ASL Character Classification

Abhinav Reddy Pannala Varshith Reddy Ganji

70314901 13832628

Department of CISE, Department of CISE,

University of Florida, University of Florida,

Gainesville, Florida. Gainesville, Florida.

pannala.abhinav@ufl.edu v.ganji@ufl.edu

Abstract-- This initiative utilizes a convolutional neural network (CNN) for classifying American Sign Language (ASL) characters. TensorFlow and Keras Sequential model combine to form this network which features Conv2D layers for extracting features, Batch Normalization for consistent learning and MaxPooling2D layers for condensing data, with dropout layers added as countermeasures against overfitting. These layers assist with extracting, normalizing and reducing data volume. This project employs data augmentation techniques which resulted in 97% classification accuracy - proof that automated ASL interpretation may soon become reality!

Key Concepts--American Sign Language (ASL), Convolutional Neural Network (CNN), Sequential Model TensorFlow Keras Image Processing Feature Extraction MaxPooling2D Dropout Layers Data Augmentation Classification Accuracy Training Testing Methods Automated ASL Interpretation.

# Introduction

Machine learning has revolutionized our approach to data, but its application in language interpretation stands out. This report introduces "ASL Character Classification," an ASL recognition system noted for its robustness, efficiency and 92% accuracy rate in character identification. Our project successfully utilized convolutional neural networks (CNNs).

TensorFlow and Keras, two industry-leading deep learning frameworks, were utilized in our advanced Sequential CNN model's design in order to interpret complex gestures for accurate ASL character classification.

As part of our process, ASL images undergo careful curating and preprocessing prior to training and testing with CNNs. For training purposes, this dataset was modified by being resized, reshaped, and normalized so as to maximize learning effectiveness.

Our CNN model's architecture was carefully planned. This configuration comprises multiple layers for convolution, batch normalization and max pooling before dense layers equipped with dropout and L2 regularization were introduced for further processing of ASL images and combatting overfitting to ensure it remained generalizable and robust. To evaluate our performance, we employed a data division strategy which facilitates comprehensive assessment with particular consideration given to accuracy loss metrics in order to gain deeper insight into its effectiveness.

# Implementation

## Libraries:

NumPy acts as our main tool for manipulating the images and for developing the model for ASL character recognition, boasting powerful array processing capability. TensorFlow serves as our machine learning library of choice; specifically, Keras API within TensorFlow provides layering functions used for loading models as well as adding various layers, such as Conv2D and MaxPooling2D which are essential components when developing convolutional neural networks (CNNs).

Adam optimization, commonly used to efficiently update weights quickly. Scikit-Learn's train\_test\_split function helps validate model performance; Keras' ImageDataGenerator feature can increase robustness by creating variations within training data; finally, the Scikit-Image resize function standardizes image dimensions across datasets.

Keras' L2 regularization technique successfully prevents overfitting by increasing model adaptability to new data sets that come its way, including ASL character recognition networks CNNs that work effectively. All these libraries contribute towards making and fine-tuning an ASL character recognition network CNN that works effectively.

## Dataset Loading:

Load data is an efficient preprocessing tool designed to load image data from NumPy files using its anti-aliased scaling capability. Reshaping and resizing images meets CNN requirements by anti-aliased scaling using Skimage; normalization scales pixel values between 0-1 for optimal neural network performance ensuring image classification models perform efficiently. For successful classification models this efficient preparation of data preparation is key.

## Model Details:

The "build model" function presents the architecture of a Sequential Convolutional Neural Network (CNN) designed specifically for ASL character recognition. This network comprises several convolutional layers which progressively increase in complexity from 64 filters up to 512 filters; this gradual increase allows the network to detect intricate features from ASL images that enable recognition of signs.

As one way of combatting overfitting in deep learning models involving complex image data, each convolutional layer in this network was equipped with batch normalization and max pooling in order to combat the tendency for deep learning models dealing with such data to overfit themselves; batch normalization helps stabilizing learning while max pooling reduces spatial dimensions of output, simplifying and decreasing network parameters accordingly.

This network features densely connected layers that play an essential part in understanding features extracted by convolutional layers and making final classification decisions. To reduce overfitting risks further, these dense layers also utilize L2 regularization (which penalizes large weights in order to promote simpler models), dropout or L2-dropout techniques; L2-regularization penalizes large weights while simultaneously encouraging simpler models while dropout randomly deactivates subset of neurons during training to stop overspecialization of training data on individual neurons themselves.

