Introduction

American Sign Language (ASL) serves as a vital means of communication within the Deaf and hard-of-hearing community in the United States and select regions of Canada. Functioning as a visual-gestural language, ASL employs a diverse range of handshapes, facial expressions, and body movements to articulate meaning. It stands as a complete and independent language, characterized by its unique grammar and syntax.

ASL plays a crucial role in facilitating communication for the Deaf and hard-of-hearing individuals, finding application in diverse scenarios such as education, social interactions, and professional environments. Acquiring proficiency in ASL can prove to be a valuable skill for those interested in delving into deaf culture or aiming to communicate effectively with members of the Deaf community.

To that end, this project explores the utilization of ASL as a dataset for training Convolutional Neural Network (CNN) and ResNet models. These models are designed for future classification tasks, further bridging the gap between sign language and emerging technologies for enhanced accessibility and communication within the Deaf community.

Motivation

Developing American Sign Language recognition systems can improve accessibility for the deaf, dumb, and hard-of-hearing community, enabling them to interact more effectively with the hearing world, making technology more inclusive. ASL recognition can empower Deaf individuals in various aspects of life, such as education, employment, and daily communication.

Our impetus behind developing American Sign Language (ASL) recognition systems within the scope of this project is deeply rooted in the technological landscape of data science programming. ASL recognition, facilitated by advanced deep learning models such as Convolutional Neural Networks (CNN) and ResNet-50, represents just one of the many applications of such models.

In a technological context, we aim to interpret and translate visual-gestural language, showcasing the potential of data science programming to decode complex visual data. The utilization of deep learning models, specifically CNN and ResNet-50 architectures, underscores the project's commitment to leveraging artificial intelligence in solving real-world problems.

Beyond the technical intricacies, the motivation extends to the practical implications of making technology more accessible. ASL recognition systems can empower the Deaf community by enabling seamless interaction with the digital world. This technological intervention aligns with the broader goals of inclusivity and accessibility, demonstrating the transformative power of data science programming in addressing societal challenges.

Business Questions

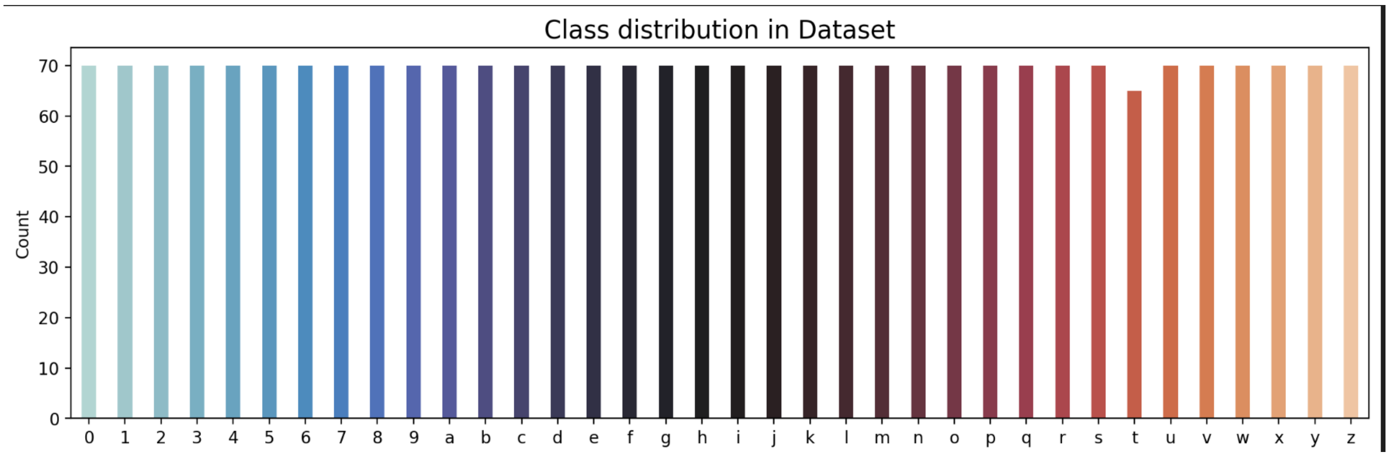
While we are only focusing on multi-class image classification, is this model reliable enough to be used for other applications simply by tweaking the dataset? Do the business applications extend beyond this ASL application and is the model even robust enough to learn and classify correctly on such a small dataset?

