AMERICAN SIGN LANGUAGE RECOGNITION USING RESIDUAL AND INCEPTION NETWORKS

by

Vu Hoang Anh Nguyen

Deep Learning

Professor Sunil Vadera

University of Salford

January 13, 2023

Table of Contents

Abstract

This study investigates the use of deep learning to the task of multi-class classification of American Sign Language (ASL). Every image in the ASL collection will be assigned one of ten digits (categories). A fully convolutional neural network (CNN) is built to perform this task on the ASL dataset. Using this as a baseline, I investigate how various CNNs might improve performance. To see how transfer learning affects performance, I train the model with pre-trained weights on the ResNet50V2 and InceptionV3 networks. In comparison to the baseline (accuracy – 79.56%) and ResNet50V2 (accuracy – 93.61%) models, I find that the InceptionV3 model (accuracy – 98.72%) outperforms. The evaluation metric accuracy is applied to compare these models, providing useful information for model selection.

1. Introduction

Over 70 million deaf individuals use sign languages globally, according to the World Federation of the Deaf (WFD). Breaking through social barriers might be made easier for hearing-impaired and sighted persons with sign recognition. American Sign Language (ASL) has been utilized as the principal language of hearing-impaired individuals in North America for name spelling, book spelling, and letter correction because it is the most straightforward and useful sign language of all (NIH, 2021). For persons who have hearing loss, ASL is therefore essential. However, the ASL recognition system's advancement is frequently disregarded. Although the majority of the established communication technologies can support spoken or written language translation, they are still insufficient for ASL. Building a reliable ASL recognition model is therefore essential for improving communication for hearing-impaired persons to help with letter correction, book spelling, and name spelling (Ma et al., 2022). For the recognition of sign language, certain deep learning approaches have been employed. The

Convolutional Neural Network (CNN), out of these techniques, often yields superior sign language recognition precision (Rastgoo et al., 2021).

2. Concepts and Theory

3.1. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a distinct variation of a multi-layer neural network inspired by the functionality of the visual cortex of living organisms. Advances in powerful GPU processors and an improved regularisation technique have enabled CNN to perform better in image classification tasks (Sultana et al., 2018). Numerous industrial leaders, like Google, Amazon, and Facebook, are using it (Pigou et al., 2015).

As illustrated in figure 1, a typical CNN is constituted of one or more blocks of convolution and subsampling layers, followed by one or more fully-connected layers, and an output layer. The CNN's convolution layers use a predetermined number of trainable weight filters to execute numerous discrete convolutions on the input. During training, the weights of the filters were adjusted. It will gather image attributes such as lines, edges, colours, and other visual components after applying filters to each channel. More complex characteristics, such as forms and patterns, can be constructed as the network depth increases. In order to develop feature maps, which can be sent to the standard neural network classifier to accurately categorize the item, all of these characteristics must be combined with the activation functions (Lum et al., 2020).

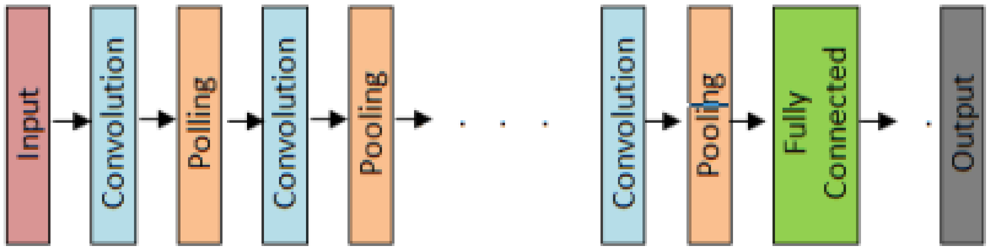


Figure 1. The architecture of a typical CNN (Sultana et al., 2018)

3.2. Transfer Learning

Transfer learning is a deep learning approach where a neural network model is first trained on a problem that is similar to the one that is being solved. A new model that is trained on the target issue then incorporates one or more layers from the trained model. Transfer learning offers the advantage of cutting down on neural network model training time and potentially lowering generalization error. The training procedure may be started using the weights in the previously utilized layers and modified to address the new issue. Transfer learning is viewed in this context as a specific kind of weight initialization approach (Brownlee, 2020).

Numerous CNN designs, including ResNet and GoogLeNet (Inception-v1), were established as a result of the major advancements made in image classification. On massive, annotated datasets like ImageNet, these models were developed and successfully tested. Leveraging transfer learning on these architectures, as opposed to building the CNN from scratch, can considerably cut down on the time needed to create a workable model for a specific challenge. Since these models were trained using ImageNet, which contains 1.2 million categorized natural images in 1000+ classes, they were also enhanced. As opposed to beginning from scratch, using the learned weights as a starting point will unquestionably speed up convergence (Lum et al., 2020). The deep architecture, which has been well-trained, can generalize on additional specialized domain problems with only a little bit of fine-tuning training, notwithstanding the mismatch between the ImageNet and the sign language images (Shin et al., 2016).

