Video Action Recognition

Job Minsak Nanang (jn00767) (6800151)
*Deeptanshu Sekhri (ds01505) (6783151)
Rutwik Shete (rs01960) (6771154)

Assignment Report on Video Action Recognition

Department of Electronic Engineering
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey, GU2 7XH, UK

May 2023

Git Hub Link: https://github.com/rutwik-shete/Action-Recognition

Weights and Bias project: https://wandb.ai/devilsteam/Video-Action-Recognition

Introduction

Action recognition, a crucial field in computer vision, has wide-ranging applications from video surveillance to human-computer interaction and sports analysis. Automatic recognition and classification of actions in video sequences can dramatically enhance our interaction with and understanding of our environment. In this report, we detail our journey using diverse state-of-the-art models for action recognition, presenting the challenges, insights, and performance metrics encountered along the way. Our exploration began with the ***TimeSFormer400*** model, setting a robust foundation for our successive experiments with other models. Building on this foundation, we experimented with various models for action recognition, each exhibiting different levels of success. We have explored ***2D ResNet with attention***, ***3D ResNet with attention***, and ***ResNet (2+1)D with attention***, all popular models known for their unique strengths in handling video data. Additionally, we experimented with **VideoMAE** which is a new concept for masking videos.The diversity of the models used in our work provided us with a broad understanding of different strategies and mechanisms used in action recognition tasks. This comprehensive exploration also offered us valuable insights into the unique challenges associated with each model. Each model was adapted to our specific domain, significantly reducing training time and improving generalization capabilities through the power of transfer learning. The detailed examination of these models will be presented in this report. Our goal is to paint a comprehensive picture of our journey in the dynamic field of action recognition, from the initial runs with TimeSFormer400 to the trials with various other promising models.

Data Pre-processing

In our action recognition project, we encountered challenges managing large volumes of video data leading to memory constraints. Initially, loading and processing the entire dataset caused out-of-memory errors. To resolve this, we changed the data structure in our pre-processing function, storing only paths to 8-frame sequences instead of loading all frames into memory as shown in **Figure 1**. This solution significantly reduced memory usage, allowing our system to function with just 10 GB of memory, down from the previous requirement of 30 GB. Furthermore, we noticed random data splits affected model performance when the code resumed from a checkpoint. We addressed this by creating .csv files for train, test, and validation splits to maintain consistency across runs, since the FastML library we used for data splitting lacked a random seed setting.

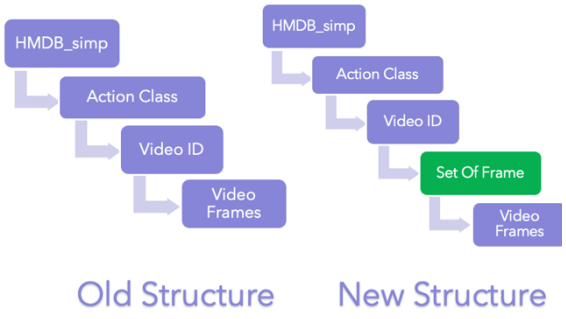


Figure 1: Change in Folder Structure

Data Analysis

The HMDB_simp dataset consists of 1,250 videos, with 25 categories, which makes it 50 videos per category. The dataset is therefore single labelled. The video files are stored as frames. On analysis we found that although there are 50 videos per category, the total number of frames per video category is not same and ranges from ~13.3k for "brush hair" to ~1.98k for "catch". Since out dataset is not balanced, out model(s) will tend to perform poorly for under-represented classes and will be biased towards the majority classes, resulting in overfitting. To analyse the effect of this skew, we will keep track of its impact on our model(s) using per class precession-recall curve and confusion matrix for test data.