Data Description/Characteristics

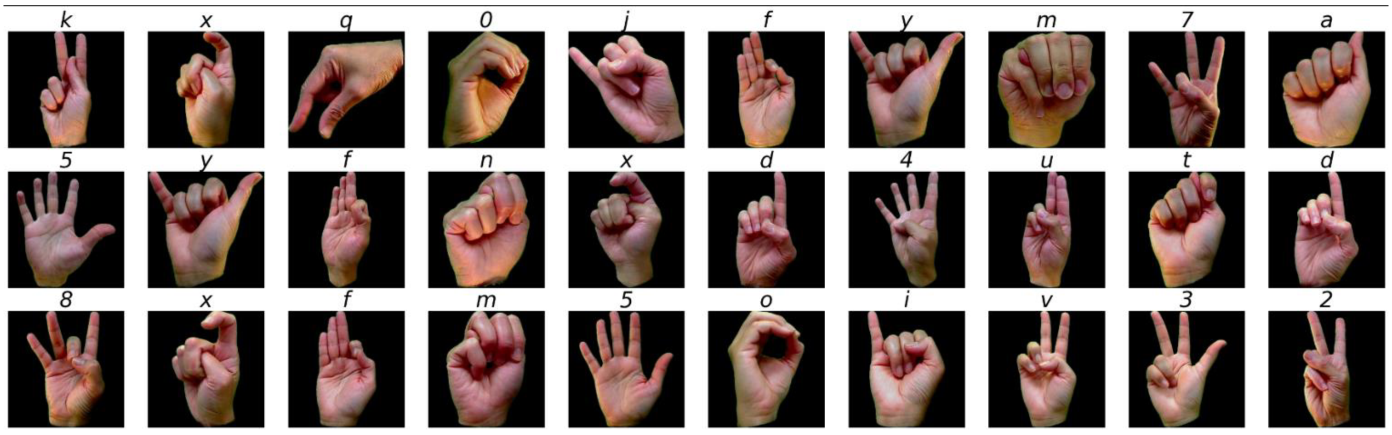
The data set is a collection of images of alphabets from the American Sign Language separated in 36 folders which represent the various classes. We have taken the dataset from Kaggle.

The data set contains 2515 images which are 400x400 pixels across 36 classes (numbers 0-9 and alphabets A-Z).

Data Distribution



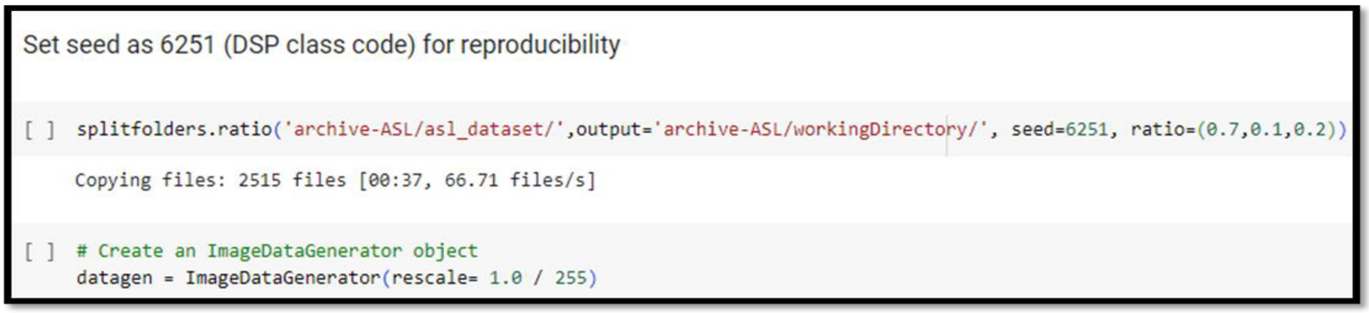
The data is distributed across 36 classes with each class having 70 records except for ‘t’, which has 65 records. We have extensively pruned our dataset from a massive ~85000 images to this final dataset of 2515 to due to computational limitations with Google Collab and Local Hardware.



Data Preprocessing

We split the data into three parts.

- Training dataset (70%)
- Validation dataset (10%)
- Testing dataset (20%)



Set seed as 6251 (DSP class code) for reproducibility

```
[ ]  splitfolders.ratio('archive-ASL/asl_dataset/',output='archive-ASL/workingDirectory/', seed=6251, ratio=(0.7,0.1,0.2))

     Copying files: 2515 files [00:37, 66.71 files/s]

[ ]  # Create an ImageDataGenerator object
     datagen = ImageDataGenerator(rescale= 1.0 / 255)
```

Choice of Hyperparameters-

Batch Size- We opted for a batch size of 32 as this choice strikes a balance between computational efficiency and model convergence. This moderate batch size allows for efficient training iterations without overwhelming system memory, supporting stable convergence during training.

Image Size-The selection of an image size of 200x200 pixels is a deliberate compromise between computational cost and the need to preserve relevant features in the ASL sign images. We determined this dimension sufficient for capturing intricate details while avoiding unnecessary computational overhead associated with larger image sizes.

