# Segment-Based Approach Using Pretrained Res-NeXt Architectures for Hand Gesture Recognition

M Lytovka (s3705609), V Kalinin (s3648214)
Leiden University
Rapenburg 70, 2311EZ Leiden, Netherlands
m.i.lytovka@umail.leidenuniv.nl
s3648214@vuw.leidenuniv.nl

## Abstract

*Action classification is a challenging task in computer vision, with numerous approaches proposed over the past decades. In this work, we focus on a specific subtask: gesture classification using the Jester dataset, emphasizing a lightweight model design. In our solution, videos are divided into fixed segments, with one frame sampled from each segment. Features are extracted from these frames using a ResNeXt-101 convolutional neural network (CNN) pretrained on the ImageNet dataset, and then fed into a multi-layer perceptron (MLP), the only trainable component of our approach. We systematically evaluate MLP architectures, frame-sampling strategies, and segment counts. Our findings demonstrate that smaller MLP architectures yield better performance, equidistant frame sampling (e.g., the first frame from each segment) is optimal, and reducing the number of segments from 8 to 4 maintains competitive accuracy while halving the training time. These results highlight an efficient pipeline for gesture classification that balances accuracy and computational demands.*

## 1. Introduction and Related Work

Our work builds upon the framework proposed by Köpüklü et al. [8], which addresses the task as follows (Figure 1). The video is divided into $n$ segments, and from each segment, the last frame is sampled to ensure equidistance. Additionally, $m$ frames preceding the last one are used to extract the Motion Fused Frames (MFFs), which are optical flow images computed using the Brox technique [11]. Consequently, for each segment, there is one RGB frame and $m$ Brox flow frames. These frames are subsequently fed into a model comprising two main components: the first component consists of $n$ parallel CNN models (one for each set of frames), pretrained on the ImageNet dataset. The second component is a Multi-Layer Perceptron (MLP), which outputs the logits for the classes. Class probabilities are computed using a softmax function applied to the logits.

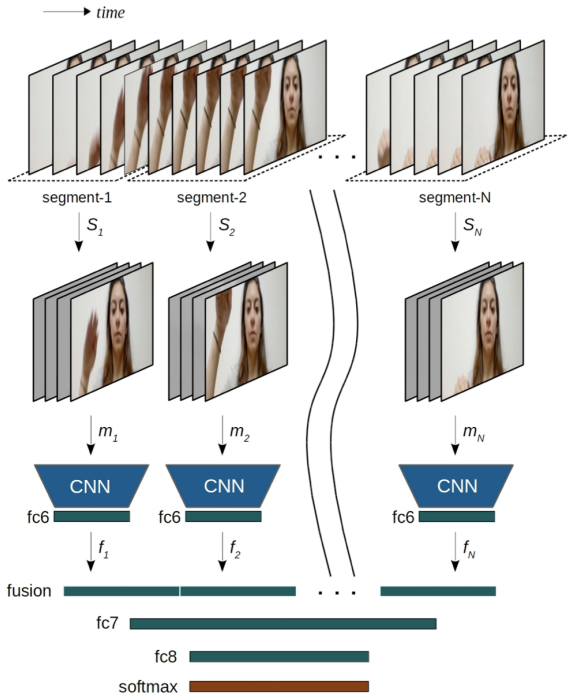


Figure 1. A visualization of the training pipeline illustrating the segmentation of video frames and using them to train CNNs. Each segment contains one RGB frame and $m$ Motion Fused Frames [8] (explained later), which is processed through a ResNeXt-101 CNN for feature extraction. These features are concatenated and input into a Multi-Layer Perceptron for classification.
*Source: Köpüklü et al. (2018) [8]*

In their study, the authors reported an accuracy of $92.9\%$ without utilizing any MFFs (i.e., setting the parameter to

0). Incorporating three MFFs increased the accuracy to 95.36%. While this improvement is notable, it represents a relatively modest gain compared to the overall accuracy. With this observation and our focus on developing a computationally efficient solution, we opted to exclude the MFFs in our implementation in favor of simplicity. Furthermore, to enhance computational efficiency, we froze the CNN parameters, using the network solely for feature extraction without any fine-tuning. Therefore, we train only MLP, which follows the feature extraction and fusion. While this approach resulted in a reduction in maximum achievable accuracy, it significantly decreased computational complexity. Also, the number of maximum number of epochs was reduced as well: from 45 to 20. It was done due to the time restrictions of the assignment.

Additionally, we employed a different CNN architecture. Whereas the original framework utilized the Inception architecture [2] with Batch Normalization (resulting in BN-Inception) [7], we chose ResNeXt-101 [12], which has demonstrated improved performance over the plain ResNet [5].