3.3. ResNet

The notion behind residual networks (ResNet) is that every additional layer should facilitate the ability to include the identity function as one of its components (He et al., 2016). The "skip connections" feature of ResNet offers a novel solution to the vanishing gradient issue. Multiple identity mappings (convolutional layers that initially do nothing) are stacked, skipped, and the activations from the preceding layer are reused. By reducing the network's number of layers, skipping accelerates initial training. The remaining network components – known as the residual parts – are then let to explore more of the input image's feature space once the network has been retrained, with all layers being enlarged (*ResNet: The Basics and 3 Resnet Extensions*, 2022)

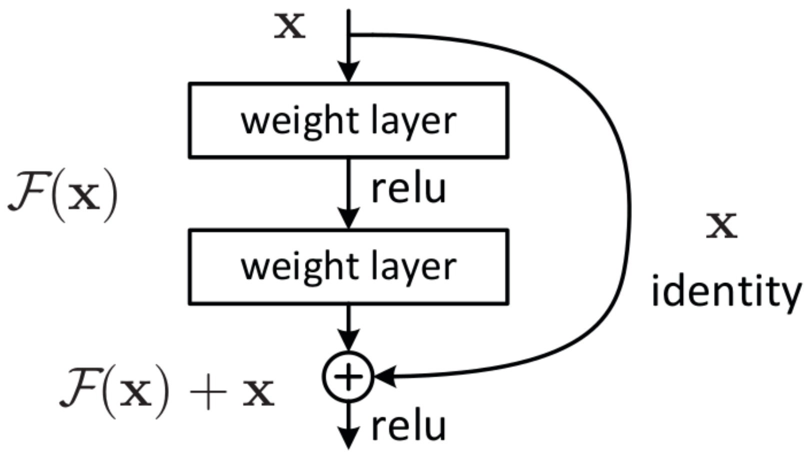

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Block diagram of ResNet architecture (He et al., 2016)

The framework allowed the layers to fit a residual mapping, designated as H(x), and it allowed the nonlinear layers to fit a different mapping, designated as F(x): = H(x)−x, resulting in a change in the original mapping to H(x):= F(x)+x, as shown in Figure 2 (Kaushik, 2020).

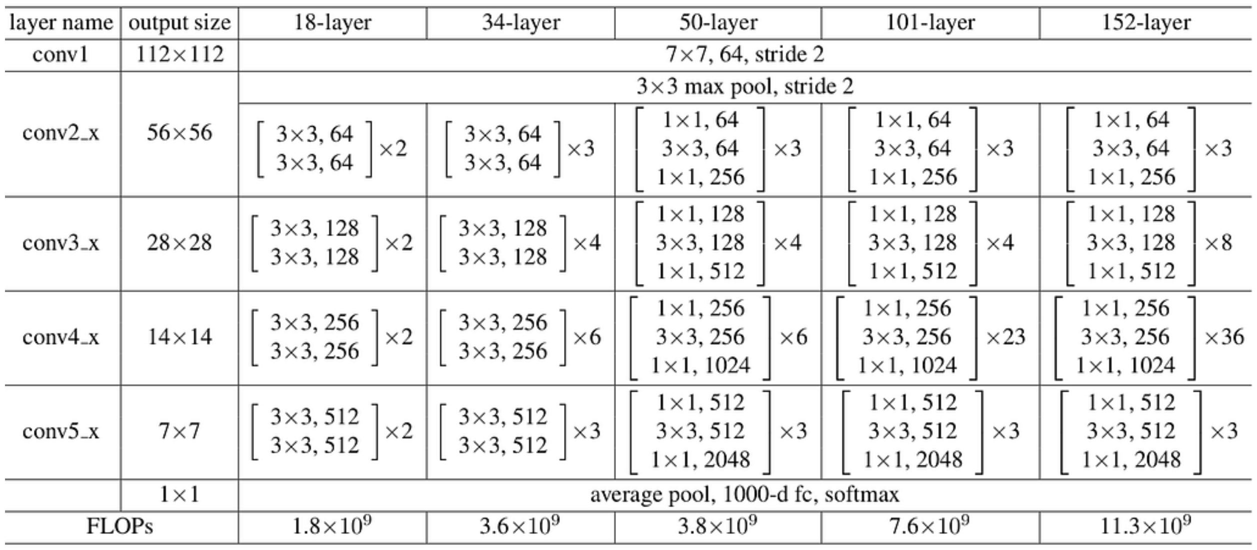
3.3.1. ResNet50

The bottleneck building block is used in the 50-layer ResNet. A bottleneck residual block, sometimes referred to as a "bottleneck", employs 11 convolutions to cut down on the number of parameters and matrix multiplications. This makes each layer's training significantly faster. Instead of using a stack of two levels, it employs three layers (*ResNet-50: The basics and a quick tutorial*, 2022).

