DAEN 429 ASL Recognition Project Report

1. Experimental Setup and Preprocessing Decisions

For this phase of the project, I utilized the required ASL Alphabet dataset, which consists of 87,000 images spanning 29 classes. My first critical decision involved the data splitting strategy. To ensure strict reproducibility and prevent data leakage between experimental runs, I implemented a stratified 80/20 split for training and validation, enforcing a fixed random seed of 429. This stratification was essential to maintain class balance across both subsets, ensuring that the validation metrics accurately reflected the model's performance across all letters rather than being skewed by class imbalances. All input images were normalized using the standard ImageNet mean and standard deviation to align the input distribution with the pre-trained ResNet-18 weights, a necessary step to maximize the efficacy of transfer learning.

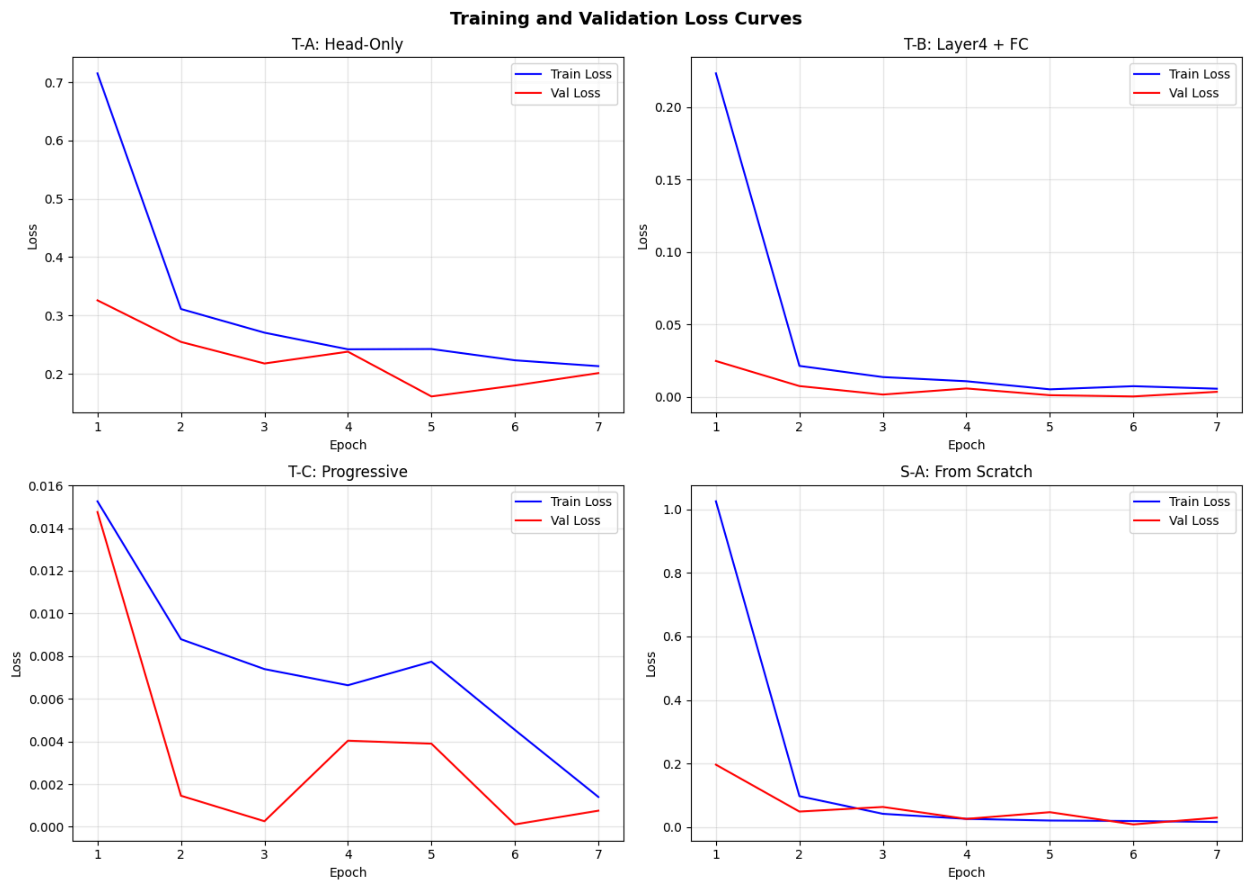
I made a data augmentation block that would improve generalization without destroying the meaning of the hand signs. I ultimately selected a combination of RandomHorizontalFlip with a probability of 0.3, RandomRotation of 10 degrees, and ColorJitter to vary brightness and contrast. I specifically chose these transformations to simulate the variance found in real-world photography, such as differing camera angles and lighting conditions. During my experimentation, I also attempted to include Gaussian Blur, hypothesizing that it might help the model handle lower-quality inputs. However, testing revealed that blurring degraded performance on my custom test set. I reasoned that ASL recognition relies heavily on high-frequency features like the sharp edges of fingers to distinguish between similar signs ('M' vs 'N'), and blurring could be hurting this critical signal. Consequently, I rejected Gaussian Blur from the final pipeline.

