Project Report : CS 7643

Adithi Minnasandran
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
aminnasandran3@gatech.edu

Jeremy Jang
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
jjang49@gatech.edu

Neel Mansukhani
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
nmansukhani6@gatech.edu

Timothy Cheung
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332
tcheung@gatech.edu

Abstract

Transfer learning has been a growing area in deep learning as an effective technique to increase performance of a model by transferring knowledge from one network to another. Furthermore, this technique can be used to decrease training resources required to obtain significant performance. The experiments displayed here aim to demonstrate how transfer learning can be applied to translating American Sign Language (ASL) letters to text to improve model performance. This is done by transferring knowledge from neural networks trained to classify ASL Digits, a task which is easier due to the simpler distinction between each class, the smaller quantity of classes, and the larger amount of data available. Experiments conducted include using a variety of pretraining data, and comparing the effects of transfer learning between simpler networks and state-of-the-art architectures. The results of these experiments all demonstrate that transfer learning is an excellent tool for teaching a neural network to classify ASL letters.

1. Introduction/Background/Motivation

As of 2022, roughly 70 million people worldwide are affected by hearing impairment or deafness. Furthermore, it is estimated that 1-2 million people across the globe use American Sign Language (ASL). Communication and social activity are important for healthy living, and thus using ML to create tools that assist people with these disabilities would be a worthwhile endeavor.

In this project, we applied transfer learning on the task of image classification for the ASL alphabet. We believe that pretraining networks on one ASL dataset will provide a good basis to learn on a second ASL dataset. Not long ago, ResNet[3] was state-of-the-art for image classification, and so we directly applied it in one of our experiments. Considering that state-of-the-art deep learning architectures such as ResNet are trained on a large dataset which may be dissimilar from our ASL datasets, we wanted to test the efficacy of transfer learning one ASL dataset onto another. Producing a robust pipeline to automatically translate ASL images to text would help users in their daily communication.

We used two main datasets, which we will deem in this paper as Digits[4] and Letters[1]. Depending on the context, we use either grayscale or 3-channel versions of these datasets. Digits contains ten classes, one for each digit 0-9, and there are 200 images of each class. The images portray hands depicting these classes with white backgrounds, and is collected from Turkey Ankara Ayranci Anadolu High School Students volunteers. Digits is used as our pretraining dataset in our experiments. Letters contains 29 classes, one for each letter A-Z as well as three additional for *space*, *delete*, and *nothing*, and there are 3000 images for each class. This dataset also contains images of hands depicting various classes and was collected by a contributor on Kaggle, but the background is not always completely clear, unlike Digits. We use Letters to measure model performance, and any mention of validation or test accuracy will refer to this dataset. We do not apply any pre- or post-processing of the images and results, except for rescaling and normalization to match the specific requirements of the network architectures we applied.

2. Approach

Our approach differs slightly from our original proposal. Initially, we proposed working on video hand gesture data using video-based network architectures. However, we

quickly found out how compute-intensive such tasks were, and thus we pivoted to an image-based project. At the end of the day, we are still working with ASL data, specifically a Digits dataset and a Letters dataset, with a focus on transfer learning from one ASL dataset to another. Our approach encompasses four overarching methodologies: a network that has been coded from scratch, an application of ResNet, an application of EfficientNet, and testing the transfer learning between datasets going both ways. We proceed to compare these different approaches and analyze the features learned by these models. There is nothing inherently "new" in our approach. Transfer learning on image datasets has been widely tried and studied across many domains[5][2]. However, as far as we are aware, we are the first to attempt transfer learning directly from one ASL dataset to another. We think this will be successful due to the similarity between the datasets, and thus the model would be able to learn good representations about one dataset, using the other.