As listed in the figure below, the 50-layer ResNet design consists of the following components:

- A convolution of a 7x7 kernel with 64 additional kernels and a 2-sized step.
- A stride of 2-sized with a maximum pooling layer.
- 9 additional layers – three 3x3,64 kernel convolution levels, one 1x1,64 kernel layer, and one 1x1,256 kernel layer. Repeating these three layers three times.
- 12 more layers – 1×1,128 kernels, 3×3,128 kernels, and 1×1,512 kernels each were added with iterations of 4.
- 18 more layers – 1×1,256 cores, and 2 3×3,256 cores and 1×1,1024 cores with 6 iterations
- 9 more layers of 1×1,512 cores, 3×3,512 cores, and 1×1,2048 cores iterated three times. (At this time, there are 50 layers in the network.)

A fully linked layer with 1000 nodes is then created using the softmax activation function, after average pooling.



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

Figure 3. The architectures of ResNet models (He et al., 2016)

3.4. Inception Network

The "naive" inception module is shown in Figure 4. On an input, convolution is performed with three different-sized filters (1x1, 3x3, 5x5). Max pooling is further utilized. The outputs are combined before being forwarded to the subsequent inception module (Raj, 2020).

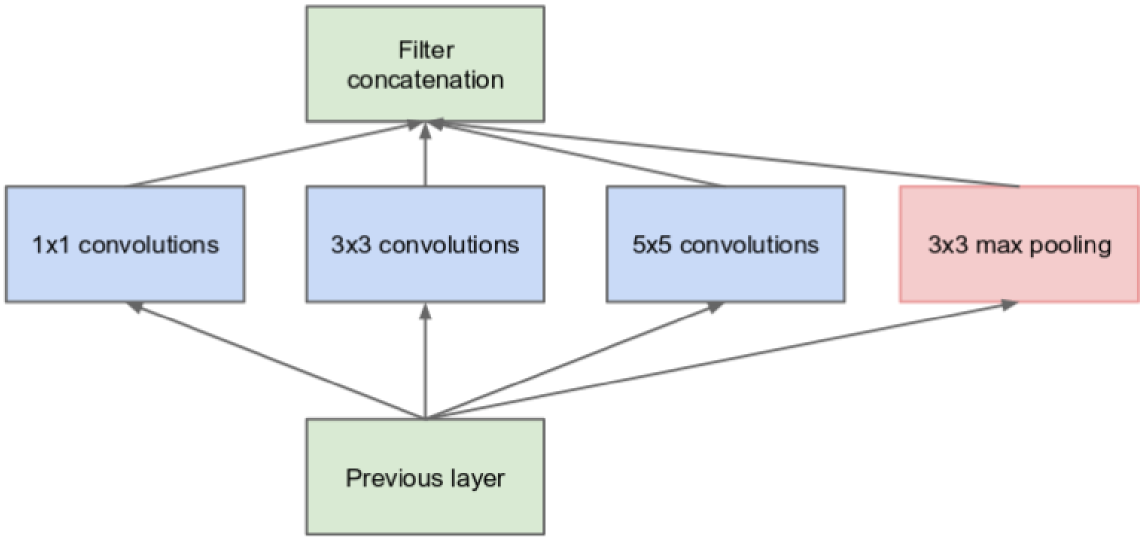


Figure 4. The naive inception module (Szegedy et al., 2015)

There are two significant additions to the inception module in addition to the layers indicated above:

- 1x1 Convolutional layer, which is utilized mostly for dimension reduction before another layer is applied.
- A parallel Max Pooling layer, which gives the inception layer another option.

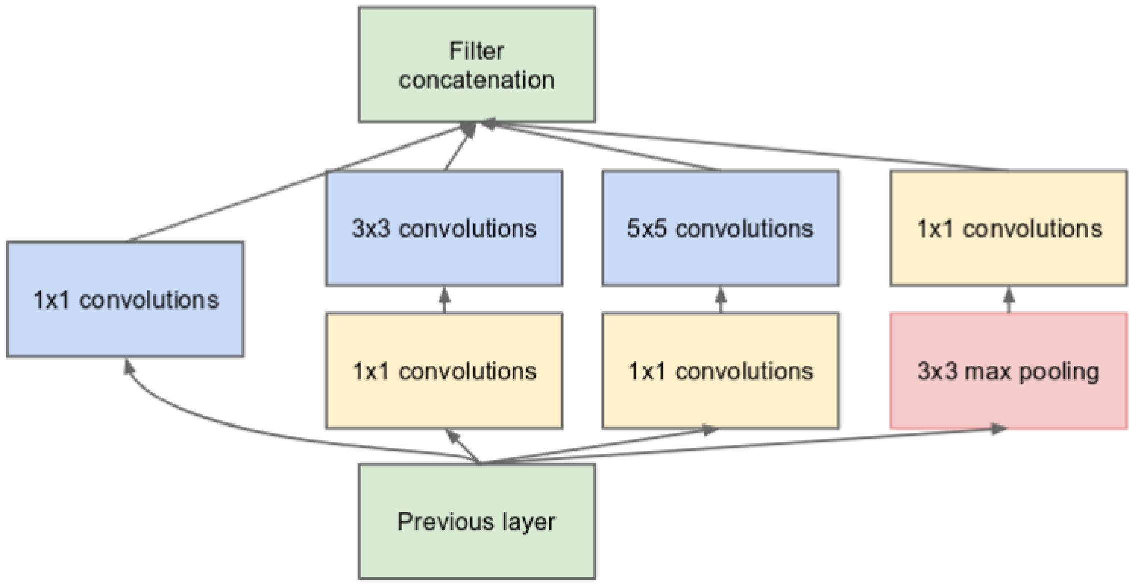


Figure 5. Inception module with dimension reduction (Szegedy et al., 2015)

3.4.1. Inception-v3

The inception network's constant evolution led to the creation of several versions of the network. Inception-v3 is a popular version.

