DSO 569 : Deep Learning For Business Applications

American Sign Language Classification using Transfer Learning : VGG16, ResNet50, InceptionV3

Akshat Rai
Report

Avantika Gargya
Report

Jaclyn Martin
Report

Rishabh Pratap Singh Sisodia
Python Code

Sam Sithimolada
Python Code

I. INTRODUCTION

Sign language is an essential mode of communication for millions worldwide, providing a linguistic lifeline for the Deaf and Hard of Hearing community. The integration of American Sign Language (ASL) interpretation into technological solutions offers significant potential to enhance communication accessibility and inclusivity. In this project, we tackle the challenge of ASL sign classification using advanced deep learning techniques, focusing on the power of transfer learning to bridge the communication gap effectively. Our primary goal is to develop a robust and efficient system capable of accurately recognizing ASL gestures from images.

Utilizing a rich dataset sourced from Kaggle, which includes a diverse array of hand gesture images representing various ASL signs, we explore the adaptability of pre-trained models—specifically Inception V3, VGG16, and ResNet—toward ASL sign recognition. By leveraging the intrinsic capabilities of these models and conducting hyperparameter tuning, we examine different architectural configurations and optimizations to enhance our system's performance. This approach not only aims to determine the most effective model architecture for ASL recognition but also to shed light on the impacts of model selection and hyperparameter adjustments on classification accuracy.

Through comprehensive experimentation and meticulous evaluation, this project seeks to underscore the efficacy of deep learning techniques in advancing ASL sign recognition, offering insights that could propel further innovations in accessible communication technologies.

II. LITERATURE REVIEW

American Sign Language (ASL) is a visual language that incorporates hand movements, facial expressions, and body postures to communicate. Recognizing ASL poses unique challenges due to the complexity and variability in gestures and expressions. Each sign involves intricate hand shapes, motions, and orientations, often accompanied by facial expressions, making automated recognition a demanding task. Efficient ASL recognition systems can facilitate better communication, educational resources, and accessibility for the Deaf and Hard of Hearing (DHH) community.

A. Historical Approaches to ASL Recognition

Early approaches to ASL recognition relied heavily on handcrafted features and rule-based methodologies. In the 1990s and early 2000s, researchers used techniques like edge detection and gesture segmentation to capture essential hand features. Statistical models like Hidden Markov Models (HMMs) were then used to identify the sequential patterns characteristic of signs. Another approach was to use sensor-based gloves to capture hand movements directly. While effective for controlled environments, these systems were impractical for broader applications due to high costs and discomfort for users. Computer vision techniques like Principal Component Analysis (PCA) and Support Vector Machines (SVMs) showed promise in recognizing isolated signs but struggled with complex, dynamic gestures.

B. Transition to Deep Learning Models

The adoption of deep learning techniques transformed ASL recognition. Convolutional Neural Networks (CNNs) gained popularity for their ability to capture spatial patterns in hand shapes and orientations, which are crucial in identifying static signs. Recurrent Neural Networks (RNNs), especially Long Short-Term Memory Networks (LSTMs), were used to recognize sequential patterns, enabling the identification of dynamic signs. Hybrid models combining CNNs and RNNs/LSTMs enhanced recognition further by leveraging spatial and temporal patterns, while transfer learning and data augmentation helped improve the models' robustness. These deep learning approaches made ASL recognition more accurate and adaptable to variations, providing a foundation for real-world applications.

C. Key papers

In developing our static ASL recognition model, we sought to incorporate insights from foundational research to design and optimize an effective architecture. One influential study by Pigou and colleagues explored using Convolutional Neural Networks (CNNs) to classify isolated signs. They emphasized that CNNs excel at capturing spatial features, such as hand shapes and orientations, which are crucial for distinguishing static ASL signs. Drawing from their approach, we incorporated multiple CNN architectures like VGG16 and ResNet50 to effectively handle the spatial features of ASL images (Pigou et al., 2015).

