Sign Language to Text Classification using Convolutional Neural Networks

Abstract

Sign language recognition is a computer vision task that aims to facilitate communication between people with hearing impairments and those who can hear. In this project, we propose to develop a deep learning-based sign language detection model that can recognize and classify sign language gestures in real-time. Convolutional Neural Networks (CNNs) are used to extract features from the input images and three architectures: ResNet-18, MobileNet-V3, and ShuffleNet-V2 have been compared. We train and evaluate our model on three publicly available sign language datasets that cover different facets of ASL. High accuracy levels on all datasets is achieved and the trade-off between accuracy and efficiency for each architecture is analyzed.The challenges and limitations of our approach and also use hyperparameter training and transfer learning to achieve the best results possible with the networks is also discussed. The study demonstrates the potential of deep nueral networks for sign language recognition and its applications for video captioning, virtual communication, and education.

1. Introduction

Hearing loss is a significant cause of disability, with about 19% of Canadian adults (4.6 million individuals) experiencing mild hearing loss in the speech-frequency range (0.5, 1, 2, and 4 kHz). Additionally, around 35% or 8.4 million individuals in Canada have some level of hearing loss in the high-frequency range (3, 4, 6, and 8 kHz), which is where age-related hearing loss typically begins. [17] The Deaf Community in Canada alone comprises over 357,000 individuals. There is a growing demand for interpreting services, currently provided by professional services through video remote human interpreting over high-speed internet connections. However, these services have limitations due to time of day and availability of interpreters. Automated solutions, such as neural networks powered by high-performance computing and computer vision techniques, can potentially address these limitations and cater to the increasing demand for interpreting services in today's digital age.

Based on the challenges stated above deep neural networks forgo handcrafting discriminatory features for classification and learn them automatically using the training data. [12]. Deep networks naturally integrate low/mid/high level features and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). In this report, we report how current state of art Convolutional Neural Network (CNN) architectures tackles these problems and how effectively these are architectures recognize the American Sign Language(ASL). The different architectures that are propose to tackle these problems are the ResNet-18 [6], ShuffleNet-v2 [20] and the MobileNet-v3 [7]. The architectures are trained from scratch as well as using pretrained weights. Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge contained in different but related source domains. Transfer learning is proven to achieve results faster and possibly better accuracy taking advantage of the known knowledge domains by the neural network architectures. [21] From the study, we find similar results for all datasets, where we achieve high accuracy with every network. The high accuracy results could imply that the images we're using are not very diverse and the neural networks even with fewer layers would work well with the datasets.

For the above problem, three static image datasets were chosen which constitutes different facets of ASL i.e. punctuation, numbers and alphabets. [4, 9, 14]. The datasets contain hand gesture images which are respectively cropped down to respectively of the hand gestures used in the ASL respectively provided by large number of users. These provide wide variability in the datasets being used and the variation in users and backgrounds helps in making the model to better learn the abstract features which in turns provide better testing accuracy. We also study in relation with the CNNs based on the hyperparameters such as batch size, learning rate and number of epochs. An ablated study choosing which hyperparameters work the best for a particular model using search based methods. The search methods find correct combination of hyperparameters gives a better accuracy and also shows what are the grievances with each of the hyperparameter.

1.1. Literature Review

A review of the literature on gesture recognition in images reveals that various methods have been used to tackle this problem. One approach, described in [3], involves using Hidden Markov Models (HMM) in combination with Bayesian Network Classifiers and Gaussian Tree Augmented Naive Bayes Classifier to recognize facial expressions from video sequences. The paper uses classical machine learning models and handcrafted techniques to train and achieve acceptable accuracy but still lacks the

recognition capability in high variation.

Akhter et al. [2] presented a method for recognizing ASL alphabets using PCA-based features combined with a Gabor filter and orientation-based hash code to represent different ASL alphabets. These features were then classified using an artificial neural network (ANN). The performance of this method was evaluated on a self-created dataset of 24 static gesture with an accuracy of 95.8% proving the use case can have a high potential to work well with neural networks.

Hussain et al. [8] used pretrained VGG-16 along with other preprocessing features such as HSV(Hue, Saturation, Value) skin algorithm to detect the hand area and then cropping the blob area. In another paper by *Molchanov et al.* [13] uses R3DCNN(Recursive 3D-Convolutional Nerual Networks) to train hand gestures using video datasets and provide almost real time detection and recognition with an accuracy of 88.4%.