Under-represented classes: *catch.*

Over-represented classes: *brush hair, kiss, shoot bow, pour, climb, smoke.*

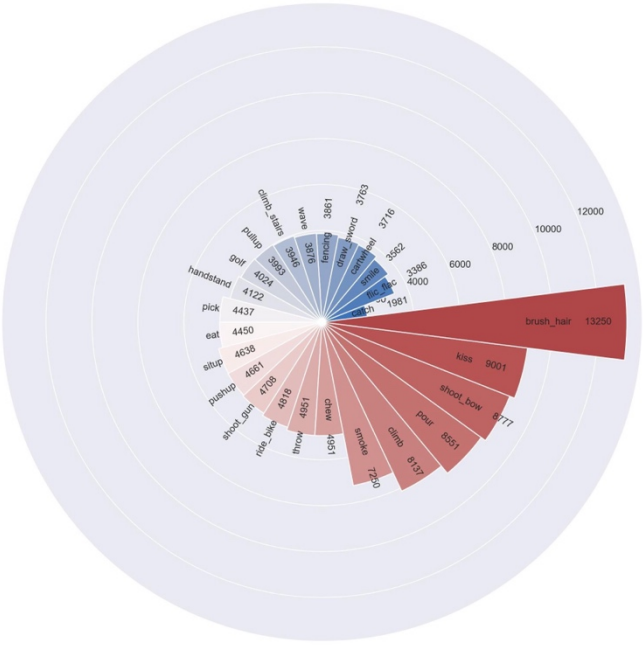


Figure 2: Action Classes and their Distribution

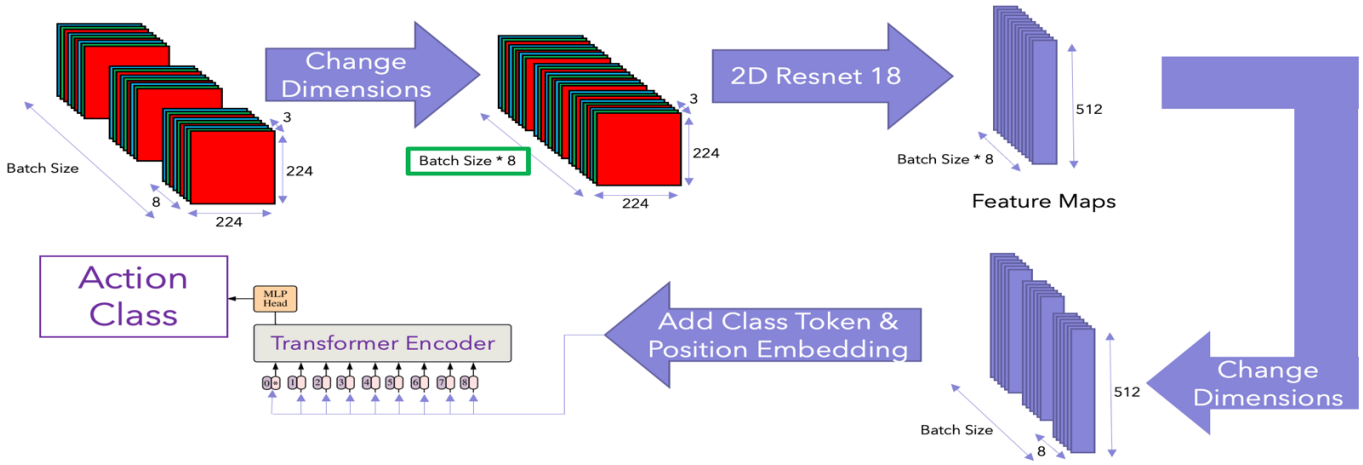


Figure 3: 2D ResNet with Attention Architecture Diagram

2DResNet With Attention

Method

In our continuous quest for improving the accuracy and efficiency of our action recognition system, we embarked on an experimental venture. We combined two distinct types of models, 2D ResNet and Transformer Encoder, each adept at extracting different types of information from video data.

The idea stemmed from the need to capture both spatial and temporal information from video sequences effectively. Videos, being a sequence of frames, contain spatial information within each frame, and temporal information lies in the sequence of frames. The challenge lies in efficiently extracting and integrating these two dimensions of information.

For spatial information extraction, we used the **2D ResNet** [1] model. However, one of the challenges we faced was the discrepancy in data dimensions. Our data is in the shape of B,T,C,H,W (where B = Batch, T = Time, C = Channel, H = Height, W = Width), but 2D ResNet accepts a 4-dimensional input in the shape of B,C,H,W. To resolve this, we reshaped our data from B,T,C,H,W to (B*T),C,H,W. This allowed us to treat each frame in the time sequence as a distinct entity in the batch dimension, effectively transforming our 5-dimensional data into a 4-dimensional format compatible with 2D ResNet. Post 2D ResNet processing, we reshape the output back to its original 5-dimensional format, B,T,C,H,W.

At this stage, we add a class token to each sequence in the batch. The class token is a standard component in transformer models, serving as a placeholder for aggregating sequence information and enabling the model to focus on classification tasks. Following this, we apply positional embeddings to the feature maps. Positional embeddings are a method of encoding the position of the frames within the sequence into the data, allowing the transformer to understand and utilize the order of the frames. Finally, we feed the class-token added, positionally embedded feature maps into the **Transformer Encoder** [2].

The Transformer, originally developed for natural language processing tasks, has demonstrated remarkable ability in handling sequential data. The Transformer Encoder takes the sequence of feature maps and encodes temporal correlations between them.

In essence, by utilizing 2D ResNet's spatial analysis capability and the Transformer Encoder's sequential processing prowess, we created an integrated model that can capture both the spatial and temporal aspects of video data. This reshaping strategy allows us to effectively use the 2D ResNet and Transformer models together, promising a new level of accuracy and efficiency in our action recognition system.

Results

Our implementation of ResNet 2D with attention resulted in impressive performance metrics as can be seen in Fig 4. After 30 epochs and using 8 attention heads, we achieved a test Top-1 accuracy of **74.42%** and a test Top-5 accuracy of **92.30%,** demonstrating the effectiveness of this model in our action recognition task.

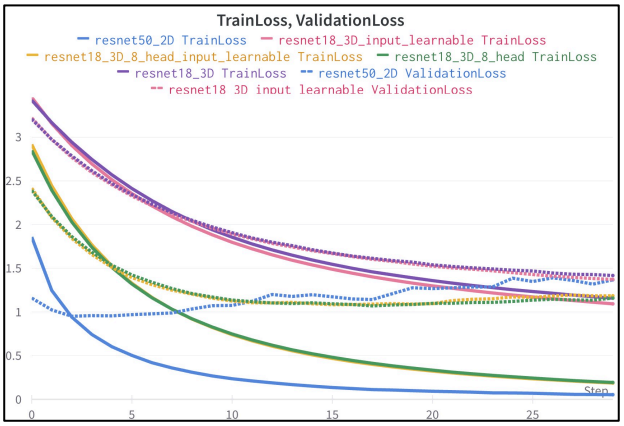


Figure 4: Training and validation loss for ResNet variants

Model	Variation	Rank1	Rank5
TimeSFormer	Trained	**84.26**	**95.77**
ResNet50 2D	Input not learnable	72.16	93.22
	Input Learnable	**74.42**	**92.30**
ResNet18 3D	No change	65.79	89.00
	Input learnable	68.52	89.72
	8 Attention heads	74.54	93.10
	16 Attention heads	74.05	94.21
	Adaptive change	**77.36**	**95.67**
ResNet18 2+1 D	No change	68.59	92.61
	8 Attention heads	**74.63**	**93.21**

Table 1: Comparing the test performance of various experiments.

