Temporal Human Action Recognition

Tran Hoang Quoc Anh, Le Hong Duc, Nguyen Quang Duc, La Dai Lam, Luu Trong Nghia
School of Information and Communication Technology
Hanoi University of Science and Technology
{anh.thq200044, duc.lh204874, duc.nq204876, lam.ld204918, nghia.lt204888}@sis.hust.edu.vn

Abstract

Human action understanding is one of the most important applications in computer vision, provided with a wide range of use in some fields such as healthcare, surveillance, security, and military technology. It consists of the computer's ability to recognize, understand and predict even the most complex action. However, there are many challenges to dealing with that problem such as large storage requirements for storing the dataset and the combination of spatial and temporal in videos themself. There are also many kinds of human action recognition, but in this project, we focus on temporal human action recognition whose input is a short video and we want to recognize the action in the video. To do that, we tried to use different deep learning models, highlight the advantages and disadvantages of each model, and demonstrate by training them on two benchmark datasets, namely UCF101 and HMDB51. Furthermore, we performed extensive and intensive analysis on models and also tried a variety of approaches to reach better results. After trial and error, we finally got 84.11% and 54.35% accuracy on the first split test set of each dataset.

1 Introduction

Besides image understanding, human action understanding is also one of the most important applications in computer vision. Its usage is widespread in every aspect of our life, as by just detecting and understanding what a person is doing, it also provides fundamental knowledge and serves as the backbones for other deep learning systems. Moreover, it can provide behavior analysis for a patient, increase the interaction between humans and robots, make a surveillance system that acts on danger, etc. But before understanding or even predicting an action, we need to recognize the action first. Over the last decade, the technology behind human action recognition has been growing at a rapid rate, especially number of the deep-learning-based models. As time goes on, there will be even more models that can do the job. But what is better at detecting what, what are the advantages, and disadvantages, and why do we use this model over that model? To answer these questions, we will test out five different models on two distinct datasets.

For the data, we used UCF101[1] and HMDB51,[2] since both of them are ubiquitous as well as the total number of videos is not too large, and all of them are short videos. They are also the benchmark datasets for many state-of-art models. For more details, we also have a separate section 2 to describe how we prepare them for modeling. Overall, we extract short clips from videos, each clip containing 16 frames (images) or each clip containing 5 frames, and also apply augmentations for each clip depending on which models we used.

IT3320E: Introduction to Deep Learning, 20221, Hanoi University of Science and Technology

For the models, we chose Late Fusion[3] as our baseline model because it is the most intuitive way of thinking as we want to train on multiple frames. After that, we chose Long-term Recurrent Convolutional Networks (LRCN)[4] next as all of our frames are in the temporal dimension. We then tried to use Vanilla 3D Convolutional Networks (C3D)[5] and Inflated 3D Convolutional Networks (I3D)[6] which could utilize the temporal and spatial information at the same time. Finally, the last model we implemented is Non-local Neural Networks.[7] For more information about the models, and why we chose them, we will describe them carefully in section 3.1.

In order to evaluate what models have learned from the datasets, we also tried to apply different methods for visualizing the results in section 3.2. We also describe training strategies and how we trained models with some tools and free online environments such as Google Colab and Kaggle in section 4.1 and present and analyze our results of each model for each dataset with different strategies in section 4.2.

However, we also met many obstacles, but among those, there are three that were hard to overcome. The first hurdle came in the shape of volume, for our datasets, which are only about 10GB and consist of short videos, but after frames extraction, the size shot up to 160GB of images. And this also leads to our second challenge, huge computational requirements. We have to deal with not only the big input size but also the enormous number of parameters for our models, as most of them are in the 3D-based model, a larger step from the 2D ones. The time of training for a model takes at least 6 hours at best and 12 hours at worst. Furthermore, there are no simple ways to increase our hardware capability. For the final one, our models need to have a good sense of context, different videos have different spatial information and it can change based on the movement of the camera. Two videos of the same action at different angles can result in different activity detection.

2 Dataset and Features

This section's work will focus on datasets for training models. HMDB-51 and UCF-101, the two datasets we utilized, will be discussed in detail in subsection 2.1. Subsection 2.2 will also present the techniques we take to process the data for model training.

2.1 Dataset

The Human Motion DataBase (HMDB-51)[2] was released in 2011. There are 6766 videos in total, grouped into 51 different categories of human actions. Videos were collected from a variety of sources, primarily movies, but also from public sources such as YouTube. Figure 1 illustrates a random frame of each HMDB-51 action class. Each type of action included at least 101 videos, and the number of videos in each category mostly varies between 101 and 160. Walk, Turn, and Run are three groups containing unusually large volumes of data, with 548, 240, and 232 videos, respectively. Videos are collected from many sources so they may differ in characteristics such as duration, and aspect ratio. However, they all share the same frame rate of 30 frames per second and a height of 240 pixels.

UCF101[1], which was released in 2012, is an extension of the UCF50 dataset. UCF101 contains 13320 videos, which is approximate twice as many as HMDB-51. These videos are classified into 101 categories of human action, with a minimum of 100 videos within each class. They are taken from Youtube and share the same 320x240 display resolution and frame rate of 25 frames per second. We show a general visualization of the UCF101 dataset in Figure 2.