The papers by *Akhter et al., Hussain et al. and Malchanov et al.* shows us a clear advantage of using deep neural networks working well with image classification problems. The idea based around deep neural networks learning high level abstractions of the dataset and then using these generalizations to unseen data has been seen work really well with image classification problems. [**?**, 7, 11, 20]

2. Methodology

2.1. Datasets

American Sign Language have quite a few datasets available and we chose 3 datasets based on different facets of ASL i.e. the punctuations, numbers and the alphabets. Some of the dataset metrics are added in the *Table* 5.

2.1.1 ASL Alphabet [4]:

This image dataset originally from NIDCD (National Institute on Deafness and Other Communication Disorders) [15] contains a total of 87,000 and 29 classes with 3000 images per class. There are 26 classes corresponding to each character in the English alphabet from A-Z and 3 classes for SPACE, DELETE and NOTHING. For modelling purposes A-Z classes were deleted and only data from the 3 remaining classes was used cutting down the dataset from 87,000 images to 9000 images. The motivation behind the creation of the dataset was to help in Sign Language Recognition.

2.1.2 Sign Language for Numbers [9]:

There are 16,500 images in this dataset and 11 classes with each class consisting of 1500 images. The 10 classes represent a number from 0-9 and there is class named unknown consisting of random images. Some of the images are shown in *figure 1*

Figure 1. Random images from Dataset 2

2.1.3 ASL Fingerspelling Images (RGB Depth) [14]

This data provided in the paper Spelling it out: Real-time asl fingerspelling recognition by Nicolas Pugeault et al. [16] contains 24 classes and more than 132,000 images corresponding to the characters in the English alphabets. This dataset was compiled from 5 different signers and each of their data is in different directory A-E providing some variability to the data. The dataset contains RGB images and depth data collected with the help of Microsoft Kinect. The problem with this dataset was the omission of classes of the j and z characters because of their reliance on movements. The motivation behind this dataset was to make versatile tools for sign language to text conversion.

Figure 2 shows the standard deviation of images by class in dataset 2 [9] and in dataset 3 [14]. Standard deviation measure the deviation of measured values or the data from its mean. When standard deviation is near zero, the measured values are near the mean and are converging. As can be seen from the images the standard deviations of the datasets are near 0, meaning that the dataset is simple. From *figure 2*, we conclude that the classes have low variability throughout the classes. Owing to these simple datasets, it was easy to train the models even reaching 100 percent accuracy for the Dataset 1. While, model was trained for high accuracies, it is not guaranteed to work well on real life complex data.

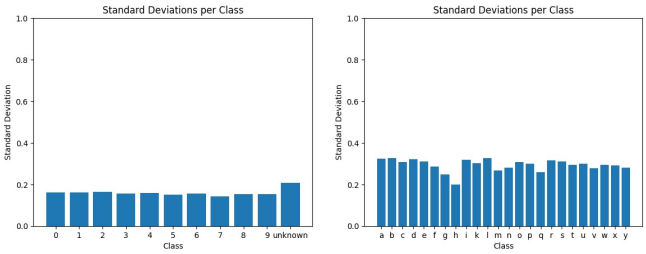


Figure 2. Standard Deviation on Dataset 2 and Dataset 3

For data preprocessing, we added certain data augmentation techniques to make the models generalize it better. Some of the transforms used were **Resize**: the images were resized such that their height and width to the required model input. **Center Crop**: the resized image was cropped from the center to obtain an image with height and width

equal to given input. **Normalization** This transformation was useful in normalizing the values of the pixels which helped in ensuring the input images had a similar distribution.

2.2. Architectures

2.2.1 ResNet 18

ResNet stands for Residual Network. ResNet 18 is a version of the ResNet model proposed in "Deep Residual Learning for Image Recognition" [6]. It is one of the 5 versions of ResNet models, which contains 18 layers. The other versions contain 34, 50, 101, and 152 layers respectively. Skip connections are introduced in Resnet18 to tackle the vanishing gradient problem in deep neural networks. Resnet18 layers also contain batch normalisation to reduce overfitting. This allows for much deeper networks to be trained effectively. Comparing VGG-19 with 19.6 billion FLOPS operations (multiply-adds), ResNet-18 has only 2.1 billion FLOPS, which makes it much easier and faster to train.

In a residual network, each layer receives two inputs: the output of the previous layer and a shortcut connection that bypasses one or more layers. The shortcut connection is added to the output of the previous layer before it is passed through the next layer. This allows gradients to flow directly from later layers to earlier layers, which can help prevent vanishing gradients. [6]

2.2.2 MobileNet-v3

MobileNet-v3 is a convolutional neural network (CNN) architecture that was introduced by Google researchers in 2019. It is a lightweight and efficient architecture designed for mobile and edge devices. The implementation of the MobileNetV3 architecture follows closely the original paper. It is customizable and offers different configurations for building Classification, Object Detection and Semantic Segmentation backbones. It was designed to follow a similar structure to MobileNet-v2 and the two share common building blocks. It encorporates squeeze and excitation blocks as backbone for the network.