In terms of hyperparameter optimization, I standardized on a batch size of 128 for all experiments. This decision was driven by a trade-off between computational efficiency and gradient stability; a batch size of 128 maximized the GPU throughput on the available hardware while providing a sufficiently stable estimate of the gradient to allow for higher learning rates. I then conducted a grid search over learning rates of 0.01, 0.001, and 0.0001 for each architecture. This tuning proved crucial because the optimization landscape varied significantly between configurations. I found that the Head-Only configuration (T-A) required a relatively high learning rate of 0.01 to drive the randomly initialized fully connected weights to convergence. Conversely, the Deep Unfreezing strategies (T-B and T-C) required a much more conservative learning rate of 0.0001. Using a higher rate for these models resulted in "catastrophic forgetting," where aggressive gradient updates destroyed the valuable feature detectors pre-learned by ResNet on ImageNet.

2. Ablation Study: Freezing Policies

The baseline configuration, T-A (Head-Only), kept the entire backbone frozen and trained only the final classification layer. T-B (Last Block) unfroze Layer 4 along with the head, allowing the model to adapt high-level spatial features to the ASL domain. T-C (Progressive) utilized a gradual unfreezing strategy, initializing with the best weights from T-B and subsequently unfreezing Layer 3 to stabilize the training of deeper features. Finally, S-A (From Scratch) served as a control, training a full ResNet-18 initialized with random weights to quantify the specific benefit of transfer learning.

The training dynamics, visualized in Figure 1, illustrate the distinct convergence behaviors of these models. T-A converged rapidly but plateaued at a slightly higher loss, due to the frozen feature extractor's inability to adapt to the specific nuances of sign language. In contrast, the S-A model began with high loss but converged reliably, proving that the 87,000-image dataset is sufficiently large to train a deep CNN from scratch, albeit at a higher computational cost than fine-tuning. The fine-tuned models (T-B and T-C) achieved the lowest final validation loss, demonstrating superior optimization stability compared to the scratch model.

