Computer Vision – Assignment 3

Nicolas Ramos Fernandez s3917886, Alexandru-Emil Gavrilut s4316061

December 2025

1 Abstract

Video classification typically requires significant computational resources and large datasets, creating a barrier to entry for limited-resource environments. In this report, we propose and analyze the impact of data preprocessing strategies on model performance under strict compute constraints (single GPU) using a small subset of the Jester dataset (23712 videos). We compare a resource-efficient 2D CNN baseline against a computationally intensive 3D ResNet, evaluating how frame-differencing strategies, specifically absolute and relative differences, affect classification accuracy. Our experiments show that while 3D architectures achieve superior performance (achieving 83.7% accuracy), they come with an 18-fold increase in training time compared to the baseline. However, we find that simple data transformation strategies can significantly boost the performance of lightweight 2D models (improving accuracy from 44% to 60.3%) by effectively filtering background noise and isolating motion features. These results suggest that strategic data preprocessing can serve as a viable alternative to complex architectures in resource-constrained settings.

2 Introduction and Related Work

Video Understanding has been a heavily studied research topic in Computer Vision. Ever since the creation of the first 3D Convolutional Neural Network (3D CNNs) by Ji et al. [3], the field has rapidly advanced with deeper architectures, Recurrent Neural Networks (RNNs), and Transformers. However, two of the most prominent issues these models must face are the vast amounts of data and computing power needed. Unlike single images, video data requires processing spatiotemporal volumes, leading to significantly higher memory footprints and training costs. This creates a barrier to entry for independent researchers and raises concerns regarding the environmental impact of large-scale model training.

In this project, we address the constraint of efficient video classification under limited computational and data resources. We restricted our experimental setup to a single NVIDIA 4060 GPU and used a subset of the Jester dataset, consisting of 23,712 training videos across 27 gesture classes. Our primary goal is to investigate how model architecture and data processing strategies can be optimized to balance accuracy with computational efficiency.

For this goal, we design and evaluate two distinct architectural approaches. As a resource-efficient baseline, we implement a 2D CNN (TinyVGG) [4] that operates on temporally averaged frames, effectively transforming the video into a single static image. This is compared against a computationally intensive 3D ResNet [1], which processes the full spatiotemporal features. These models represent two extremes of the efficiency spectrum: TinyVGG contains 2.7 million parameters, whereas the 3D ResNet has 33 million. Finally, for both the baseline and 3D models, we explore how applying transformations on each frame, before feeding it to the model, can lead to significant increases in performance without significantly increasing computational needs and without requiring any new data.

3 Experiment Results and Discussion

3.1 Baseline Model

For the baseline, we implemented a model using the TinyVGG architecture. This model accepts a 2D image input and outputs a probability vector for the gesture classes. The architecture consists of 10 2D convolutional layers, each followed by batch normalization and a ReLU activation. 2D max pooling is applied before the 3rd, 5th, 7th, and 9th convolutional layers. The classification head consists of a linear layer followed by ReLU, a dropout layer (p=0.1), and a final linear layer.

We evaluated three different data transformation methods. In all cases, the first and last 30% of frames were trimmed to isolate the core gesture movement. The remaining frames were resized to 100 by 150 pixels. The final input to the model was generated by calculating the mean value of each pixel across the processed frames. For increased efficiency, we cached all the resulting datasets, which reduced the training time from minutes for training and 3 minutes for validating per epoch, to 30 seconds for training and 20 seconds for validation. Caching is necessary for both 2D and 3D images, as without it, the program would have to load every frame in memory, compute the resulting image with its respective strategy, and only after that pass it to the GPU, which is simply inefficient.

The first approach, which we named "simple mean", calculates the pixel-wise mean of the trimmed frames without further modification. After 20 epochs, this method achieved a maximum validation accuracy of 44%. This result indicates that a significant amount of gesture information can be retained even when the temporal dimension is completely collapsed. However, we wanted to see whether some improvements could be made to the image we provide to the model that would increase accuracy.

