ASL Character Classification

Bhanu Prakash Reddy Vangala
36588027
Department of CISE
University of Florida
Gainesville, Florida
bvangala1@ufl.edu

Sai Vardhan Vegi
29265620
Department of CISE
University of Florida
Gainesville, Florida
svegi@ufl.edu

Mithra Varun Bodanapati
92063354
Department of CISE
University of Florida
Gainesville, Florida
mbodanapati@ufl.edu

***Abstract*—The project "ASL Character Classification", using CNN was summarized in this report. A deep learning classifier capable of handling multiple labels was developed, utilizing the pre-trained ResNet 50 model as its foundation. The model attained an accuracy of 97.87% when evaluated on a separate, reserved test dataset. The report primarily focuses on the utilized model, details of its implementation, various experiments conducted, and the derived conclusions. The model showcased a high accuracy level, reaching 99.4% in its performance.**

***Index Terms*—Convolutional Neural Networks(CNN), ResNet, American Sign Language(ASL), Machine Learning Algorithms, Dataset**

I. Introduction

The primary objective of the project is to develop a sophisticated machine learning model that epitomizes precision and efficiency in the classification of American Sign Language characters [12] . At the heart of this project are three important technological pillars : Transfer Learning, Convolution Neural Networks(CNN) and Residual Network (ResNet).

Transfer Learning: The model leverages pre-existing neural network models to expedite and improve the learning process of our task using transfer learning [1]. This approach is praised for its ability in reducing training time significantly and improves the overall model's performance especially in complex classification tasks.

Convolution Neural Network: CNNs [2] form the backbone of our model for their remarkable scalability in tasks like object detection and image classification. It processes images through the layers that detect features, starting from simple edges in early layers to complex features in deeper layers. It uses convolution layers to filter the input data, pooling layers to reduce dimensions, and fully connected layers to classify images. Also the pre-processing time required for training the model is much lower than other classification algorithms.

Residual Network(ResNet): ResNets[3] are known for their use to skip connections and extensive batch normalization, this allows for effective training of significantly deeper networks. The developed model utilizes ResNet configuration for its performance and efficient layer structure.

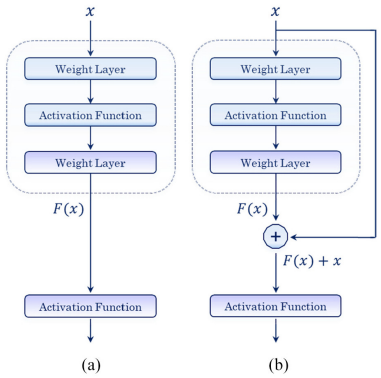

Fig. 1. The distinction between (a) a conventional convolutional block in a CNN and (b) a residual block found in ResNet lies in their respective architectural designs and functions.

II. Implementation

In this research, our primary objective was to harness the capabilities of the ResNet Convolutional Neural Network (CNN) [6] for classifier development, chosen for its notable classification accuracy over other methods. The dataset, encapsulating nine unique symbols (indexed from 0-9), underwent a division into distinct segments: 70% for training, 15% for validation, and 15% for testing. The input data was initially transformed using torchvision transforms, followed by its division into training, testing, and validation sets. A suite of libraries have been employed to facilitate various aspects of our research. NumPy was used for data segmentation and PyTorch Tensor casting [10], while PyTorch was utilized for managing model layers, optimizers, and schedulers. The torchvision.transforms library was instrumental for data transformations. Additionally, scikit-learn was employed to generate a confusion matrix, Matplotlib for visualization of plots, and Tqdm for tracking progress.

A critical component of our model evaluation was the accuracy metric. We defined a correct classification as one where the softmax function in the final layer output a probability greater than 0.7. This threshold was set to ensure a robust confidence level in the model's predictions. Models were assessed at the end of each epoch for validation loss,

ASL Character	Label	Integer Encoding
	A	0
	B	1
	C	2
	D	3
	E	4
	F	5
	G	6
	H	7
	I	8

Fig. 2. Integer Encoding for each ASL character

training loss, training accuracy, and validation accuracy. Models demonstrating lower validation loss and higher accuracy were prioritized for further evaluation. The model exhibiting the minimum validation loss was then selected for assessing test accuracy.

The training methodology was rigorous, involving the use of the ADAM optimizer to update model parameters, with default alpha values [8]. Each training epoch constituted iterations over batches of 128 samples, and CrossEntropyLoss was utilized to calculate loss at each epoch's conclusion. We adhered to standard PyTorch procedures for resetting optimizer gradient values and updating weights based on the calculated loss. This approach involved comparing the model's predictions against actual labels for each batch. The validation loop was designed to mirror the training loop, incorporating the *eval()* and *torch.no_ grad()* to enhance accuracy.