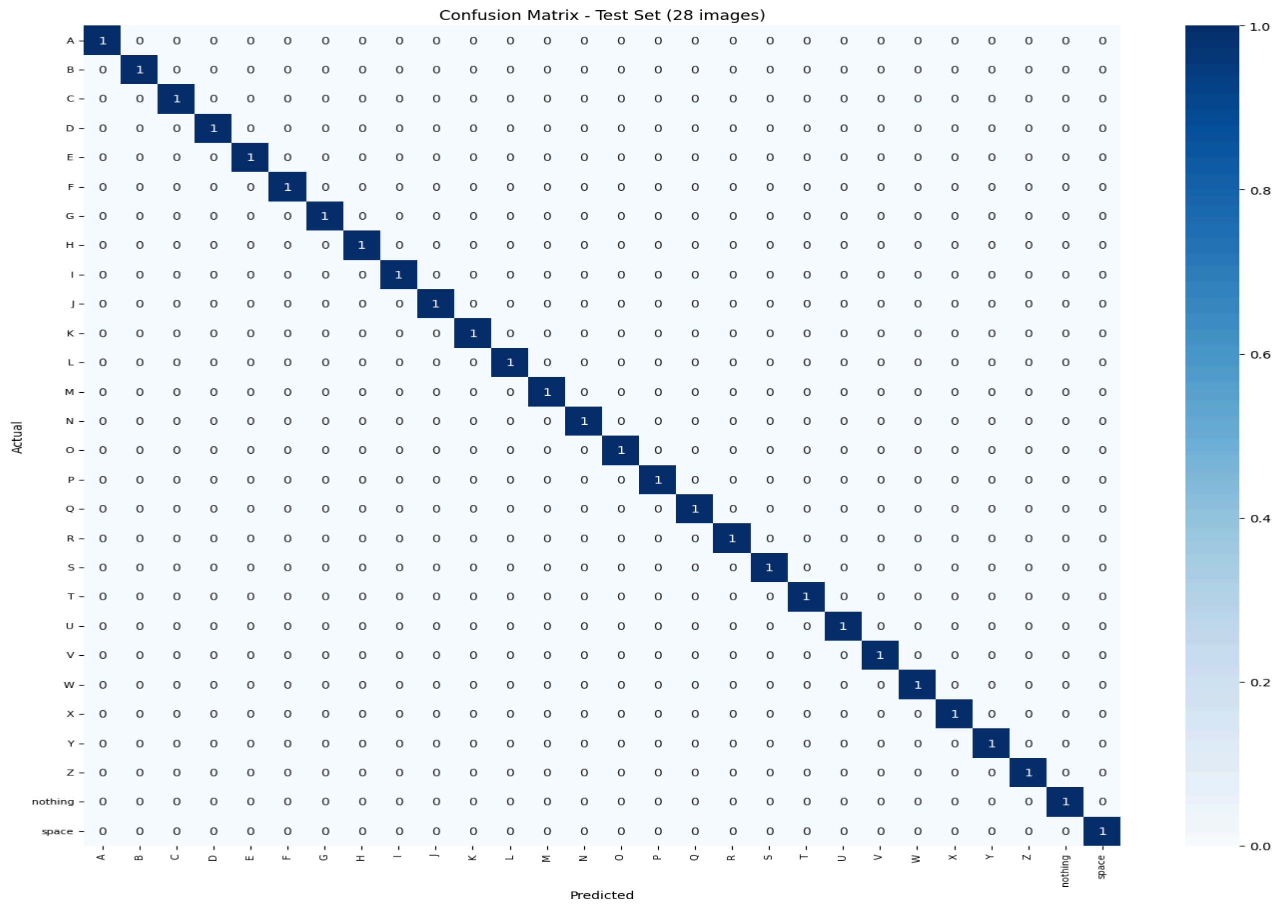


The comparative analysis on the validation set, summarized in the table below, identified the Progressive Fine-Tuning model (T-C) as the superior candidate. Empirically, T-C achieved the highest raw metrics across the board, with an accuracy of 0.99994 compared to T-B's 0.99988. While the numerical difference between the fine-tuned models is marginal, T-C's slight edge validates the hypothesis that deeper fine-tuning yields better feature adaptation. Furthermore, T-C theoretically possesses a more robust feature hierarchy due to the additional fine-tuning of Layer 3. This deeper adaptation suggests a higher likelihood of generalizing to unseen data distributions, such as the custom test set, making T-C the clear choice for final testing.

