Dancing in Style: Classifying Dance Videos By Style

Esteban Nathan Guzman
Stanford University
nate18@stanford.edu

Han Dao
Stanford University
handao@stanford.edu

Sophie Wu
Stanford University
swulee@stanford.edu

Abstract

The task of classifying human actions in videos has been a significant area of computer vision research, with dance being a culturally rich and stylistically diverse subset. In this paper, we present a novel approach to classify dance videos by style using an enhanced Two-Stream Inflated 3D ConvNet (I3D) model. We leverage the Kinetics-700 and Let's Dance datasets, combining them to create a robust dataset for training and evaluation. Our method and experimentation involve extensive hyperparameter tuning to improve the model performance. Additionally, we experimented with transfer learning by fine-tuning two different existing I3D models, originally based on the ResNet50 and the ResNet50 (NonLocal Dot Product) backbones, on our dance-focused dataset. Experimental results demonstrate the effectiveness of our approach, achieving accuracy in classifying various dance styles. This work not only aids in the preservation and study of dance but also provides insights into the distinguishing features of different dance styles.

1. Introduction

The task of classifying human action in videos has long since been a major area of computer vision research. While there exists many frameworks and models aimed at recognizing human action in general, we aim to focus in on a narrower subset: learning to recognize different styles of dance. Dance is an integral part of cultures all around the world, and each style of it is a product of unique regional, historical, and social influences. Being able to automatically and systematically classify these different forms of dance would not only help in the preservation and further study of them, but potentially also reveal new insights on what aspects make each dance style its own.

1.1. Problem Statement

More precisely, the problem we aim to address is as follows: given a set of unique video clips, each depicting human subjects performing a specific style of dance, how to best train and fine-tune an action recognition model to categorize these videos into the correct styles of dance with as high accuracy as possible. For each video clip input, our model outputs a label corresponding to a specific dance style that it believes the video to be of. The baseline accuracy we will compare our model to is one that has only been trained on a general human action dataset.

1.2. Literature Review

One of the most foundational papers in the area of human action recognition is "Two-Stream Convolutional Networks for Action Recognition in Videos" by Simonyan and Zisserman [8]. In it, Simonyan and Zisserman explore an extension to deep CNNs for still images in order to perform action recognition in videos. One of the key ideas is to have decompose the videos into two separate "streams": the spatial and the temporal. Each stream would then be learned by a separate deep ConvNet, which would then return separate scores that are combined by late fusion. This method was able to obtain impressive accuracy on the UCF-101 and HMDB-51 video datasets, achieving roughly 88% and 59.4% respectively.

This idea of Two-Stream ConvNets has also been applied and evaluated on the more specific task of classifying dance videos. In "Let's Dance: Learning from Online Dance Videos" [4], Castro *et al*. introduce a 1000 video dataset comprised of multiple dance styles, then compare the performance of a several state-of-the-art action recognition models on this new dataset. While the Two-Stream ConvNet models perform well on the original UCF-101 dataset, the performance is rather lower on the Castro *et al*.'s "Let's Dance" dataset, achieving roughly 69% accuracy. In their work, they also propose a Three-Stream CNN—in which the temporal stream is further split into two, one that works based off of optical flow, and one that is derived from skeletal pose estimation. This model also also achieves roughly 69% accuracy on the "Let's Dance" dataset.

In light of this, we decided to build upon a different method extending Two-Stream ConvNets for dance recognition. In their work, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset" [2], Carreira and Zisserman propose a Two-Stream Inflated 3D ConvNets (I3D), where ”filters and pooling kernels of very deep image classification ConvNets are inflated into 3D”. This allows for the model be ”very deep, naturally spatio-temporal classifiers”. The I3D models were able to outperform practically all other state-of-the-art action recognition models, including on the much larger and more challenging Kinetics dataset. Thus, we hope to leverage I3D model's architecture and apply it to our more specific task of classifying dance videos.