This architecture strikes an optimal balance between depth and complexity on one hand and regularization techniques on the other. Reaching such equilibrium is essential to optimizing a model's performance while remaining generalizable enough for various image classification tasks involving ASL characters being identified as classifying images.

A close-up of hands with letters

Description automatically generated

Figure 1: Hand Signs

## Data Augmentation:

Load data is an essential step when training models; this function imports images and labels needed for training purposes into one place for easy upload. Once this has been done, data augmentation occurs to increase diversity within the training set.

Keras ImageDataGenerator class offers an effective means of augmenting data. Conceived specifically as an effective solution to augmenting images during real time training sessions, this tool works by making small modifications such as random rotations (up to 15 degrees), horizontal/vertical shifts and zooms (up to 10%) that introduce variability into datasets for training processes themselves.

This approach offers several key advantages; among them are increasing robustness by using moderate levels of processing to enrich data without altering essential features essential to accurate image classification, while including variations that improve ASL character recognition across conditions and orientations, leading to more robust models capable of meeting real world challenges when confronted by images that differ substantially from its training set.

## Training:

Within ASL character classification projects, data sets are typically separated into a training set and validation set to test model accuracy against new information while providing reliable metrics of its performance. With this ASL character classification project in particular, these sets were divided evenly with 20% set aside as validation data to make certain models worked effectively against unknown material to provide reliable metrics of its performance.

Adam Optimizer was selected due to its effective handling of sparse gradients while adapting its learning rate during training, thus offering gradual yet precise weight adjustment during complex classification tasks such as ASL recognition. Furthermore, its loss function supports multiple class classification scenarios - essential since this model aims to classify multiple ASL characters.

EarlyStopping, ReduceLROnPlateau and ModelCheckpoint utilities all help optimize model training while preventing overfitting by actively monitoring model's performance and stopping training when validation accuracy no longer improves; ultimately avoiding overfitting. Model Checkpoint utility will save iterations efficiently during training iterations or loss by saving its best version and maintaining this best version during this training phase or before completion. This utility saves iterations efficiently to preserve optimal training results by saving its best performing version according to what its best performing version would have been had this training been completed successfully by saving its best performing version as per its best version due.

Training of this model takes place across 40 epochs and once complete is saved with the filename A90.h5. This could represent its accuracy rate (e.g. 91%) or another predetermined naming convention.

Through careful data partitioning, model construction and training enhancement techniques, a robust CNN for ASL character recognition was successfully built. Thanks to such techniques, its output became highly accurate at accurately recognizing ASL characters.

## Data Augmentation Techniques:

As part of data augmentation techniques used to strengthen ASL character recognition models, certain data augmentation techniques such as random rotation may be implemented as data augmentative techniques. It allows a model to recognize hand signs from various orientations; width/height shifts simulate various hand placements while mirroring does not alter meaning; width/height shifts work equally well for ASL signs without changing meaning, while horizontal flips may also be employed; zooming adjusts for various hand sizes/distance from camera settings - these strategies ensure accurate real-world applications! These enhancement strategies train models to accurately recognize ASL characters under multiple orientations--key factors when deployed into practice environments.

## Model Training Parameters:

Adam Optimizer has proven itself an indispensable choice when selecting training parameters for models to ensure effective learning and overall performance. Automatic rate changes as training progresses have further maximized its efficiency here.

Model training involves breaking data down into three distinct sets for training, validation, and testing purposes: training data is typically allocated 80% toward this purpose with validation/testing being split equally (20% per set), to ensure enough data sets can be tested against for generalizability/overfitting issues to identify and resolve before moving onto final production data sets for use during production testing.

Two critical parameters not explicitly set during training but nonetheless remain essential are batch size and the number of epochs. Batch size determines how many samples pass through an network at every iteration simultaneously - this number has an enormous influence on training speed and model performance; number of epochs define how often dataset is passed forward/backward during sessions; selecting an ideal value helps prevent overfitting issues.

Loading "best\_model.h5" signals use of ModelCheckpoint callback function during training process. This function monitors model performance on validation set, saving "best\_model.h5" whenever it outperforms previous iterations to preserve an optimized version for final assessment; testing and real-world applications of optimized models then become feasible.

# Experiments

1. Dataset

The training dataset contains a total of 8443 samples. The test dataset has the same format as the training data: 270,000×100 NumPy array, where 100 is the number of test samples.

A collage of different images of hand gestures

Description automatically generated

Figure 2: Example of Dataset

1. Architecture

To get high accuracy for ASL classification projects, we used different models with variable architecture which helped us to gather the strengths and weaknesses of each model on ASL classifying documents like CNN which is great at managing the image data with precision, AlexNet did not have advanced feature extraction capability like CNN. Random Forest models may not effectively capture spatial hierarchies within image data and give lower accuracy percentage during ASL classification tasks compared with CNN models.