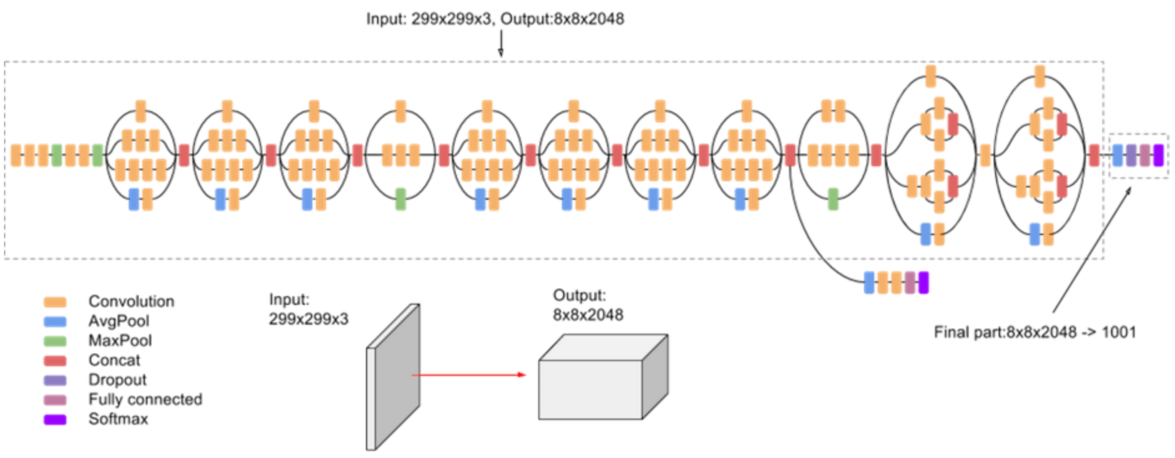


Figure 5. Inception-v3 model (Faisal et al., 2020)

Three components make up the Inception-v3 model: the classifier, an enhanced Inception module, and the fundamental convolutional block. For feature extraction, the fundamental convolutional block, which alternates convolutional and max-pooling layers, is utilized. The enhanced Inception module's architecture is built on Network-In-Network, where multiple scale convolutions are performed simultaneously and the outputs from each branch's convolution are further combined. The employment of auxiliary classifiers leads to more consistent training outcomes, improved gradient convergence, and simultaneous relief from overfitting and vanishing gradient problems. Inception-v3 makes extensive use of 1x1 convolutional kernels to cut down on the number of feature channels and speed up training. Additionally, the large convolution is split up into smaller convolutions, lowering the complexity and cost of calculation. As a result of its distinctive Inception architecture, Inception-v3 provides outstanding performance for object identification (Lin et al., 2018).

In total, the inception V3 model is made up of 42 layers. The Inception-v3 model's main components are shown in Table 1. The size of each module's output serves as its corresponding module's input in this instance.

Type	Patch / Stride Size	Input Size
Conv	3×3/2	299×299×3
Conv	3×3/1	149×149×32
Conv padded	3×3/1	147×147×32
Pool	3×3/2	147×147×64
Conv	3×3/1	73×73×64
Conv	3×3/2	71×71×80
Conv	3×3/1	35×35×192
3 × Inception	Module 1	35×35×288
5 × Inception	Module 2	17×17×768
2 × Inception	Module 3	8×8×1280
Pool	8 × 8	8 × 8 × 2048
Linear	Logits	1 × 1 × 2048
Softmax	Classifier	1 × 1 × 1000

Table 1. The components of Inception-v3

3.5. Early Stopping

Overfitting of the training dataset can occur when there are too many epochs, and underfitting can occur when there are too few. Early stopping is a technique that helps to define an arbitrarily large number of training epochs and terminate training as soon as the model performance on a hold-out validation dataset stops increasing (Brownlee, 2020).

4. Methodology

4.1. Dataset

The data set contains 2477 images with 10 classes for digits (1 to 9). The data is preprocessed by resizing to 128x128 pixels, using a random shuffle and normalization of the data. The original images consist of RGB coefficients in the 0-255, but such values would be too high for the models to process. Therefore, to target values between 0 and 1 instead, scaling with a 1/255 factor is used. The images are split into 2 sets of training and validation. The training set has 1929 images, and the validation set has 548 images.