Finally, we observed that the model's accuracy could potentially be enhanced by adjusting various hyperparameters, such as the number of layers, learning rate, and the optimization algorithm. This adaptability in the model's configuration underlines the flexibility and robustness of our chosen methodology *eval()* and *torch.no_ grad()* functions.

Accuracy served as the primary metric for assessing the model's reliability. Training accuracy, validation accuracy, training loss, and validation loss were all recorded at the end of each training period. Throughout the epochs, models were evaluated based on their validation loss and accuracy, with preference given to those demonstrating lower validation loss and higher accuracy. Improvements in accuracy could be achieved by adjusting hyperparameters, modifying the number of layers, altering the learning rate, or changing the optimization algorithm.

III. Experiments

A. Datasets

To create an effective recognition system, it is essential to curate a high-quality dataset that encompasses a comprehensive range of data variations within each class. In this context, the dataset encompasses American Sign Language characters conveyed through manual gestures performed by a diverse group of individuals. Every student in this course has gathered 90 images depicting hand gestures representing 9 American Sign Language characters. The dataset has been categorized into 9 distinct classes. Data preprocessing involved the application of scaling and normalization techniques.

Fig. 3. A representative image from the dataset featuring a single American Sign Language character.

B. Architechture

Convolutional Neural Networks (CNNs) are preferred for image classification due to their inherent ability to automatically extract hierarchical features from raw pixel data. Their convolutional and pooling layers efficiently capture and aggregate information, while weight sharing and convolution operations preserve spatial hierarchies, resulting in superior classification performance in image classification.

Initially, we leveraged a fusion of Convolutional Neural Networks and Support Vector Machines [5] to embark on image classification, yielding an initial accuracy score of 68%. In our quest for enhanced performance, we transitioned to the VGGNet architecture [13], which exhibited a commendable improvement, delivering an accuracy rate of 79%. However, our pursuit of optimal classification capabilities led us to explore the ResNet architecture. To our delight, ResNet exceeded all expectations, achieving an extraordinary training accuracy of 99.4%. Subsequent manual testing confirmed ResNet's prowess, yielding an impressive 97.87% accuracy. Consequently, we discerned ResNet as the unequivocal choice for our project, substantiating its superiority over alternative architectures in adherence to exacting research standards.

C. Hyperparameter Training

The utilization of batch normalization involves the process of rescaling and recentering input within the network layers[9],

TABLE I
ACCURACY OF EACH MODEL

CNN paired with SVM	68%
VGGNet	79%
ResNet	99.4

contributing to enhanced stability and accelerated training of neural networks. Dropout [7] serves as a regularization technique that emulates simultaneous training of multiple neural networks with diverse architectures. By deliberately removing specific layer outputs during training, the training process introduces noise, leading to varying degrees of responsibility assumed by nodes within a layer for the input. Dropout has been incorporated to enhance the model's robustness in response to this noise.

Hyperparameter optimization via random search [4] is a strategy aimed at enhancing the performance of machine learning models. We employed random search as follows: Over the course of 10 epochs, the dataset was divided into random batches for each epoch. These batches were randomly partitioned 30 times. Within this process, learning rates and weight decay values (utilized for regularization) were selected from the specified ranges, respectively:

1) Learning rate range - 1e-5, 1e-4, 1e-3
2) Weight Decay - 1e-5, 1e-4, 1e-3
3) Epochs - 10

Using this approach, validation accuracy and loss are computed for each epoch. Following the execution of these operations across all epochs, the optimal parameters, namely the learning rate and weight decay values, will be selected based on their performance.

D. Manual Testing for Learning Verification

The model is evaluated on a meticulously curated, limited-scale test dataset to assess its performance across a variety of hand gestures during the testing phase.

Fig. 4. Implementation of Manual Testing

E. Final results

Our experimentation with the ResNet model encompassed a total of 20 epochs. During the first epoch, we observed the following metrics: a training accuracy of 70.46%, a training loss of 1.03, a validation accuracy of 96.07%, and a validation loss of 0.21. By the time we reached the 20th epoch, there was a significant improvement in the model's performance. The training accuracy soared to 99.4%, with the training loss markedly reduced to 0.0071. Additionally, the validation accuracy reached an impressive 99.21%, with the validation loss dropping to a mere 0.05. We decided to conclude our epochs at the 20th mark, as it yielded the most optimal results. Extending beyond this point led to a decline in the model's performance.