Spatio-temporal Convolutions with Attention

Spatio-temporal convolutions have been a popular approach for action recognition[3]. This experiment incorporates contextual information (using attention mechanism[2]) into the convolution operations. We compare variations of two different backbone models to gather spatio-temporal convolutions.

3D convolution performs convolution in both space and time simultaneously. This enables the extraction of spatial and temporal features together. A 3D convolution model expects a 5D input with shape [B, C, T, H, W] where B = Batch Size, C = Channel Depth, T = temporal depth, H & W being height and width, respectively. T defines the temporal depth of the convolution. Truan et. al [4] tried different values of T from 1,3,5,7 and empirically concluded that T=3 performs the best. A 3x3x3 kernel requires fewer parameters and compute. Also, it covers a small local neighbourhood, capturing finer details that might get averaged out on a more significant value. Additionally, complex patterns can be retrieved by stacking up many 3x3 layers.

Two-Pathway network(2+1D) or two stream networks use separate temporal and spatial streams that capture motion cues (extracts motion information) and spatial information (extracts appearance details) separately. They are fused to form one stream containing both extracted features. These can be thought as a convenient approximation of 3D convolution where 2D convolution[1, C, H, W] is applied on spatial domain, and 1D convolution in temporal domain[T, 1, 1, 1]

We implement two different ResNet experimental setups with variants: ResNet18_3D and ResNet18_2Plus1D. Both are pre-trained on Kinetics 400[5]. An additional self-attention layer is stacked at their outputs (to gather context). Baseline training was done for 30 epochs at a learning rate $3x10^{-6}$ and batch size of 20. We conducted various experiments on this base setup and report the results. In Fig 4 we plot the validation and training loss for the variations and compare their performance in table 1.

ResNet18 3D(R3D) vs 2Plus1D(R2+1D): The 2Plus1D variant marginally outperforms the 3D variant on both training and validation. We conclude that the model can better understand temporal modelling by doing temporal convolution separately.

3D convolution with Attention: we enhance our model by adding a transformer encoder layer to apply self-attention and learn contextual relations. We used eight attention heads with an FC layer of 512. This gave excellent results, with the model performing well on training and validation. The model quickly converged on the losses; however, after some time, the model started to overfit. This can be attributed to the data-hungry nature of transformer models and the need for the same.

Increasing Attention heads: We increased the attention heads from 8 to 16. This did not improve performance. Conclusively, additional heads were providing diminishing returns, as the existing heads were enough to capture aspects of input.

Making Convolution Input learnable: improves model performance by allowing the model to learn more appropriate features for the specific task. For us, the performance marginally improved while increasing the training time considerably. Reason being that the gradients were being propagated back to the top layer. Also, since our learning rate was quite low, the input layer was not learning due to vanishing gradient. Improved performance was seen with a faster learning rate, albeit at the cost of transformer layer learning.

We also **changed the adaptive average pooling** layer of the backbone network to allow it to pass on a larger feature extraction, and with this change we got the best Rank 1 test performance of **77.36%** on this architecture with 37M parameters.

VideoMAE

In this report, we present VideoMAE, an advanced video classification model that leverages the powerful Timesformer architecture [6], 13D convolutional layers, temporal pooling, weight initialization techniques, masking, and data augmentation. VideoMAE is designed to improve the accuracy and robustness of video classification tasks by effectively capturing spatio-temporal patterns, modeling long-term temporal dependencies, and enhancing the model's generalization capabilities.

Model Architecture

VideoMAE has three main components: feature extractor, temporal modeling module (Timesformer), and the classifier.

The feature extractor incorporates a 13D convolutional layer, which performs spatio-temporal convolutions on the input video frames. This layer captures both spatial patterns and temporal dependencies within the video frames. Batch normalization and ReLU activation are applied to enhance the feature representation. Temporal pooling is then performed using a 3D max pooling layer to extract key features from different time intervals.

The temporal modeling module is based on the Timesformer architecture, which utilizes self-attention mechanisms to capture interactions between frames in the video. This module enables the model to focus on important frames and effectively model the temporal relationships within the video sequence. By incorporating Timesformer, VideoMAE enhances its ability to understand the temporal dynamics and capture complex temporal patterns in the video data.

The output from the feature extractor and the temporal modeling module is combined and passed through the classifier. The classifier consists of a sequence of layers, including layer normalization, dropout regularization, and a linear layer. The linear layer performs the final classification based on the extracted features, producing predictions for the input video.

Weight Initialization and Masking

Weight initialization plays a crucial role in training deep learning models effectively. In VideoMAE, the weights of the model, including those of the 1-3D convolutional layer and the Timesformer module, are initialized using Xavier uniform initialization [7]. This initialization technique sets the initial weights such that the variance of the outputs of each layer remains consistent across different layers. By ensuring appropriate weight initialization, the model facilitates effective gradient propagation during training, contributing to faster convergence and improved performance.