2. Related Work

In this section, we review existing literature on action recognition in videos, categorizing the approaches into three main categories: shallow high-dimensional encodings, two-stream convolutional networks, and 3D convolutional networks. We discuss the strengths and weaknesses of representative methods in each category and highlight the state-of-the-art techniques.

2.1. Shallow High-Dimensional Encodings

Shallow high-dimensional encoding methods have been widely used in early video action recognition research. These methods typically involve extracting local spatio-temporal features, such as Histogram of Oriented Gradients (HOG) and Histogram of Optical Flow (HOF), and encoding them using techniques like Bag of Features (BoF) or Fisher Vectors (FV).

Improved Dense Trajectories (IDT) by Wang and Schmid (2013) represents a significant advancement in this category. IDT extracts dense trajectories from videos and computes several descriptors, including HOG, HOF, and Motion Boundary Histograms (MBH), along the trajectories. These descriptors are encoded using Fisher Vectors to create a high-dimensional representation of the video. The method achieves impressive results on benchmarks like UCF-101 and HMDB-51 by compensating for camera motion and integrating multiple descriptors [10, 6]. However, the reliance on hand-crafted features and the computational cost of dense trajectory extraction are notable limitations.

2.2. Two-Stream Convolutional Networks

Two-stream convolutional networks, introduced by Simonyan and Zisserman (2014), leverage both spatial and temporal information from videos. The spatial stream processes individual video frames to capture appearance information, while the temporal stream uses dense optical flow to capture motion information. The outputs of both streams are combined, typically using late fusion, to recognize actions.

Two-Stream ConvNet is a pioneering approach in this category. It demonstrated that combining spatial and temporal streams significantly improves performance compared to using either stream alone. This architecture was competitive with state-of-the-art hand-crafted features on UCF-101 and HMDB-51 [8]. However, training two separate networks and computing optical flow for the entire dataset can be resource-intensive.

2.3. 3D Convolutional Networks

3D convolutional networks extend 2D convolutional networks by adding a temporal dimension to the convolutional filters, allowing the network to learn spatio-temporal features directly from raw video frames. This approach avoids the need for pre-computed optical flow and can potentially capture more complex motion patterns.

C3D by Tran et al. (2015) is a notable example of a 3D ConvNet. It applies 3D convolutions over 16-frame video clips, capturing both spatial and temporal information. The model achieved state-of-the-art performance on several benchmarks, including Sports-1M and UCF-101, and showed that 3D ConvNets can learn powerful spatio-temporal features without relying on hand-crafted descriptors [9]. The main challenge with 3D ConvNets is the significantly larger computational cost and the need for large-scale annotated video datasets for training.

I3D (Inflated 3D ConvNet) by Carreira and Zisserman (2018) further advances 3D ConvNets by inflating the filters of a 2D ConvNet pre-trained on ImageNet into 3D. This approach leverages the successful architecture design and learned parameters of 2D ConvNets, making it possible to train very deep spatio-temporal networks. Pre-training on the large-scale Kinetics dataset and fine-tuning on smaller benchmarks resulted in substantial performance improvements, achieving state-of-the-art results on UCF-101 and HMDB-51 [2].

Quo Vadis, Action Recognition? by Carreira and Zisserman (2018) introduces a new model, Two-Stream Inflated 3D ConvNet (I3D), based on 2D ConvNet inflation. This model expands filters and pooling kernels of deep image classification ConvNets into 3D, enabling seamless spatio-temporal feature extraction from videos while leveraging successful ImageNet architectures and parameters. Pre-training on Kinetics significantly boosts performance on smaller benchmarks [2].

Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification by Feichtenhofer et al. (2019) explores various architectures for balancing speed and accuracy in video classification. They highlight the potential of mixed Convolutional Neural Networks (CNNs) that combine 2D and 3D convolutions to achieve state-of-the-art performance with optimized computational efficiency [5].

