SIGN LANGUAGE RECOGNITION IN DEEP LEARNING: A COMPARATIVE STUDY OF CUSTOM CNN MODEL AND PRE-TRAINED ARCHITECTURES

Vanessa Huynh
University of California, Santa Cruz
vanessahuynhtvh@gmail.com

ABSTRACT

The goal of this research is to successfully train different Convolutional Neural Network (CNN) models to identify sign language images, compare the performance of each model, and figure out the best model for image recognition and classification. The dataset is collected from a public domain called Kaggle and contains 2515 images representing English alphabet letters and digits from 0-9 in American Sign Language. We use three different deep learning architectures to approach this computer vision problem. Specifically, we implement a custom CNN model and fine-tune two pre-trained models called InceptionV3 and ResNet50. These models were trained using TensorFlow and Keras. Each model consists of numerous convolutional and fully connected layers and is trained to classify images into one of 36 different classes. The models' architecture facilitates feature extraction and classification tasks, making it well-suited for image recognition problems. The performance of the custom CNN model, InceptionV3, and ResNet50 are 94.17%, 96.68%, and 98.67%. All three models generate extremely high accuracy which can contribute to helping researchers choose the best deep-learning method for Sign Language recognition and other image classification tasks.

INTRODUCTION

According to the World Health Organization (WHO), there are 20% of the global population, or approximately 430 million people with disabling hearing loss [1]. It is estimated that this number could grow to over 2.5 billion people – or 1 in every 10 people – who will experience some degree of hearing loss by 2050 [2]. In the U.S., approximately 30 million people over age 12 live with hearing disability [3]. There are several reasons attributed to hearing loss including genetics, illness, aging, or frequent exposure to loud noises [4]. Besides the significant reduction in hearing ability, these people also suffer from the challenge of communicating with normal-hearing people because those who suffer from hard of hearing or deafness use sign language while normal-hearing people use regular spoken and written language.

BACKGROUND

Those with hearing loss or deaf people communicate in many ways including using sign language, lip reading, hearing aids, etc. However, hearing aids or cochlear implants can be expensive to acquire. Over the past decades, researchers and engineers have been working together to figure out solutions to help close the gap in communication between these people and normal-hearing people.

Some previous work that incorporated Sign Language into hardware is DeepASL, which created special glasses that capture the hand movements of the signer and the voice of the speaker they are communicating with [5]. Wearable tech-glove translates speech into sign language in real time [6]. Kinect Sign Language translator using motion-sensing equipment to translate Chinese Sign Language into spoken and written words [7]. Other approaches use software such as Real-Time, Automatic Sign Language Detection for Video Conferencing [8], or On-Device, Real-Time Hand Tracking with MediaPipe by Google Research [9]. While these are complete products, this paper will focus on how to apply machine learning techniques to create image recognition systems that apply to future software and hardware technologies with improved features.

APPROACH

We approach this problem by training the Sign Language image dataset with three different Convolutional Neural Network (CNN) models in hope of providing diverse methods for researchers to develop sign Language translation systems through deep learning. For simplicity, we collected 2515 colored-hand-gestures images with different hand sizes for each letter and number with a consistent background. We propose to customize a CNN model and compare its performance with two other pre-trained models called InceptionV3 and ResNet50. InceptionV3 and ResNet50 were trained on ImageNet dataset which contains over 1.2 million images across 1000 different classes. We want to experiment how our custom model performs compared to these two well-trained models on a medium dataset.