4.2. Baseline CNN

A baseline model is a simple model that can be developed quickly with minimum technical knowledge and can deliver acceptable results on a task (Li et al., 2020). A simple CNN is built for the purpose of sign language classification. For the goal of classifying sign language, a straightforward CNN is constructed. The model's overall architecture includes a Convolutional layer to learn the features of the input images, a Max Pooling layer to reduce the number of parameters to learn and aid in translation invariance, and a Flatten layer to flatten the convolutional layer's output into a single long feature vector. The model continues with a Dense neural network layer of 512 neurons with ReLu as the standard activation function. The final layer is the Dense layer which provides a classification for 10 classes of images, using a softmax activation function.

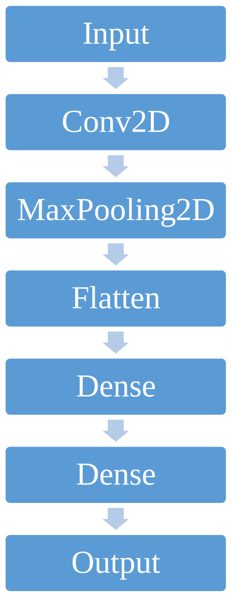


Figure 6. Experimented baseline model.

4.2. ResNet50V2 Model

The ResNet50V2 is preloaded with weights trained by ImageNet, continuing with a Dense neural network layer of 512 neurons. This dense layer uses ReLU as its activation function. The final dense layer with softmax activation gives classification for 10 groups of images.

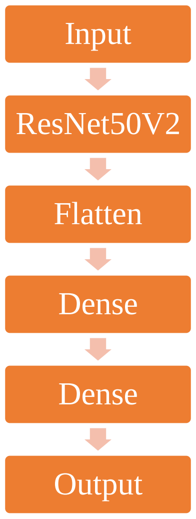


Figure 7. Experimented ResNet50V2 model.

4.3. InceptionV3 Model

Following preloaded InceptionV3 with pre-trained weights by ImageNet is a Dense neural network layer of 512 neurons with ReLU activation. Then images are classified by the final dense layer with softmax activation.

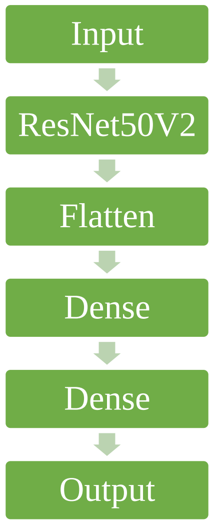


Figure 7. Experimented InceptionV3 model

5. Results and Discussion

5.1. Baseline CNN

Below are the results of the baseline CNN model. The model obtains a 79.56% test accuracy.