2.4. State-of-the-Art and Future Directions

The current state-of-the-art in action recognition is dominated by deep learning approaches, particularly two-stream and 3D convolutional networks. The introduction of large-scale video datasets like Kinetics has enabled the training of more complex models, leading to significant performance gains. However, the computational cost and the need for extensive annotated data remain challenges.

In summary, two-stream and 3D ConvNets represent the most promising directions for action recognition in videos. The use of pre-trained 2D ConvNets as a starting point for 3D models, as seen in I3D, is a particularly clever approach that leverages existing image classification knowledge. Future research may focus on optimizing these models for efficiency and exploring new ways to capture and utilize spatio-temporal information in videos.

3. Dataset/Features

3.1. Creating the Datasets

To develop a comprehensive dataset for dance video classification, we utilized two primary sources: the Kinetics-700 dataset and the Let's Dance dataset. The following steps outline the process of creating and refining our dataset:

3.1.1 Filtering Kinetics-700

We began by filtering the Kinetics-700 dataset to extract videos specifically related to dance. This involved identifying and selecting videos labeled with dance-related actions. The identified dance categories included 19 unique labels.

3.1.2 Combining with Let's Dance Dataset

Next, we incorporated the Let's Dance dataset, which contains detailed annotations for various dance styles. We mapped the dance labels from Let's Dance to align with those from Kinetics-700, ensuring consistency across our combined dataset.

3.1.3 Data Augmentation

To increase the size and variability of our dataset, we applied several data augmentation techniques to the training split. These techniques included:

- **Horizontal Flip**: Mirroring the video frames horizontally.
- **Vertical Flip**: Mirroring the video frames vertically.
- **Color Jitter**: Randomly adjusting the brightness and saturation of the frames.
- **Noise Addition**: Adding random noise to the frames to simulate different video qualities.

These augmentations could potentially enhance the model's ability to generalize to various real-world scenarios.

3.1.4 Splitting the Dataset

The final merged dataset was split into three sets:

- **Training Set (90%)**
- **Validation Set (5%)**
- **Test Set (5%)**

This split ensured a sufficient amount of data for training while maintaining separate sets for validation and testing to evaluate the model's performance.

For the purpose of transfer learning and finetuning, we also created smaller subsets of the dance videos filtered out solely from the Kinetics-700. The following subsets for finetuning were created as follows, where each video in the original dataset had an equal probability of being present in the subset. Hyperparameter tuning was not performed for our experiments with transfer learning, so the subsets are split solely into training and test, and are as follows.

Subset 1 (1500 Videos)

- **Training Set (93%)**
- **Test Set (7%)**

Subset 2 (525 Videos)

- **Training Set (90%)**
- **Test Set (10%)**

The two subsets differ in size in an effort to experiment with the tradeoff between training time, epochs, and memory constraints.

3.1.5 Annotations

We generated annotation files for each split, listing the video names and their corresponding integer-encoded labels. This step was impoertant for the MMAction2 pipeline to process and classify the videos accurately.

The class labels were mapped to the following dance styles:

- 0: belly dancing
- 1: breakdancing
- 2: country line dancing

- 3: cumbia
- 4: dancing ballet
- 5: dancing charleston
- 6: dancing gangnam style
- 7: dancing macarena
- 8: jumpstyle dancing
- 9: krumping
- 10: moon walking
- 11: mosh pit dancing
- 12: robot dancing
- 13: salsa dancing
- 14: square dancing
- 15: swing dancing
- 16: tango dancing
- 17: tap dancing
- 18: zumba

Dataset	Year	Actions	Clips	Total	Videos
Kinetics-600	2018	600	min 600	492,000	492,000
Kinetics-700	2019	700	min 700	650,000	650,000
Let's Dance	2020	20	avg 100	10,000	10,000

Table 1. Details of the datasets used in our research.

4. Methods

4.1. Overview

In this section, we describe the learning algorithms and the proposed model utilized in our study. Our primary model, the Two-Stream Inflated 3D ConvNet (I3D), builds on state-of-the-art image classification architectures by extending them into the spatio-temporal domain. This section includes the mathematical formulations of our input, output, and loss functions, and details the modifications made to existing models to enhance performance on action recognition tasks.