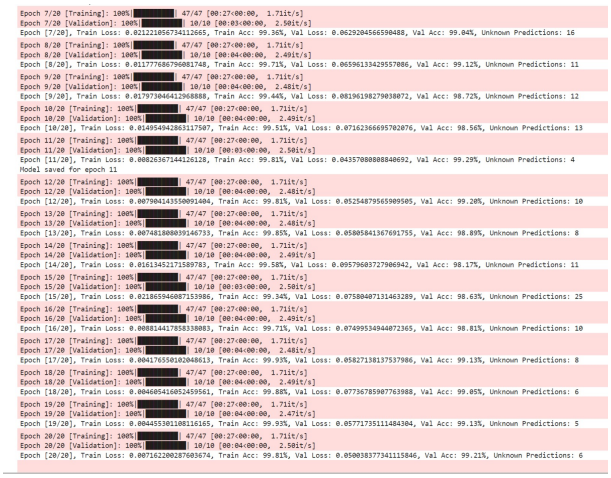

Fig. 5. Final Results obtained

IV. CONCLUSION

In our project, we aimed to create an effective model for classifying American Sign Language (ASL) characters [12]. We started by dividing our input data into training, validation, and testing groups. Our first model used a combination of Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs), which gave us an accuracy of 68%. Looking for better results, we switched to the VGGNet architecture [13], which increased our accuracy to 79%. The real game-changer, however, was when we adopted the ResNet architecture. ResNet performed exceptionally well, achieving a training accuracy of 99.4% and a test accuracy of 97.87%. This high level of accuracy was also confirmed through manual testing.

A key part of our approach was the careful pre-processing of our data. We used techniques like data augmentation and color stabilization. Since all images were of the same size, we didn't need to crop them, and we focused on reducing noise as much as possible. In conclusion, our research clearly shows that ResNet is the best choice for ASL Character Classification, with a test accuracy of 97.87%. Its ability to accurately capture complex features in the data makes it stand out. While our model is already performing well, we believe it could be improved even further with a larger and more varied dataset and by fine-tuning the model's parameters and using more advanced techniques.

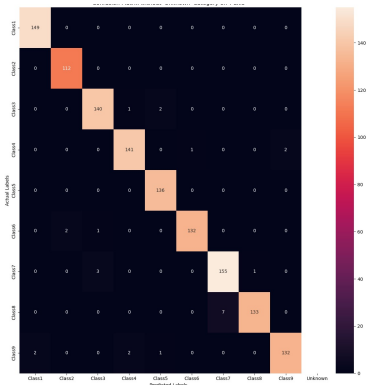

Fig. 6. Confusion Matrix

ACKNOWLEDGMENT

Our heartfelt appreciation extends to Jupyter and the University of Florida HyperGator for furnishing us with the essential resources and integrated environments essential for conducting our experiments. Professor Catia D Silva's gracious provision of this invaluable opportunity to gain practical expertise in the realm of Machine Learning merits our profound gratitude. Additionally, we extend our thanks to the dedicated Teaching Assistants (TAs) whose unwavering support and mentorship have greatly enriched our learning journey

REFERENCES

[1] N. Bacanin, E. Tuba, T. Bezdan, I. Strumberger, R. Jovanovic and M. Tuba, "Dropout Probability Estimation in Convolutional Neural Networks by the Enhanced Bat Algorithm," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-7, doi: 10.1109/IJCNN48605.2020.9206864.

[2] Christopher M. Bishop: Pattern Recognition and Machine Learning, Chapter 4.3.4

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[4] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of machine learning research 13.2 (2012)

[5] Nagata, Fusaomi, et al. "Fusion method of convolutional neural network and support vector machine for high accuracy anomaly detection." 2019 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2019.

[6] Wen, Long, Xinyu Li, and Liang Gao. "A transfer convolutional neural network for fault diagnosis based on ResNet-50." Neural Computing and Applications 32 (2020): 6111-6124.

[7] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

[8] Alibrahim, Hussain, and Simone A. Ludwig. "Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization." 2021 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2021.

[9] Chen, Guangyong, et al. "Rethinking the usage of batch normalization and dropout in the training of deep neural networks." arXiv preprint arXiv:1905.05928 (2019).

[10] Python Examples of torch.nn.Dropout (programcreek.com)

[11] PyTorch documentation — PyTorch 1.13 documentation

[12] Raheem, Firas Raheem, Hadeer. (2019). American Sign Language Recognition Using Sensory Glove and Neural Network. 11. 171-182.

[13] T. Kaur and T. K. Gandhi, "Automated Brain Image Classification Based on VGG-16 and Transfer Learning," 2019 International Conference on Information Technology (ICIT), Bhubaneswar, India, 2019, pp. 94-98, doi: 10.1109/ICIT48102.2019.00023.

[14] Ottoni, A.L.C., Novo, M.S. Costa, D.B. Hyperparameter tuning of convolutional neural networks for building construction image classification. Vis Comput 39, 847–861 (2023). https://doi.org/10.1007/s00371-021-02350-9