Koller et al. (2016) investigated sign language recognition using hybrid statistical models for continuous sequences. Their work demonstrated that combining CNNs

with statistical models could improve the recognition of varied input data across signers. Although our approach is focused on static sign recognition, we applied their recommendations by using multiple pretrained models to compare performance across InceptionV3, VGG16, and ResNet50.

Camgoz et al. (2017) proposed specialized sub-networks, or SubUNets, for recognizing static hand shapes, showcasing the importance of specialized feature extraction for accurate classification. While they primarily focused on continuous recognition, we adapted their feature extraction techniques through pre-trained models to detect isolated hand gestures effectively.

By integrating key findings from these foundational studies into our static ASL recognition model, we were able to create a robust architecture that leverages the spatial feature extraction capabilities of multiple pre-trained CNN models, ensuring accurate and efficient classification of isolated signs.

Despite significant advancements in ASL recognition, current models still face challenges. First, while many models achieve high accuracy with static signs, they often struggle with generalization across diverse signers and conditions. Second, hybrid models combining CNNs with sequence models like RNNs or LSTMs are underexplored for static recognition, especially for signs with nuanced spatial patterns. Additionally, many studies use limited datasets, which reduces a model's robustness and real-world applicability. By addressing these gaps through the development of models that handle varied signers and incorporate hybrid architectures with diverse training data, future research can achieve more generalized and practical ASL recognition systems.

III. METHODOLOGY

Our methodology for developing a static asl recognition model involves detailed data preparation, the selection of appropriate model architectures through transfer learning, rigorous training processes, and comprehensive evaluations to ensure accuracy and robustness. Below, we outline each key phase of our methodology.

A. Dataset Description

Our dataset comprises static images representing various ASL signs. It includes a wide range of hand signs for the American Sign Language alphabet (A-Z) and numerical digits (0-9), totaling 36 classes. There are a total of 2515 image files. Each image features a single hand gesture against a uniform background, aimed at simplifying the task of feature extraction by focusing on the shape and orientation of the hand.

B. Data Preparation, Visualization and Organization

The dataset, comprising images representing various ASL signs, was initially stored in a zip file. We utilized Python's zipfile library to extract these images into a structured directory format suitable for further processing. We visualized the dataset using matplotlib and tensorflow libraries to understand its characteristics, such as sign variety and distribution. (Fig 1 describes the distribution of 36 classes).

We converted this data into a pandas dataframe containing the file paths, labels, and images as numpy arrays. We analyzed the class distribution to detect any imbalance, which is vital for determining the necessity for data augmentation.

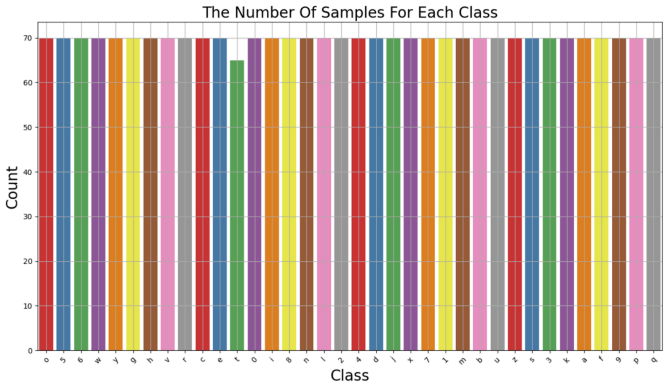


Fig 1: Class distribution in the ASL Dataset

C. Data Augmentation

To enhance generalization and address potential class imbalances, we implemented data augmentation using TensorFlow's ImageDataGenerator. The parameters set in the generator included:

- rescale=1./255 to normalize pixel values,
- rotation_range=40 for random rotations up to 40 degrees,
- width_shift_range=0.2 and height_shift_range=0.2 for random horizontal and vertical shifts respectively,
- shear_range=0.2 for shearing transformations,
- zoom_range=0.2 for random zooming,
- horizontal_flip=True for horizontal flips,
- fill_mode='nearest' to fill in newly created pixels after a transformation.