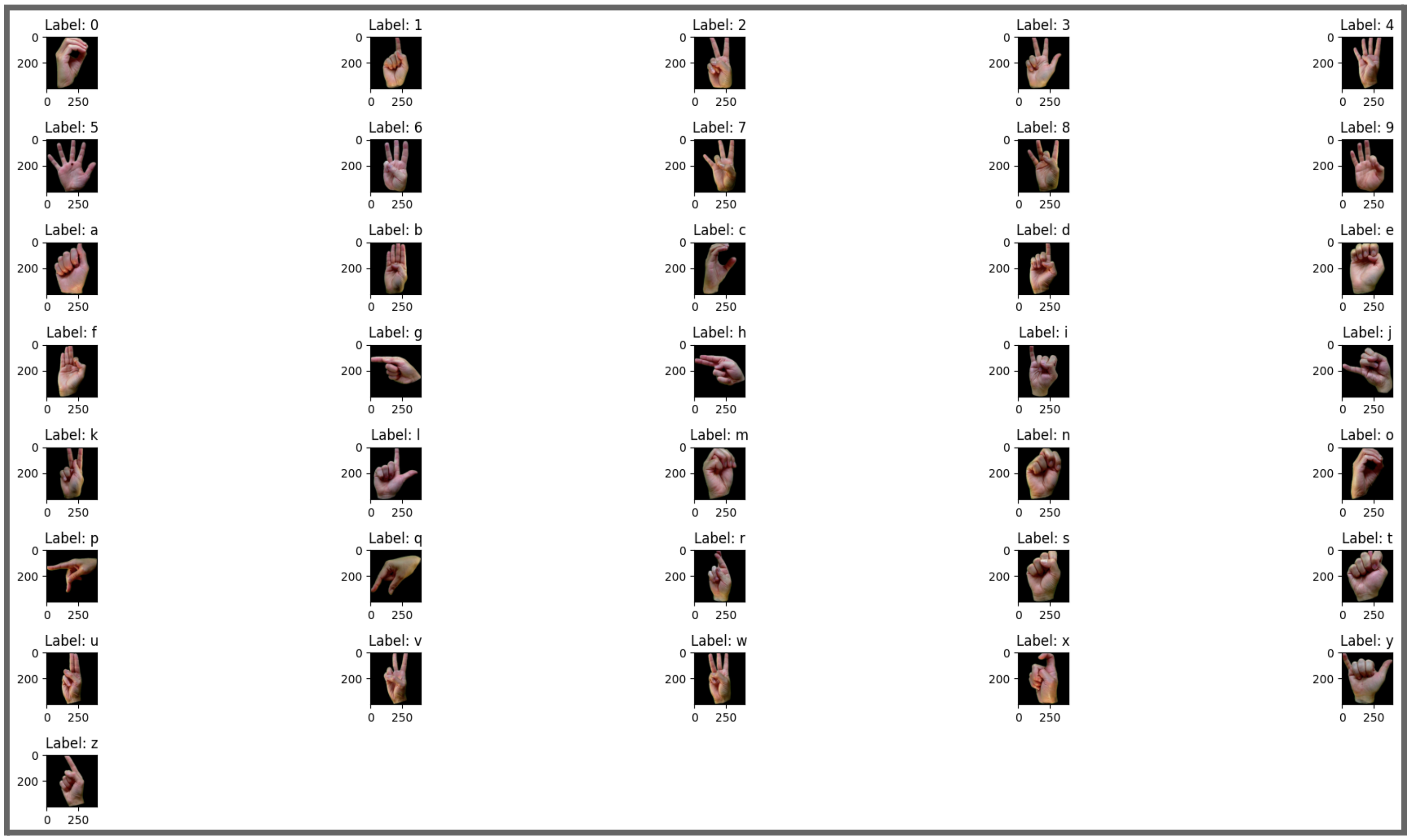


Figure 1: One sample image per class

DATASET

We collected 2515 images from a public domain and each image contains a hand signing a letter from a-z, or a number from 0-9. There are 70 images per letter and number, except the letter "t" has 65 images. The 70 images show different hands signing the same letter or number, which helps training models recognize different skin tones, hand gestures, sizes, and height distribution of hands for each class.

After uploading images and extracting labels to make a data frame, there are some crucial steps to preprocessing these data. First of all is image preprocessing. We resized images to a more manageable size from 400x400 to 224x224 for our working environment. Then we rescaled images to range 0-1 because CNNs are sensitive to the scale of pixel values in images. Rescaling ensures that our images have pixel values within an expected range to help with numerical stability and convergence. After preprocessing, we split data into training sets and testing sets. For all three models including our custom CNN model, InceptionV3, and ResNet50, we split 70% data into training and 30% into testing. After that, we encoded the letter labels into numerals to make them machine-readable.

METHOD

Our method applies deep learning to train a custom CNN model and fine-tune InceptionV3 and ResNet50. Deep learning neural networks, or artificial neural networks, attempt to mimic the human brain through a combination of data inputs, weights, and biases. These elements work together to accurately recognize, classify, and describe objects within the data[10]. Convolutional Neural Networks (CNNs), also known as ConvNets or Convolutional Networks, are a specialized class of deep neural networks designed for processing grid-like data, such as images and videos. They have been highly successful in various computer vision tasks, including image recognition, object detection, image segmentation, and more. CNNs are inspired by the human visual system and are designed to automatically learn and extract hierarchical features and patterns from visual data.

CUSTOM CNN:

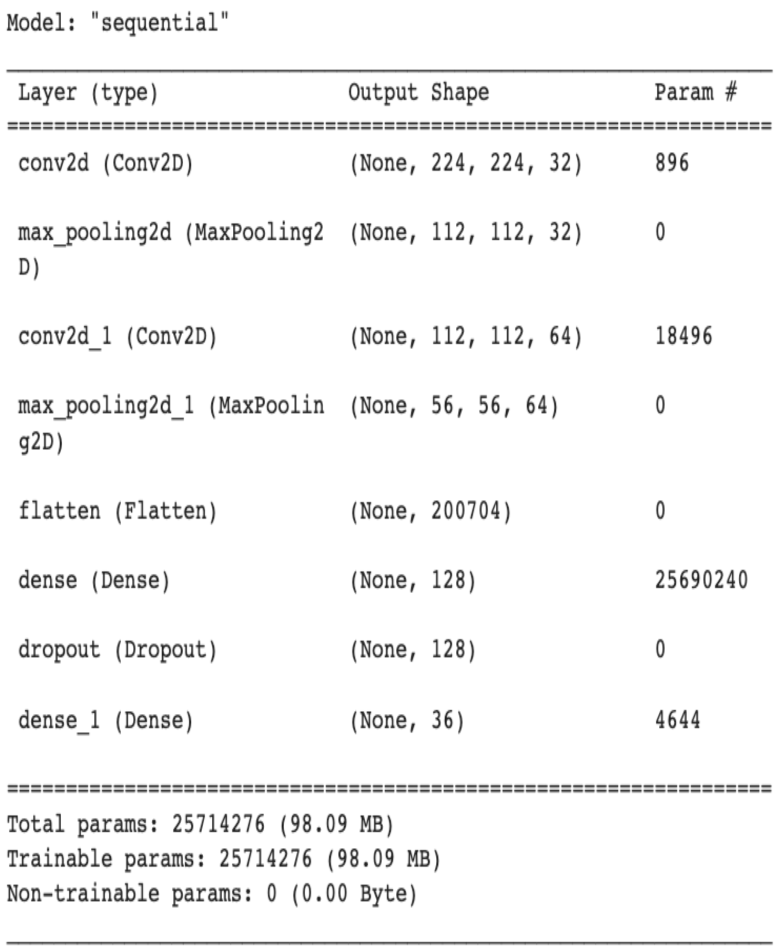
Our custom CNN model contains 32 kernels, each with a size of 3x3. Following the first convolutional layer, there is a max-pooling layer that reduces the spatial dimensions of the feature maps, which is the reason the output shape is reduced by half after the first Conv2D layer. The second Conv2D layer has 64 filters with a size of 3x3. Following that, there is another max-pooling layer that continues to reduce the spatial dimensions into half from the previous Conv2D layer. The "flatten" layer reshapes the 3D output from the previous layer into a 1D vector. In this case, it transforms the (None, 56, 56, 64) shape into a 1D vector with 200,704 elements. Our dense layer is fully connected with 128 units followed by a dropout layer to prevent overfitting. The final dense layer has 36 units that are responsible for producing predictions.