In addition to weight initialization, VideoMAE incorporates a masking technique inspired by the paper "Maskformer: A Self-supervised Attention Model for Video Representation Learning" [8]. This paper proposes a method to sparsify the attention maps within the self-attention mechanism, enhancing the model's ability to attend to relevant video frames and improve computational efficiency. In VideoMAE, a similar masking approach is applied to the final linear layer. A random mask is generated based on a specified mask percentage; the weights of the linear layer are element-wise multiplied with the mask. This masking process introduces sparsity in the weight matrix, encouraging the model to focus on a subset of features during classification.

Data Augmentation

To enhance the model's robustness and generalization capabilities, VideoMAE incorporates various data augmentation techniques. These techniques include resizing the video frames to a fixed size of 224x224 pixels, random cropping of regions, random horizontal flipping, and random rotation. By applying these transformations, VideoMAE introduces diversity in the input data, exposing the model to a wide range of variations that it may encounter during inference. This augmentation strategy helps VideoMAE handle changes in appearance, viewpoint, and minor variations within the videos, ultimately leading to improved performance on unseen videos.

Conclusion

In conclusion, VideoMAE is an advanced video classification model that combines the strengths of the Timesformer architecture, 13D convolution, temporal pooling, weight initialization techniques, masking, and data augmentation. By effectively capturing spatio-temporal patterns, modeling long-term temporal dependencies, and enhancing the model's generalization capabilities, VideoMAE demonstrates improved accuracy and robustness in video classification tasks. The incorporation of weight initialization techniques, masking of the final linear layer, and data augmentation strategies further

contribute to the model's performance and generalization capabilities. VideoMAE offers a state-of-the-art solution for video classification, with potential applications in action recognition, surveillance, and video analysis domains.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, Accessed: May 24, 2023. [Online]. Available: https://arxiv.org/abs/1512.03385v1

[2] A. Vaswani *et al.*, "Attention Is All You Need," *Adv Neural Inf Process Syst*, vol. 2017-December, pp. 5999–6009, Jun. 2017, Accessed: May 24, 2023. [Online]. Available: https://arxiv.org/abs/1706.03762v5

[3] D. Tran, H. Wang, L. Torresani, J. Ray, Y. Lecun, and M. Paluri, "A Closer Look at Spatiotemporal Convolutions for Action Recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, Nov. 2017, doi: 10.1109/CVPR.2018.00675.

[4] "[1412.0767] Learning Spatiotemporal Features with 3D Convolutional Networks." https://arxiv.org/abs/1412.0767 (accessed May 24, 2023).

[5] W. Kay *et al.*, "The Kinetics Human Action Video Dataset," May 2017, Accessed: May 24, 2023. [Online]. Available: https://arxiv.org/abs/1705.06950v1

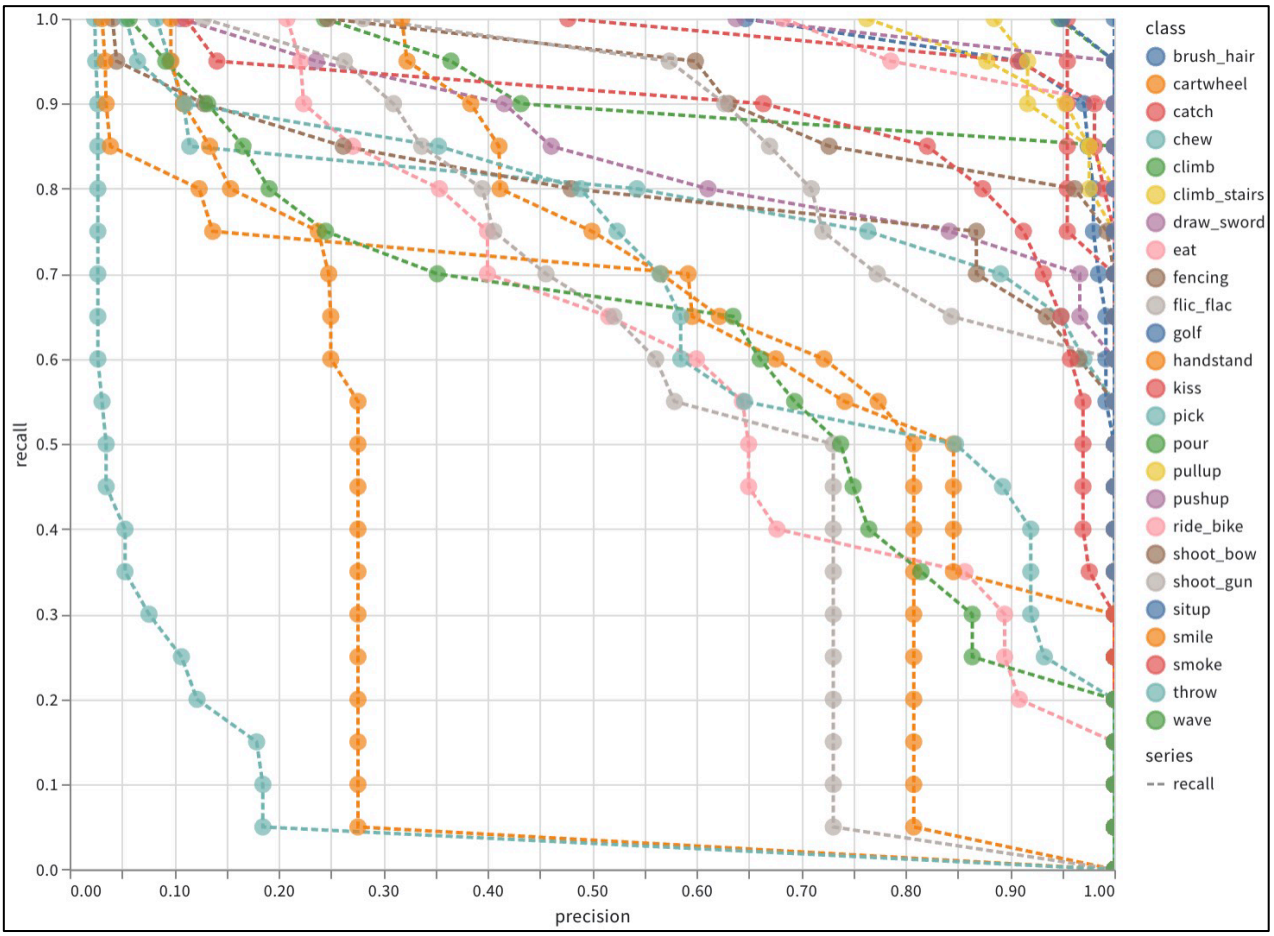
[6] G. Bertasius, H. Wang, and L. Torresani, "Is Space-Time Attention All You Need for Video Understanding?," Feb. 2021, Accessed: May 24, 2023. [Online]. Available: https://arxiv.org/abs/2102.05095v4