D. Training, Test and Validation Split

We divided the dataset into training (70%) and Testing (30%). This was further divided into: validation (15%), and testing (15%) sets to ensure thorough training and unbiased evaluation.

E. Model Development

We utilized transfer learning to leverage pre-trained models known for their high efficiency and accuracy in image recognition tasks. We explored three models for this project - VGG16, ResNet50 and InceptionV3.

After removing the top layers of each pre-trained model, custom layers tailored for ASL recognition were added. These included: a batch Normalization layer for faster and stable training, a dense layer using 'relu' activation, a dropout layer to prevent overfitting, a final dense output layer using 'softmax' activation. (See Table 1)

The architectural configurations and model parameters were tuned as below -

- Activation L1 = Relu
- Activation L2 = Softmax
- Loss = Categorical Cross Entropy

- Metric = Accuracy
- Drop out = 0.2
- Learning Rate = 0.01
- Batch Size = 16
- Epoch = 50
- Patience = 10

The model is compiled using the Adam optimizer and categorical cross-entropy loss function. EarlyStopping was set with a patience of 10 epochs, and ModelCheckpoint was used to save the best model based on validation loss. The model is trained on the augmented training data with validation performed on the validation set, with a maximum of 50 epochs due to computational constraints. Table 1 below describes the model architecture -

Model	Layer 1	Dropout	Layer 2
VGG 16	Dense(1024, activation= 'relu')	Dropout= 0.2	Dense(36, activation= 'softmax')
ResNet 50	Dense(1024, activation= 'relu')	Dropout= 0.2	Dense(36, activation= 'softmax')
Inception V3	Dense(1024, activation= 'relu')	Dropout= 0.2	Dense(36, activation= 'softmax')

Table 1 : Architecture of models under evaluation

F. Model Evaluation and Performance Analysis

Training history is visualized through plots of training vs. validation accuracy and loss to assess model convergence and overfitting. The best-performing models from InceptionV3, VGG16, and ResNet are evaluated on both the training and validation sets to measure accuracy and loss. Model predictions are generated on the test set, and classification metrics such as accuracy, loss, and a confusion matrix are computed to evaluate performance and identify misclassifications.

This comprehensive methodology ensures our ASL recognition model is not only accurate in recognizing static signs but also robust against variations and capable of generalizing well across diverse representations of ASL signs.

IV. INSIGHTS

After determining the best parameters for the models using Early Stopping, for each of the transfer learning models : VGG16, ResNet50 and InceptionV3 , we compared their performance relative to each other.

We plotted the training and validation errors and loss against epochs :