While our current implementation predominantly relies on the work of Köpüklü et al. [8], we also reviewed another paper by Bolei et al. [13], which considers Temporal Relation Reasoning in videos. The idea presented in the paper is straightforward: sequences of frames are taken, and multiple subsets are subsampled with $n = \{2, 3, 4, \dots\}$ frames in their appropriate order (Figure 2).

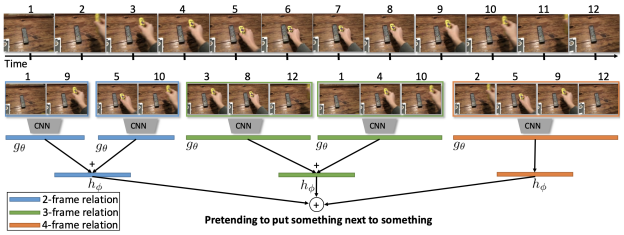


Figure 2. Temporal relation reasoning framework, depicting sequences of frames sampled from video clips. The illustration highlights the formation of subsets of frames (n = 2, 3, 4, ...) to establish temporal relationships. The figure emphasizes the difference between this method and the sequential frame selection used in the MFF study.
*Source: Bolei et al. (2017) [13]*

This approach establishes temporal relations between frames with varied spans and numbers of data points. Our approach is similar but differs in that we use a single sequence of $n$ frames instead of combining sequences of varied lengths. The authors reported achieving 95.31% accuracy with their best model, which is encouraging for the potential success of our solution.

## 2. Methodology

In this work, we propose a sequential approach to optimize the hyperparameters of the network architecture. In each round, we optimize one hyperparameter by comparing models based on their performance during the first 20 epochs of training and selecting the best-performing model. Once a hyperparameter is optimized, it is fixed for subsequent rounds, and the next hyperparameter is optimized. This process is repeated for three hyperparameters, although our approach has certain limitations, which are discussed in Section 4. The experiment resulted into 6 different models, outlined in the Appendix A. From now on, we will reference the models by their names in the table. The three hyperparameters optimized in this study are:

- The architecture of the MLP module of the network - comparison of models 1-3
- Selection of the frame from the video (uniform vs 1st frame) - comparison of models 3-4
- Number of the segments ($n$) we divide original video into - comparison of models 4-6

### 2.1. Dataset and Segmentation

We utilize the Jester dataset [9], which consists of short video clips of hand gestures, averaging 37 frames per video at 12 FPS. Our approach divides each video into a fixed number ($n$) of temporal segments, selecting one representative frame from each segment, either randomly or as the first frame.

### 2.2. Data Augmentation and Feature Extraction

To increase the diversity of the training videos and improve model generalization, various data augmentation techniques are applied. The augmentation pipeline consists of several random transformations:

- **Elastic Transformations [10]:** Random elastic deformations are applied to the frames of the video (20% probability). The deformation parameters include pixel displacement ($\alpha = 50$) and the standard deviation of the Gaussian smoothing kernel ($\sigma = 9$).
- **Color Jittering:** The brightness, contrast, saturation, and hue of the images are randomly adjusted (20% probability). The parameters for the color jitter are set to $\pm 0.1$ for each channel.
- **Random Sharpness Adjustment:** The sharpness of the image is adjusted randomly (50% chance) with a sharpness factor ranging of 2.
- **Autocontrast:** Random (50% chance) application of an autocontrast.

- **Random Equalization:** Random (50% chance) equalization is applied.

Then, the images are put into a preprocessing pipeline:

- **Resize:** The input frames are first resized to a resolution of $232 \times 232$ pixels using bilinear interpolation.
- **Random Crop:** After resizing, a random crop is applied to each frame to extract a patch of size $224 \times 224$ pixels.
- **Normalization:** The cropped images are normalized using ImageNet [3] mean and standard deviation values:

$$\text{Mean} = \begin{bmatrix} 0.485 \\ 0.456 \\ 0.406 \end{bmatrix}, \quad \text{Std} = \begin{bmatrix} 0.229 \\ 0.224 \\ 0.225 \end{bmatrix}.$$

- **Interpolation:** Bilinear interpolation is used throughout the resizing and cropping operations.

The augmentation and preprocessing are applied to the training dataset, while the validation and test datasets undergo only preprocessing.

### 2.3. Training, Evaluation, and Implementation

Training is performed with the Adam optimizer (learning rate: 0.001, $\beta_1$: 0.9, $\beta_2$: 0.999, $\epsilon$: $10^{-8}$, and L2 penalty: 0) applied to a mini-batch of 8 videos with a cross-entropy loss. After each training epoch, we evaluate the model on a validation set and save metrics and checkpoints. All experiments were conducted on a system equipped with an Intel Core i7-14700 processor (20 cores: 8 performance cores and 12 efficient cores; performance cores are Hyper-Threaded), 32 GB of RAM, an NVidia GeForce GTX 4060 GPU with 8 GB of VRAM, and a 1 TB NVMe SSD. We make our code publicly available [1] for the reproducibility of the results.

