EE 541 Computational Introduction to Deep Learning

Project Report

Abid Hassan, Benjamin Fein-Ashley

1 Introduction

In this work, we utilize several deep learning algorithms to classify American Sign Language (ASL) alphabet letters from images, as well as real-time classification using live video. The dataset used for this project is the ASL Alphabet dataset [1], which contains images of hand signs representing each letter of the ASL alphabet. The goal is to build a model that can accurately classify these hand signs into their corresponding letters. We examine ResNet18 and DenseNet121 architectures, and compare their performance in terms of accuracy on a test dataset provided separately from the training dataset. The training data set contains 87,000 images which are 200x200 pixels. There are 29 classes, of which 26 are for the letters A-Z and 3 classes for space, delete, and nothing, which are intended to improve performance in real time classification.

1.1 Literature Review

A variety of machine learning and deep learning methods have been applied to sign language recognition with varying success. Principal Component Analysis (PCA) combined with SVM classification achieved around 94% accuracy, demonstrating a baseline for evaluating more complex deep learning architectures [2].

A study utilizing multiple deep learning models, including ResNet-50 and EfficientNet, showed significant success using transfer learning techniques, achieving accuracies up to 99.98% with ResNet-50 [2].

MobileNetV2 was employed as a lightweight architecture, achieving 98.77% accuracy, suitable for real-time classification on resource-constrained devices [3].

A DeepCNN model combined with extensive data augmentation techniques was proposed, achieving 99.67% accuracy [4].

A novel hardware-based approach used neuromorphic sensors coupled with an artificial neural network implemented on an FPGA, achieving 79.58% accuracy [5]. Although the accuracy of this method is lower than others, it represents a more practical system for real time application.

Table 1: Accuracy Results from the Literature Review

Model	Accuracy (%)
ResNet-50 (Transfer Learning) [2]	99.98
MobileNetV2 [3]	98.77
DeepCNN [4]	99.67
Neuromorphic Approach [5]	79.58
PCA + SVM [2]	94.00

Although computationally expensive, ResNet models are able to achieve near perfect classification on the ASL test dataset. Several lightweight and embedded architectures also achieve high accuracy. Non deep-learning methods such as PCA + SVM have lower accuracy.

2 Methods

2.1 Data Preprocessing

The dataset is preprocessed by resizing the images to a uniform size of 200x200 pixels and normalizing the pixel values to be between 0 and 1. Data augmentation techniques such as rotation, flipping, and zooming are applied to increase the diversity of the training data and improve the model's generalization ability.

2.2 Model Architecture

We implement two deep learning architectures, ResNet18, and DenseNet121. These architectures are chosen for their proven performance in image classification tasks. The models are trained using the Adam optimizer with a learning rate of 0.01 and a batch size of 64. The loss function used is categorical cross-entropy. ResNet (Residual Network) employs skip (or residual) connections that allow the network to bypass one or more layers. This design aids in mitigating the vanishing gradient problem, thus facilitating the training of very deep networks. In contrast, DenseNet (Dense Convolutional Network) creates direct connections between all layers within a dense block, where each layer receives the concatenated output of all preceding layers. This dense connectivity enhances feature reuse and typically results in improved parameter efficiency.

For comparison to non-deep learning methods, we also implement an SVM classifier with PCA dimensionality reduction. The PCA is set up to keep 95% of the variance in the data, and the SVM is trained using a radial basis function (RBF) kernel.

2.3 Training and Evaluation

The models are trained on the training dataset for 20 epochs. The provided training data is randomly split in an 80 to 20 ratio of training to validation data. The model at the epoch with best validation accuracy is saved as the final model to be evaluated. The performance of each model is evaluated on a separate test dataset, and metrics such as accuracy, precision, recall are calculated. The data is split into 80 percent training and 20 percent validation. The test set is provided separately from the training data.

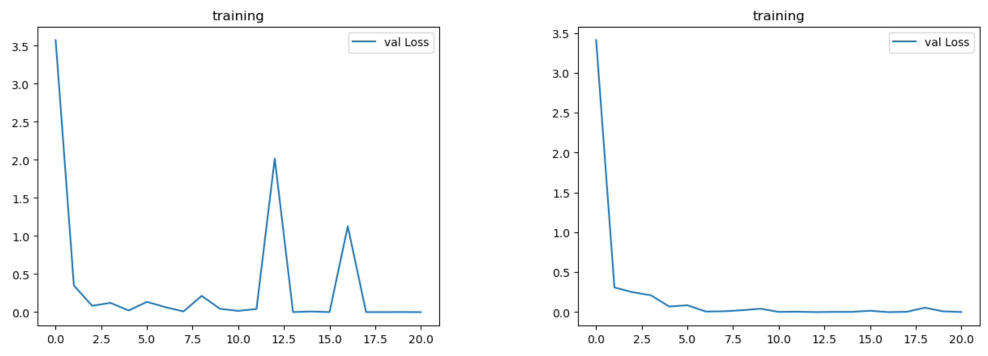
3 Results

The results of the experiments are summarized in Table 2.

Table 2: Model Performance on Test Dataset

Model	Test Accuracy	Precision	Recall
SVM + PCA	89.3%	0.88	0.89
ResNet18	100%	1.0	1.0
DenseNet121	99.98%	1.0	1.0

The ResNet18 model achieved perfect accuracy on the test set, while DenseNet121 achieved almost perfect accuracy, misclassifying two images in the test set. SVM + PCA performed significantly worse. The SVM architecture is less capable of capturing local features in the images, limiting its performance.