Image Channels- We set the image channel value to 3, corresponding to RGB color images. This choice is logical as ASL signs, like many real-world images, exhibit color variations. Leveraging color information can potentially enhance the model's ability to discern subtle differences in sign patterns, contributing to improved classification performance.

Number of Classes- The hyperparameter n_classes = 36 was chosen to encompass the comprehensive set of ASL signs. ASL includes 26 letters, and 10 other classes to cover numbers from 0-9.

Callback Functions:

Early Stopping Callback (early_stopping):

The inclusion of the Early Stopping callback serves as a crucial mechanism to prevent overfitting and optimize training efficiency. By monitoring the validation loss (monitor='val_loss'), the model's performance is assessed during training. The min_delta parameter is set to 0.001, ensuring that only substantial improvements are considered.

The patience parameter, set to 2, indicates that training will cease if there is no improvement for two consecutive epochs. The option to restore_best_weights ensures that the model reverts to the epoch with the lowest validation loss, preserving the most optimal state. This strategy aligns with best practices in deep learning to prevent unnecessary training epochs that might lead to overfitting, especially when the model has already achieved its best performance on the validation set.

Learning Rate Reduction Callback (reduce_learning_rate):

The Learning Rate Reduction callback is implemented to fine-tune the model's learning rate during training based on the validation accuracy (monitor='val_accuracy'). A patience value of 2 signifies that if there is no improvement in accuracy for two consecutive epochs, the learning rate will be reduced. The factor of 0.5 implies a 50% reduction in the learning rate upon activation, allowing the model to adjust to subtle changes in the optimization landscape. This adaptive learning rate strategy is beneficial for navigating the model out of plateaus and potentially achieving better convergence. The verbose parameter set to 1 provides informative updates on learning rate adjustments during training.

Choice of Other Parameters:

Optimizer: Adam Optimizer; We chose the Adam optimizer for its efficiency and effectiveness in optimizing deep learning models. Adam combines the benefits of both the Adagrad and RMSprop optimizers, offering adaptive learning rates and momentum. This adaptability is particularly advantageous in scenarios where the dataset has varying gradients or noisy gradients. By dynamically adjusting the learning rates for each parameter, Adam helps accelerate convergence and navigate complex optimization landscapes.

Loss Function: Categorical Cross entropy; Categorical Cross entropy is a suitable choice for multi-class classification tasks, which aligns with the nature of the ASL recognition problem. It is a standard and widely used loss function for scenarios where each input belongs to a single class out of multiple classes. The model is trained to minimize the categorical cross entropy, ensuring that the predicted probability distribution aligns closely with the true distribution of class labels. This choice is consistent with best practices for classification tasks and helps the model learn effectively from the labelled data.

Metrics: Accuracy; Accuracy is selected as a metric to monitor the model's performance during training and evaluation. In classification tasks, accuracy represents the ratio of correctly predicted instances to the total number of instances. It provides a straightforward and easily interpretable measure of how well the model is performing across all classes.

Model Description & Justification

Justification for using CNN:

- **Local Feature Learning**: CNNs are designed to recognize patterns and features in images. They use convolutional layers to scan the image in small regions, known as receptive fields. This allows them to capture local patterns and features, which are crucial for image classification.
- **Translation Invariance**: CNNs can recognize features regardless of their position in the image. This property is known as translation invariance. It's especially useful for image classification, where the same object or feature can appear in various parts of an image.
- **Hierarchical Feature Extraction**: CNNs have multiple layers that learn increasingly complex features as you move deeper into the network. The early layers may recognize simple features like edges and corners, while deeper layers learn to recognize more complex structures. This hierarchical feature extraction is well-suited for image analysis.
- **Parameter Sharing:** CNNs use shared weights for the convolutional filters. This sharing significantly reduces the number of parameters compared to fully connected networks, making CNNs more computationally efficient and effective for image data.
- **Pooling Layers**: Pooling layers reduce the spatial dimensions of the feature maps, focusing on the most salient information while discarding irrelevant details. This helps in reducing overfitting and makes CNNs more robust.

CNN Model Architecture:

Our American Sign Language Image Classification CNN model consists of the following:

Convolutional Layers:

- Three sets of convolutional layers with increasing filters (32, 64, 128) capture hierarchical features.

Max Pooling and Dropout:

- Max pooling reduces spatial dimensions, retaining essential information.
- Dropout layers (0.2, 0.3, 0.4) prevent overfitting by randomly dropping units during training.

Fully Connected Layers:

- Flattened output from convolutions passes through Dense layers (512, 128) for predictions.
- Additional dropout layers ensure robust feature learning.