This brought us to the idea of taking the first frame of the video (before trimming) and then subtracting it from every other frame in the video after trimming. After subtracting, we took the absolute value of each pixel, since images cannot have negative values. This can be expressed by the formula

$$\text{result} = |\text{current frame} - \text{first frame}| \quad (1)$$

This idea is similar to that of temporal derivatives ([5] section 19.4.2), and the general idea is that the first frame can be understood as the "base" of the video, and by subtracting it to every other frame we only capture the differences from the "base". This helps remove some irrelevant information such as the background or the gender or race of the person, making the model able to capture the motion explicitly. We call this the "absolute difference" approach. It achieved 56.2% validation accuracy. However, there are still some problems to this approach. Most importantly, since the absolute value is calculated, a negative change in pixel values is interpreted the same as a positive change of the same magnitude. In order to try to solve this problem, instead of calculating the absolute value of the difference, we performed the following operation:

$$\text{result} = \frac{\text{current frame} - \text{previous frame}}{2} + 0.5 \quad (2)$$

This operation moves the difference to the [0, 1] range while still being able to differentiate between negative and positive differences, in the ranges $[0, 0.5)$ and $(0.5, 1]$ respectively. This approach resulted in an accuracy of 55.0%.

Finally, we tried a third type of data modification. In the previous approach, all frames are compared to the first, so a video of a person moving their hand to the left or to the right will produce very similar images, possibly leading the model to struggle distinguishing gestures that depend on direction. Instead,

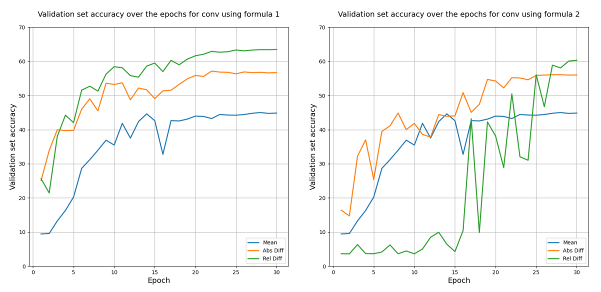

Figure 1: Figure comparing the accuracy over time of the baseline models, with differences calculated using formula 1 (left) vs formula 2 (right). It can be observed that the mean and absolute diff models have similar performance using both formulas, but that relative difference has a significantly different curve, constantly going up and down. It also achieves a lower final validation accuracy.

we tried trimming the video and then subtracting each frame from its previous frame before averaging. Since each frame depends on the previous frame, the images encode some temporal features. We call this the "relative difference" approach. We first tried using formula 1, and it yielded 63.5% validation accuracy. Subsequently, we tried formula 2 and achieved 60.3%.

As illustrated in figure 1, the relative difference curve is significantly different. This could be due to the images having very tough to understand features. Since most of the background would be gray as it would be 0.5 with relatively minimal noise, this does not let the relative difference model have empty, or very color-sensitive features, and it makes much worse guesses in the beginning. Consequently, as the absolute difference model has a completely black background, it is color-sensitive and able to distinguish between different gestures much easier at first, but struggling to grasp nuances.

The results of training the baseline models are shown in the dashed lines of figure 2. It is clear that the simple mean image achieved the lowest accuracy. The relative difference model got the highest accuracy, with absolute difference coming in close second. This suggests that the difference models were either able to capture part of the temporal dimension, and that the removal of irrelevant characteristics from the background, e.g. race, helped the model focus on the gesture. Looking at figure 3, the difference models still struggled to distinguish between gestures where direction plays an important role, but displayed better performance than simple mean.

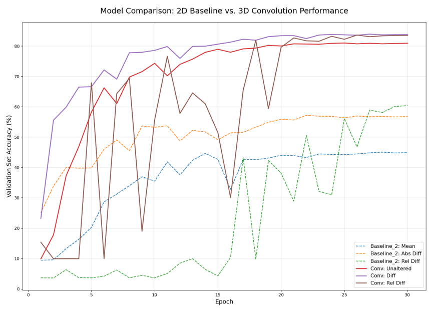

Figure 2: Figure showing the validation set accuracy over the epochs for the baseline and 3D convolutional models, each using the 3 approaches defined in their sections, with differences calculated using formula 1, except the relative difference models, which use formula 2. The solid lines mark the convolutional models, while the dashed lines mark the baseline models. Focusing on the dashed lines (baseline models), we can observe that the mean approach reached the lowest final accuracy, and it was significantly lower than the other two approaches throughout the whole training; it is also noticeable that the mean approach stabilized around epoch 15, while the other two approaches kept improving slightly after that point. Relative difference had the highest accuracy at the end, closely followed by absolute difference. The solid lines (3D Conv) all reach higher final validation accuracies. It can be observed that the relative difference 3D Conv curve is also jagged, with steep increases and decreases, like the relative difference baseline curve

