Building a Convolutional Neural Network to Recognize Sign Language

Group 6: Ilsa Qadir, Krishma Kapoor, Jasmine Hanjra

Introduction

The purpose of this project is to build a convolutional neural network (CNN) that will accurately recognize sign language gestures. The system will convert the gestures into text, making communication between signers and non-signers efficient and accessible. This deep learning solution aims to bridge communication for people who are deaf or hard of hearing and foster inclusivity in various social settings.

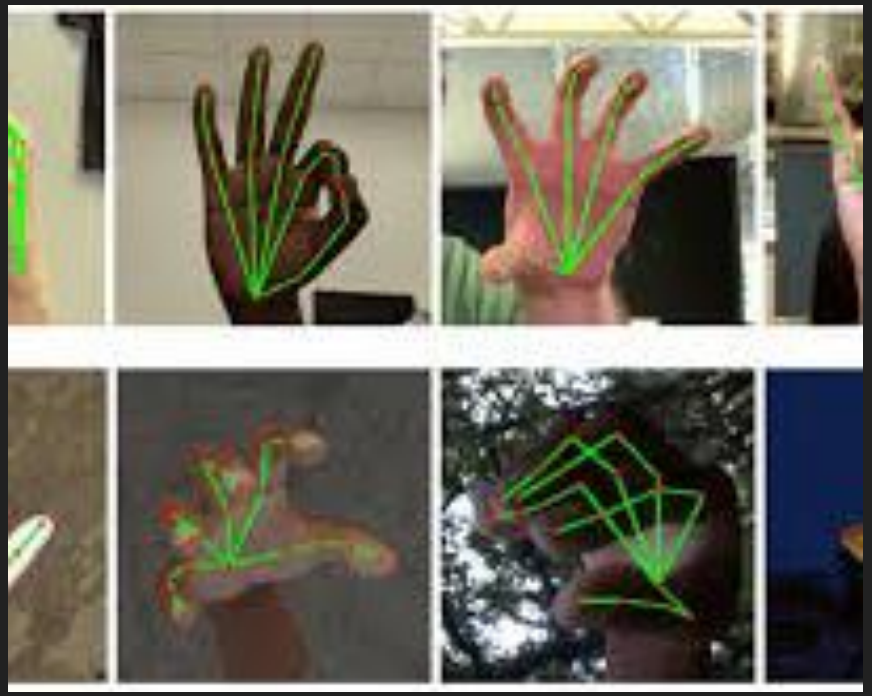
Motivation

The lack of accessible tools for real-time sign language translation inspired us to take on this project. While professional interpreters and manual translation apps exist, they can be costly, time-consuming, and require expertise.

Our system allows **anyone** to input ASL gestures in image format and receive accurate text-based translations, creating a bridge between signers and non-signers.

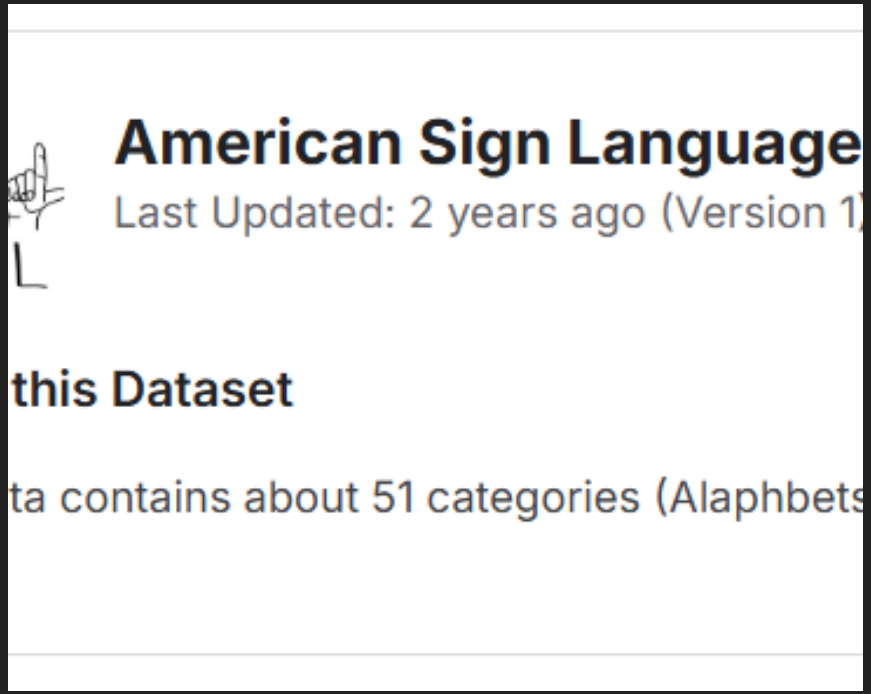
The Goal

- The goal of this project is to:
  - Modify an existing CNN model so that it can classify more difficult hand gestures from images.
  - Maintain high accuracy in recognizing these hand gestures.
  - Improve the user experience to easily input ASL images