Output Layer with SoftMax:

- 36-neuron output layer (for 36 classes) with SoftMax activation for multi-class classification.

Justification for using ResNet-50:

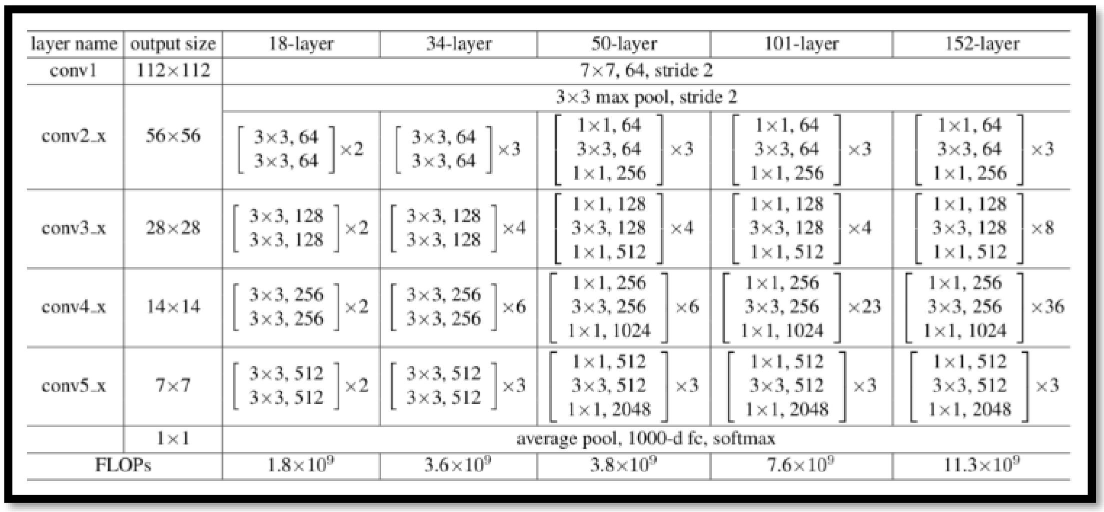
We opted for the ResNet (Residual Networks) architecture for our sign language recognition project due to several key advantages:

- **Handling Complex Signs**: Sign language involves intricate gestures and variations. ResNet is skilled at recognizing these complex patterns, which is crucial for our project.
- **Overcomes Vanishing Gradient Problem:** ResNet introduces the concept of skip connections or residual connections, which allow gradients to bypass certain layers. This helps address the vanishing gradient problem, making it easier to train very deep networks.
- **Easier Training**: ResNet's unique "residual connections" make it simpler to train deep networks. This is essential for understanding the diverse range of signs in sign language.
- **Quick Learning**: ResNet is designed to learn swiftly, speeding up the development of accurate models.
- **Transfer learning**: Pre-trained ResNet models are readily available and widely used in transfer learning. Researchers and practitioners can take a pre-trained ResNet model and fine-tune it for their specific image classification tasks with relatively small datasets.
- **Proven Excellence**: ResNet has a strong track record in computer vision, excelling at recognizing complex patterns, a necessity in sign language recognition.
- **Leveraging Pre-trained Models**: Starting with pre-trained models accelerates our model's learning and enhances its performance in sign language recognition.

ResNet-50 Model Architecture:

ResNet-50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer). The ResNet architecture follows two basic design rules. First, the number of filters in each layer is the same depending on the size of the output feature map. Second, if the feature map's size is halved, it has double the number of filters to maintain the time complexity of each layer.
The 50-layer ResNet architecture includes the following elements, as shown in the table below:

- A 7×7 kernel convolution alongside 64 other kernels with a 2-sized stride.
- A max pooling layer with a 2-sized stride.
- 9 more layers 3×3,64 kernel convolution, another with 1×1,64 kernels, and a third with 1×1,256 kernels. These 3 layers are repeated 3 times.
- 12 more layers with 1×1,128 kernels, 3×3,128 kernels, and 1×1,512 kernels, iterated 4 times.
- 18 more layers with 1×1,256 cores, and 2 cores 3×3,256 and 1×1,1024, iterated 6 times.
- 9 more layers with 1×1,512 cores, 3×3,512 cores, and 1×1,2048 cores iterated 3 times. (up to this point the network has 50 layers)
- Average pooling, followed by a fully connected layer with 1000 nodes, using the SoftMax activation function.