## 3. Experimental Results

Following the methodology and training 6 models for 20 epochs, we obtained results outlined in Figures 3, 4, and 5.

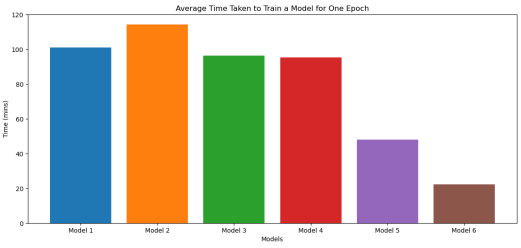


Figure 3. Bar chart showing the training time per epoch for different models. Hardware information may be found in the section2.3

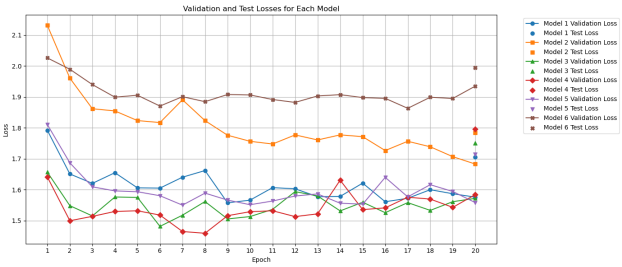


Figure 4. Line graph presenting the validation and test losses across 20 training epochs for six models. Each line corresponds to a specific model configuration.

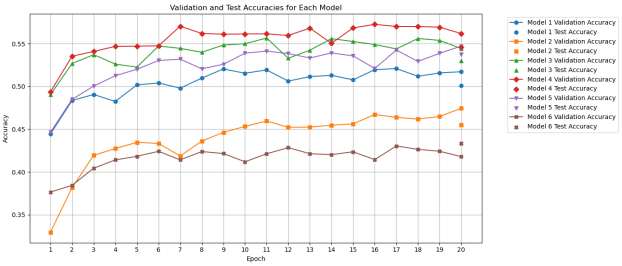


Figure 5. Line graph comparing validation and test accuracies across 20 training epochs for six models.

| Model | Test Accuracy | Test Loss |
|---|---|---|
| 1 | 0.499 | 1.708 |
| 2 | 0.456 | 1.783 |
| 3 | 0.531 | 1.742 |
| 4 | 0.546 | 1.796 |
| 5 | 0.537 | 1.714 |
| 6 | 0.433 | 1.995 |

Table 1. Test Accuracy and Test Loss for different models.

### 3.1. Round 1: Architecture Selection

In this experiment, we compare Models 1 to 3, each employing distinct configurations for the MLP component of the model. Details of the architectures can be found in the Appendix A. The results indicate that Model 2, the largest architecture, exhibited the poorest performance. In contrast, Model 3, the smallest architecture, achieved the best results. Although Model 1, the medium-sized architecture, required nearly the same computation time as Model 3, it delivered inferior results, as evidenced by the loss and accuracy plots. Consequently, Model 3 was selected for the next round.

### 3.2. Round 2: Frame Sampling Technique

In this experiment, we evaluate Models 3 and 4, which share the same architecture but differ in frame sampling

techniques. Model 3 samples frames uniformly, whereas Model 4 selects the first frame from each segment. The results demonstrate that equidistant frame selection, as implemented in Model 4, yields better performance. Thus, Model 4 was chosen for subsequent experiments.

### 3.3. Round 3: Segment Size

This experiment investigates the impact of different segment sizes on performance by comparing Models 5 and 6. The results indicate clear underfitting in both cases. However, while Model 5 converges more slowly, its test accuracy is comparable to that of Model 4. Additionally, significant differences in training time were observed, which will be discussed in the following section.

### 3.4. Explanation and Thoughts

**Time.** A clear twofold reduction in training time is observed between Models 4 and 5, as well as between Models 5 and 6. This reduction is primarily attributed to the time required for feature extraction by the Res-NeXt backbone, which constitutes the most computationally intensive part of the training process. In contrast, the MLP module trains relatively quickly. Notably, a significant increase in the number of parameters from 75.5M to 257.3M resulted in a slight increase in computation time between Models 1 and 2.

**Loss.** A clear decreasing trend in loss convergence is observed for Model 2, with a similar but less pronounced trend for Model 1. This suggests that 20 epochs may be insufficient for a fair comparison, highlighting a limitation of our approach. However, this issue does not appear to affect the smaller models, such as Models 5 and 6, to the same extent.

**Accuracy.** An increasing trend in accuracy is observed for both Models 5 and 2. For Model 2, this suggests that its wide and deep architecture may require additional training epochs to reach its full potential, with the possibility of outperforming the other models over extended training. Similarly, Model 5 may also yield surprisingly positive results, given that its MLP component is half the size of Model 4's, yet it does not exhibit significant drops in performance.