Config	Parameters	Learning Rate	Batch Size	Accuracy	Macro-F1
T-A	14,877	0.01	128	0.945	0.944985482
T-B	8,408,605	1e-4	128	0.999885057	0.999885057
T-C	10,508,317	1e-4	128	0.999942528	0.999942528
S-A	11,191,389	1e-4	128	0.998333333	0.998333097

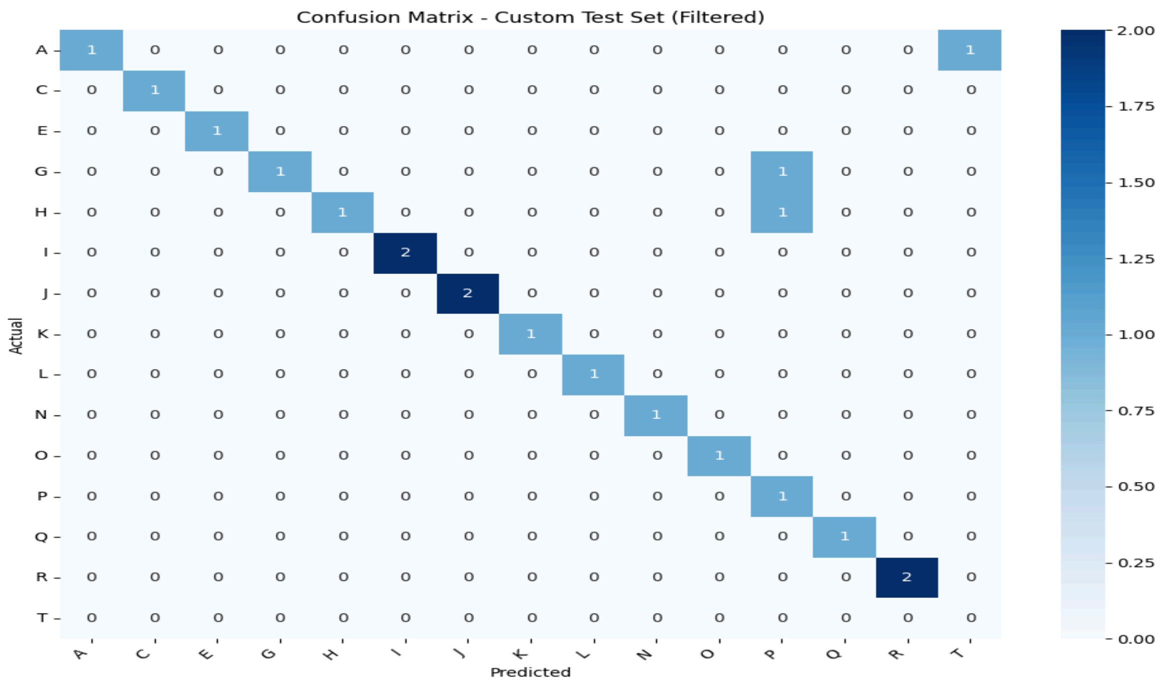
3. Final Evaluation & Analysis

I first evaluated the selected T-C model on the provided Kaggle test set of 28 images. The model achieved perfect performance with an accuracy and Macro-F1 of 1.0. While this confirmed the model was functioning correctly, the test set contained only one image per class, which was deemed insufficient for a rigorous stress test of the model's robustness.