The squeeze-and-excitation (SE) block is a neural network module that recalibrates feature maps by adaptively enhancing informative channels while suppressing less relevant ones. Introduced in 2018, SE blocks have been widely adopted in computer vision tasks, showing improved performance in tasks such as image classification, object detection, and semantic segmentation. SE blocks are lightweight, versatile, and easily integrated into existing architectures, making them a powerful tool for enhancing feature representations in deep learning models. [1]

2.2.3 ShuffleNet-v2

ShuffleNet V2 is a convolutional neural network (CNN) architecture that was introduced in 2018. It is an efficient and lightweight architecture designed for mobile devices. The paper "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design" by *Ningning Ma et al.* discusses how the neural network architecture design is mostly guided by the indirect metric of computation complexity, i.e., FLOPs.

ShuffleNet V2 builds upon the ideas introduced in ShuffleNet V1, but introduces several new techniques to improve performance while reducing computational complexity. These techniques include the use of channel shuffle operations, group convolutions, and a new efficient head design. The result is an architecture that achieves state-of-the-art performance on several benchmarks while being significantly smaller and faster than other architectures.

Model	Learnable parameters	FLOPS
ResNet18	11.69M	1.82B
MobileNetV3-small	2.97M	219M
ShuffleNet V2	9.11M	1.34B

Table 1. Comparison between ResNet18, MobileNetV3 and ShuffleNet V2

2.3. Optimization Algorithm

An optimization algorithm finds the value of the parameters (weights) that minimize the error when mapping inputs to outputs. An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rates. Thus, it helps in reducing the overall loss and improving accuracy. Choosing the right weights for the model is a daunting task, as a deep learning model generally consists of millions of parameters. It raises the need to choose a suitable optimization algorithm for your application. There are various optimizers which widely used for training deep neural networks. Some of them being Adam, RMSProp, AdaDelta and AdaGrad [5].

For our study we're using Adam optimizer [10] for training our networks. Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. ADAM can be considered as the combination of RMS-prop with momentum. So it combines both adaptive learning rates with momentum. It also records the exponentially weighted average of previous gradients similar to momentum. We can say

$$Adam = Momentum \oplus RMS - prop \oplus initadjustment \quad (1)$$

The update rule for Adam is given by

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}} + \epsilon}\hat{m}_t \tag{2}$$

$$m_t = \beta_1 m_t + (1 - \beta_1)\nabla_\theta f(\theta_t) \tag{3}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\nabla_\theta f(\theta_t^2) \tag{4}$$

The hyperparameters of the Adam optimizer are:

- η - The learning rate determines how fast the optimizer should learn. A smaller learning rate will take more time to converge but will be more accurate, while a larger learning rate will converge faster but may overshoot the minimum.
- β_1 : The exponential decay rate for the first moment estimates (default value is 0.9).
- β_2 : The exponential decay rate for the second-moment estimates (default value is 0.999).
- ϵ : A small constant used to prevent division by zero in the implementation of Adam (default value is 10^{-8}).

For calculating lossm we're using the cross entropy loss. Cross entropy loss is a commonly used loss function in machine learning and deep learning tasks, including classification problems. It measures the dissimilarity between predicted probabilities and actual labels. The cross entropy loss penalizes the model more for larger discrepancies between predicted probabilities and true labels, resulting in a higher loss value. Cross entropy loss is an important tool for optimizing model performance during the training process, allowing the model to learn from the discrepancies between predicted and true labels.

3. Results

3.1. Experiment Setup

The datasets were transformed using the PyTorch package. Dataset-specific mean and standard deviation values are produced to achieve adequate data normalisation. However, resource constraints were encountered during the first phases of training the models, and in order to overcome this obstacle, Google Colab Pro was chosen to utilise computational resources. For all models, hyperparameters are standardised to guarantee that they are comparable and that the search for optimal values can be undertaken evenly across various models and datasets.The model was trained using a range of learning rates from 0.01 to 0.001. The learning rate is an important hyperparameter in training neural networks that determines the step size at each iteration of the optimization process.In addition, the model was trained with different batch sizes of 16, 32, 64, and 128.To evaluate the performance of the model, accuracy computed for each combination of learning rate and batch size. These metrics provide a quantitative measure of how well the model is able to generalize to new data and classify correctly.The results of the evaluation were used to select the optimal combination of hyperparameters that produced the best performance on the validation set. A batch size of 64 and a learning rate of 0.001 are specified for all models. Standardize augmentations techniques as mentioned in section [18] are applied to all the datasets and in all the models

3.2. Main Results

After conducting a comparative analysis of the ShuffleNet, MobileNet, and ResNet-18 architectures, it was observed that ShuffleNet outperformed MobileNet in terms of accuracy, despite having a similar computational time. This suggests that the ShuffleNet architecture may be more efficient at utilizing computational resources compared to MobileNet.