2.1. Experiment 1 - Model from Scratch

For our first experiment, a neural network with 4 layers of convolution, leaky ReLU, batch normalization, and max-pooling, followed by a fully connected layer was trained from scratch on the ASL letter dataset. The same model was also pretrained on the ASL Digits dataset, and then trained on the ASL Letters dataset. All weights besides the ones from the fully connected layer from the network trained on Digits were used as the initialization when training on the Letters dataset, and none of the features were frozen. An equal number of images were used from both datasets (16 batches of 128 images), and all images were rescaled to be 64 by 64 pixels. An Adam optimizer, and cross entropy loss were used for both networks for 20 epochs. The goal of this experiment was to keep as many factors as possible constant between the pretrained model, and the one trained from scratch. With many factors remaining constant, it would become easier to determine if transfer learning played a large part in improving performance. This experiment was done using a less than state-of-the-art convolutional neural network. The purpose of this was so that the efficacy of transfer learning could be seen at multiple levels of strength of networks. The first attempt at a weaker model was done using one or two layers, but this created an issue of being too weak of a model for the task at hand, providing abysmal accuracy scores and poor feature representation for both datasets. So, rather than trying transfer learning on an incredibly weak model, it made more sense to test it on a model that while still not being state-of-the-art, can obtain decent feature representations.

2.2. Experiment 2 - ResNet

In our second experiment, we applied the ResNet18 architecture[3] in several ways. First, we trained the network from scratch on the ASL Letters dataset, allowing all parameters to be updated. Second, we pretrained the network on the ASL Digits dataset, then transfer learned the network on ASL Letters, allowing only the final fully connected layer to be updated in this step. Third, we used a network that has been pretrained on the ImageNet 1K dataset, then transfer learned the network on ASL Letters, allowing only the final fully connected layer to be updated. Fourth, we used a network that has been pretrained on the ImageNet1K dataset, further pretrained the network on all parameters on the ASL Digits dataset, then transfer learned the network on ASL Letters, allowing only the final fully connected layer to be updated.

As in Experiment 1, we trained over 20 epochs, each with 16 batches of 128 images, using an Adam optimizer and cross entropy loss. However, we used 3-channel images (instead of grayscale) that were rescaled to 224 by 224 pixels and normalized according to ResNet specifications. Also, like Experiment 1, we modified only the final fully connected layer. However, we froze all parameters except for the final fully connected layer in the transfer learning step. Additionally, we used Gradcam[6] to help us understand how different models focused on different parts of the images.

2.3. Experiment 3 - EfficientNet

Next, we applied EfficientNet[8], introduced by Mingxing Tan et al in 2019. EfficientNet is derived from MobileNet using AutoML approach[7]. We made use of EfficientNet in two different ways - with and without pretrained weights.

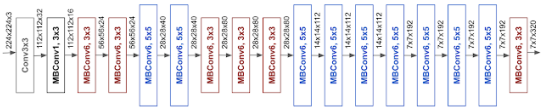


Figure 1. Architecture of EfficientNet, [7]

For the control group, we used a pretrained EfficientNet-b0 (the baseline model of EfficientNet) followed by fully connected layers with ReLU activation functions for the 29 classes of Letters. Like in previous experiments, the model was trained using 16 batches of 128 images. For the experimental group, we used a pretrained EfficientNet to train on the Digits dataset. This model is also followed by fully connected layer with same architecture of previous step but for the 10 classes. The fully connected layer of trained model is then replaced with fully connected layer with same architecture for 29 class classification while all other weights are subject to freeze and performed same training and validation method used as control group. Then, same procedure was conducted again with same architecture of EfficientNet,

but the existing weight came from EfficentNet that is pre-trained from other images were resetted with random initial weights.

2.4. Experiment 4 - Changing Direction

Considering the large size of deep architectures and the high computation time of training such networks, we decided for this experiment that it would be worthwhile to try transfer learning with a shallow network. Hence, a shallow convolutional neural network was built using PyTorch. It was separately trained on the Digits dataset and on the Letters dataset. The results from training the model from scratch on these datasets were considered the baseline. The next step was to transfer learn the Letters dataset on the model that was trained on Digits, and transfer learning the Digits data on the model that was trained on Letters. The hyperparameters including learning rate, number of epochs, loss criterion, and optimizer were kept the same so that the experiment's results would solely indicate how effective transfer learning was. Transfer learning speeds up training and can achieve higher accuracy than training from scratch. The datasets used were an ASL Letters dataset containing 3000 images per class and a total of 29 classes, and a Digits dataset with around 200 images per class and a total of 10 classes. For this experiment, a subset of the Letters data and all the data from the Digits dataset was used for training. As in the previous experiments, 16 batches of 128 images were used for training.