## 4. Discussion

This study has several limitations. Below, we outline them, along with potential solutions for future research:

- **Number of epochs:** The training was limited to 20 epochs, which may be insufficient for larger models, such as Model 2, leading to underfitting. Future studies could address this by increasing the number of training epochs to allow models to reach their full potential.
- **Regularization techniques:** Implementing regularization strategies, such as weight decay, dropout layers, and a gradual reduction of dropout ratios, may enhance training efficiency and model performance.
- **Alternative CNNs:** Exploring alternative architectures for feature extraction, such as Inception-BN, which has demonstrated high accuracy in related tasks ( [8], [13]), could provide valuable benchmarks against ResNeXt-101.
- **Structured hyperparameter tuning:** Replacing manual hyperparameter tuning with a systematic approach, such as Bayesian optimization, would enable a more comprehensive exploration of the parameter space and reduce reliance on human intervention.
- **Global training:** Fine-tuning the pretrained CNN (Res-NeXt 101 in this case) as part of the overall model could further optimize feature extraction, provided sufficient computational resources are available.
- **Enhanced MLP architecture:** Investigating more sophisticated architectures, such as Recurrent Neural Networks ( [4]), especially Long Short-Term Memory networks ( [6]), as replacements for the simple MLP component, may improve the second stage of the model.

## 5. Conclusion

Our work explored ResNeXt-101 and an MLP for hand gesture recognition on the Jester dataset, achieving reasonable accuracy with limited training resources. Key findings include:

- The smallest MLP architecture among those evaluated demonstrated the best performance in the architecture comparison, favouring the simplicity in the context of the present task.
- Equidistant frame selection improved loss convergence over random sampling, highlighting the importance of systematic approaches.
- Reducing segments from 8 to 4 halved training time with minimal impact on accuracy, suggesting 4 segments as a practical choice.

Due to the limitations such as underfitting in larger models and the absence of regularization, there is a need for extended training, diverse datasets, and advanced temporal reasoning for future improvements.

## References


[1] Github repository. https://github.com/Reymer249/CompVision_ass3.

[2] Yangqing Jia Pierre Sermanet Scott Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Andrew Rabinovich Christian Szegedy, Wei Liu. Going deeper with convolutions, 2014.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[4] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

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

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[8] Okan Köpüklü, Neslihan Köse, and Gerhard Rigoll. Motion fused frames: Data level fusion strategy for hand gesture recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2184–21848, 2018.

[9] Qualcomm. Jester dataset. https://www.qualcomm.com/developer/software/jester-dataset.

[10] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 958–963, Redmond, WA, 2003. IEEE. Microsoft Research, One Microsoft Way, Redmond WA 98052.

[11] N. Papenberg T. Brox, A. Bruhn and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *European conference on computer vision, pages 25–36. Springer*, 2004.

[12] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. 2016.

[13] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. 2017.


## Author Contributions

M. Lytovka was responsible for implementing all code related to server connections via SSH, Linux configurations required to facilitate training and data preprocessing. V. Kalinin focused on frame sampling strategies, implementing the code for the MLP architecture, and generating result plots. Both authors contributed equally to the development of architectures, result analysis, and writing the report.

## A. Models Evaluation Tables

| Model | Max Training Accuracy | Min Training Loss | Test Accuracy | Test Loss |
|---|---|---|---|---|
| 1 | 0.52 | 1.56 | 0.50 | 1.71 |
| 2 | 0.47 | 1.68 | 0.46 | 1.79 |
| 3 | 0.56 | 1.48 | 0.53 | 1.74 |
| 4 | 0.57 | 1.46 | 0.55 | 1.80 |
| 5 | 0.54 | 1.55 | 0.54 | 1.71 |
| 6 | 0.43 | 1.86 | 0.43 | 2.00 |

Table 2. Performance metrics of different models on training and testing datasets. The table shows the maximum training accuracy, minimum training loss, test accuracy, and test loss for each model.

| Model | MLP - num. of parameters | Frame Selection Type | Segments | Code Links |
|---|---|---|---|---|
| 1 | 75.5 M | Random (uniform) | 8 | Model 1 |
| 2 | 257.3 M | Random (uniform) | 8 | Model 2 |
| 3 | 8.4 M | Random (uniform) | 8 | Model 3 |
| 4 | 8.4 M | 1st - equidistant | 8 | Model 4 |
| 5 | 4.2 M | 1st - equidistant | 4 | Model 5 |
| 6 | 2.1 M | 1st - equidistant | 2 | Model 6 |

Table 3. Model configuration details, including the number of parameters in the multi-layer perceptron (MLP), the frame selection method used, the number of segments in the input, and links to additional resources for each model.