On the other hand, ResNet-18, a deep neural network, required a greater computational time than the other architectures under consideration. This is likely due to the increased depth of the network and the resulting increase in the number of parameters that need to be trained. The three architectures were compared in order to get a better understanding of the architectures and their behaviour on a varied dataset.

As seen in Figure 3, the ResNet18 model works admirably on Dataset 1 (900 test pictures), attaining flawless test accuracy, precision, recall, and F1-score. On the other hand, the model's performance on Dataset 2 (1650 test pictures) and Dataset 3 (2784 test images) is somewhat lower, with a test accuracy of 93% and precision, recall, and F1-score of 94%. This implies that as the size of the dataset grows, accuracy suffers slightly, which may be mitigated by fine-tuning the hyperparameters. Figure 10 depicts the ResNet18 training and validation accuracies on datasets 1, 2, and 3. Figure 13 depicts the train and validation loss of ResNet18 on datasets 1, 2, and 3.

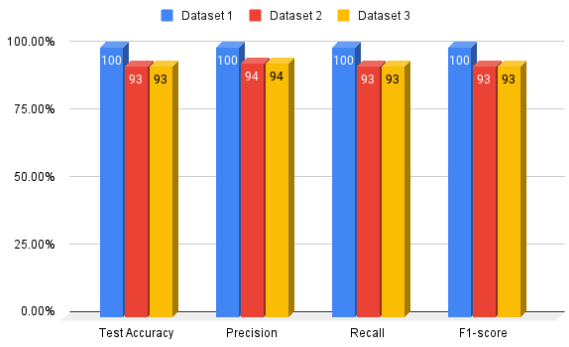


Figure 3. Classification report of ResNet18

Transfer learning is a machine learning technique where a model is trained on a large dataset and then fine-tuned on a smaller, related dataset to achieve better performance. This project aimed to evaluate the impact of pre-trained weights on the performance of two different neural network architectures - ShuffleNet and ResNet-18 - through transfer learning experiments. The objective was to compare the performance of both architectures before and after transfer learning was employed.

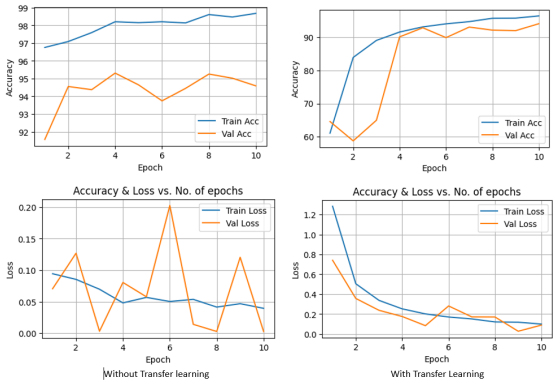


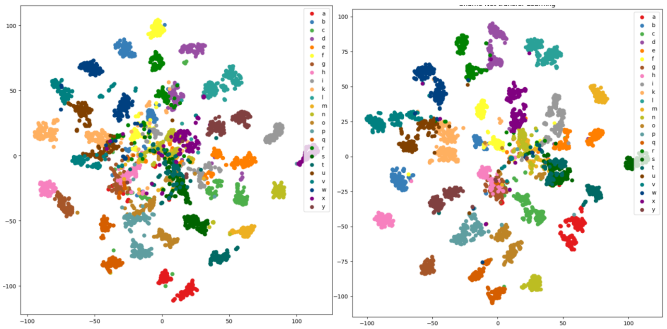
Figure 4. ResNet-18 Accuracy and Loss

In ShuffleNet, the results of the experiments showed a significant reduction in validation loss when transfer learning was utilized as compared to not using it.14 Additionally, the test accuracy improved from approximately 91% to 95% after employing transfer learning.

Regarding ResNet-18, the validation accuracy was observed to be low, and the validation loss fluctuated significantly without transfer learning. However, with transfer learning, the validation accuracy improved significantly,4 and the validation loss was reduced. These observations suggest that the utilization of pre-trained weights can improve the performance of ResNet-18 by enhancing its ability to extract relevant features from the data.