3. Experiments and Results

We measured success using prediction accuracy on the validation dataset, and the speed (number of epochs) at which the model achieves good validation accuracy.

3.1. Experiment 1 - Model from Scratch

The test accuracy score, as well as the training curves for validation accuracy were used as measures of success in these experiments. When trained from scratch on the ASL Letters dataset, the model had a subpar accuracy score of 69.126%, but saw a significant improvement when pretrained on the Digits dataset. When using transfer learning in this experiment from the model pretrained on the Digits dataset with a test accuracy of 90.072%, the model's performance on the harder task with the ASL Letters dataset was a comparable 85.747%. This created a notable 16.621% increase in accuracy when using transfer learning. On top of this, when looking at the training curves in figure 1, it is clear that the feature representation from the Digits dataset helped the pre-trained network learn faster, as it has a significantly higher accuracy score in the first epoch, which continues to improve and score higher than the model trained from scratch.

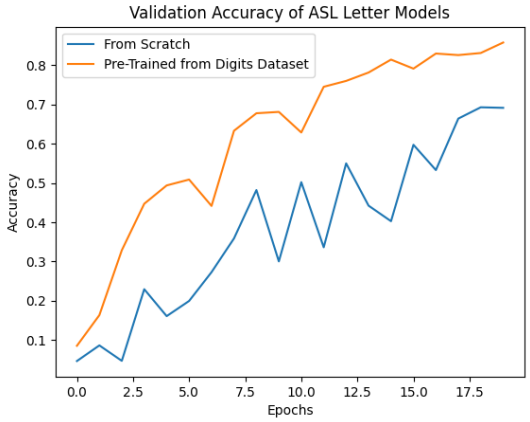


Figure 2. Validation accuracy for Experiment 1

3.2. Experiment 2 - ResNet

With the ResNet18 architecture, we trained a model from scratch in order to establish a baseline. It achieved the best validation accuracy, likely because it was the only model where all parameters were being updated. Figure 3 shows the validation accuracies of the four models trained in the experiment.

For the network pretrained on only the Digits data, the model was only able to achieve a validation accuracy of about 38.477%. This is due to the network not having enough pretraining data to learn good representations, since the Digits dataset is small compared to the Letters dataset. As a result, the model sometimes focuses on the wrong part of the image, as shown in Figure 4 (right), where Gradcam shows that the model was focused on the edges of the ceiling rather than the hand.

With a network pretrained on ImageNet1K, the model performs very well, achieving performance slightly under the baseline despite only updating final layer parameters. Gradcam shows the model focusing on all salient portions of the hand.

With a network pretrained on both ImageNet1K and Digits, we surprisingly saw worse performance than pretrained on just ImageNet1K. It could be that the model "forgot" things it had learned before as it was being trained on Digits. Gradcam shows that the model ignores parts of the hand from the palm down, which could be a behaviour it learned while training on Digits. This showcases how training a network on new data does not generalize it further if the old data (or some form of it) is not also present.