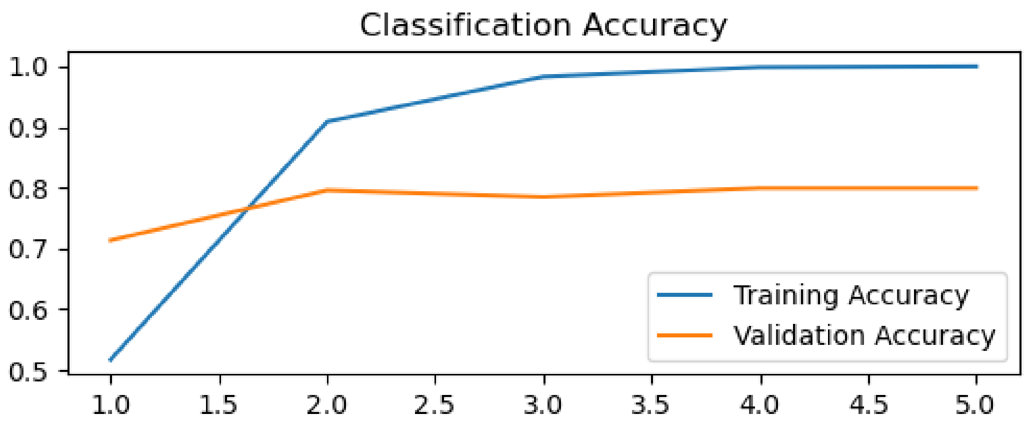


Figure 8. Baseline model accuracy

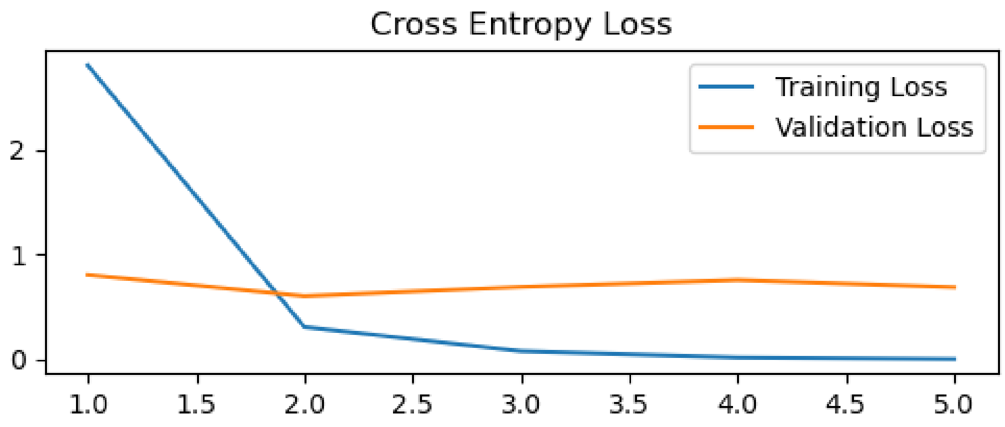


Figure 9. Baseline model loss

5.2. ResNet50V2

The results of training ResNet50V2 are displayed below. 93.61% accuracy was attained.

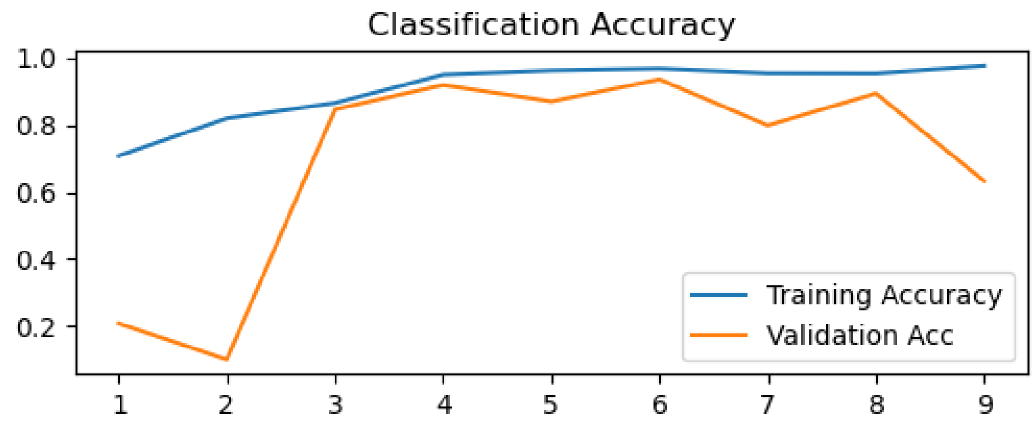


Figure 10. ResNet50V2 model accuracy

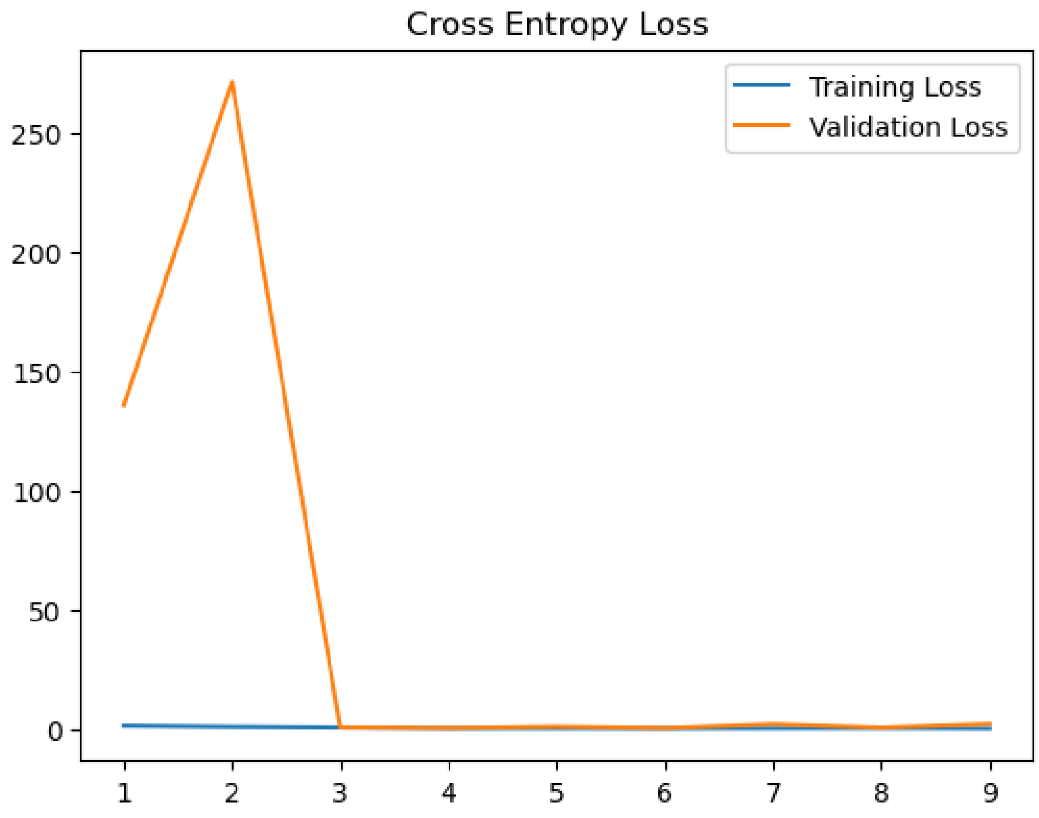


Figure 11. ResNet50V2 model loss

5.3. InceptionV3

The Inception training results are shown in the area below. The accuracy of the model, which is 98.72%, is the highest of all the models employed in this work.