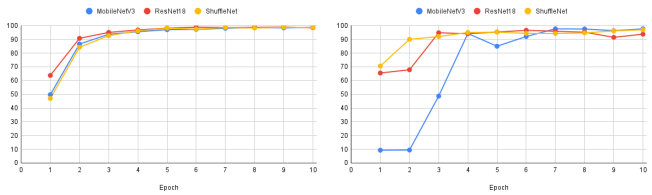
To assess the effectiveness of the transfer learning approach, the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique was utilized. The resulting t-SNE plot showed that with transfer learning, the data points were more accurately represented in clusters and were easily distinguishable from one another.5 This suggests that transfer learning facilitated the identification of relevant features and improved the overall performance of the neural network architectures

Figure 6 depicts the training accuracy on the training data. Training accuracy typically improves, with occasional swings in accuracy across epochs. In the instance of MobileNetV3, the starting accuracy in epoch 1 was 49.77%, suggesting that the model could only properly categorise around half of the training data at the outset. However,



(a) Without Trabsfer Learning (b) With Transfer Learning

Figure 5. T-SNE for ShuffleNet on Dataset 3



(a) Training Accuracy (b) Validation Accuracy

Figure 6. Metrics of all models on dataset 2

after 10 training epochs, the accuracy climbed to 98.97%, showing that the model accurately classified nearly 99% of the training data. Similarly, ResNet18 and ShuffleNet improve in accuracy with training, with ResNet18 attaining the greatest overall training accuracy of the three models. Training accuracy is merely one measure of a model's performance and does not always correspond to its performance on unseen data. The validation accuracy of three distinct models (MobileNetV3, ResNet18, and ShuffleNet) after ten training epochs. All models begin with relatively low validation accuracy, and as the number of epochs rises, the accuracy of each model steadily improves. ShuffleNet has the highest validation accuracy for the majority of epochs, followed by ResNet18 and MobileNetV3. ShuffleNet obtains the greatest validation accuracy of 97.03% in epoch 10, followed by ResNet18 at 93.82% and MobileNetV3 at 97.69%. Whereas, a smaller validation loss 13 suggests that the model is more capable of generalising to new data. We can observe that the validation loss lowers as the epochs advance for all three models. This shows that the models are growing better at generalising new data over time. Throughout all epochs, ShuffleNet has the lowest validation loss.

The stated in figure 7 test accuracies for the three models are 93.27%, 96.79%, and 97.3% for ResNet18, ShuffleNet, and MobileNetV3, respectively. ResNet18, ShuffleNet, and MobileNetV3 have recorded accuracy values of 94%, 97%, and 97%, respectively. ResNet18, ShuffleNet, and MobileNetV3 recall scores are 93%, 97%, and 97%,

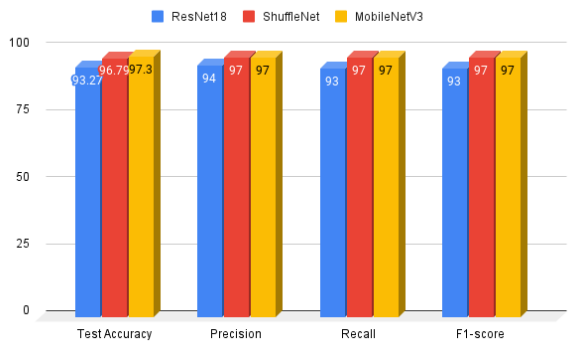
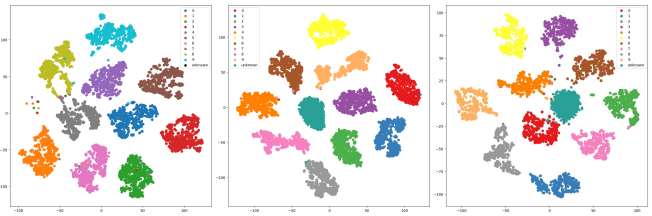


Figure 7. Classification report of all models on dataset 2



(a) ResNet18 (b) MobileNetV3 (c) ShuffleNet

Figure 8. T-SNE on Dataset 2 with all models

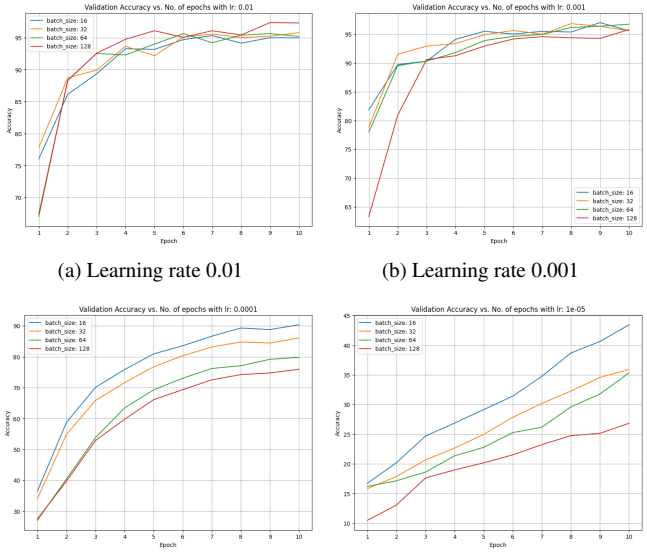
respectively. ResNet18, ShuffleNet, and MobileNetV3 have reported F1-scores of 93%, 97%, and 97%, respectively. Overall, the three models have similar accuracy, recall, and F1-score values.