4.2. Input and Output

The input to our model consists of video frames sampled at a rate of 25 frames per second. For the RGB stream, we use 64-frame snippets of size 224×224 pixels. For the optical flow stream, we compute flow fields for 64 consecutive frames.

We used the K700-2020 dataset, which includes 700 human action classes, each with at least 700 video clips from different YouTube videos. Each clip is approximately 10 seconds long. For our study, we filtered out only the dance videos from this dataset [3, 1]. Additionally, we incorporated the Let's Dance dataset, which contains detailed annotations for various dance styles. We mapped the dance labels from Let's Dance to align with those from Kinetics-700, ensuring consistency across our combined dataset.

The output is a probability distribution over the selected action classes, computed by applying a softmax activation to the final layer of the network.

4.3. Two-Stream Inflated 3D ConvNet (I3D)

The I3D model inflates 2D ConvNet filters and pooling kernels into 3D, enabling the network to learn spatio-temporal features directly from video data. The architecture is based on the Inception-v1 network, which we extend by converting 2D filters $N \times N$ into 3D filters $N \times N \times N$. The model comprises two parallel streams: one for RGB frames and one for optical flow.

Mathematical Formulation: Let $x^{(t)}$ denote the input frame at time t. The convolution operation for a 3D filter W is defined as:

$$y^{(t)} = \sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{l=1}^{k} W_{ijl} x_{p+j,q+l}^{(t+i)}$$

where k is the kernel size, and (p, q) denotes the spatial dimensions.

The pooling operation for a 3D kernel is defined similarly:

$$y_{pool}^{(t)} = \max_{i,j,l} \left(x_{p+j,q+l}^{(t+i)} \right)$$

Loss Function: The loss function used is the cross-entropy loss, defined as:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

where C is the number of classes, y_i is the ground truth label, and $\hat{y}_i$ is the predicted probability.

4.4. Training Procedure

We train the I3D model on the K700-2020 dataset, focusing specifically on dance videos. To implement our training

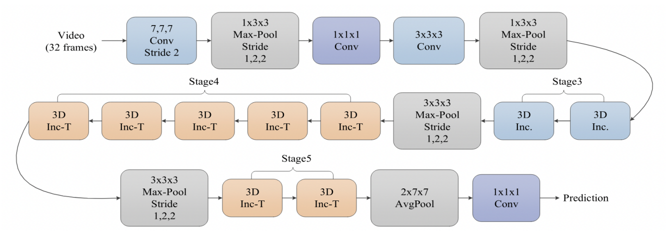

Figure 1. Diagram of the I3D architecture.

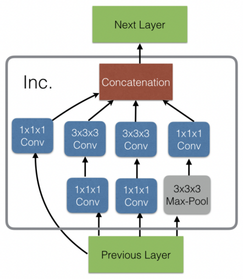

Figure 2. Detailed view of the inception submodule within the I3D architecture.

procedures, we built on top of MMAction2 [7], an open-source toolbox for video understanding based on PyTorch. More specifically, we utilized some of their modules and API calls to augment videos and create and modify new layers to our models.

Algorithm Steps:

1. **Initialization:** Inflate 2D filters from a pre-trained Inception-v1 model into 3D filters.

2. **Data Augmentation:** Apply random cropping, resizing, and flipping to the input frames.

3. **Forward Pass:** Compute the spatio-temporal features using 3D convolutions and pooling.

4. **Loss Computation:** Calculate the cross-entropy loss between the predicted and true labels.

5. **Backpropagation:** Update the network weights using gradient descent with momentum.

6. **Testing:** Evaluate the model on test sets by averaging predictions across video frames.

In addition to this training procedure, we performed hyperparameter tuning using Optuna to identify the optimal learning rate, dropout rate, and optimizer for our model. The hyperparameter tuning process involved running 4 trials, with each trial executing for 10 epochs. We used a train-validation split to evaluate model performance and selected the best model based on the top-1 accuracy on the validation set.