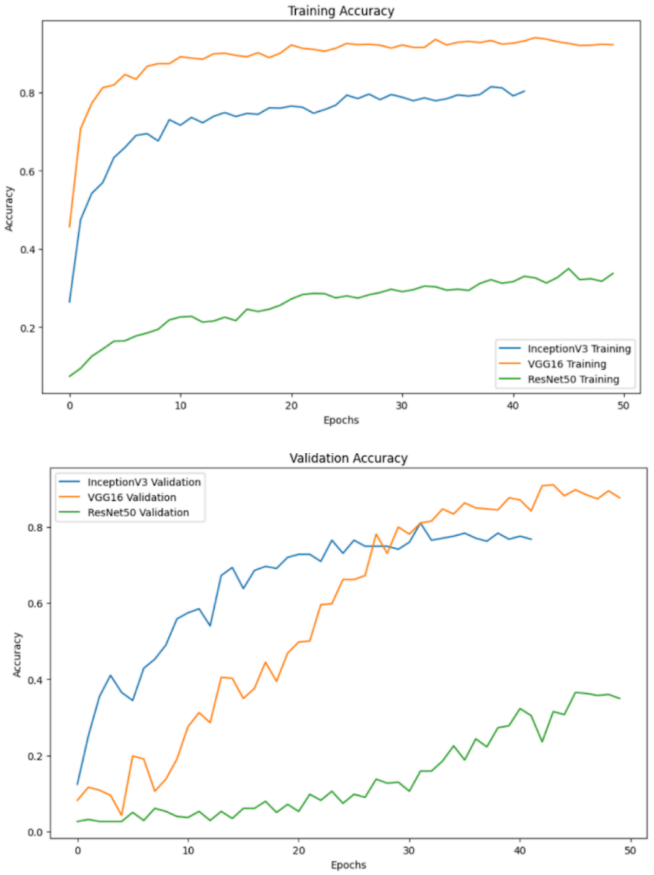
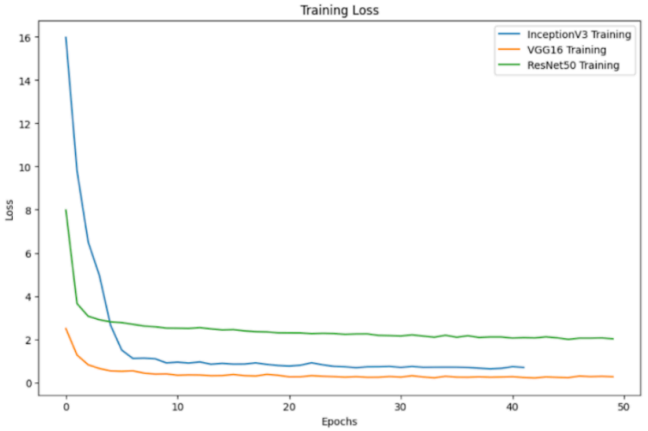


Fig 2 : Training and Validation Accuracy comparison across our three models

There are distinct differences between the models in terms of their performance on this dataset. Evidently, VGG16 shows the highest training accuracy, followed by InceptionV3 and Resnet50 which performs much poorly in comparison to the other two. Validation curves indicate that up to roughly 30 epochs, InceptionV3 outperforms the VGG16 after which VGG16 delivers a better accuracy on validation data. ResNet50, again.

Similar trends are enforced by (both training and validation) loss curves -



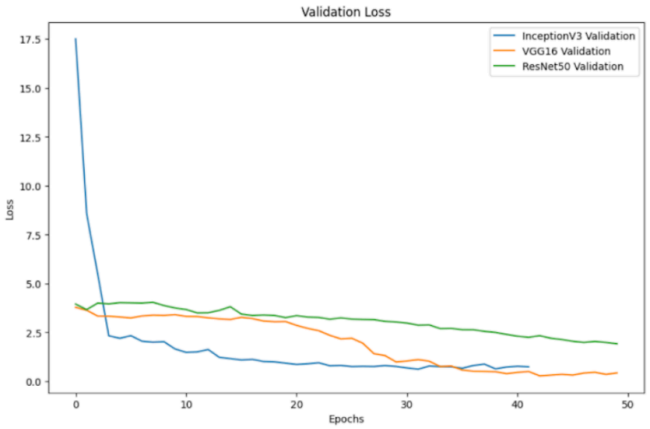


Fig 3 : Training and Validation Loss comparison across our three models

The VGG16 overall trains the best, providing the lowest training error throughout. InceptionV3 beats ResNet50 early on, at around the 5-6 epoch mark, making it second best for the remaining process. When it comes to validation loss, VGG16 outshines InceptionV3 only after ~35 epochs, which is similar to the trend observed in the accuracy plots. ResNet has a higher loss throughout.