(a) ResNet18 Training Loss (b) DenseNet121 Training Loss

Figure 1: Training Loss Comparison for ResNet18 and DenseNet121

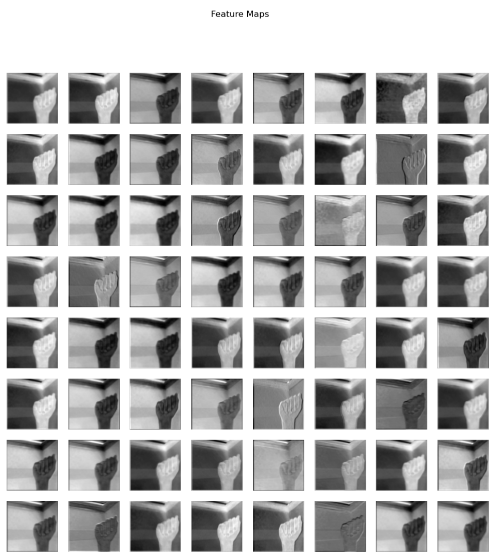


Figure 2: Feature Maps from the ResNet18 model

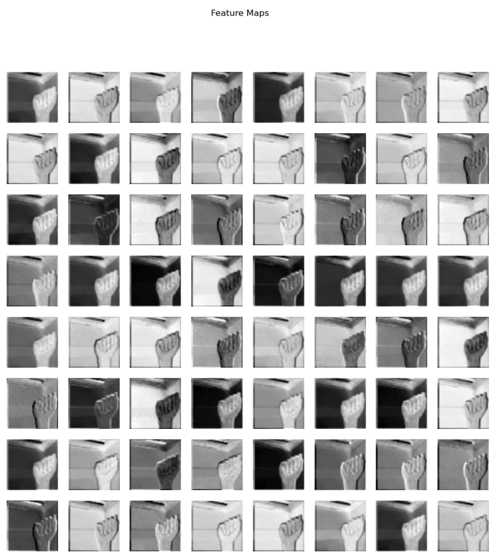
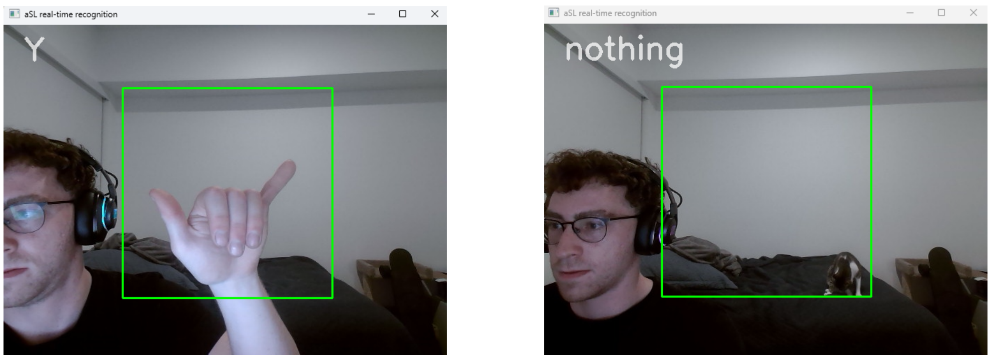


Figure 3: Feature Maps from the DenseNet121 model

The models were also tested on live video input, achieving real-time classification. The input utilizes a webcam and a defined bounding box to classify signs made by the user.



(a) Signing ”Y” (b) Nothing

Figure 4: Example live classification of video input

4 Conclusion

4.1 Extensions/Challenges

There were few challenges in terms of the performance of deep learning models on this dataset. Our ResNet18 transfer learning model achieved perfect accuracy on the test set, and our DenseNet121 model was also very close to perfect. This accuracy is very similar to literature accuracies for these models. The challenge for this dataset comes from creating lightweight models that can be used in real time or embedded systems. We used our model for live classification, which works fairly well but is far from perfect. More work could be done on processing the real time image so it is ready for classification, such as learning to place the bounding box in the correct position. Currently the software requires careful hand positioning to generate accurate classification, which is not ideal for a final product. Labels are generated very quickly and there is not a noticeable delay for live classification.

4.2 Final Remarks

The project successfully demonstrates the application of deep learning models for ASL classification. The ResNet18 and DenseNet121 architectures achieved high accuracy, making them suitable for real-time applications. Future work could focus on improving the robustness of the models in real-world scenarios, such as varying lighting conditions and hand positioning.

References

[1] Grassknoted, "American sign language alphabet," 2022, kaggle dataset. [Online]. Available: https://www.kaggle.com/datasets/grassknoted/asl-alphabet?resource=download

[2] Mdpi, "American sign language alphabet recognition using deep learning and transfer learning techniques," *Sensors*, vol. 23, no. 18, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/18/7970

[3] Techscience, "A lightweight mobilenetv2 for american sign language recognition," *Intelligent Automation & Soft Computing*, vol. 35, no. 3, pp. 493–508, 2022. [Online]. Available: https://www.techscience.com/iasc/v35n3/49360

[4] NCBI, "American sign language recognition using a deep convolutional neural network with data augmentation," *Journal of Medical Internet Research*, vol. 24, 2022. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9078784/

[5] Mdpi, "Neuromorphic vision-based recognition of american sign language letters using an fpga," *Sensors*, vol. 17, no. 10, p. 2176, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/10/2176