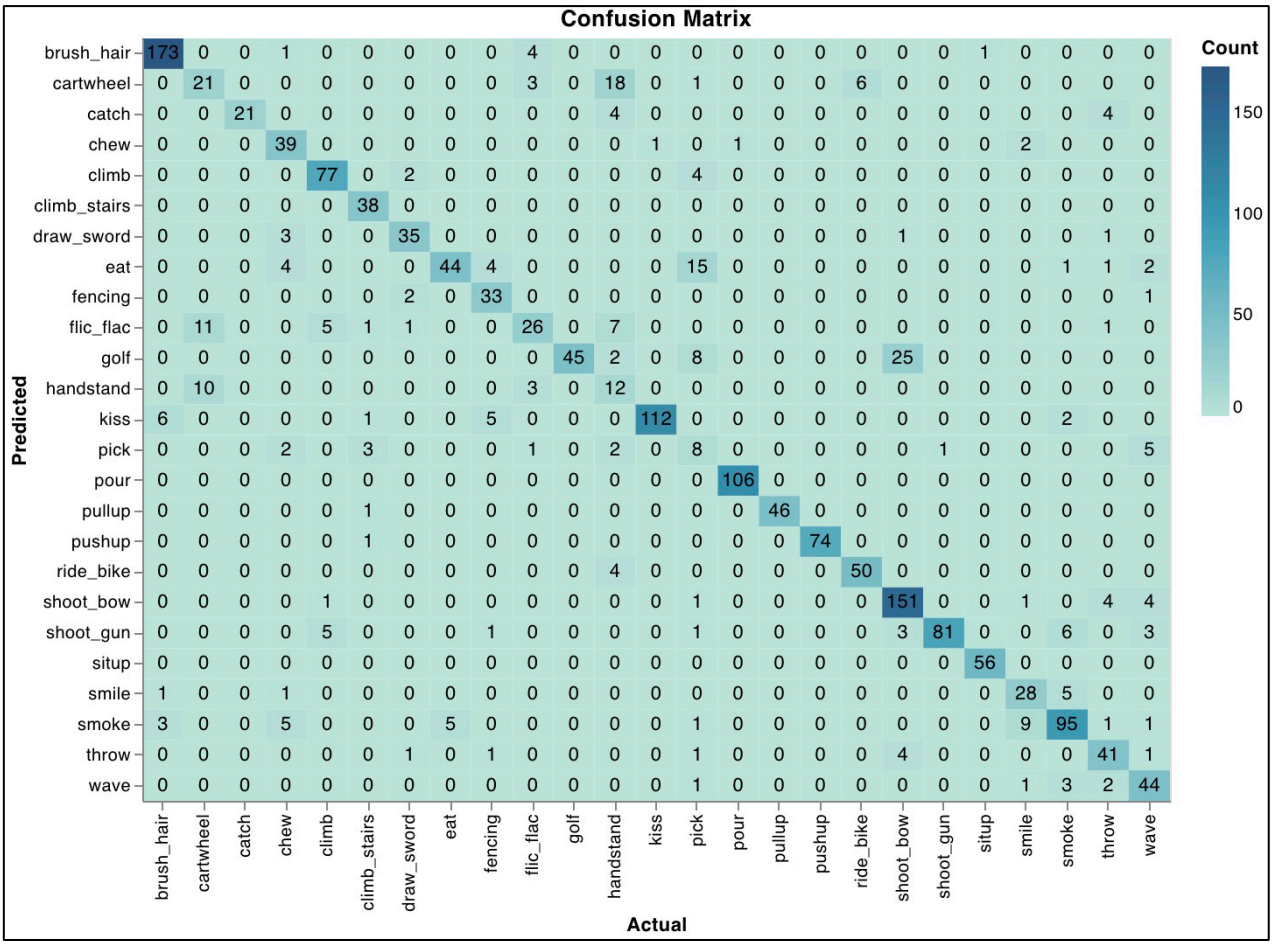
[7] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", Accessed: May 24, 2023. [Online]. Available: http://www.iro.umontreal.