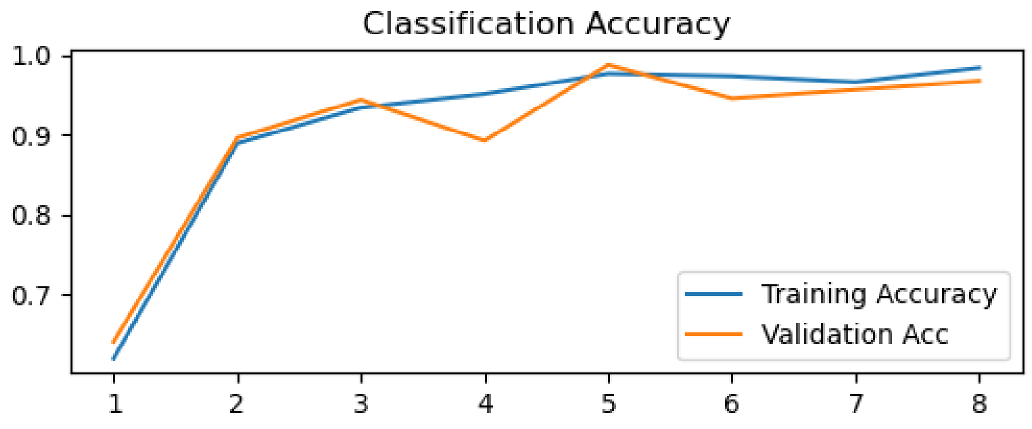


Figure 12. InceptionV3 accuracy

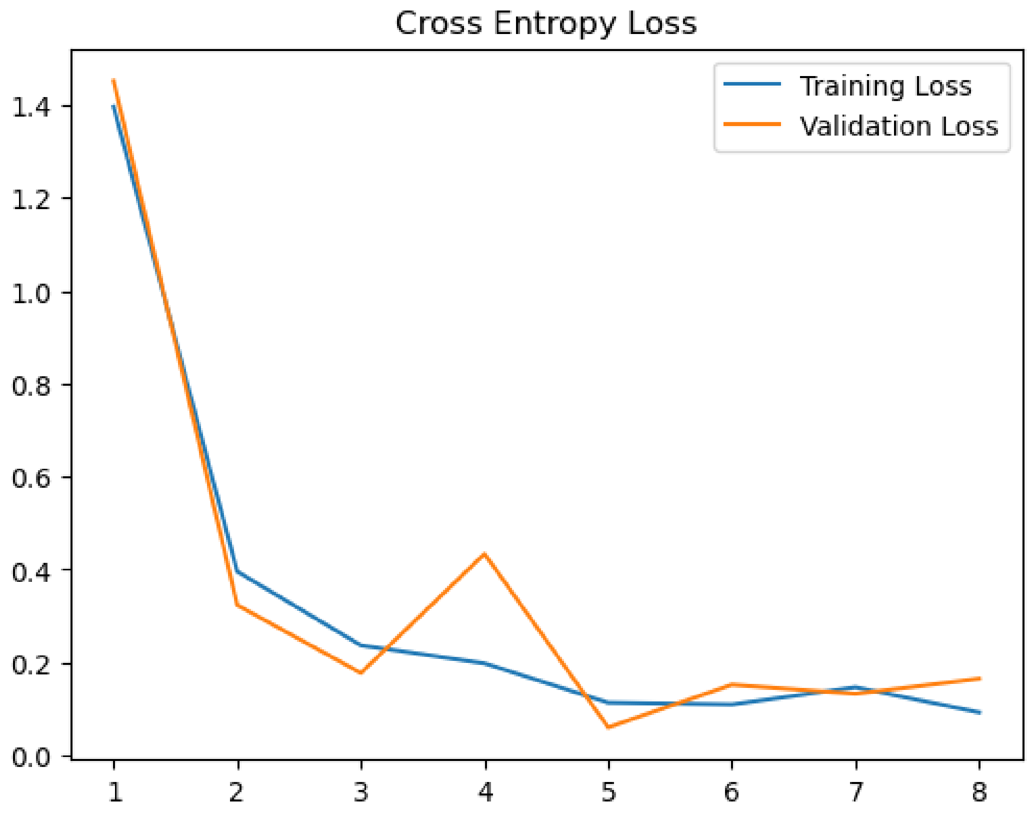


Figure 13. InceptionV3 loss

5.4. Comparison

The comparison table for all the tested models is shown in Table 2 as a summary of the findings. The baseline model performs poorly because of its simplicity. Results for ResNet50V2 and InceptionV3 models are typically excellent thank to transfer learning. However, ResNet50V2 uses higher computational costs (2s/step). Therefore, InceptionV3 outperforms all the models with the highest accuracy and decent computing resources (1s/step).