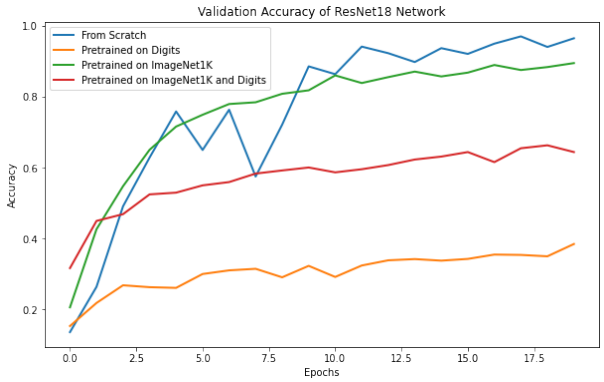


Figure 3. Model performance of applications of ResNet18

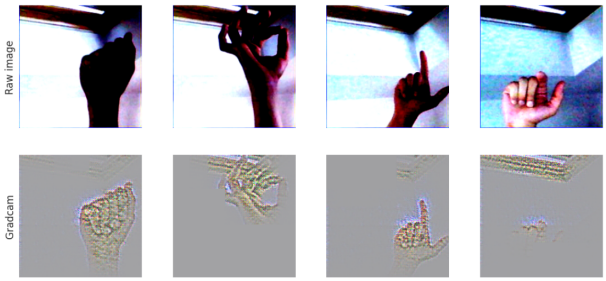


Figure 4. Raw image and Gradcam of "Pretrained on Digits"

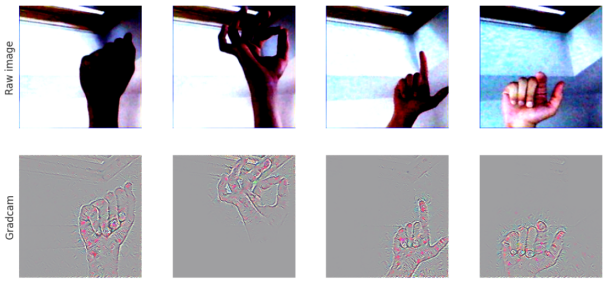


Figure 5. Raw image and Gradcam of "Pretrained on ImageNet1K"

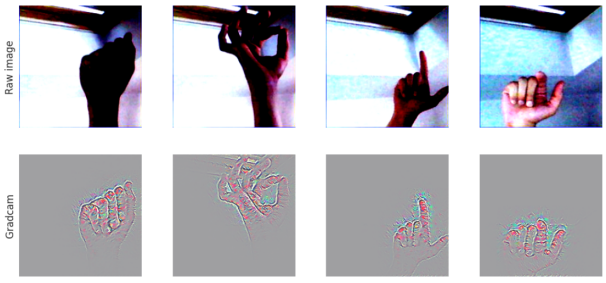


Figure 6. Raw image and Gradcam of "Pretrained on ImageNet1K and Digits"

3.3. Experiment 3 - EfficientNet

In this experiment, we made use of EfficientNet. For the models using weights presetted from EfficientNet packages, the training and validation accuracies for two models, trained from scratch and transferred learning from digit dataset, showed rapid convergence to 100% accuracy in 3 epochs. The accuracy of model trained from scratch showed smoother and more stable convergence than the model that had transferred convolution layers but the validation accuracy of the model with transferred learning from Digits dataset shows higher performance in the iteration across most of the point in time. Over 20 epochs, both model performed better than 95% while the model with transferred convolutional layers scored higher at all epochs except for third epoch. At the end of training, epoch 20, the model trained from scratch scored 95.9% accuracy while the model with learning transferred scored 97.33% accuracy.

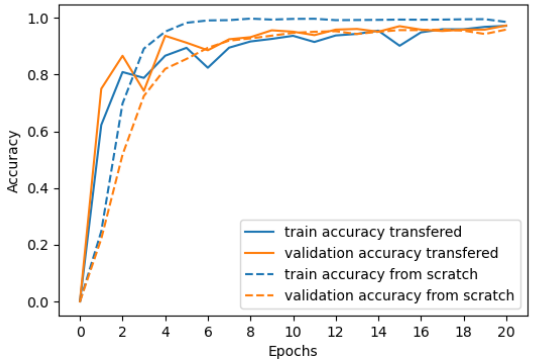


Figure 7. Training and validation accuracy for model trained from scratch vs model using transferred learning using pretrained EfficientNet weights.

The other test using randomly resetted weights for EfficientNet performed worse than the previous experiment as expected. The model that is trained from scratch using EfficientNet with randomly initialized weights seems to be learning the features slowly as the training accuracy gradually, almost linearly increases until epoch 10 with 85.286%, then the learning and increment of the accuracy slows down and scored 92.786% by epoch 20. The validation accuracy of the same model scored around 3.5% (which is about 1/29 where 29 is the number of classes) until 14th epoch, then slowly picked up the features, and ended with 67.891% accuracy by epoch 20. The model that has been pretrained using Digits dataset from EfficientNet with randomly initialized wieghts performed much better than the model trained from scratch. Both, training and validation accuracies of this model picked up the featured after 3 epoches. At 20th epoch, training accuracy reached 97.734% while validation accuracy reached 87.188%.