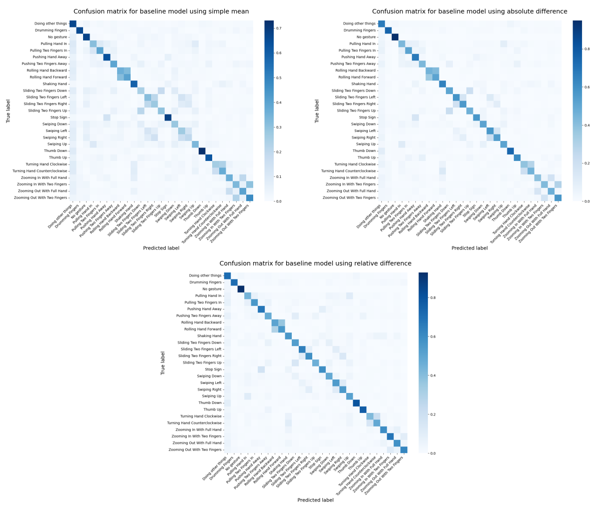

Figure 3: Confusion matrices of the three baseline approaches after 30 epochs. It can be seen that all three approaches struggled differentiating rolling hand backward vs forward, and distinguishing turning hand clockwise and counterclockwise, features dependent on directional, continuous movement. Simple mean struggled actions such as sliding two fingers left vs right, which the other two approaches did not struggle as much with.

3.2 3D ResNet model

Having built solid baseline models, we experimented with a 3D ResNet (Residual Networks) architecture. These allow the model to *explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions*, as expressed in the original paper [2]. We adapted the ResNet architecture to 3D following the paper by changing all the 2D kernels to 3D kernels. Our model has a very similar structure to the one presented in the paper [1]. We trained for 30 epochs using batch size 64, and used cross entropy loss with label smoothing of 0.1, the AdamW optimizer with a learning rate of 0.001, and a cosine annealing learning rate scheduler. We will be referring to this model interchangeably as ResNet or 3D CNN. The reason for choosing a convolutional network over transformer model is that transformers tend to work best when provided with very large amounts of data and computing power. This makes it inadequate for this experiment.

Similarly to the baseline model, we explored three ways of modifying the original video data to help the model make better predictions. The approaches were very similar to those in the baseline, except averaging all the frames after trimming and transformation has not necessary. The frames were stacked into a 4-dimensional tensor of shape (num_frames, 3, 100, 150), so that the model would have access to the

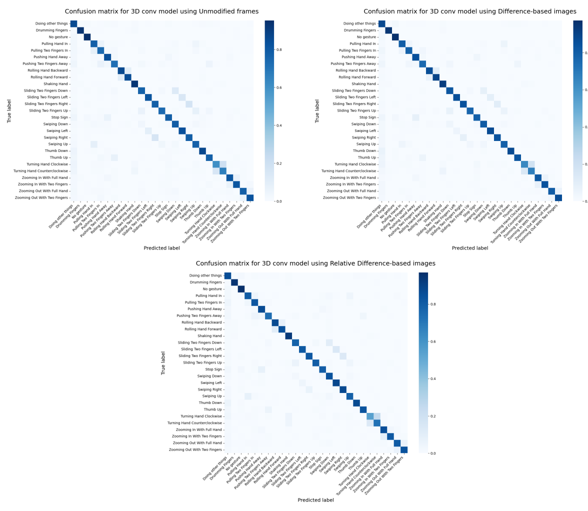

Figure 4: Figure depicting the confusion matrices of the three 3D CNN model approaches after 30 epochs. It can be seen that all three models can sometimes struggle differentiating pulling hand in vs pulling two fingers in, sliding two fingers down/left/right vs swiping down/left/right, turning hand clockwise and anticlockwise, and zooming in/out with full hand vs zooming in/out two fingers. This is especially noticeable in the unmodified frames model.

temporal dimension. To increase efficiency in both compute and training time, we saved the tensor of every image in the cache, reducing our training time by almost 66%, lowering from 36 minutes for a single train loop and 16 minutes for a validation loop down to 11 and 7 minutes respectively: a mere 18 minutes per epoch. Finally, to preserve consistency with the baseline models and provide a quality comparison, both formula 1 and formula 2 were used.