We prevent overfitting by including the dropout layer with a substantial dropout rate. It prevents the network from relying too heavily on any single neuron and encourages the model to learn more robust and generalized representations of the data. This simple technique is an effective way to enhance the model's performance and prevent overfitting in deep learning[11].

INCEPTIONV3

After implementing and training our custom CNN model, we apply Transfer Learning and fine-tune two pre-trained models called InceptionV3 and ResNet50. Transfer Learning is a method of reusing an existing model, also known as a pre-trained model, and applying the knowledge that the pre-trained model learned from prior training and testing to a new task. In other words, Transfer Learning gives us the ability to share learned features across different learning tasks[12]. Fine-tuning is one of the approaches within transfer learning, and it involves modifying the pre-trained model to fit the specific requirements of our problem.

InceptionV3 is a deep convolutional neural network architecture for image classification and computer vision tasks. It is part of the Inception family of models developed by Google and was introduced in 2015. InceptionV3 is designed to be both computationally efficient and highly accurate, making it suitable for a wide range of image-related tasks. The model is the culmination of many ideas developed by multiple researchers over the years. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al. [13]



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896

 max_pooling2d (MaxPooling2  (None, 112, 112, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 56, 56, 64)        0
 g2D)

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 128)               25690240

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 36)                4644

=================================================================
Total params: 25714276 (98.09 MB)
Trainable params: 25714276 (98.09 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 2: Custom CNN model

The InceptionV3 model begins with a 299x299x3 input image and goes through a series of layers, including convolutional and pooling layers, to process and extract features. The key innovation lies in its Inception modules, which consist of parallel convolutional pathways of varying filter sizes, enabling the network to capture features at multiple scales and abstractions. InceptionV3 features three Inception Module 1, five Inception Module 2, and two Inception Module 3, each enhancing the network's ability to understand complex visual information[14]. Global average pooling is employed to reduce the spatial dimensions, and the model concludes with a linear layer and softmax classifier for final classification, often with 1000 classes.