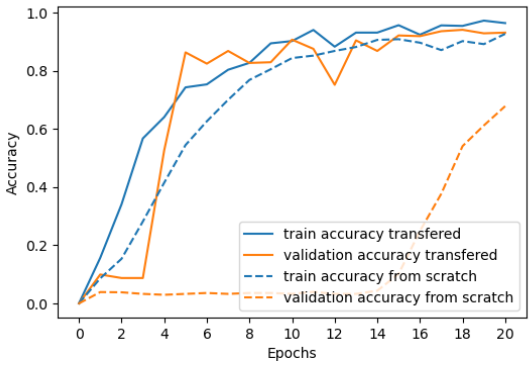


Figure 8. Training and validation accuracy for model trained from scratch vs model using transferred learning using randomly initialized EfficientNet weights

3.4. Experiment 4 - Changing Direction

We measured success by the learning curves of training and validation data and comparing the accuracy curves of the models. This experiment can be broken into two parts. Part one being training a model from scratch on the Digits dataset, training the same model from scratch on the Letters dataset, and then training the Letters dataset on the Digits model. Part two of the experiment was training a model on the Letters dataset, training the same model on the Digits dataset, and then training the Digits data on the Letters model.

The learning curves for part 1 show that in the case of transfer learning, the initial and final training losses are lower, as seen in Figure 9. With transfer learning Letters data on the Digits model, I started to notice overfitting around epoch 10 i.e., the validation loss remained constant while the training loss kept decreasing, hence part 1 needed to be terminated at 8 epochs. However, the final accuracy of the transfer learned Letters model was approximately the same after 8 epochs as the final accuracy of the model trained from scratch on the Letters data for 14 epochs. This shows that transfer learning made learning go faster (Figure 10), and as a result overfit the data at an earlier epoch.
For part 2, apart from lower losses in transfer learning (Figure 11), there was also a marked increase in accuracy after 14 epochs (Figure 12).

This experiment was a success because in both part 1 and part 2, transfer learning performed better in terms of accuracy as well as loss. The final accuracy for the baseline Digits model was 88.35%, whereas the final accuracy when Digits was transfer learned on Letters was 95.15% (part 2 of experiment). Similarly, the accuracy of the baseline Letters model after 8 epochs was 82.73%, while the final accuracy of transfer learning Letters on the Digits model was 88.53%

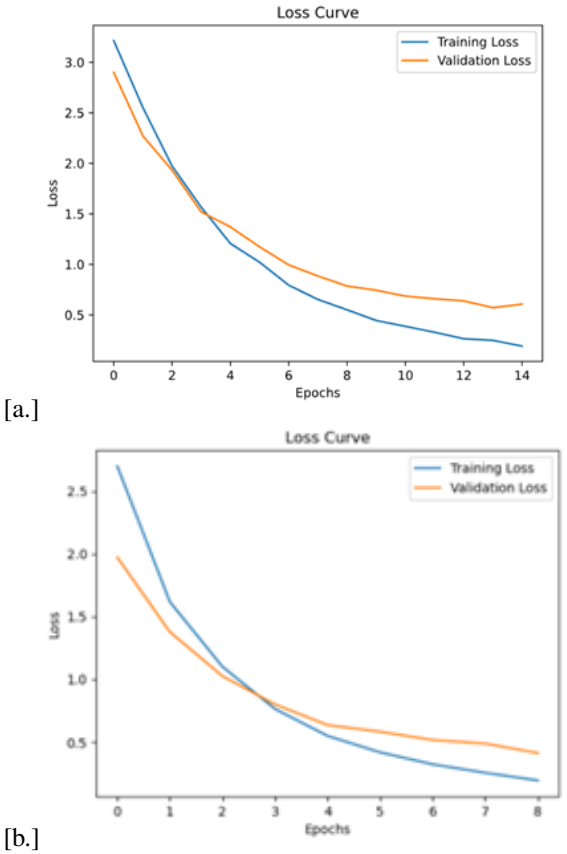


Figure 9. Exp part 1 learning curves. Training model on Letters (a). Transfer learning Letters on digits (b).