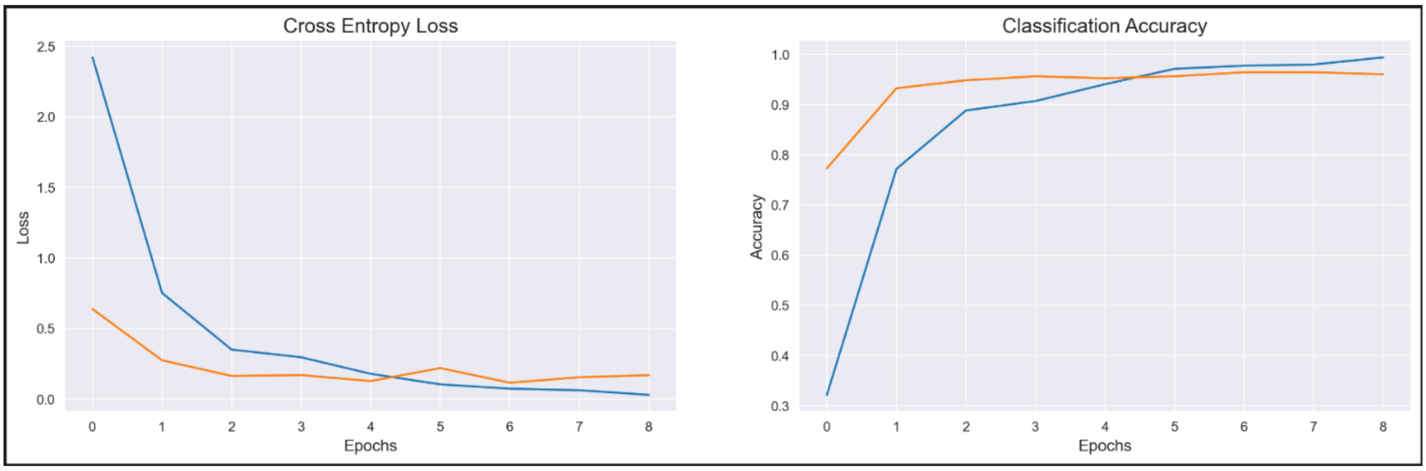
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Evaluation

In this comparative study of two deep learning models, a Convolutional Neural Network (CNN) and a ResNet-50 model, we explored their architectures, training processes, and performance in the context of American Sign Language (ASL) gesture recognition. The analysis encompassed various aspects, including model training and optimization, performance evaluation, and a direct comparison of the results.

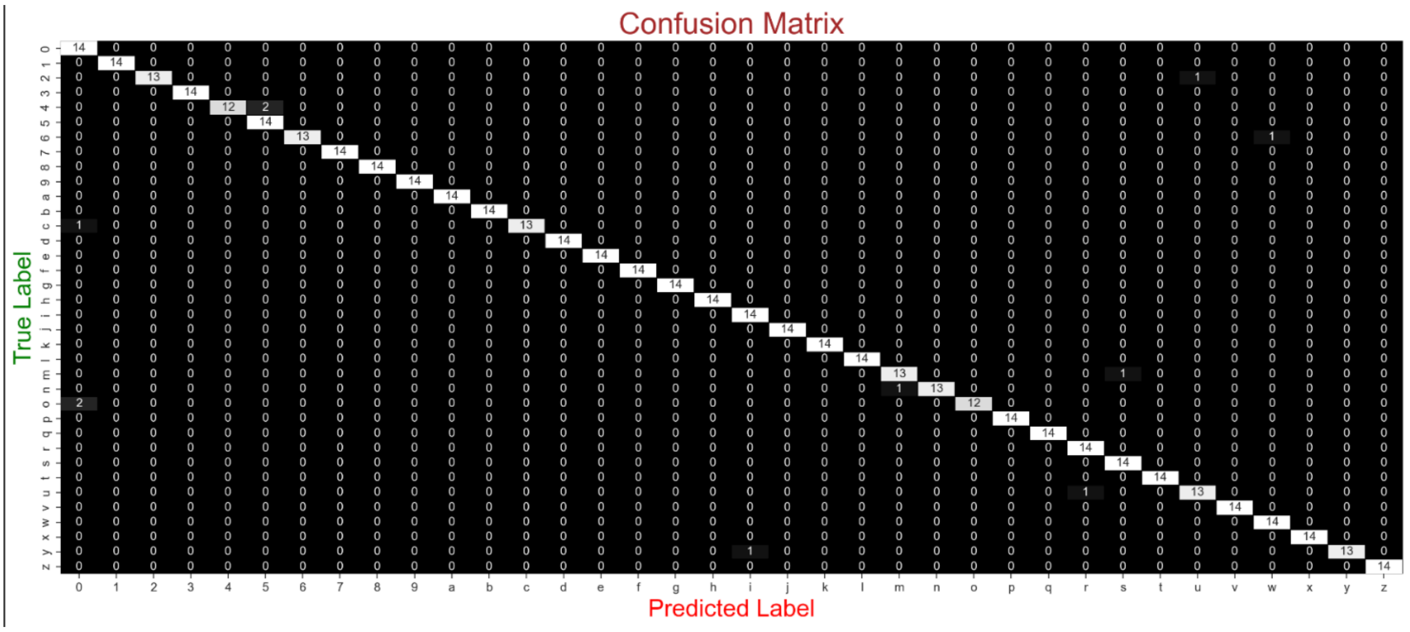
CNN Model:

Model Architecture: The CNN model featured three convolutional blocks with dropout layers, enabling it to capture hierarchical features.