We also further experimented with transfer learning, which involved finetuning on existing I3D models from the work done in “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset” [2] and ”Non-Local Neural Networks” [11]—which both utilize a ResNet50 based backbone for feature extraction, ImageNet for pretraining, and Kinetics-400 as its main train set—on a small subset of our dance-focused dataset. Here we utilized MMAction2, modifying and building on top of their existing configuration files in order to create training and testing pipelines similar to those used by Carreira and Zisserman in [2] and Wang *et al*. in ” [11].

Taking those pipelines as a base, we then followed the same training procedure as stated above in Section 4.4, with the exception of step **1. Initialization**. Rather, in order to leverage transfer learning, initialization was performed as follows:

1. **Initialization:** Load in pre-trained weights from an existing I3D model. Freeze the lower layers, and re-initialize the final classification layer to output classifications for our custom dance-specific dataset.

From there, training proceeded as previously stated.

5. Experiments and Results

5.1. Evaluation Metrics

We primarily used the following metric to evaluate our model’s performance:

- **Accuracy:** The proportion of correct predictions out of the total number of predictions.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

5.2. Hyperparameter Tuning

We performed hyperparameter tuning using Optuna to identify the optimal learning rate, dropout rate, and optimizer for our model. The hyperparameters were chosen based on the following considerations:

- **Learning rate (lr):** We searched in the range of 1×10^{-5} to 1×10^{-2} on a logarithmic scale to balance the speed of convergence and the stability of training.

- **Dropout rate (dropout):** We varied the dropout rate between 0.2 and 0.7 to prevent overfitting while ensuring sufficient model capacity.

- **Optimizer:** We evaluated both Adam and SGD optimizers to determine the best optimization strategy for our model.

The hyperparameter tuning process involved running 4 trials, with each trial executing for 10 epochs. We used a train-validation split to evaluate model performance and selected the best model based on the top-1 accuracy on the validation set.

The best performing hyperparameters were:

- **Learning rate:** 7.87×10^{-4}
- **Dropout rate:** 0.2966
- **Optimizer:** SGD

5.2.1 Hyperparameter Tuning Results

Table 2 summarizes the training and validation losses, as well as the validation accuracy over 10 epochs:

Epoch	Training Loss	Validation Loss	Validation Accuracy (%)
1	2.90550	2.81120	15.32
2	2.88860	2.79430	15.47
3	2.87170	2.77740	15.41
4	2.85480	2.76050	15.48
5	2.83790	2.74360	15.44
6	2.82100	2.72670	15.45
7	2.80420	2.70980	15.46
8	2.78730	2.69300	15.43
9	2.77040	2.67600	15.49
10	2.75350	2.65910	15.45

Table 2. Training and Validation Losses and Accuracy over 10 Epochs

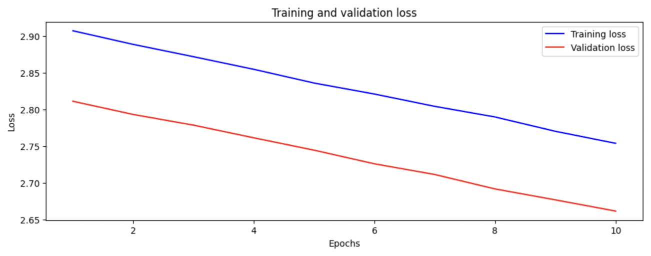

Figure 3. Training and Validation Loss

The training and validation losses exhibit a decreasing trend, indicating that the model is learning effectively. The validation accuracy starts at 12.73% and increases gradually, suggesting that the model is improving its ability to generalize to unseen data.

Figure 3 shows that the training loss consistently decreases, suggesting that the model is effectively learning from the training data. Similarly, the validation loss also decreases, indicating that the model's performance on unseen data is improving. The gap between the training and validation loss is relatively small, suggesting that there is no significant overfitting. To prevent overfitting, we employed dropout regularization with the tuned dropout rate. Additionally, we monitored the validation loss and accuracy to detect any signs of overfitting early.