Model	Accuracy
Baseline	79.56%
ResNet50V2	93.61%
InceptionV3	98.72%

Table 2. Models' accuracy

6. Conclusion and Future Work

Finding the most accurate model to do the multi-class classification of American Sign Language is the goal of this study. For the purpose of ASL hand gesture identification, 3 distinct CNN models are compared in this paper. Overall, utilizing InceptionV3 with transfer learning produces the greatest result. The model can be highly useful for ASL translation these classification networks should be used, developed, and combined with temporal data and recurrent neural networks to learn word and sentence sequences.

References

American sign language (2021) *National Institute of Deafness and Other Communication Disorders*. U.S. Department of Health and Human Services. Available at: https://www.nidcd.nih.gov/health/american-sign-language (Accessed: January 13, 2023).

Brownlee, J. (2020) *Transfer Learning in Keras with Computer Vision Models*, *Machine Learning Mastery*. Available at: https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/ (Accessed: January 13, 2023).

Brownlee, J. (2020) *Use early stopping to halt the training of neural networks at the Right Time*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/ (Accessed: January 13, 2023).

Faisal, M. et al. (2020) "IHDS: Intelligent harvesting decision system for date fruit based on maturity stage using Deep Learning and Computer Vision," IEEE Access, 8, pp. 167985–167997. Available at: https://doi.org/10.1109/access.2020.3023894.

He, K. *et al.* (2016) "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [Preprint]. Available at: https://doi.org/10.1109/cvpr.2016.90.

Ji, Q. *et al.* (2019) "Optimized deep convolutional neural networks for identification of macular diseases from Optical Coherence Tomography images," *Algorithms*, 12(3), p. 51. Available at: https://doi.org/10.3390/a12030051.

Kaushik, A. (2020) *Understanding ResNet50 Architecture*, *OpenGenus IQ: Computing Expertise & Legacy*. OpenGenus IQ: Computing Expertise & Legacy. Available at: https://iq.opengenus.org/resnet50-architecture/ (Accessed: January 13, 2023).

Li, D., Hasanaj , E. and Li, S. (2020) *Baselines* , *CMU ML Blog*. Available at: https://blog.ml.cmu.edu/2020/08/31/3-baselines/ (Accessed: January 31, 2023).

Lin, C. *et al.* (2018) "Transfer learning based traffic sign recognition using inception-V3 model," *Periodica Polytechnica Transportation Engineering*, 47(3), pp. 242–250. Available at: https://doi.org/10.3311/pptr.11480.

Lum, K.Y., Goh, Y.H. and Lee, Y.B. (2020) "American Sign Language Recognition Based on MobileNetV2," *Advances in Science, Technology and Engineering Systems Journal*, 5(6), pp. 481–488. Available at: https://doi.org/10.25046/aj050657.

Ma, Y., Xu, T. and Kim, K. (2022) "Two-stream mixed convolutional neural network for American Sign Language recognition," *Sensors*, 22(16), p. 5959. Available at: https://doi.org/10.3390/s22165959.

Pigou, L. *et al.* (2015) "Sign language recognition using convolutional neural networks," *Computer Vision - ECCV 2014 Workshops*, pp. 572–578. Available at: https://doi.org/10.1007/978-3-319-16178-5_40.

Raj, B. (2020) *A simple guide to the versions of the inception network*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202 (Accessed: January 13, 2023).

Rastgoo, R., Kiani, K. and Escalera, S. (2021) "Sign language recognition: A deep survey," *Expert Systems with Applications*, 164, p. 113794. Available at: https://doi.org/10.1016/j.eswa.2020.113794.

ResNet-50: The basics and a quick tutorial (2022) *Datagen*. Available at: https://datagen.tech/guides/computer-vision/resnet-50/ (Accessed: January 13, 2023).

ResNet: The Basics and 3 Resnet Extensions (2022) *Datagen*. Available at: https://datagen.tech/guides/computer-vision/resnet/ (Accessed: January 13, 2023).

Sagnik, M. and Ramaprasad, P. (2019) "Comparative study of Convolutional Neural Networks," *International Journal of Electronics and Communication Engineering*, 6(8), pp. 18–21. Available at: https://doi.org/10.14445/23488549/ijece-v6i8p103.

Shin, H.-C. *et al.* (2016) "Deep convolutional neural networks for computer-aided detection: CNN Architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, 35(5), pp. 1285–1298. Available at: https://doi.org/10.1109/tmi.2016.2528162.

Sultana, F., Sufian, A. and Dutta, P. (2018) "Advancements in image classification using convolutional neural network," *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)* [Preprint]. Available at: https://doi.org/10.1109/icrcicn.2018.8718718.

Szegedy, C. *et al.* (2015) "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [Preprint]. Available at: https://doi.org/10.1109/cvpr.2015.7298594.

World Federation of the Deaf (2022) *WFD*. Available at: http://wfdeaf.org/ (Accessed: January 13, 2023).