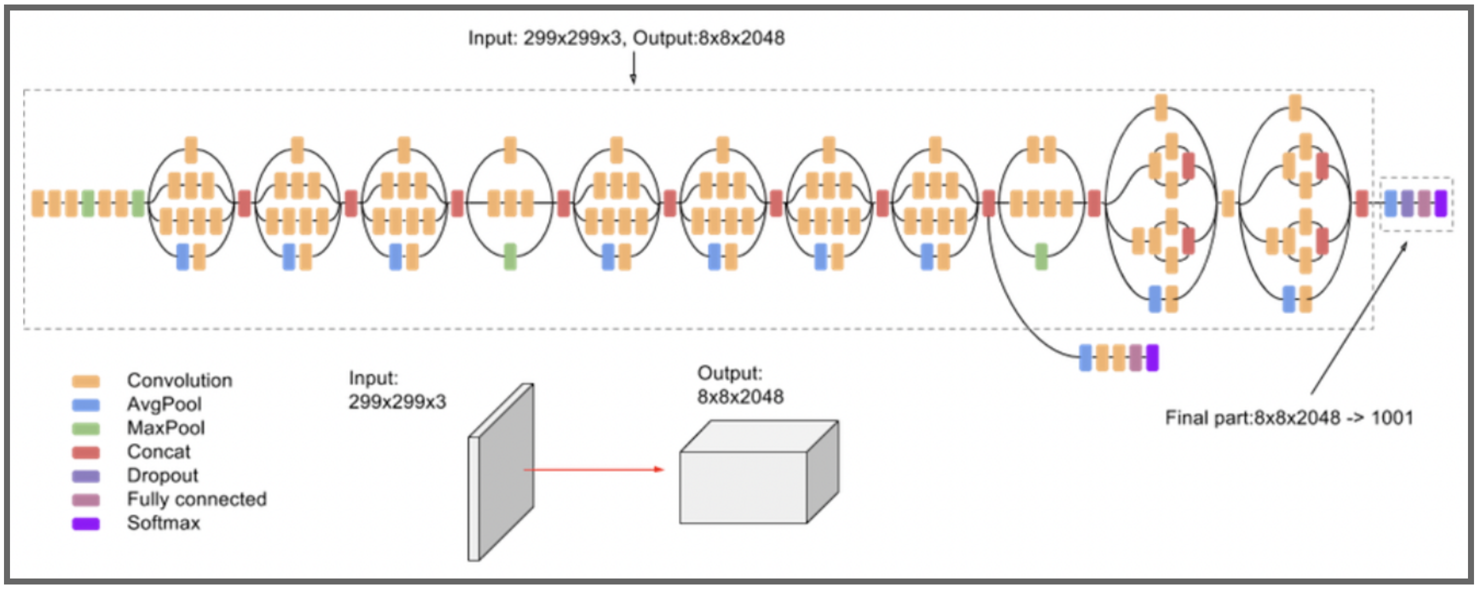


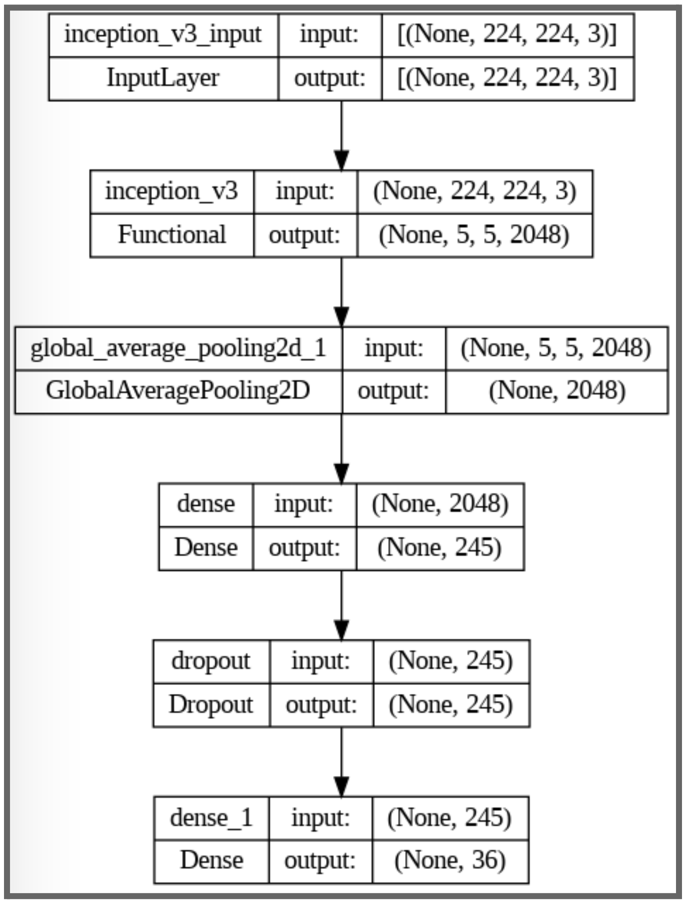
Figure 3: InceptionV3 architecture. Source: Towards Data Science

Our approach to InceptionV3 involved importing InceptionV3 as a base model. We froze the pre-trained model weights and then trained two subsequent fully-connected layers on our dataset. To customize the model for our specific task, we removed InceptionV3's original top classification layer and replaced it with a GlobalAveragePooling2D layer to downsize spatial dimensions. We then added two fully connected (Dense) layers, with the last layer employing a softmax activation function for multi-class classification. The model was prepared for training through compilation using the Adam optimizer and the sparse categorical cross-entropy loss function. This method allowed us to leverage InceptionV3's feature extraction capabilities while adapting it for our ASL image classification task.

RESNET50

ResNet is one of the popular Convolutional Neural Network architectures in machine learning. In 2015, Microsoft Research introduced Residual Networks (ResNet in short) in this paper [15] and achieved significant advancements and set new records in the field of deep learning. ResNet50 is a variant of the ResNet model that has 48 Convolutional layers along with 1 maxPooling and 1 Average Pooling Layer.

The ResNet50 architecture including six parts: Input Pre-processing, Cgg[0] blocks, Cgg[1] blocks, Cgg[2] blocks, Cgg[3] blocks, and Fully-connected layer.



inception_v3_input	input:	[(None, 224, 224, 3)]
InputLayer	output:	[(None, 224, 224, 3)]
inception_v3	input:	(None, 224, 224, 3)
Functional	output:	(None, 5, 5, 2048)
global_average_pooling2d_1	input:	(None, 5, 5, 2048)
GlobalAveragePooling2D	output:	(None, 2048)
dense	input:	(None, 2048)
Dense	output:	(None, 245)
dropout	input:	(None, 245)
Dropout	output:	(None, 245)
dense_1	input:	(None, 245)
Dense	output:	(None, 36)