T-distributed Stochastic Neighbor Embedding is a data visualisation and machine learning approach that lowers high-dimensional data to a lower-dimensional space while keeping the data's local structure, exposing patterns and correlations that were not obvious in the original data [19]. The t-SNE technique can be used as a measure of how well the data was trained on a CNN architecture.The t-SNE visualisations in the figure 8 show that the bulk of the datapoints in both datasets are categorised based on their attributes. MobileNetV3's categorization is more accurate than the previous two models, this is also supported by test accuracies presented in figure 6 and figure 7.

3.3. Ablative Study

Ablative studies are beneficial for comprehending complex systems and models as well as directing future development and improvement efforts. They may also be used to uncover possible flaws or vulnerabilities in a system, such as over-reliance on a certain component or vulnerability to certain sorts of errors or failures.

ShuffleNet-v2 was chosen for the ablative research because it achieves good accuracy while employing fewer parameters than other deep learning architectures like ResNet or VGG. ShuffleNet becomes more memory-efficient and faster to train per epoch. ShuffleNet was used on dataset 2. The learning rate and batch sizes were changed to see how changing hyperparameters affected the models and then observed the results. Models were trained with batch sizes ranging from 16 to 32, 64, and 128 with learning rates ranging from 0.01, 0.001, 0.0001, and 0.00001. As seen in Figure 9, validation accuracy decreases as the learning rate increases for all batch sizes.



(a) Learning rate 0.01 (b) Learning rate 0.001

(c) Learning rate 0.0001 (d) Learning rate le-05

Figure 9. Validation accuracy vs Learning rate

As the results can be improved with batch size depending on the complexity of the data. Validation accuracy decreases as the learning rate decreases for all batch sizes. Overfitting in terms of learning rate can result in a decrease in validation accuracy, which is not relevant in our instance because it occurs when the model is too complicated or the training data is inadequate. Because the optimizer makes higher steps in terms of learning rate during each iteration, it overshoots the minimum of the loss function, preventing it from finding the optimal solution and decreasing validation accuracy. The validation loss measurements are attested in Figure 12.

The amount of time required for each epoch varies with batch size. According to the research, a batch size of 16 takes an average epoch time of 56.2 seconds, but a batch size of 128 requires an average epoch time of 35.2 seconds. At a learning rate of 0.01 with a batch size of 128 highest validation accuracy was achieved of 97.364% after 9 epochs. After 10 epochs validation accuracy decreased by 0.06% due to overfitting. Observations can be seen in table 2. The change in time per epoch due to batch size is due constraints on the GPU architecture, due to transfer of batch to GPU compute engine.

References

[1] Channel Attention & Squeeze-and-Excitation Networks. https://blog.paperspace.com/channel-attention-squeeze-and-excitation-networks/, Sept. 2020. Accessed: 2023-04-16.

[2] Arif-Ul-Islam and Shamim Akhter. Orientation hashcode and articial neural network based combined approach to recognize sign language. In *2018 21st International Conference of Computer and Information Technology (ICCIT)*, pages 1–5, 2018.

[3] Ira Cohen, Nicu Sebe, Ashutosh Garg, Lawrence S. Chen, and Thomas S. Huang. Facial expression recognition from video sequences: temporal and static modeling. *Computer Vision and Image Understanding*, 91(1):160–187, 2003. Special Issue on Face Recognition.

[4] Grassknoted. Asl alphabet. `https://www.kaggle.com/grassknoted/asl-alphabet`, 2017. Accessed: 2022-04-11.

[5] Gousia Habib and Shaima Qureshi. Optimization and acceleration of convolutional neural networks: A survey. *Journal of King Saud University - Computer and Information Sciences*, 34(7):4244–4268, 2022.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] A. Howard, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[8] Soeb Hussain, Rupal Saxena, Xie Han, Jameel Ahmed Khan, and Hyunchul Shin. Hand gesture recognition using deep learning. In *2017 International SoC Design Conference (ISOCC)*, pages 48–49, 2017.

[9] Muhammad Khalid. Sign language for numbers. `https://www.kaggle.com/datasets/muhammadkhalid/sign-language-for-numbers`, 2021. Accessed: 2022-04-11.

