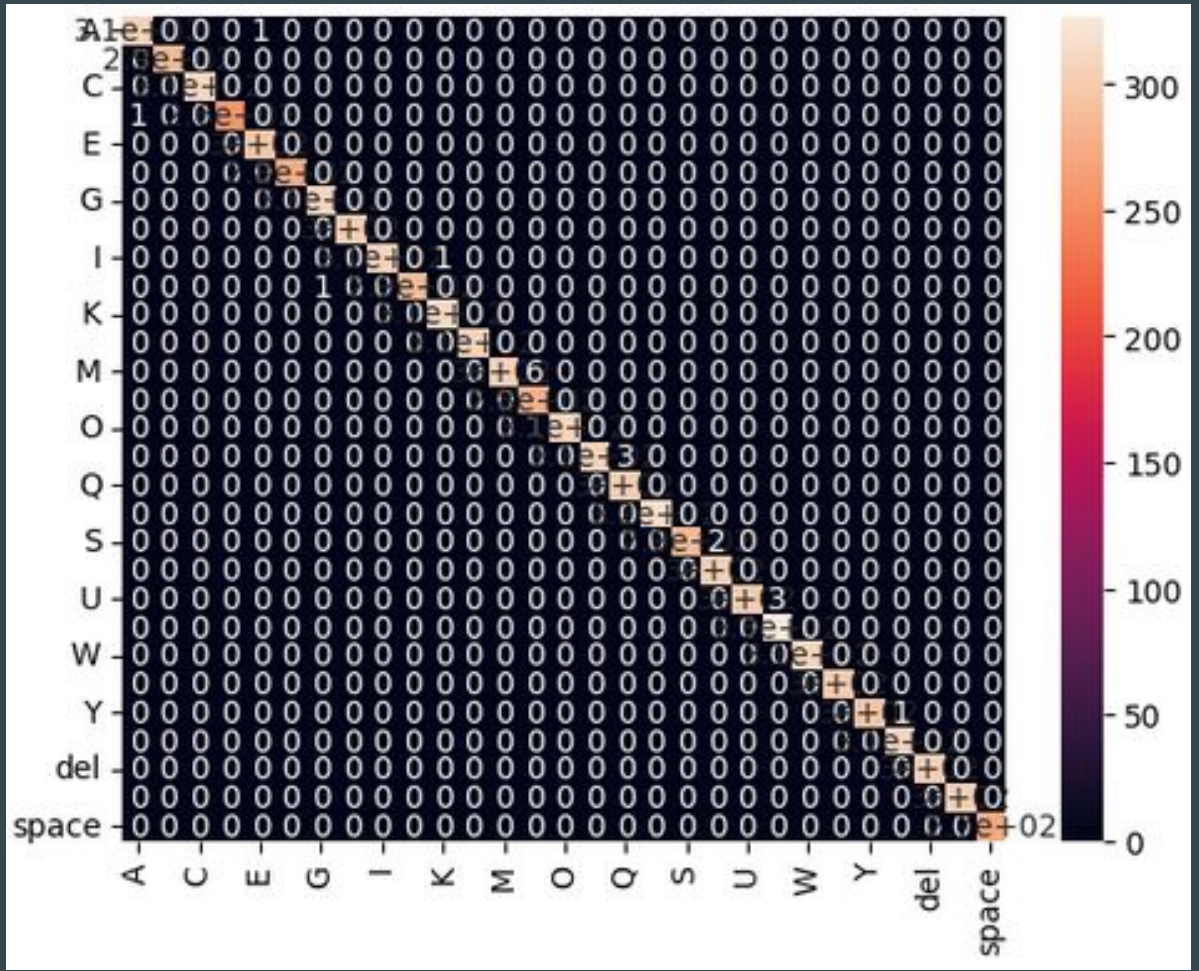
Confusion Matrix

Each row represents the true class, and each column represents the predicted class.

The values in the cells represent the counts of instances falling into each class.

The diagonal elements of the matrix (top-left to bottom-right) represent the correct predictions.

While the off-diagonal elements represent the errors made by the model.

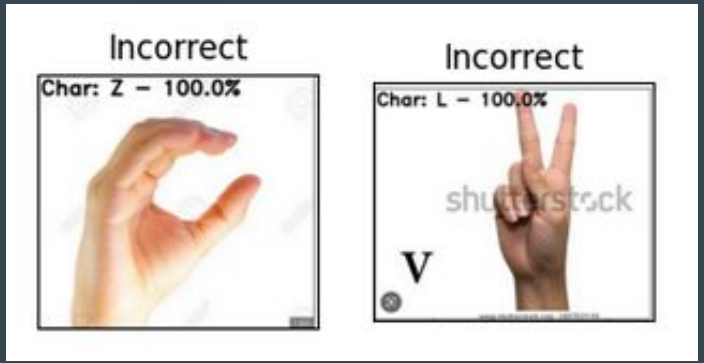


SIGNIFICANCE of RESULTS

The model exhibited a notable **mean validation accuracy of 95.48%** (15 epochs) as well as a high level of accuracy on the **testset 99.77%**, in the identification of American SignLanguage (ASL) alphabets.

These results strongly suggest the efficacy of deep learning architectures for ASL recognition tasks.

But there were some **problems in accurately identifying certain similar shaped letters in ASL in live data**, namely A and M, which suggests a necessity to enhance the model's architecture and dataset selection, or alternatively, to transition towards utilizing landmark data.



CONCLUSION

In conclusion, our study successfully implemented a CNN and TensorFlow model for ASL alphabet recognition, achieving high accuracy rates.

However, certain letters posed challenges, highlighting the need for refining the model architecture and dataset selection.

Moving forward, incorporating hand landmark data could enhance the model's precision and resilience by providing detailed hand positioning information.

Further improvements can be made by refining the model's architecture and expanding the dataset. We are just setting the foundation for advancements in ASL recognition technology, promoting accessibility and efficacy in real-world scenarios, such as education and communication for the deaf and hard of hearing.

A video walkthrough of project workflow and a live demo: (LINK)