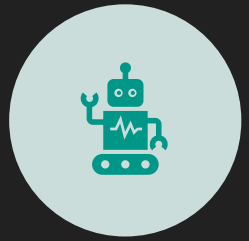


Data preparation

- We started with a dataset containing 24,000 images, limited to the ASL alphabet (A-Z).
- While this provided a foundation, it lacked versatility and real-world applicability, such as recognizing common words or numbers.



Expanded dataset



To improve the model’s utility, we added images of numbers, frequently used words like "Help" and "Friend," and additional gestures.

The updated dataset included **203,000 images**, making it robust and comprehensive for real-world applications.

Expanded dataset continued

- To improve the model’s utility, we added images of numbers, frequently used words like "Help" and "Friend," and additional gestures.
- The updated dataset included 203,000 images, making it robust and comprehensive for real-world applications.
- Class Categories: 51 distinct classes, covering:
- Letters (A-Z)
- Numbers (0-9)
- Words such as "Baby," "Stop," and "Nothing."

Method

Our Method consisted of:

- **Using a pre-existing model on a new dataset.**
- **Model Training:** Training the CNN based on this new data to accurately determine the hand shapes and gestures presented from the images.
- **Model Testing:** The model was tested on performance and quality.

We used:

- Programming Language: Python
- Deep Learning/Machine Learning Libraries: PyTorch, TensorFlow, Keras, Sklearn
- Development Environment: Kaggle

Model Training

WE BUILT UPON AN EXISTING CNN ARCHITECTURE BY MODIFYING KEY COMPONENTS TO IMPROVE PERFORMANCE:**DROPOUT ADJUSTMENTS:** ADJUSTED DROPOUT RATES TO BALANCE OVERFITTING AND GENERALIZATION.

LAYER ADJUSTMENTS: INCREASED FILTERS IN THE SECOND CONVOLUTIONAL LAYER TO ENHANCE THE MODEL’S ABILITY TO RECOGNIZE COMPLEX PATTERNS.

EARLY STOPPING: IMPLEMENTED A PATIENCE PARAMETER OF 5 EPOCHS TO PREVENT UNNECESSARY TRAINING CYCLES.

Library and Tools

- We used Python as the primary programming language, along with libraries such as TensorFlow, Keras, and PyTorch.
- Development was conducted on Kaggle, providing access to GPUs and streamlined integration of datasets.

Details

Original Dataset:

Only includes the ASL alphabet

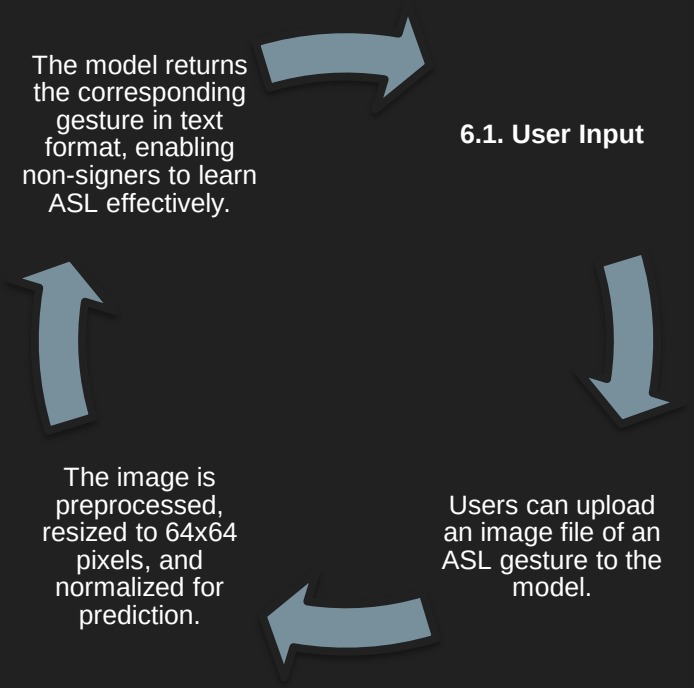
24,000 images

New Dataset:

Includes, numbers, words, and the ASL alphabet

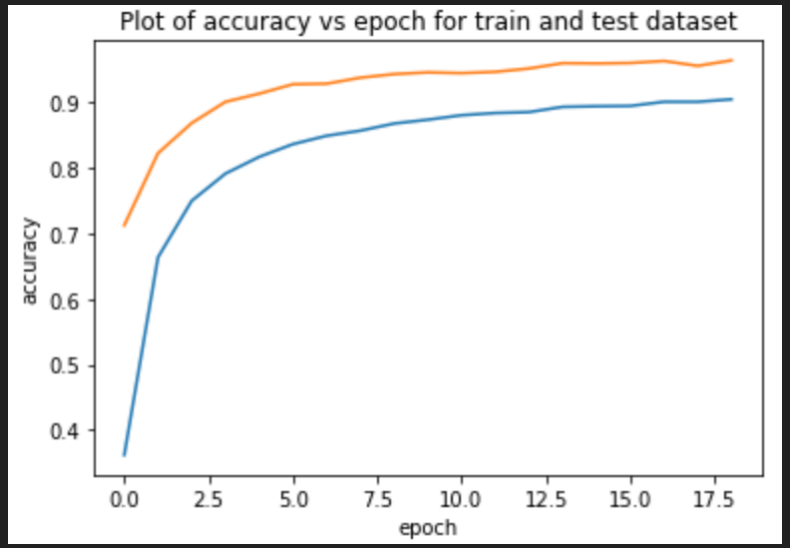
203,000 images

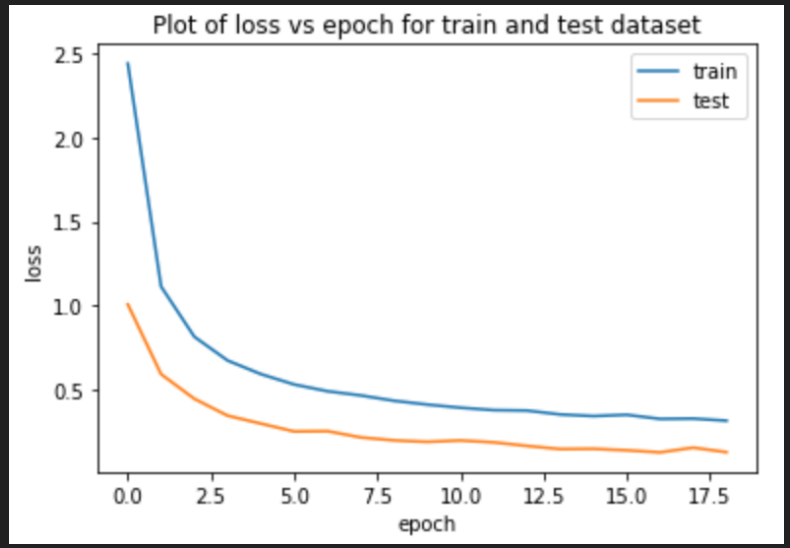
Features and Functionality



Before Modification

Accuracy Before: 96.4%





What we modified:

Dropout

Decreased the Dropout value for the second layer to 0.2 instead of 0.25 to reduce overfitting and to better generalize the model.

Increased the Dropout values for the third layer and the dense layer to 0.3. Since dense layers have more parameters, higher dropout rates are more effective.

Layer Adjustments:

Changed the second convolutional layer from 64 to 128 to increase the number of filters. This would help the model learn more complex patterns.

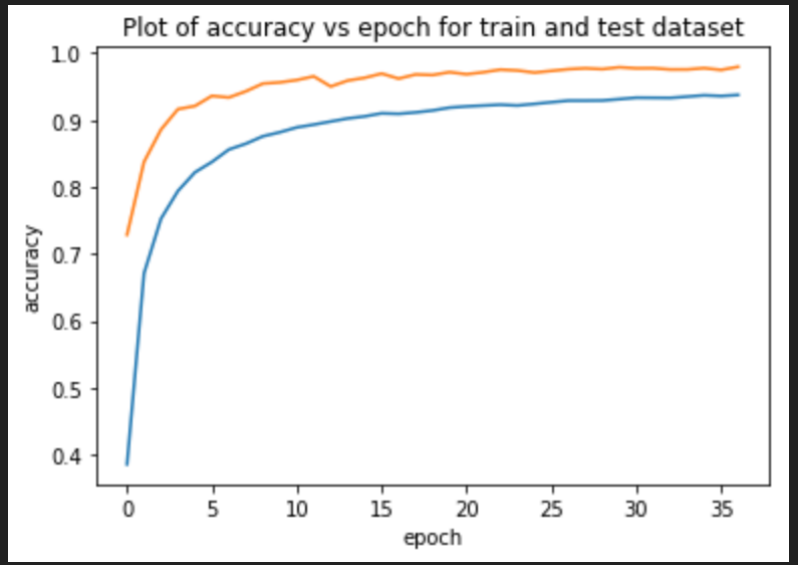
Conv2D(64, (3, 3)) to Conv2D(128, (3, 3))