[8] Z. Tong, Y. Song, J. Wang, and L. Wang, "VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training," Mar. 2022, Accessed: May 24, 2023. [Online]. Available: https://arxiv.org/abs/2203.12602v3

TimeSFormer – Rank 1 – 84.26%

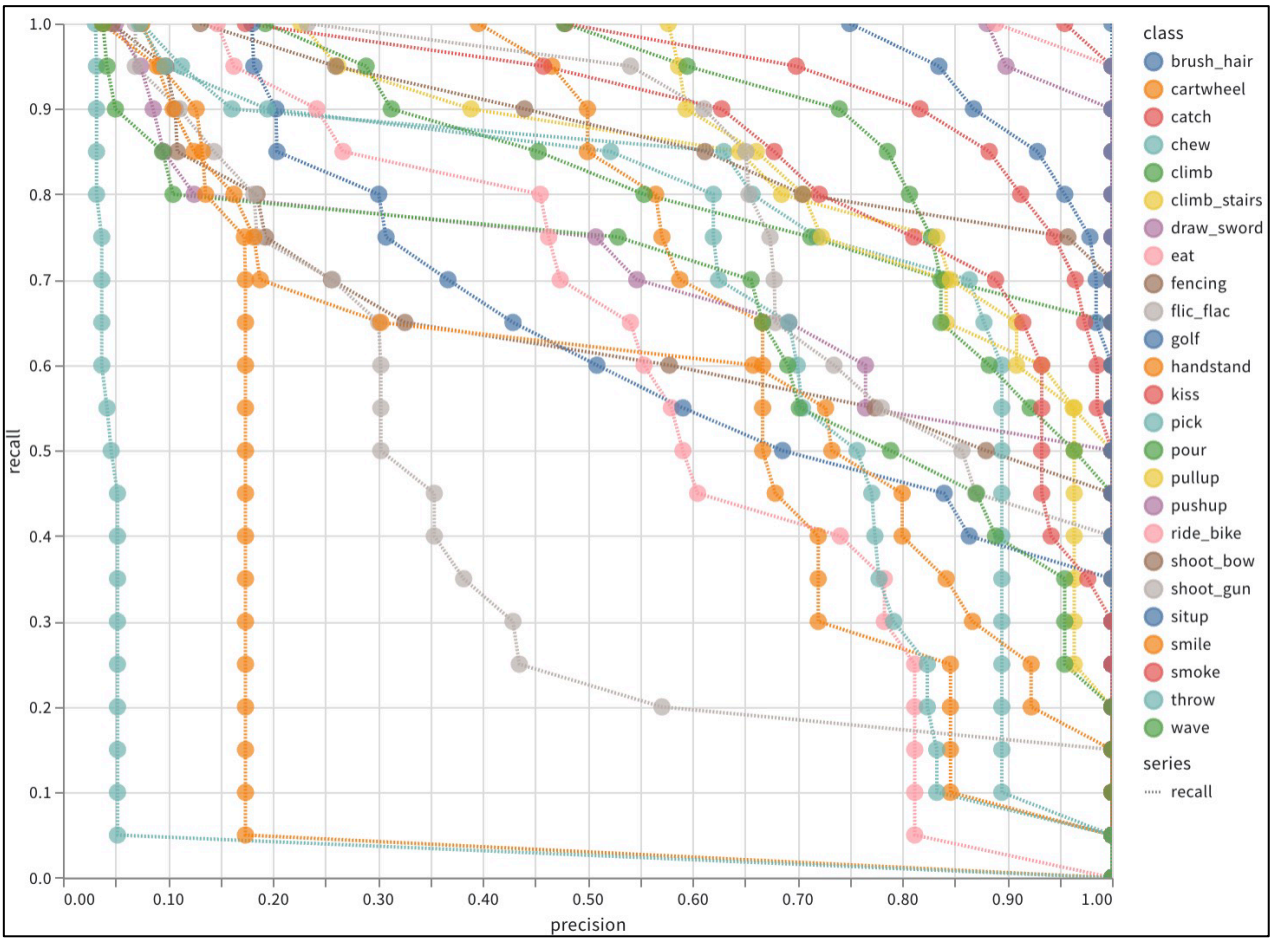
Precision & Recall & Confusion Matrix

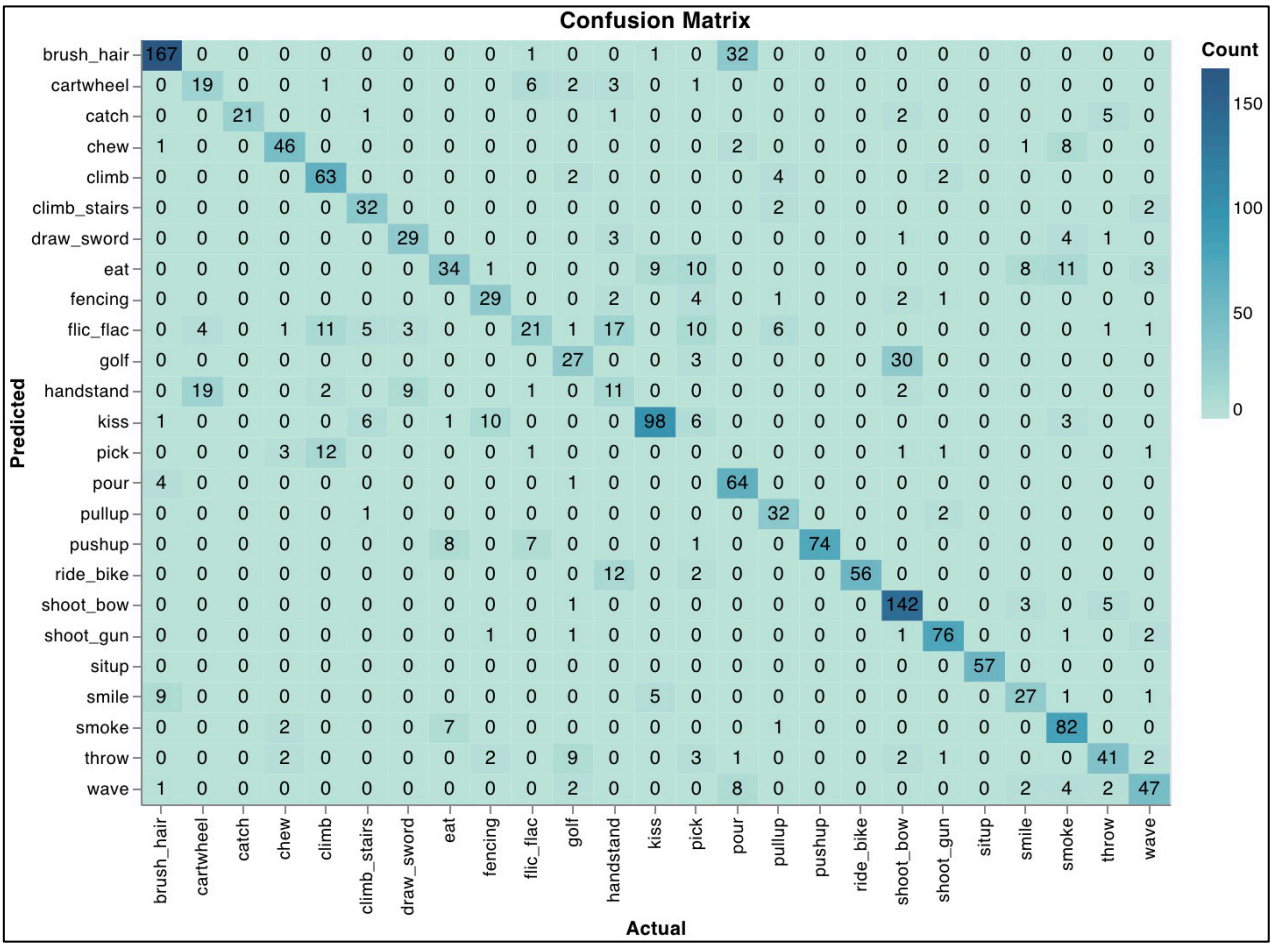




ResNet50 2D Input Learnable– Rank 1: 74.42%