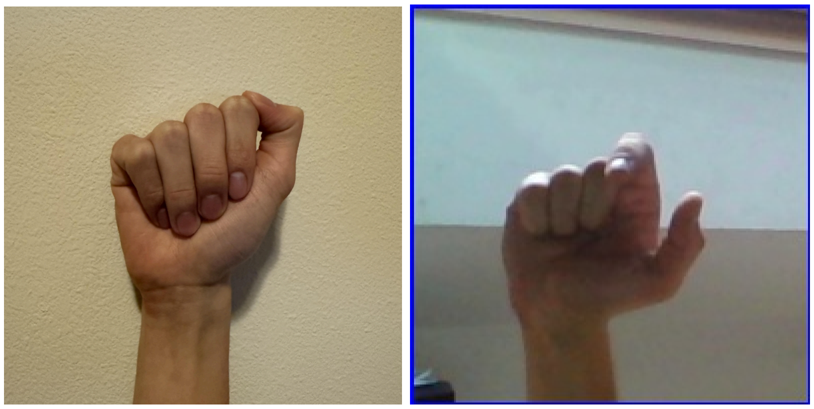


To test generalization capabilities properly, I created a custom dataset of 20 images featuring varied backgrounds and lighting conditions. My initial testing on this set yielded a surprisingly poor accuracy of 0.20 and a Macro-F1 of 0.15. This severe performance drop, despite 99% validation accuracy, prompted me to investigate the preprocessing pipeline for domain shifts. I hypothesized three potential causes: orientation mismatches, aspect ratio distortion, and lighting variance.

I began by investigating the orientation issue. I discovered that my phone was storing rotation information in EXIF metadata, which the default PyTorch image loader was ignoring and thus reading 90 degree rotated versions of my images. This meant the model was effectively seeing "sideways" hands. By implementing an EXIF transpose operation to align the orientation before processing, accuracy immediately jumped from 0.20 to 0.75. Next, I addressed aspect ratio distortion. The ResNet model expects square inputs (224x224), but simply resizing rectangular phone photos was squashing the hand features and distorting the geometric patterns. I implemented a Center Square Crop before resizing to preserve geometric integrity, which further improved accuracy to 0.85. Finally, I tested the lighting hypothesis by attempting to normalize the pixel intensity distribution of the custom images to match the training set mean. Surprisingly, this yielded no change in accuracy. This finding was significant as it suggested the model generalizes well to lighting variations, likely due to the ColorJitter augmentation I employed during training, and that the primary hurdles were geometric. **The final custom set evaluation yielded an accuracy of 0.85 and a Macro-F1 of 0.83.**



3.1 Misclassification Analysis: 'A' vs. 'T'



The above pictures show my custom 'A' image (left), where the thumb rests vertically against the side, being incorrectly classified as 'T' (right), which involves the thumb being out. This misclassification reveals that the model prioritized the dominant "closed fist" silhouette shared by both signs over the fine-grained position of the thumb. Likely due to lighting or angle obscuring the necessary depth cues, the model failed to resolve the subtle thumb placement and defaulted to the visually similar 'T' class.

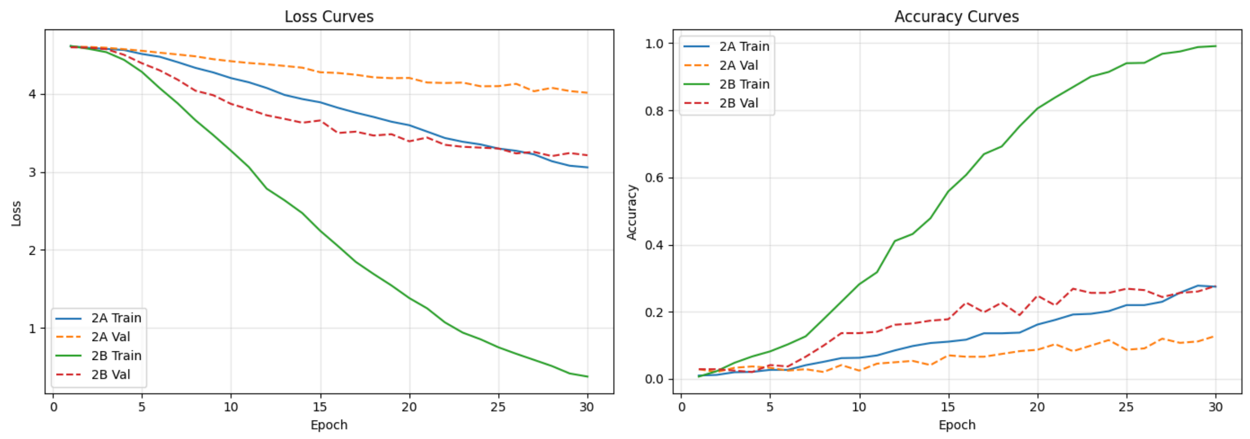
4. Conclusion (Phase 1)