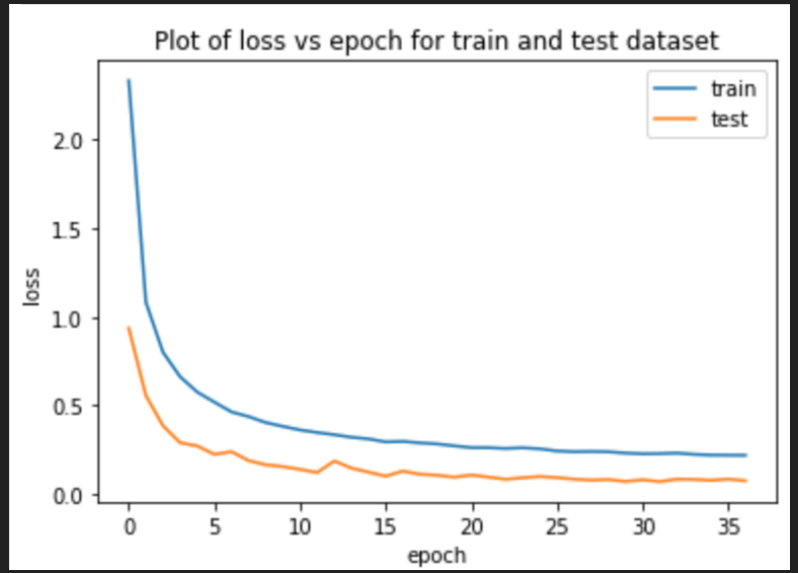
Early Stopping

Changed the patience value to 5 so that it would halt training if there were no improvements after 5 epochs instead of 2.

Results: After Modification

Accuracy after: 97.8%





Results

Before Modification: The original model achieved an accuracy of 96.4%.

After Modification: Post-tuning, the model's accuracy increased to **97.8%** on the expanded dataset.

This improvement demonstrates the effectiveness of our adjustments and highlights the model’s ability to generalize across diverse gestures.

7.2. Performance Metrics

We evaluated the model using metrics like precision, recall, and F1-score to ensure balanced performance across all classes.

The confusion matrix showed that most misclassifications occurred with gestures having similar hand shapes, such as "O" and "0."

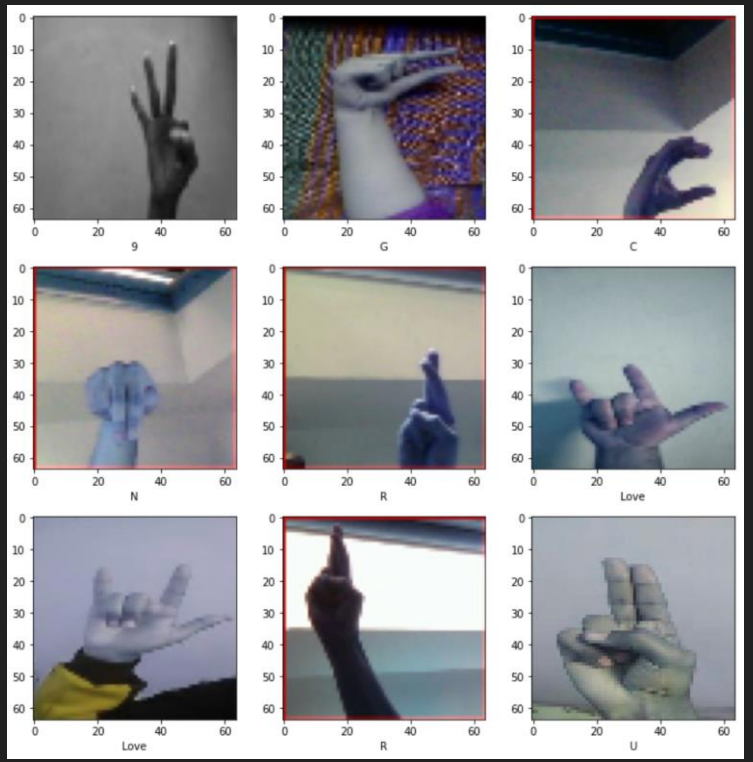
Training the Model

```
Epoch 30/50
1111/1111 - 10s - loss: 0.2327 - accuracy: 0.9312 - val_loss: 0.0731 - val_accuracy: 0.9788
Epoch 31/50
1111/1111 - 10s - loss: 0.2285 - accuracy: 0.9333 - val_loss: 0.0820 - val_accuracy: 0.9772
Epoch 32/50
1111/1111 - 10s - loss: 0.2296 - accuracy: 0.9332 - val_loss: 0.0721 - val_accuracy: 0.9774
Epoch 33/50
1111/1111 - 10s - loss: 0.2326 - accuracy: 0.9330 - val_loss: 0.0855 - val_accuracy: 0.9755
Epoch 34/50
1111/1111 - 10s - loss: 0.2254 - accuracy: 0.9349 - val_loss: 0.0842 - val_accuracy: 0.9755
Epoch 35/50
1111/1111 - 10s - loss: 0.2216 - accuracy: 0.9369 - val_loss: 0.0794 - val_accuracy: 0.9775
Epoch 36/50
1111/1111 - 10s - loss: 0.2209 - accuracy: 0.9358 - val_loss: 0.0858 - val_accuracy: 0.9748
Epoch 37/50
1111/1111 - 10s - loss: 0.2202 - accuracy: 0.9374 - val_loss: 0.0768 - val_accuracy: 0.9794
```