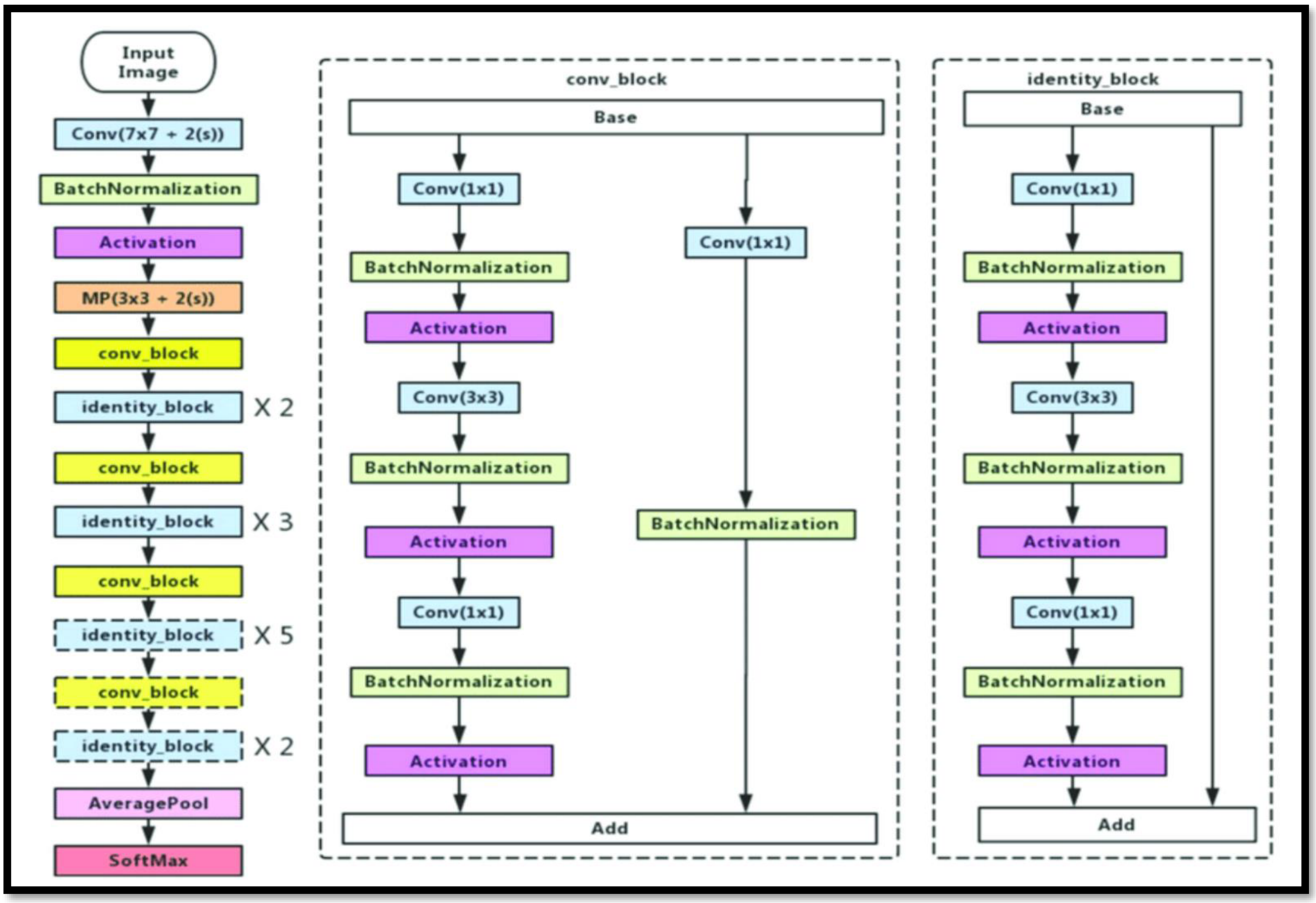
Performance Metrics: The CNN model showed commendable performance, with a training accuracy of 99.94%, a validation accuracy of 96.41%, and a testing accuracy of 97.82%. The training process demonstrated a consistent convergence rate and minimal loss.