Next, we calculated the accuracy and loss for all the models over our testing dataset :

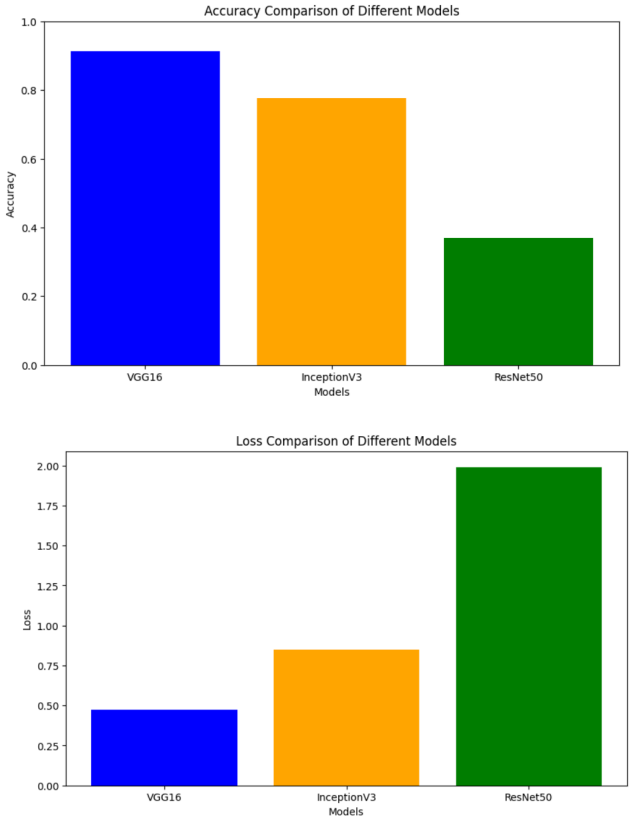


Fig 4 : Testing accuracy and loss across the models

The Table below (Table 2) summarizes the classification output and accuracy for our models. We used these to finally determine our best performing model.

Model	Correctly Identified Data	Misclassified data	Accuracy score
VGG 16	328	49	0.87
ResNet 50	130	247	0.34
Inception V3	296	81	0.79

Table 2 : Model performance on ASL dataset

Visibly, we achieve the best accuracy through the VGG16 transfer learning model

V. RESULTS

Based on the comparison on accuracy and loss metrics above, final model selected - VGG16 transfer learning model.

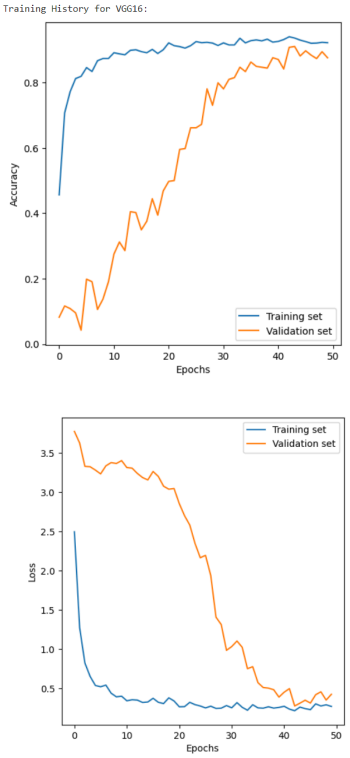


Fig 5:Training and Validation accuracy and loss for VGG 16

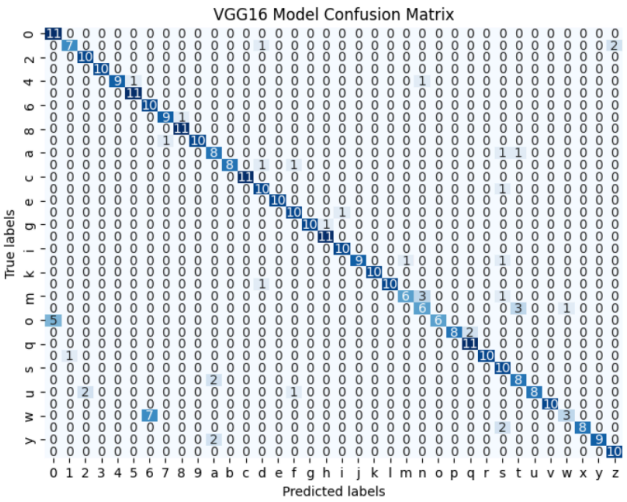


Fig 6 : Confusion Matrix for the VGG16 Model

The confusion matrix (Fig 6) illustrates that 328 images were correctly classified and 49 test images were misclassified by our model, providing an accuracy of 87%.