Looking back at figure 2, we notice that the 3D ResNet model achieved significantly higher accuracies: 80.9% 83.7%, and 83.5% for unaltered frames, absolute difference, and relative difference respectively. This suggests that using 3D convolutions offers a significant advantage over calculating the mean of all the frames and performing 2D convolutions on that, which was the expected result. The difference between unaltered frames and the two difference methods is more narrow than the same difference in the baseline models. A possible reason is that the 3D ResNet models are inherently able to capture temporal differences between frames, reducing the necessity to alter the frames to highlight the differences. However, given that they come with close to zero additional computing costs, they are still a useful tool.

Figure 4 depicts the confusion matrices of the 3D Convolutional models using the 3 approaches. The three matrices are quite similar, and it can be observed that all three tend to make mistakes between the same few classes. The misclassified videos are very different from those made by the baseline model: while the baseline tended to confuse the same type gesture in different directions, i.e. sliding two fingers left or right, the 3D CNN model is generally able to distinguish the direction of the gestures, but instead tends to make mistakes in gestures that follow similar paths, such as sliding two fingers down vs swiping down. This suggests that our 3D ResNet model takes advantage of the temporal aspect of videos by utilizing 3D convolutions to capture the difference between moving in different directions. However, the model may over-rely on the paths followed by hands in the gestures to make the classification, which may lead it to mix up gestures that follow a similar path. The only gestures that both the baseline and 3D CNN models performed less well were turning hand clockwise and counterclockwise.

4 Conclusion

In this work, we demonstrate that efficient video classification is a balance between architectural complexity and data preprocessing strategies. Our experiments show a distinct trade-off: the 2D baseline models, while computationally efficient (training in less than 1 minute per epoch), plateaued at 60.3% accuracy. In contrast, the 3D ResNet models achieved a significantly higher accuracy of 83.7%, effectively capturing temporal dynamics that the 2D models missed, though at the cost of an 18-fold increase in training time. Furthermore, we found that simple pixel-wise transformations, specifically relative 2 and absolute differencing 1, can drastically improve the performance of simpler models by removing static noise (background, user appearance) and isolating motion. Although 3D architectures remain superior for capturing complex temporal dependencies, our results highlight that strategic data transformations can bridge the gap for environments with limited resources.

References

[1] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition, 2017.

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

[3] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[5] A. Torralba, P. Isola, and W.T. Freeman. *Foundations of Computer Vision.* Adaptive Computation and Machine Learning series. MIT Press, 2024.

5 Individual Contribution

For this project, we worked together, in equal manner, to achieve our results and finish this report. Alexandru wrote the code for the simple mean approach for the baseline, while Nicolas updated it for the 2 remaining strategies: absolute and relative difference. All changes were shared through GitHub (https://github.com/Zawezu/gesture-recognition) and, at your request, we can submit our git history, showing equal contribution. For the 3D CNN, Nicolas designed the ResNet used, and wrote all the code necessary to train the relative difference model, while Alexandru handled the data for the unaltered and absolute difference images, and trained and analyzed the results of the models generated. Training was done on the computers available in the DM0 laboratory rooms at Leiden University, and, for 2 baselines only, on Nicolas's personal computer. For this reason, the code includes a function to easily switch between the 2 operating systems, Windows 11 and Ubuntu Linux. Subsequent updates, checking for data integrity and path validity, as well as easily switching between cache directories used and saving models in the train function were made by both of us, at different times while we worked on the project. The report was first drafted by Nicolas, updated by Alexandru, and final touches were, once again, made by Nicolas. This was a combined effort, one that we are both proud of!