Figure 4: Fine-tuned InceptionV3

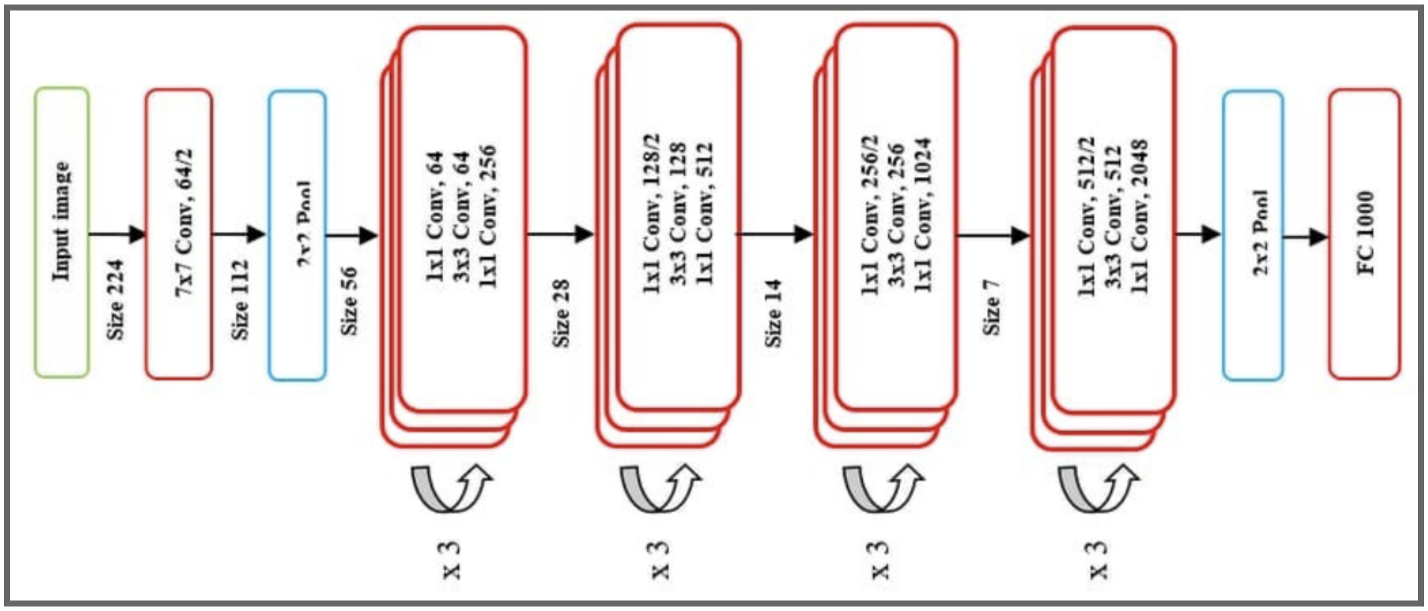


Figure 5: ResNet50 architecture. Source: Towards Data Science

In the case of our ResNet50 integration, we initiated the process by importing the pre-trained ResNet50 model with weights from ImageNet. Customization was achieved by adding some layers, such as GlobalAveragePooling2D, for spatial dimension reduction. Furthermore, we added Dense layers with ReLU activation, including a dropout layer with a high rate of 0.9 to mitigate overfitting, and a final Dense layer with a softmax activation function for multiclass classification. Our model was configured using the Adam optimizer and the sparse categorical cross-entropy loss function. This approach combined the strengths of ResNet50's pre-trained feature extraction with fine-tuning to achieve impressive accuracy in our ASL image classification task.

RESULTS

All three models obtained remarkable results during both training and testing processes. The custom CNN model achieved a training accuracy of 87.95%, and the testing accuracy was 94.17%. The fine-tuned InceptionV3 model achieved a training accuracy of 90.80%, and the testing accuracy was 96.68%. The fine-tuned ResNet50 model achieved a training accuracy of 94.55%, and the testing accuracy was 98.68%.

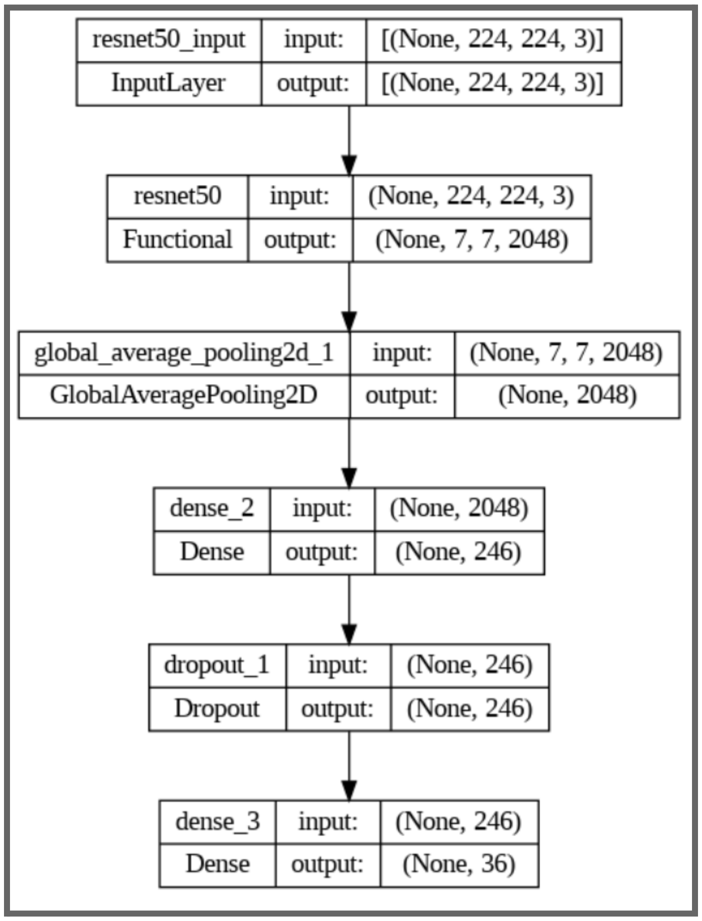
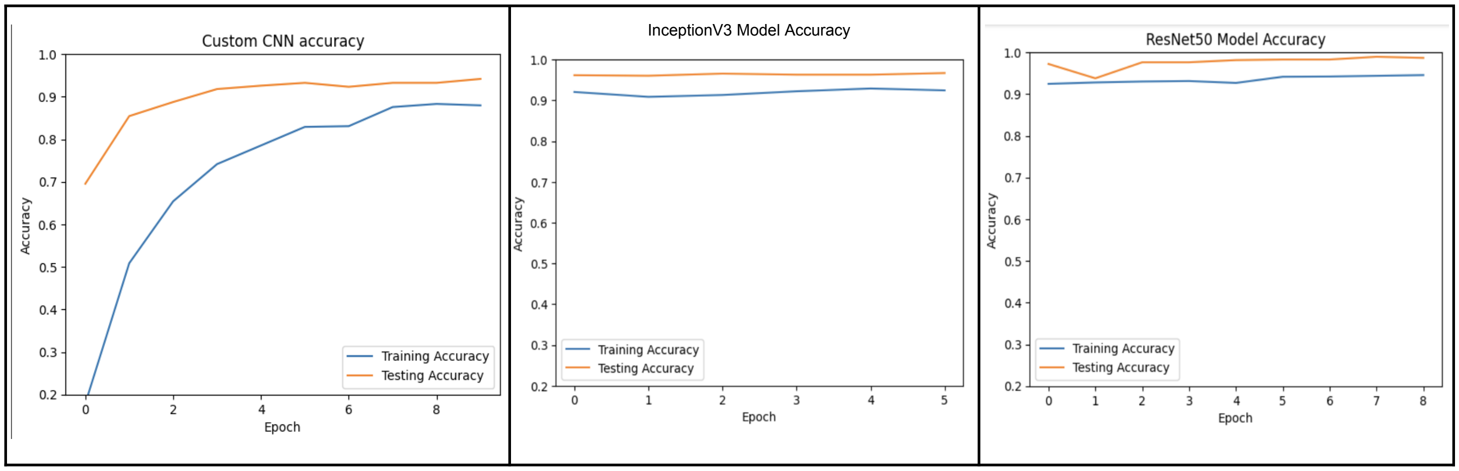