If we look at this metric in comparison,

Training Accuracy	Validation Accuracy	Testing Accuracy
0.936	0.910	0.870

Table 3 : Accuracy for the final model (VGG16+)

Here is a sample output when we run this model -

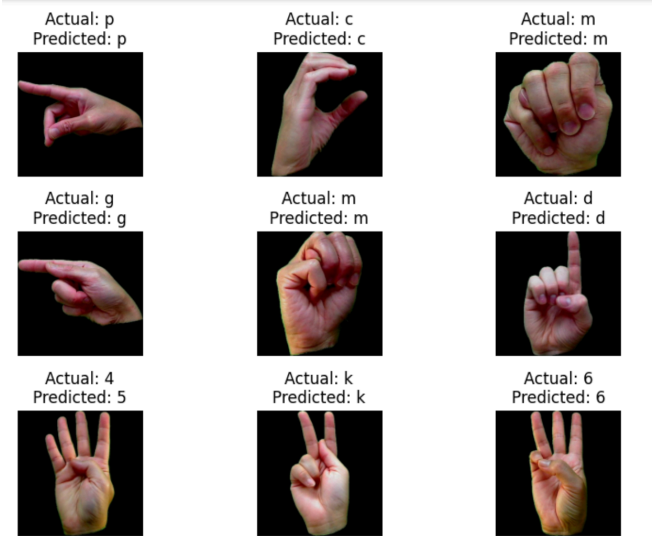


Fig 6 : Actual classes compared with final predicted classes from the VGG16 model

VI. CONCLUSION

The outlined experiment provides a structured approach to developing a deep learning model for ASL recognition. It combines rigorous data preparation, systematic model training, and comprehensive evaluation to create a tool that performs well statistically and holds significant practical value in real-world applications. This approach aims to harness the power of AI to bridge communication gaps, highlighting the transformative potential of deep learning in enhancing accessibility and inclusion.

The project involved building an American Sign Language (ASL) recognition model using a dataset comprising 2515 static images of ASL signs. Data preparation involved extracting and organizing images, followed by visualization and analysis of class distribution. Data augmentation was implemented to address class imbalances and enhance generalization. Transfer learning with VGG16, ResNet50, and InceptionV3 was employed, with custom layers added for ASL recognition. Model parameters were optimized, and training was performed with early stopping and model checkpointing. Evaluation included training history analysis, performance metrics, and model predictions on the test set. VGG16 emerged as the best-performing model, showcasing robustness and accuracy in recognizing diverse ASL signs.

We were able to enhance the existing architecture on this subject by adding our model onto the other transfer learning models (VGG16, ResNet50, InceptionV3) , which delivered an accuracy of 87% on unseen data.

REFERENCES

[1] M. Pigou et al. (2015), "Sign Language Recognition Using Convolutional Neural Networks"

[2] Koller, O., Zargaran, S., Ney, H. (2016). Continuous Sign Language Recognition: Towards Large Vocabulary Statistical Recognition Systems Handling Multiple Signers. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[3] Camgoz, N. C., Hadfield, S., Koller, O., Bowden, R. (2017). SubUNets: End-to-End Hand Shape and Continuous Sign Language Recognition. Proceedings of the IEEE International Conference on Computer Vision (ICCV)

[4] C. Huang et al. (2015), "Sign Language Recognition Using Deep Neural Networks"

[5] A. Molchanov et al. (2016), "Hand Gesture Recognition with 3D Convolutional Neural Networks"

[6] P. Kumar et al. (2019), "Vision-Based Static Hand Gesture Recognition Using Convolutional Neural Networks"

[7] https://www.kaggle.com/datasets/ayuraj/asl-dataset

[8] https://www.kaggle.com/code/themlphdstudent/sign-classification-using-tl-inceptionv3