[10] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[12] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Kari Pulli. Multi-sensor system for driver's hand-gesture recognition. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–8, 2015.

[13] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4207–4215, 2016.

[14] MrGeislinger. Asl rgb-d depth fingerspelling - spelling it out. `https://www.kaggle.com/datasets/mrgeislinger/asl-rgb-depth-fingerspelling-spelling-it-out`, 2021. Accessed: 2022-04-11.

[15] National Institute on Deafness and Other Communication Disorders. NIDCD website. `https://www.nidcd.nih.gov/`. Accessed: Apr. 16, 2023.

[16] Nicolas Pugeault and Richard Bowden. Spelling it out: Real-time asl fingerspelling recognition. pages 1114–1119, 11 2011.

[17] Pamela Ramage-Morin, Rex Banks, Dany Pineault, and Maha Atrach. Unperceived hearing loss among canadians aged 40 to 79. *Health reports*, 30:11–20, 08 2019.

[18] Pablo Ruiz. Understanding and visualizing ResNets. https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8, Apr. 2019. Accessed: 2023-04-17.

[19] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to Use t-SNE Effectively. *Distill*, 1(10):e2, Oct. 2016. Accessed: 2023-04-16.

[20] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 06 2018.

[21] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019.

4. Appendix

lr/batch size	16	32	64	128
0.01	55.5	39.8	36.1	35.4
0.001	56.7	40.4	36.1	35.5
0.0001	56.4	39.5	36.0	35.2
0.00001	56.2	39.6	35.5	35.1

Table 2. Average epoch time (seconds) with respect to batch sizes and learning rate

Name	Operator	H, W	IC, OC	K, S
C1	conv2d	224, 224	3,64	7, 2
C2	conv2d	56, 56	64,64	3, 1
C3	conv2d	56, 56	64,64	1, 1
C4	conv2d	56, 56	64,128	3, 2
C5	conv2d	56, 56	64,128	1, 2
C6	conv2d	28, 28	128,128	3, 1
C7	conv2d	28, 28	128,256	3, 2
C8	conv2d	28, 28	128,256	1, 2
C9	conv2d	14, 14	256,256	3, 1
C10	conv2d	14, 14	256,512	3, 2
C11	conv2d	14, 14	256,512	1, 2
C12	conv2d	7, 7	512,512	3, 1

Table 3. Configurations of all conv2d operators in ResNet-18. H/W denotes height and width, IC input channels, OC output channels, K kernel size, and S stride size. All ops use "SAME" padding.

Layer Type	Output Shape	Number of Parameters
Input	(224, 224, 3)	0
Conv2D	(112, 112, 16)	432
Bottleneck	(112, 112, 16)	1040
Bottleneck	(56, 56, 24)	5832
Bottleneck	(56, 56, 24)	8784
Bottleneck	(28, 28, 40)	15560
Bottleneck	(28, 28, 40)	23144
Bottleneck	(14, 14, 80)	42344
Bottleneck	(14, 14, 80)	74896
Bottleneck	(14, 14, 80)	74896
Bottleneck	(14, 14, 112)	161160
Bottleneck	(14, 14, 112)	180568
Bottleneck	(7,7, 160)	319184
Conv2D	(7, 7 ,960)	153600
AvgPool2D	(1, 1, 960)	0
Conv2D	(1, 1, 1280)	1230080
Conv2D	(1, 1, 1000)	1281000

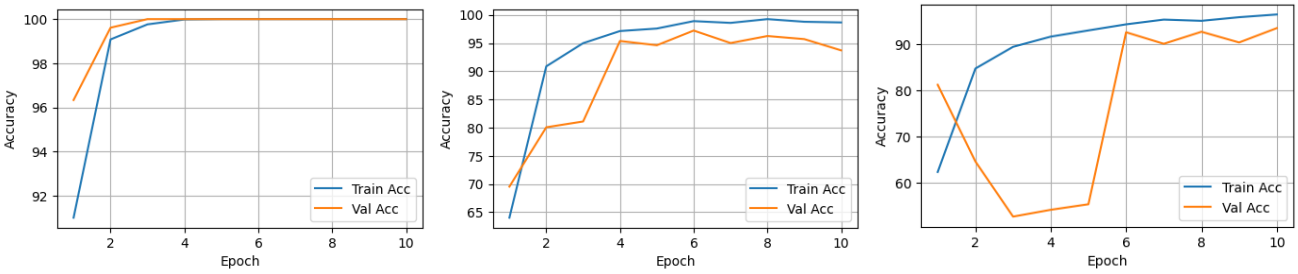
Table 4. MobileNet v3 architecture

Algorithm 2 Adam

Input: α: Stepsize
Input: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Input: $f(\theta)$: Stochastic objective function with parameters θ
Input: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second raw moment estimate)
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while