Advantages: The model displayed good classification abilities and efficient convergence, making it an asset for ASL gesture recognition applications.



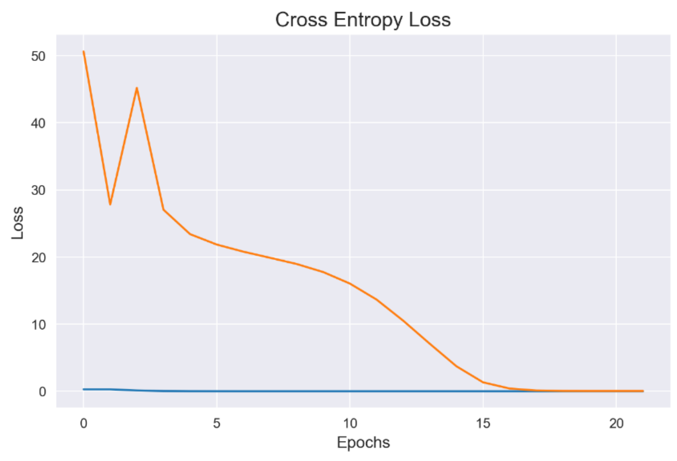
ResNet-50 Model

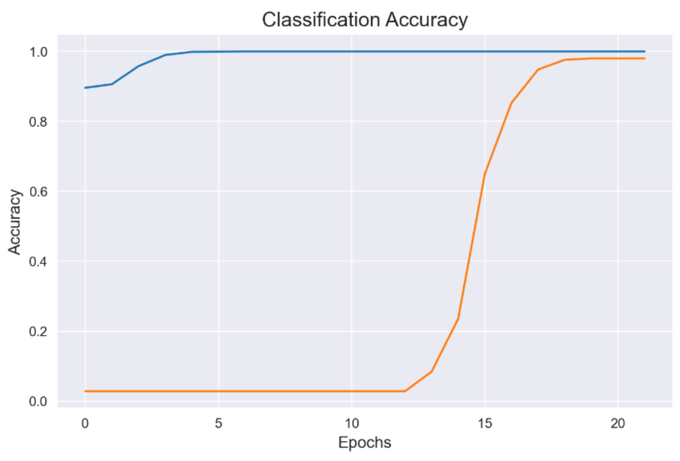
Model Architecture: The ResNet-50 model leveraged deep residual blocks to capture intricate features and promote gradient flow effectively.