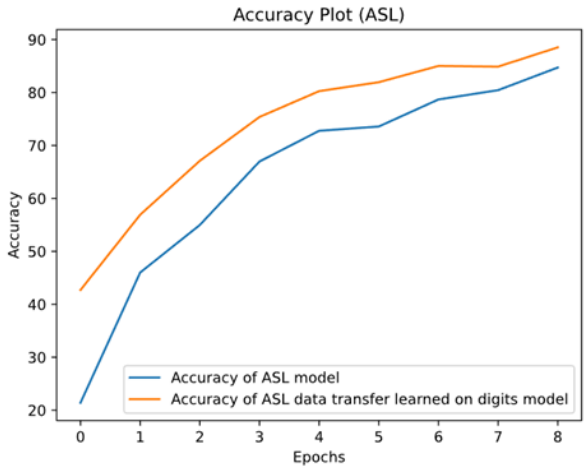
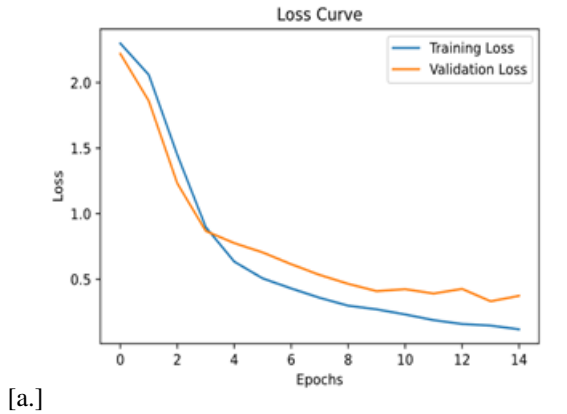


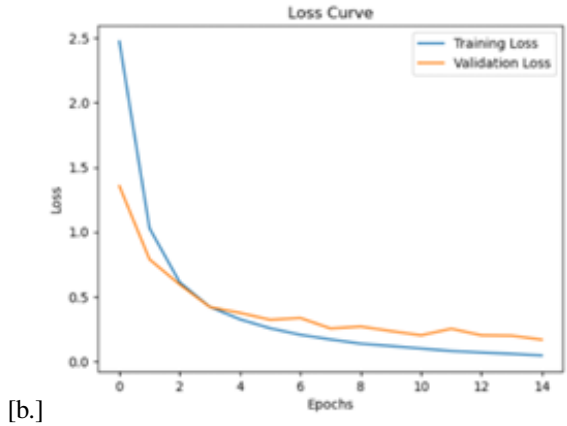
Figure 10. Part 1 Accuracies.

(part 1 of experiment).

Key design decisions for the experiment are as follows. After doing some experimentation, we landed on resizing the images to 64x64 to make the training go faster. The original size of the images was 224x224 which made training take several hours. We wanted to give more importance to



[a.]



[b.]

Figure 11. Exp part 2 learning curves. Training model on digits (a). Transfer learning digits on Letters (b).

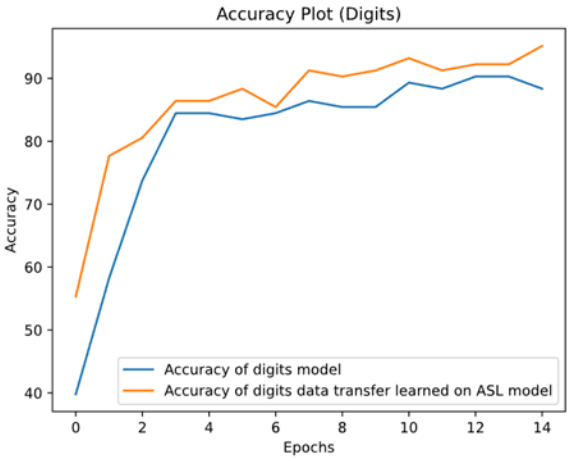


Figure 12. Part 2 Accuracies.