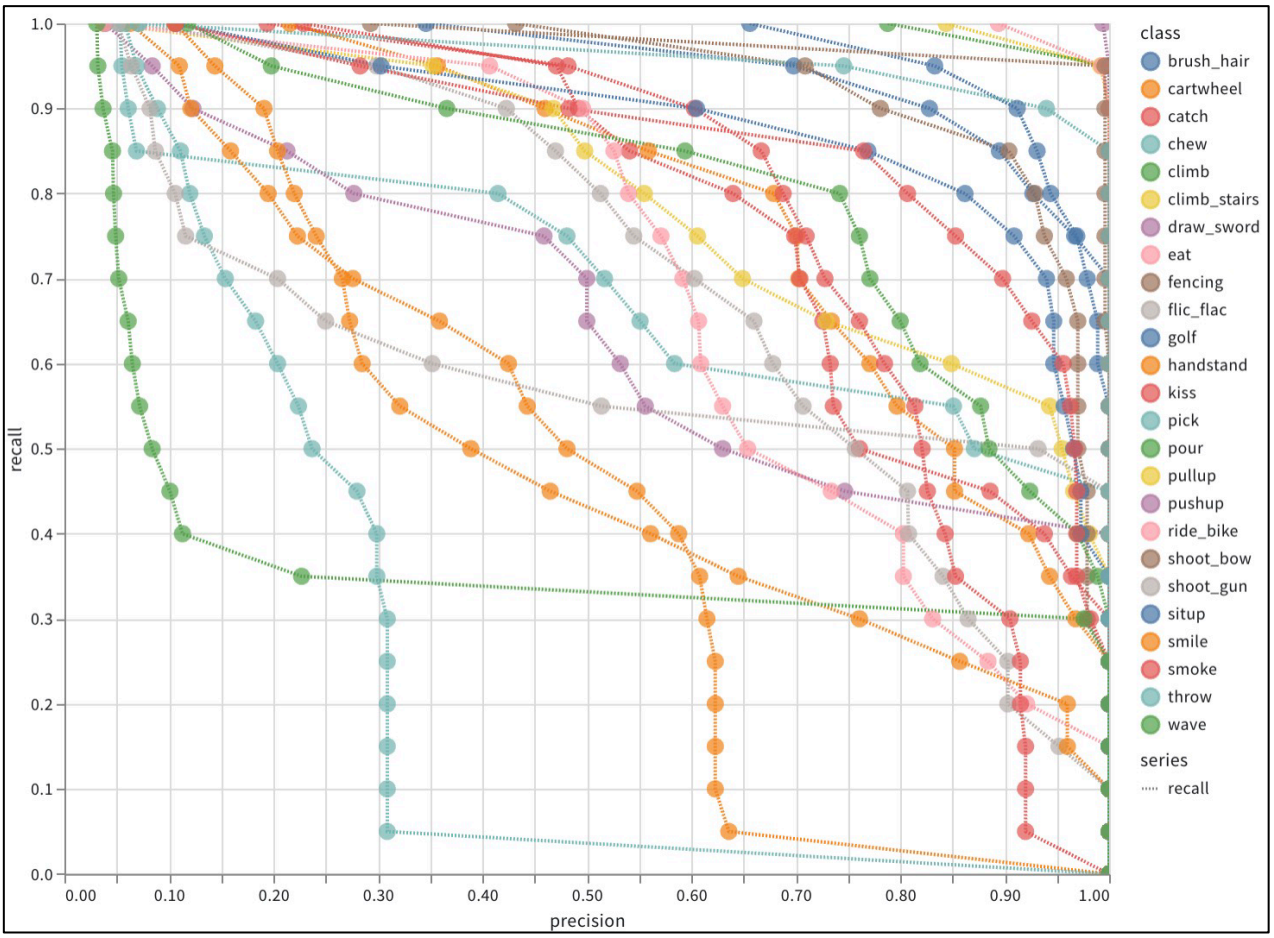
Precision & Recall & Confusion Matrix

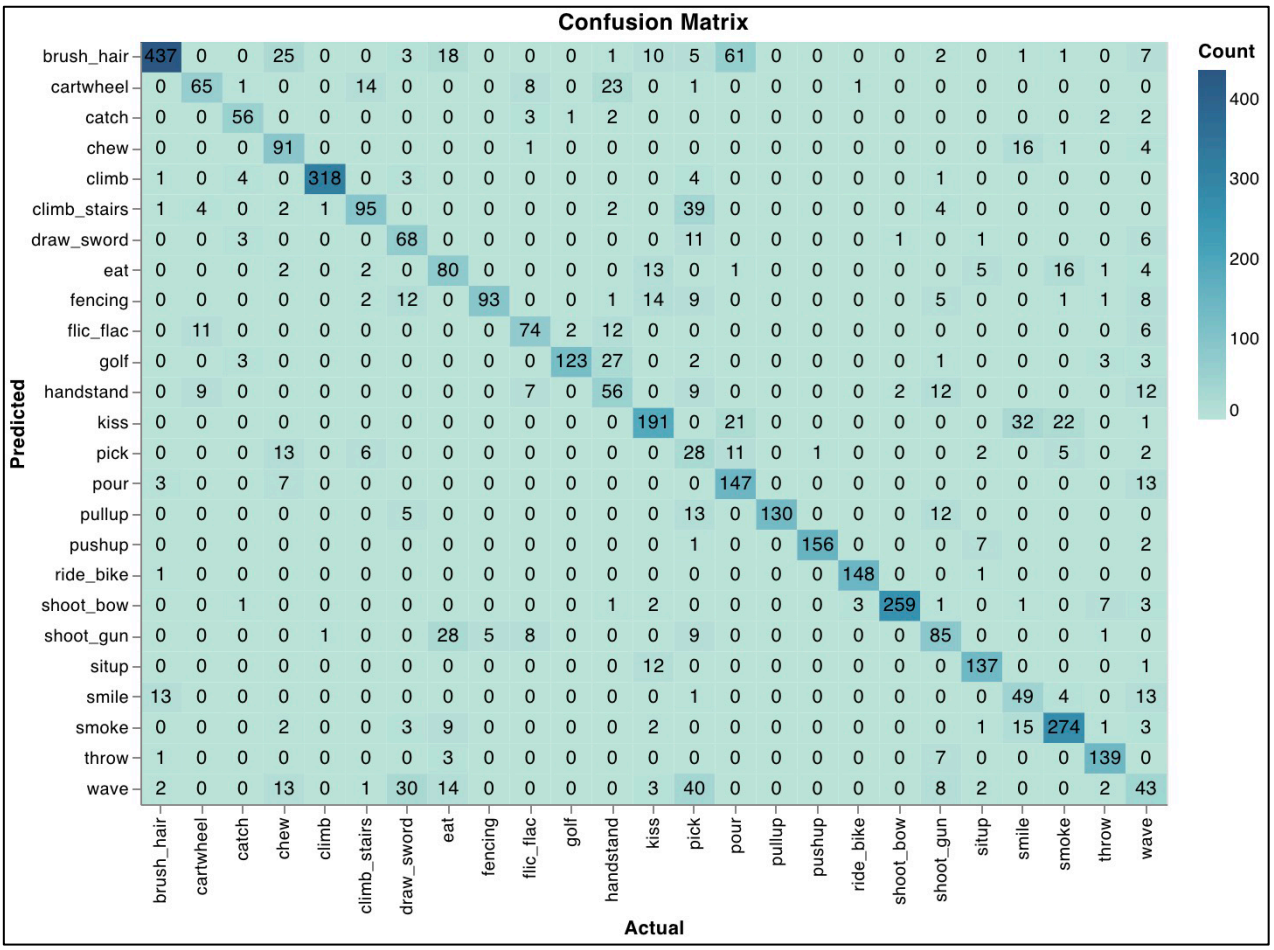




ResNet18 3D – Rank 1: 77.36%

Precision & Recall & Confusion Matrix





ResNet18 2+1D – Rank 1: 74.42%