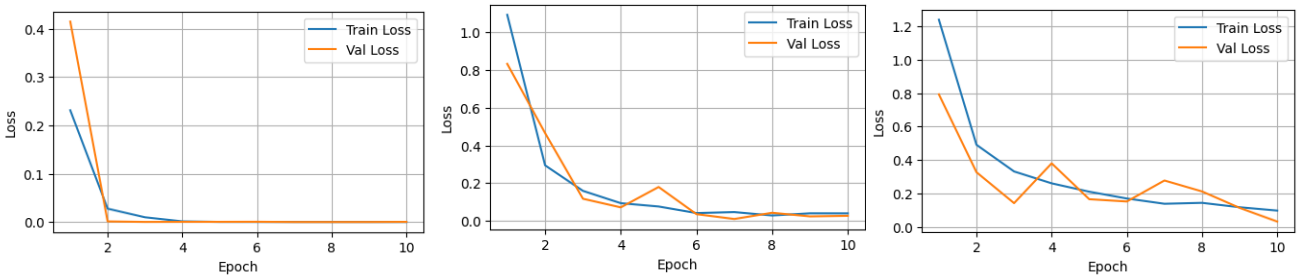
Parameters	Dataset 1	Dataset 2	Dataset 3
Total Classes	29	11	24
Reduced Classes	3	11	24
Total Images	87k	16.5k	132k
Reduced Images	9k	16.5k	30k
Image Size	200x200	400x400	variable
Format	.jpg	.jpg	.jpg

Table 5. Datasets



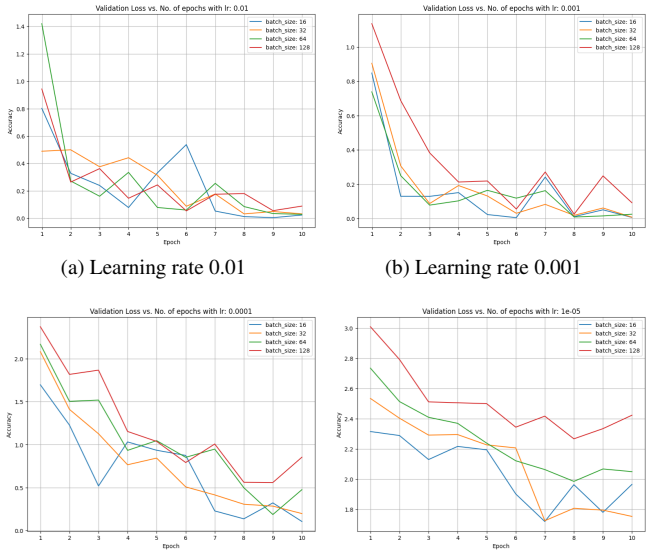
(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 10. Train and Validation Accuracy of ResNet18



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 11. Train and Validation Loss of ResNet18



(a) Learning rate 0.01 (b) Learning rate 0.001

(c) Learning rate 0.0001 (d) Learning rate le-05

Figure 12. Validation Loss vs Learning rate

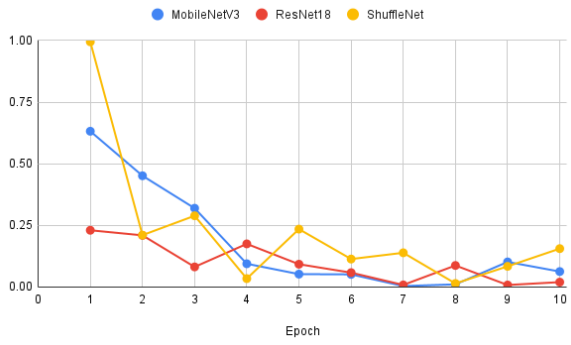


Figure 13. Validation Loss of all models on dataset 2

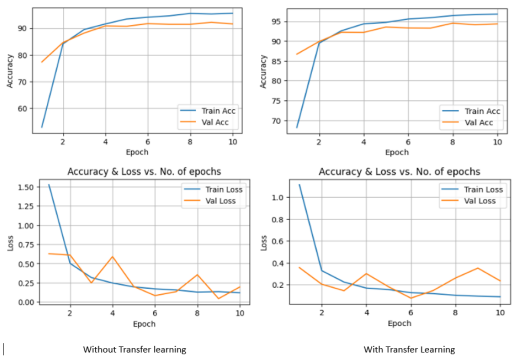


Figure 14. Shuffle Net Accuracy and Loss