resnet50_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
resnet50 | input: | (None, 224, 224, 3)
Functional | output: | (None, 7, 7, 2048)
global_average_pooling2d_1 | input: | (None, 7, 7, 2048)
GlobalAveragePooling2D | output: | (None, 2048)
dense_2 | input: | (None, 2048)
Dense | output: | (None, 246)
dropout_1 | input: | (None, 246)
Dropout | output: | (None, 246)
dense_3 | input: | (None, 246)
Dense | output: | (None, 36)

Figure 6: Fine-tuned ResNet50



DISCUSSION

Our findings indicate notable variations in model performance within our study. The custom CNN model demonstrated a very high accuracy of more than 94%. In comparison, both InceptionV3 and ResNet50 models provided even higher accuracy rates, nearly 97% and 99%, respectively.

However, it's worth noting that the initial stability of the custom CNN model was less consistent when compared to InceptionV3 and ResNet50. This can be attributed to the fact that the custom CNN model was created from scratch, while InceptionV3 and ResNet50 were fine-tuned from pre-existing models with extensive training and testing datasets.

Despite the impressive performance of InceptionV3 and ResNet50, we observed a small gap in terms of performance that allowed ResNet50 to perform slightly better(2% higher accuracy). Further investigation revealed to us that there is an interesting subtle difference in what InceptionV3 vs ResNet50 found from our dataset.

We observed that both models have identical prediction results. However, a slight distinction occurred in the 24th class prediction. The mapping value at the predicted class to the original class represents how many instances the model predicted correctly. The confusion matrix of InceptionV3 and ResNet50 shows that InceptionV3 predicted the 24th

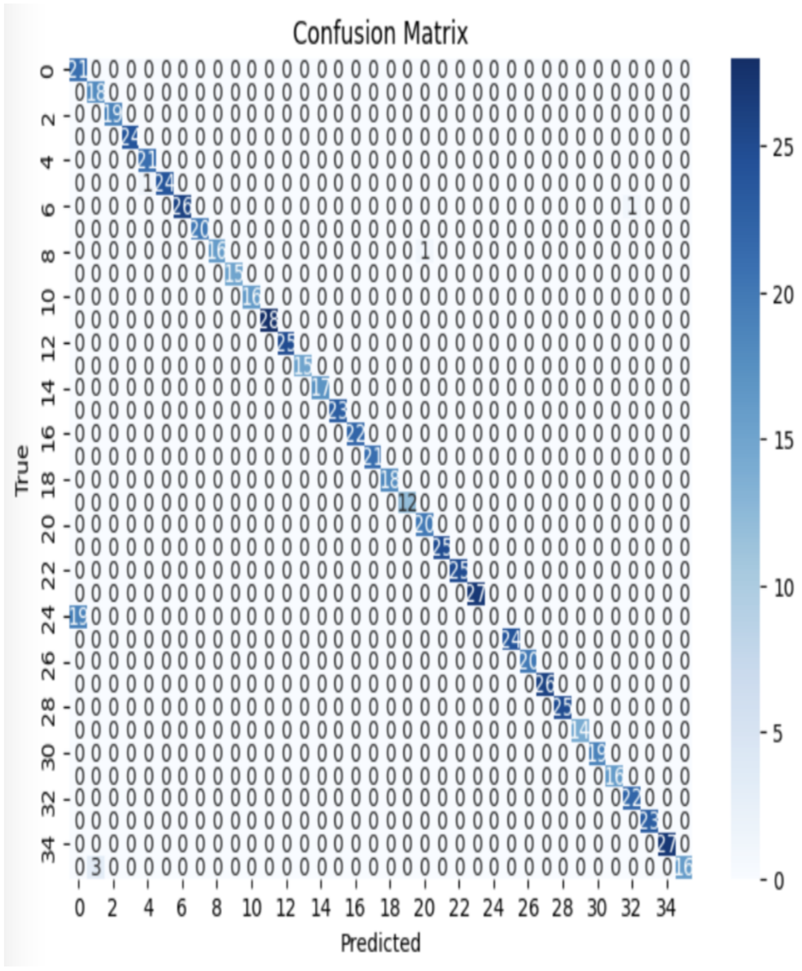
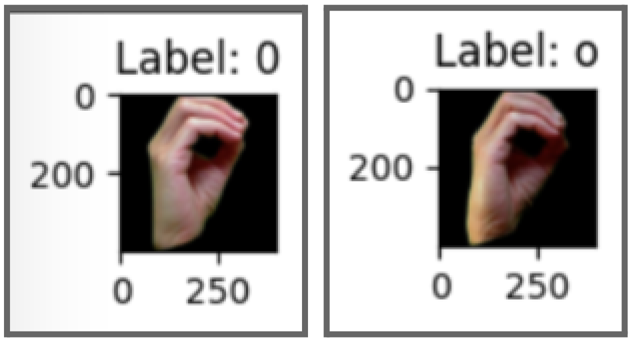