tuning hyperparameters and modifying layers of the model, so we made the decision to downsize the images. The optimal learning rate for the model was found to be 0.0005. This value was optimal to avoid overfitting and underfitting in both parts of the experiment. The loss criterion chosen was cross entropy, the optimizer was Adam, and the number of epochs chosen for part 1 was 8, and the number of epochs chosen for part 2 was 14. The hyperparameters and the model were kept the same for each part of experiment 4.

4. Discussion

The experiments show that transfer learning improved the final accuracy and losses of the models. It was found in some instances that transfer learning also increased the speed of training by producing the same accuracy in fewer epochs compared to training a model from scratch. In experiment 1, there was a 16.621% increase in accuracy when the Letters data was transfer learned on to a model trained on Digits. In experiment 2, when the ASL data was transfer learned on the pretrained ImageNet1K ResNet18 model, the performance was slightly under the baseline, despite only updating the final layers of the model. This indicates that transfer learning reduces computation time on a new task since fewer parameters needed to be trained to achieve similar results. In experiment 3, which used EfficientNet, it was noted that transfer learning provided a marginally higher final accuracy as compared to training the model from scratch. However, it was also seen that the model trained from scratch showed a smoother convergence compared to transfer learning. In experiment 4, it was seen in both part 1 and part 2 of the experiment that transfer learning provided higher final accuracies on both the Digits as well as Letters transfer learning tasks. It was also found that transfer learning provided similar results in fewer epochs on the task of training on Letters data.

5. Conclusion

In conclusion, it was found that transfer learning provided higher accuracies and faster convergence when compared to training from scratch. A key lesson from these experiments is that transfer learning's performance depends on the model's architecture and the data that the model is pretrained on. In experiment 4, since the initial model had too few parameters (specifically an insufficient number of convolutional layers) for the Letters dataset, there was overfitting at just 10 epochs. However, the same model gave significant improvement in accuracy on the task of transfer learning Digits, since the model had an adequate number of parameters for the Digits task. Further, since the Digits dataset has fewer classes than the Letters dataset, transfer learning the Letters data on a model trained on Digits is only effective if the convolutional layers of the model are not frozen since the Letters data requires different/more feature to be extracted by convolutional layers than the Digits data. Transfer learning the Letters data on ResNet18 (with all but the final layers frozen) trained on ImageNet1K produced results similar to training the model from scratch as observed in experiment 2.

6. Work Division

Details of contributions are provided in Table 1.

References

[1] Akash. Kaggle asl alphabet dataset. 1

[2] Sasank Chilamkurthy. Transfer learning for computer vision tutorial. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015. 1, 2

[4] Arda Mavi. A new dataset and proposed convolutional neural network architecture for classification of american sign language digits, 2020. 1

[5] Jo Plested and Tom Gedeon. Deep transfer learning for image classification: a survey. 2022. 2

[6] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. 2019. 2

[7] Mingxing Tan. Efficientnet: Improving accuracy and efficiency through automl and model scaling, May 2019. 2

[8] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019. 2

Student Name	Contributed Aspects	Details
Neel Mansukhani	Experiment 1	Built a CNN from scratch using Pytorch. Completed corresponding sections of the report and Abstract. Gathered code and data from all 4 experiments, generated a common .yaml file, and put files in a linux runnable form.
Timothy Cheung	Experiment 2	Worked on experiments with the ResNet18 architechture and Gradcam analysis. Completed Sections 1, 2, and sections pertaining to Experiment 2 of report.
Jeremy Jang	Experiment 3	Worked on experiments with the EfficientNet architechture. Setup the References section and sections pertaining to Experiment 3 of report.
Adithi Minnasandran	Experiment 4	Built a CNN from scratch and worked on transfer learning Letters data on Digits and vice versa. Completed sections 4, 5, and sections corresponding to Experiment 4 of report.

Table 1. Contributions of team members.