The following table summarizes the different hyperparameters tried during the hyperparameter tuning process and their respective accuracies, including the original run before hyperparameter tuning. These are the hyperparameters tried other than the one that gave the top accuracies:

Trial	Learning Rate	Dropout Rate	Optimizer	Validation Accuracy (%)
Original	0.00100	0.50000	Adam	7.42000
0	0.00079	0.29661	SGD	9.61000
1	0.00707	0.53317	Adam	7.85000
2	0.00236	0.21482	SGD	10.84000
3	0.00049	0.66175	SGD	11.87000

Table 3. Hyperparameter Tuning Results

5.3. Transfer Learning and Finetuning

Finetuning was performed by loading pretrained weights from two different I3D models, referred to as ResNet50 and ResNet50 (NonLocal Dot Product) from 'Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset" [2] and "Non-Local Neural Networks" [11] respectively. Finetuning for both models followed the same training procedure and settings for data augmentation. However, experiements were done with the size of the dataset the two pretrained models were done on, as well as the hyperparameters.

The following finetuning was done on the **ResNet50** pre-trained model, using **Subset 1** data as described in Section 3. Training was performed with the following settings and produced the following results:

pretrain	lr	epochs	top1 acc %	top5 acc %
ResNet50	1×10^{-2}	3	51.61	82.80

Similarly, the following finetuning was done on the **ResNet50 (NonLocal Dot Product)** pre-trained model, using **Subset 2** data (525 videos) as described in Section 3. Training was performed with the following settings and produced the following results:

pretrain	lr	epochs	top1 acc %	top5 acc %
ResNet50 (NonLocal)	Scheduled	10	48.28	79.31
ResNet50 (NonLocal)	Optimal 7.87×10^{-4}	10	56.90	86.31

For the learning rate, we experimented with two different variations.

- **Scheduled:** A multi-step learning rate that started at 1×10^{-4} in epoch 1 and was multiplicatively decreased by $\gamma = 0.1$ at epochs 4 and 8.
- **Optimal:** This was the best learning rate (7.87×10^{-4}) found in the Section 5.2 Hyperparameter Tuning, and was used in conjunction with the SGD optimizer.

The predictions of the ResNet50 NonLocal model trained with the optimal hyperparamaters are shown below. We can see that the most significant confusions are labels 13 (salsa dancing) with 15 (swing dancing) and 16 (tango dancing). This makes sense, as these three styles are qualitatively similar. All three are partner or social dances and salsa and tango share Latin influences. Similarly, the next largest source of confusion, label 1 (breakdancing) and label 9 (krumping), also share some stylistic influences, as both are styles of street dance that fall under the hip-hop umbrella.

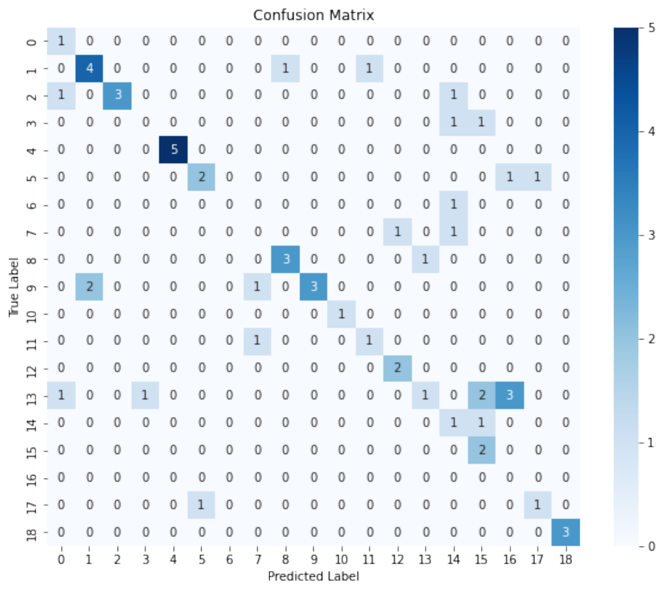

Figure 4. Confusion Matrix for ResNet50 Nonlocal with Optimal Hyperparamaters

However, despite these confusions, the models show promise. We see significant gains in accuracy when finetuning on top of existing I3D models. It also appears that models trained on smaller datasets can also perform similarly to or even better than those trained on larger datasets, given more training time and the "correct" hyperparameters.