Figure 7: InceptionV3 Confusion Matrix

class zero instances correctly, and ResNet50 predicted 17 instances correctly.

At the original 24th class, InceptionV3 predicted that class was the zero class 19 times, and ResNet50 predicted that class was the zero class 2 times. We inspected further by recalling the labeling step, the 24th class is the letter “o” and the zero class is the number 0.



These signs clarify that because the signs for “0” and the letter “o” are completely identical, InceptionV3 could not distinguish between these two classes and predicted 0 to be “0”, and the letter “o” to also be “0”. On the other side, ResNet50 correctly differentiated between “0” and the letter “o” resulting in the confusion matrix.

While InceptionV3 performed well in most cases, the inability to identify a single case could be a significant disadvantage, especially in computer vision (CV) problems that involve distinguishing among similar targets, such as regression or classification tasks. On the other hand, ResNet50 has demonstrated its robustness in various cases, and with further enhancement could make it excel in future CV tasks including image classification, motion detection, and more.

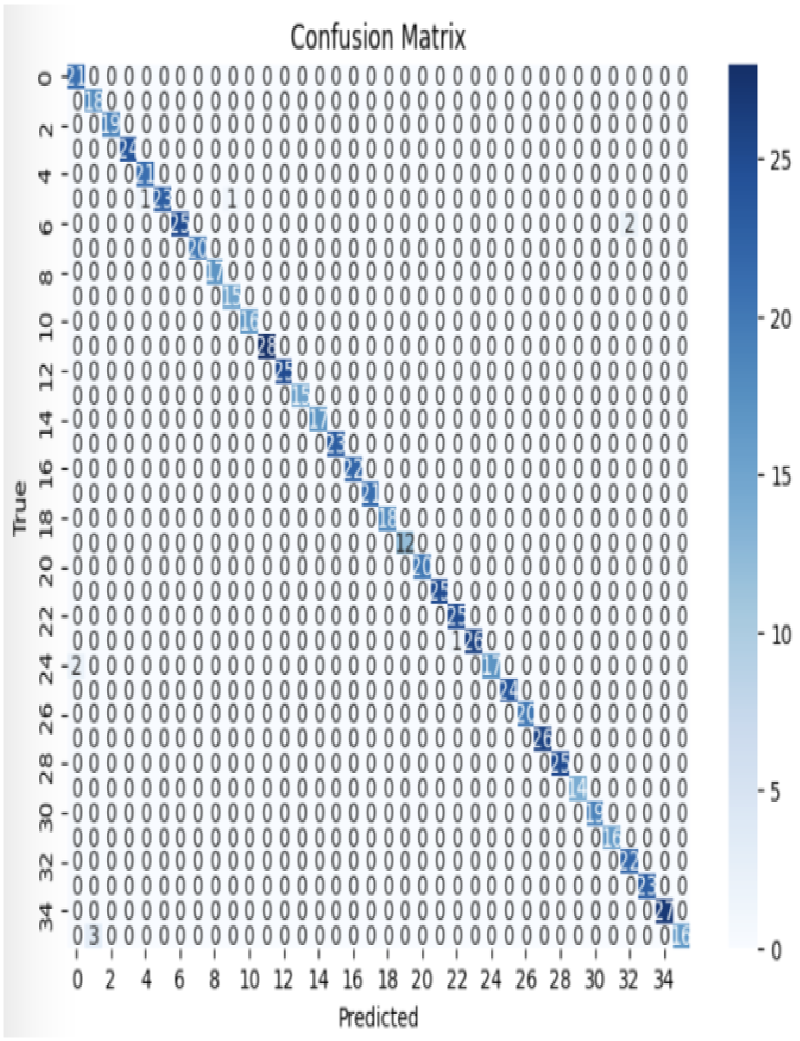


Figure 8: ResNet50 Confusion Matrix

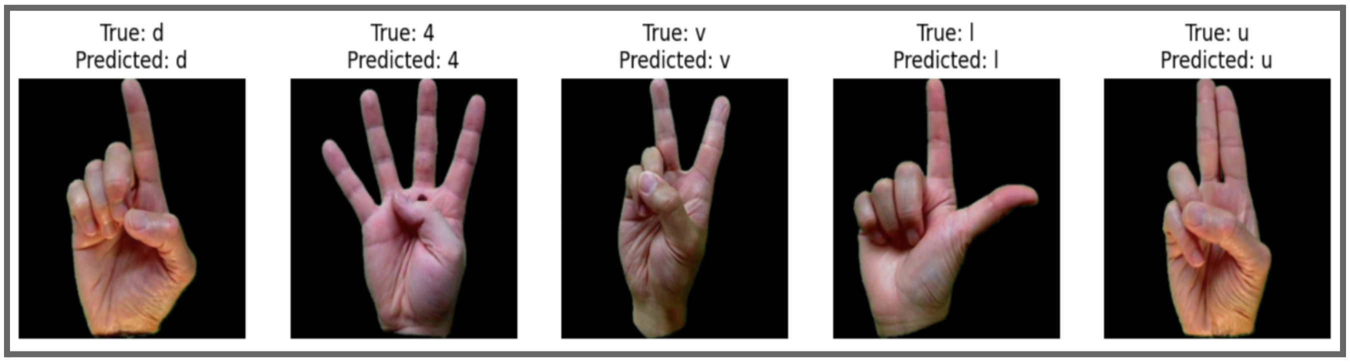


Figure 9: Examples of ResNet50 prediction

CONCLUSION

We perceived that ResNet50 outperformed InceptionV3 and our custom CNN model, achieving an impressive 99% accuracy. This makes ResNet50 a top choice for tasks like image classification and motion detection. Its ability to tell apart similar-looking characters, such as "0" and "o," shows that ResNet50 has high potential in recognizing more intricate and complex cases. Using ResNet50 in future projects could bring more accurate and reliable results in computer vision, helping researchers in various areas.

FUTURE IMPROVEMENT

Some future work that we can improve on includes tuning the custom CNN model to produce higher accuracy than the current performance (94% accuracy). We believe that this is achievable by investing more time into exploring different parameters. The second thing that we can improve on is figuring out a way to enhance InceptionV3 performance, especially in recognizing identical objects, or targets such as the letter "o" and "0" that are signed the same in ASL but have different meanings depending on context. Some approaches may be data augmentation or additional data. Besides improving our models, we also hope to experiment with more diverse datasets. Knowing that our models work well on image datasets, we could try working on more complex datasets such as sign language signed by different-size hands, or even two hands, we could also try training our models on more complex words, or even video datasets by extracting videos into frames, then extract a series of images of those frames and train on those series of images like how we did in this research.

LIMITATIONS

There are three limitations to this research. The first one is a limited dataset because our dataset doesn't have signers sign complex words with two hands or signs in different backgrounds or settings. The second limitation is computational resources. This research is a personal project and isn't funded by any organizations which prevents us from exploring more approaches that require more computational power. This is also part of the reason why we trained on a moderate dataset. The third limit is the time scope of this research is only 10-12 weeks which shortened our time to explore additional datasets, test with different models, and provide more solutions.