When selecting model architectures like CNNs for classification tasks accuracy should always be our goal for maximum precision.

Accuracy for each model when adjusted is as follows: Random Forest Classification - 74.67%

AlexNet Classification - 81.23%

ResNet Classification - 88.00%

CNN Classification - 97.37%

1. Hyperparameter Tuning

Batch normalization techniques provide neural networks with increased stability and speed by regularizing layer input by scaling, recentering and rescaling input layers of each neural network in turn; Dropout regularization methods enable concurrent training of multiple networks simultaneously. At each training step, specific outputs were removed, making the training process noisier; different nodes within each layer taking different amounts of responsibility for inputs. To strengthen resilience even further, dropout was introduced. Adaptive Learning Rate was selected due to its dynamic adaptation during training, aiding model convergence. To do this, an LR scheduler gradually altered the learning rate hyperparameter. Furthermore, one-stop policy modifications were applied to modify multiple hyperparameters simultaneously.

1. Batch Size [32,64,128,256]-32 is considered ideal as higher values don't significantly decrease training times.

2. To provide enough material for training purposes, our dataset was split in training data percentages of 10-20-30 for training purposes and 10-20-10-10-10-10 as testing material; 30% being our total split percentage.

3. Checkpointing - Checkpoints were recorded at each checkpoint to identify which of the 60 epochs with minimal validation loss was superior, whilst

4. Updating was completed prior to each epoch to guarantee greater accuracy utilizing fixed learning rate values for optimal precision.

1. Results

At first, we ran 60 Epoch's of our total machine learning model. For each Epoch (first Epoch: 12% and train loss:5.2616); validation accuracy 12.37% and Valid loss:6.0595 for same Epoch time period, then gradually it increased and in 44th Epoch it reached 88.93% with train loss being only 0.993, validation 93.61% with validation loss being 0.7874 and finally 93.37% test accuracy with test loss being only 0.5088 (from which final test accuracy being 97.37 and test loss being 0.558% of final test accuracy being 97.37 and test loss=0.558). This allowed for gradual increase throughout running of each Epochs until finally reaching 97.37% test accuracy with test loss being 0.58808.

A screen shot of a computer

Description automatically generated

Figure 3: Overall Results

# Conclusion

Convolutional Neural Network (CNN) creation for American Sign Language (ASL) categorization represents an outstanding feat at the intersection of machine learning and assistive technology. This project showcased its powerful abilities to interpret complex image data such as ASL signs essential for communication among deaf and hard-of-hearing communities; accuracy and loss rates results show efficient training against real data with incredible predictions being made below as:

A screenshot of a computer

Description automatically generated

Figure 4: Predictions of the model

Accuracy gained in recognizing American Sign Language signs is essential to real-time translation technologies that will revolutionize how those reliant on this form of communication connect; an essential step toward more inclusive technology solutions that bridge communication gaps and promote accessibility.

Future applications resulting from this initiative could involve more sophisticated uses. Future improvements might involve expanding the model's ability to read dynamic sign language sequences or merging it with real-time video processing for live translation purposes, among many other improvements. Examining how applicable this technology might be across other sign languages is also highly relevant in expanding its global impact.

Combining natural language processing and translation could result in comprehensive systems capable of translating American Sign Language (ASL) to intelligible spoken and written languages - using sophisticated neural network topologies like Recurrent Neural Networks or Long Short-Term Memory networks for greater temporal insight into ASL's temporal dynamics.

# References

1. 2. Isaacs J, Foo S (2004) Hand pose estimation for american sign language recognition, System Theory, 2004. In: Proceedings of the thirty-sixth southeastern symposium on. IEEE, pp 132–136.
2. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks", Computer Vision – ECCV 2014, pp. 818-833, 2014.
3. Núñez Fernández, D., Kwolek, B.: Hand posture recognition using convolutional neural network. In: Mendoza, M., Velastín, S. (eds.) CIARP 2017. LNCS, vol. 10657, pp. 441–449. Springer, Cham (2018).
4. Barczak AL, Reyes NH, Abastillas M, Piccio A, Susnjak T (2011) A new 2D static hand gesture colour image dataset for ASL gestures. Res Lett Inf Math Sci 15:12–20.
5. Bheda V, Radpour D (2018) Using deep convolutional networks for gesture recognition in American sign language. arXiv preprint arXiv:1710.06836.

|  |
| --- |
|  |