6. Conclusion and Future Work

In this study, we explored various experimentations, including hyperparameter tuning, to improve the performance of our model. Through extensive experimentation, we identified optimal hyperparameters that significantly enhanced model performance. The key hyperparameters included a learning rate of 7.87×10^{-4}, a dropout rate of 0.2966, and the use of the SGD optimizer.

The model showed consistent improvement in training and validation losses, with the validation accuracy fluctuating around 15.45% over 10 epochs. The results indicate effective learning and generalization without significant overfitting.

Moreover, we also saw that transfer learning is indeed applicable to this specific domain, and that general action recognition models can be finetuned with decent success to classify and recognize more specific variations of actions—such as styles of dance.

For future work, we would explore the following:

- **Data Augmentation:** We were in the process of augmenting the dataset using techniques such as vertical flipping and color jittering. However, due to time constraints, we were unable to generate the full augmented dataset and experiment with it. Implementing and experimenting with these data augmentation techniques in the future could significantly increase the robustness and diversity of our dataset, potentially leading to improved model performance.
- **Extended Training Time:** Increasing the number of epochs could further enhance the model's performance, allowing it to learn more complex patterns in the data.
- **Larger Hyperparameter Search Space:** Conducting a more extensive search over a broader range of hyperparameters could potentially yield better results.
- **Ensemble Methods:** Combining multiple models could improve robustness and accuracy, leveraging the strengths of different algorithms.
- **Further Pretraining and Finetuning:** More extensive finetuning on top of different pretrained models that we did not explore here could yield better results.
- **Additional Regularization Techniques:** Exploring further regularization methods, such as weight decay and batch normalization, could further reduce overfitting.

7. Appendices

The two pretrained models used in finetuning and their descriptions can be found here in the I3D documentation. The ResNet50 model referred to here is the fourth row of the table provided in the README; the ResNet50 (NonLocal) model is the first.

8. Contributions and Acknowledgements

- Sophie: Transfer learning, pretraining and finetuning explorations; quantitative and qualitative analysis of final predictions; introduction, literature review, methods
- Han: Hyperparameter tuning, literature review, methods, quantitative analysis of results
- Nate: Dataset preparation, processing, and filtering; additional explorations with dataset augmentation; dataset and features

8.1. Public Code Utilized

We made use of the mmaction2 open-source project which provided essential tools and frameworks for our work in action classification and pose estimation.

8.2. Non-CS231N Collaborators

There were no non-CS231N collaborators involved in this project.

References

[1] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman. A short note on the kinetics-700-2020 human action dataset. *arXiv preprint arXiv:2010.10864*, 2020. 4

[2] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017. 1, 2, 5, 6

[3] J. Carreira and A. Zisserman. Kinetics-700 dataset. *arXiv preprint arXiv:1907.06987*, 2019. 4

[4] D. Castro, S. Hickson, P. Sangkloy, B. Mittal, S. Dai, J. Hays, and I. A. Essa. Let's dance: Learning from online dance videos. *CoRR*, abs/1801.07388, 2018. 1

[5] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. *IEEE International Conference on Computer Vision (ICCV)*, pages 6202–6211, 2019. 2

[6] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. 2

[7] MMAction2 Contributors. OpenMMLab's Next Generation Video Understanding Toolbox and Benchmark, July 2020. 5

[8] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014. 1, 2

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015. 2

[10] H. Wang and C. Schmid. Action recognition with improved trajectories. *IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013. 2

[11] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *CVPR*, 2018. 5, 6