ACKNOWLEDGEMENT

I want to acknowledge my mentor, Matan Gans from InspiritAI, for his guidance and support throughout this project. I also want to thank Jeremiah Ockenfuss, a Ph.D. student in physics at the University of California, Santa Barbara, for clarifying machine learning concepts and reviewing this paper. Additionally, I want to thank Dr. Kerry Veenstra from the University of California, Santa Cruz, in the Department of Computer Engineering for giving helpful feedback, evoking further discussions about this research domain, and sharing his research experience.

REFERENCES:

[1]"Launch of the World Report on Hearing," www.who.int. https://www.who.int/news-room/events/detail/2021/03/03/default-calendar/launch-of-the-world-report-on-hearing

[2] "WHO: 1 in 4 people projected to have hearing problems by 2050," *www.who.int*. https://www.who.int/news/item/02-03-2021-who-1-in-4-people-projected-to-have-hearing-problems-by-2050

[3]"Deafness Facts and Statistics," Verywell Health. https://www.verywellhealth.com/facts-about-deafness-6362569

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," arXiv.org, 2015. https://arxiv.org/abs/1512.00567

[14] "Inception V3 Model Architecture," OpenGenus IQ: Computing Expertise & Legacy, Sep. 05, 2021. https://iq.opengenus.org/inception-v3-model-architecture/

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015. Available: https://arxiv.org/pdf/1512.03385.pdf

[4] Mayo Clinic, “Hearing Loss - Symptoms and Causes,” Mayo Clinic, Mar. 30, 2023. https://www.mayoclinic.org/diseases-conditions/hearing-loss/symptoms-causes/syc-20373072

[5] B. Fang, J. Co, and M. Zhang, “DeepASL: Enabling Ubiquitous and Non-Intrusive Word and Sentence-Level Sign Language Translation,” Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, pp. 1–13, Nov. 2017, doi: https://doi.org/10.1145/3131672.3131693.

[6] Z. Zhou et al., “Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays,” Nature Electronics, vol. 3, no. 9, pp. 571–578, Sep. 2020, doi: https://doi.org/10.1038/s41928-020-0428-6.

[7] L. Clawson, “Kinect Sign Language Translator - part 1,” Microsoft Research, Oct. 29, 2013. https://www.microsoft.com/en-us/research/blog/kinect-sign-language-translator-part-1/?from=https://research.microsoft.com/en-us/collaboration/stories/kinect-sign-language-translator.aspx&type=exact

[8] “Developing Real-Time, Automatic Sign Language Detection for Video Conferencing,” blog.research.google, Oct. 01, 2020. https://blog.research.google/2020/10/developing-real-time-automatic-sign.html

[9] “On-Device, Real-Time Hand Tracking with MediaPipe,” blog.research.google, Aug. 19, 2019. https://blog.research.google/2019/08/on-device-real-time-hand-tracking-with.html

[10] Amazon, “What is Deep Learning? Deep Learning Explained - AWS,” Amazon Web Services, Inc., 2023. https://aws.amazon.com/what-is/deep-learning/

[11] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014, Available:https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[12] “Hands-on Transfer Learning with Keras and the VGG16 Model,” www.learndatasci.com. https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/