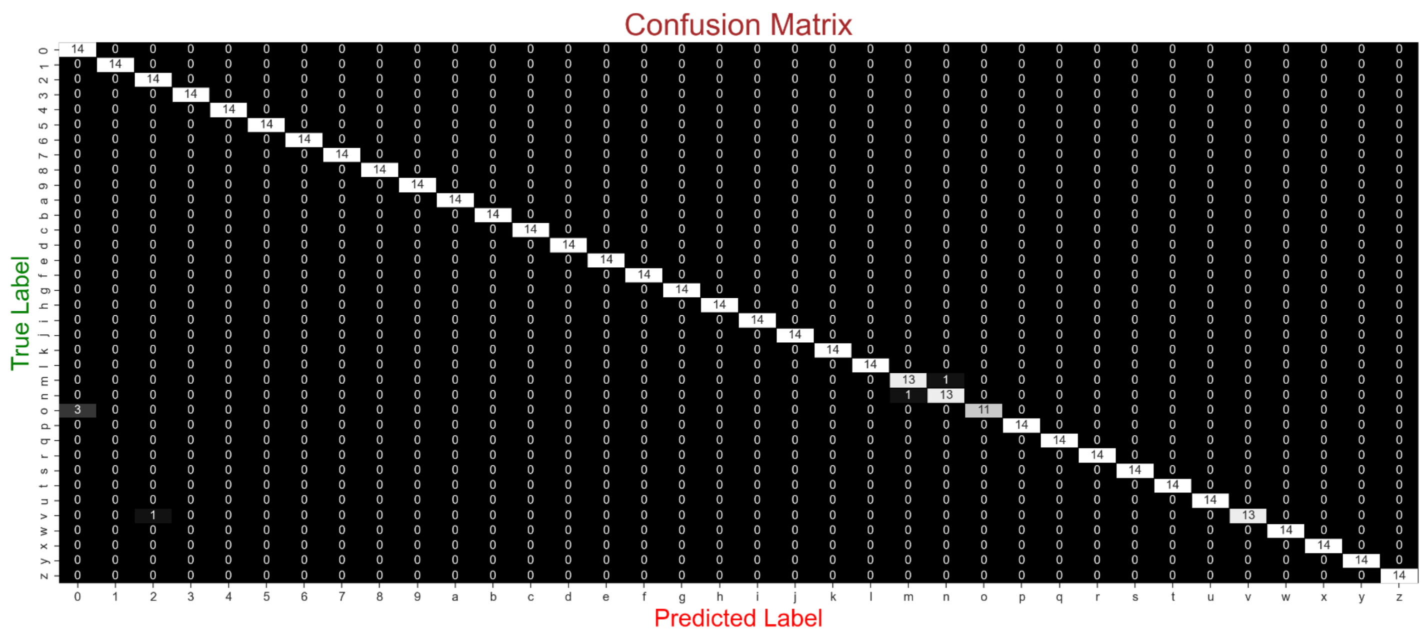


Performance Metrics: The ResNet-50 model achieved outstanding results, boasting a training accuracy of 100.0%, a validation accuracy of 98.01%, and a remarkable testing accuracy of 98.81%. The incorporation of residual blocks enhanced the model's feature extraction capabilities.

Advantages: The ResNet-50 model stood out with its unparalleled accuracy, especially on the testing set. It demonstrated robust generalization and excelled in complex image recognition tasks.







Conclusion

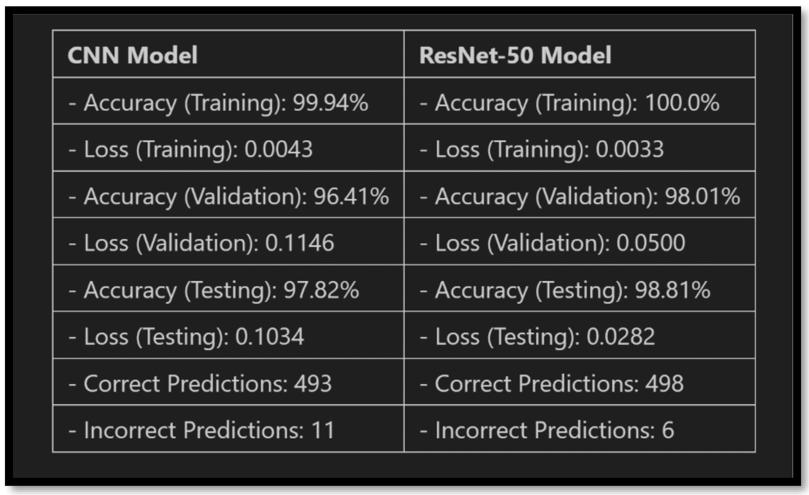
While this model has been designed to classify ASL images, its business applications can be further extended to other image classification applications such as satellite image classification, real-estate, farming/crop monitoring, and more.

1. Market Size and Growth Potential

The performance of both our custom CNN and the ResNet-50 model stands as a testament to the growing demand and potential of ASL recognition technology. Achieving high testing accuracies of 97% and beyond reiterates the readiness of such technology to address the needs of Deaf and hard-of-hearing customers.

2. Integration into Existing Products or Services

The adaptability of deep learning models for integration into existing products or services is a pivotal aspect of our project. Both the custom CNN and ResNet-50 models are versatile tools that can be seamlessly integrated into a wide range of applications.



CNN Model	ResNet-50 Model
- Accuracy (Training): 99.94%	- Accuracy (Training): 100.0%
- Loss (Training): 0.0043	- Loss (Training): 0.0033
- Accuracy (Validation): 96.41%	- Accuracy (Validation): 98.01%
- Loss (Validation): 0.1146	- Loss (Validation): 0.0500
- Accuracy (Testing): 97.82%	- Accuracy (Testing): 98.81%
- Loss (Testing): 0.1034	- Loss (Testing): 0.0282
- Correct Predictions: 493	- Correct Predictions: 498
- Incorrect Predictions: 11	- Incorrect Predictions: 6

In conclusion, both the CNN and ResNet-50 models demonstrate strengths in ASL gesture recognition. However, the ResNet-50 model outperforms the CNN model in terms of accuracy, especially on the testing set. With a testing accuracy of 98.81%, the ResNet-50 model showcases remarkable generalization capabilities, making it the preferred choice for applications that demand superior accuracy and robust recognition of complex ASL gestures.

References:

Pytorch Resnet. The Basics and a Quick Tutorial [Pytorch Resnet Tutorial]
Author: *runAI*
Publication date: March 2023

Sharma, T. Detailed Explanation of Resnet CNN Model. Medium [Resnet CNN Model Explanation]
Author: *T. Sharma*
Publication date: March 6th, 2023

ASL Dataset on Kaggle. [ASL Dataset on Kaggle]
Author: *Ayush Thakur*
Publication date: October 2018

ChatGPT – OpenAI. [OpenAI - ChatGPT]
Author: *OpenAI*
Accessed: November 7th, 2023