The results of this study indicate that the Progressive Fine-Tuning approach (T-C) slightly outperformed Training from Scratch (S-A), achieving 99.99% validation accuracy versus 99.83%. Beyond the raw metrics, the pre-trained model demonstrated significantly faster convergence, reaching near-perfect accuracy almost immediately, whereas the scratch model required several epochs to learn low-level edge detectors. This translated into computational efficiency as well where T-C required less computation time than S-A to achieve more. The comparable performance of S-A on validation suggests that the large dataset size (87k images) provided sufficient data diversity to learn robust features without pre-training. While the S-A model performed exceptionally well due to the large dataset size, the Transfer Learning approach provided a more stable training process with lower variance in loss. I conclude that the Progressive Fine-Tuning approach offers the best balance of training stability and final accuracy, as the pre-trained weights act as a strong regularizer against overfitting to the specific noise patterns of the training data. Furthermore, my experience with the custom test set highlights that preprocessing alignment, specifically regarding orientation and aspect ratio, is just as critical to real-world deployment as the model architecture itself. To further improve, implementing an object detection pre-processing step (like YOLO) to crop the hand region before classification would eliminate background noise and improve robustness against domain shifts.

5. Phase 2: Dynamic Word Recognition Bonus

5.1 Architecture and Methodology In the second phase of the project, the system was extended to recognize dynamic words using the WLASL100 dataset, which consists of video clips representing 100 different classes. Instead of training a video classification model from scratch, a transfer learning approach was adopted by utilizing the best performing static model from Phase 1, Configuration T-C, as a spatial feature extractor. The classification head of the ResNet 18 was removed to expose the 512 dimensional output of the Global Average Pooling layer. To handle the temporal dimension, videos were processed by uniformly sampling 16 frames per clip. Each frame was passed through the CNN encoder to generate a sequence of feature vectors with a shape of 16 by 512. These sequences were then fed into a 2 layer Long Short Term Memory LSTM network with a hidden dimension of 256 and a dropout rate of 0.3. The LSTM aggregated the temporal information, and its final hidden state was passed to a fully connected layer with 0.5 dropout to predict the final word class.

5.2 Ablation Study: Training Strategies To determine the optimal fine tuning strategy for this hybrid architecture, two distinct training configurations were compared over 30 epochs using the Adam optimizer with a learning rate of 1e-4. The first configuration, 2A Frozen CNN, kept the ResNet backbone completely frozen, training only the LSTM and the new classifier head. This setup tested the hypothesis that static alphabet features are sufficient for dynamic words without modification. The second configuration, 2B Light Adaptation, unfroze the last convolutional block Layer 4 of the ResNet backbone, allowing it to train alongside the temporal head. This strategy aimed to adapt the high level spatial features to specific video artifacts such as motion blur, which are not present in static images.



5.3 Results and Analysis The results demonstrated in the figure above show a clear advantage for the second strategy. Configuration 2A achieved a Validation Macro F1 of only 0.1043, whereas Configuration 2B significantly outperformed it with a Validation Macro F1 of 0.2342. Figure 2 illustrates the training dynamics, showing that while the frozen model 2A plateaued early, the unfrozen model 2B showed consistent improvement in both loss reduction and accuracy. These findings suggest that static ASL features alone are insufficient for dynamic recognition and that the model requires the flexibility to adjust its deeper spatial representations to the new domain of in the wild video data. Based on these validation metrics, Configuration 2B was selected as the final model. Evaluation on the held out test set yielded an Accuracy of 0.0750 and a Macro F1 of 0.0608. The significant drop in performance from validation to testing highlights the substantial challenge posed by the WLASL dataset, which features diverse lighting, backgrounds, and signer variations compared to the controlled static environment of the Phase 1 dataset.