Calculating Accuracy

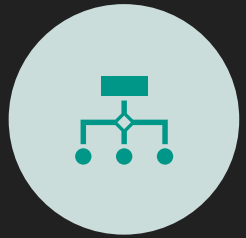
```
952/952 [==============================] - 2s 3ms/step - loss: 0.0771 - accuracy: 0.9785
Test results - Loss: 0.07713638991117477 - Accuracy: 97.8489339351654%
```

Images From Dataset



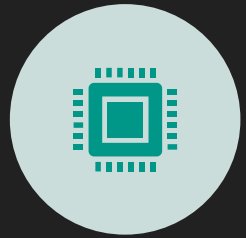
User Input

TO IMPROVE USER EXPERIENCE, WE ADDED A NEW FEATURE THAT WOULD ALLOW A USER TO INPUT THE FILE PATH OF AN IMAGE.



THE MODEL WOULD THEN RETURN THE LETTER, NUMBER, OR WORD THAT CORRESPONDS WITH THE IMAGE.

THIS ENABLES THE USER TO LEARN ASL EFFECTIVELY.



BY INPUTTING ANY IMAGE, A USER CAN DETERMINE WHAT AN ASL SYMBOL MEANS.

Challenges

Overfitting: Initially, the model overfitted on the training data due to the high complexity of dense layers. Dropout adjustments and regularization mitigated this issue.

Dataset Imbalance: Certain gestures, such as "Z," had fewer samples, which we addressed by data augmentation techniques like rotation and flipping.

Lighting Variability: Differences in lighting and hand positioning affected model accuracy. We tackled this by normalizing the dataset and adding diverse samples.

What have we learned from this?

- How CNN actually works.
- How to train an existing CNN with a new dataset.
- How changing even the smallest things can have an affect on the overall accuracy and efficiency.
- Not all models will work with maximum accuracy on any dataset.
- What tools can be used to modify a model to work with a new dataset.

Future Directions

- 10.1. Dynamic Gesture Recognition
- Extend the system to recognize sequential gestures, enabling sentence-level translations.
- Incorporate temporal models like Recurrent Neural Networks (RNNs) or Transformers for video-based input.
- 10.2. Real-Time Application
- Develop lightweight versions of the model for integration into mobile devices or augmented reality glasses.
- Potential applications include:
  - Live ASL interpretation during meetings or classes.
  - Educational tools for teaching ASL to children and adults.
- 10.3. Collaboration with Communities
- Partner with organizations serving the deaf community to ensure the model addresses real-world needs.
- Expand the dataset to include gestures from other sign languages, such as British Sign Language (BSL).

Broader Impact

Inclusivity: This project promotes equal opportunities for communication, allowing signers to participate more fully in society.

Education: By making ASL more accessible, we can encourage more people to learn sign language and connect with the deaf community.

Technology for Good: This work exemplifies how deep learning can solve societal challenges and foster understanding between diverse groups.

Conclusion

- Our project showcases the potential of deep learning to transform lives by bridging communication gaps.
- By addressing both technical challenges and user accessibility, we've created a system that not only achieves high accuracy but also promotes inclusivity.
- This is just the beginning. With further enhancements, we hope to make sign language translation seamless and universal.

Sources

01

Modified Model:

- https://www.kaggle.com/code/ilsaqadir/deep-learning-project-asl-to-text-cnn

02

Original Model:

- https://www.kaggle.com/code/raanaramadan/asl-model

03

Dataset:

- https://www.kaggle.com/datasets/alhasangamalmahmoud/american-sign-language-asl