2.2 Data Preprocessing

One of the difficulties when solving the problem of human action recognition is how to handle the input. A single raw video could be up to 15 seconds with approximately 30 frames per second, if using the whole video as the input for our model, it will be computationally inefficient. Furthermore, because each video has a variable amount of frames, the inputs can not all have the same dimension, which also challenges model implementation. This implies that we cannot utilize raw videos as training inputs. The solution to the input problem is that training on short clips with low FPS and low spatial resolution. More precisely, from an untrimmed video, we extract some of its frames and create a short clip using these frames. The number of frames and spatial resolution in short clips taken from full-length videos will be identical. Following that, we will make sure the dimensions of inputs are

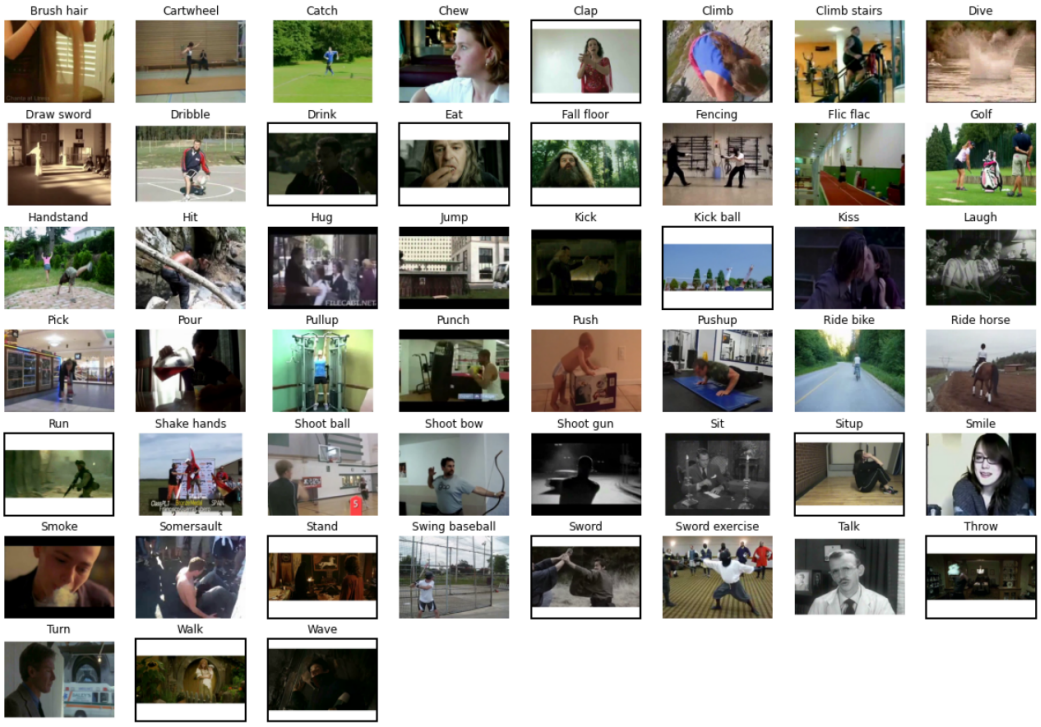


Figure 1: 51 human actions included in HMDB-51 shown in a sample frame. The black frame borders specify the difference in aspect ratios of HMDB-51 videos.

consistent. When testing, we run the model on various clips before averaging the results to arrive at the final predictions.

As we carefully examined the data set and began extracting the frames, we became aware of a number of factors that might limit the model's performance. Firstly, it is challenging to identify the action's beginning and ending positions given the length of video.

As illustrated in figure 3a, we have taken 8 equally spaced frames from a video that lasted for 13 seconds. A person standing with their hands is displayed in the first six frames, and this image resembles the frames of class HandstandWalking (as shown in figure 3b). The action is only performed in the final seconds, and the actual label of the video is ParallelBars. This demonstrates how the model's performance might be harmed by the irrelevant first frames.

An another challenge when extracting clips from the whole video is the presence of camera motion. This occurs frequently and accounts for a fairly large portion of videos of HMDB51 dataset. When the video has camera movement, the background will change and will affect the local motion calculation. This makes the model more sensitive and difficult to identify important features. Figure 4a illustrates how the background changes as the camera tracks the ball's motion.

Finally, another factor that affects the extraction of frames is the video's scene transition. A scene transition is shown in Figure 4b, where a character wearing a red shirt is recorded in the first two frames, and a new character with a different background is captured in the remaining frames. It is possible to conclude that the scene changes between frame two and frame three of the figure. These are the three key issues that should be properly taken into account when processing data. We will now discuss some of the techniques we use to prepare the data for training.

Talking about extracting frame, taking the five frames separated by a fixed time interval is the easiest way to obtain the frames. This allows us to cover the whole length of the video. This strategy, however, is ineffective when the raw video contains transitions or has camera movement. The second technique we used involving taking a shorter clip with 16 sequential frames . When the background changes, we believe this approach becomes more robust. Finally, we apply another data processing technique that combines the first two strategies. In particular, one video can be used to produce five clips, where

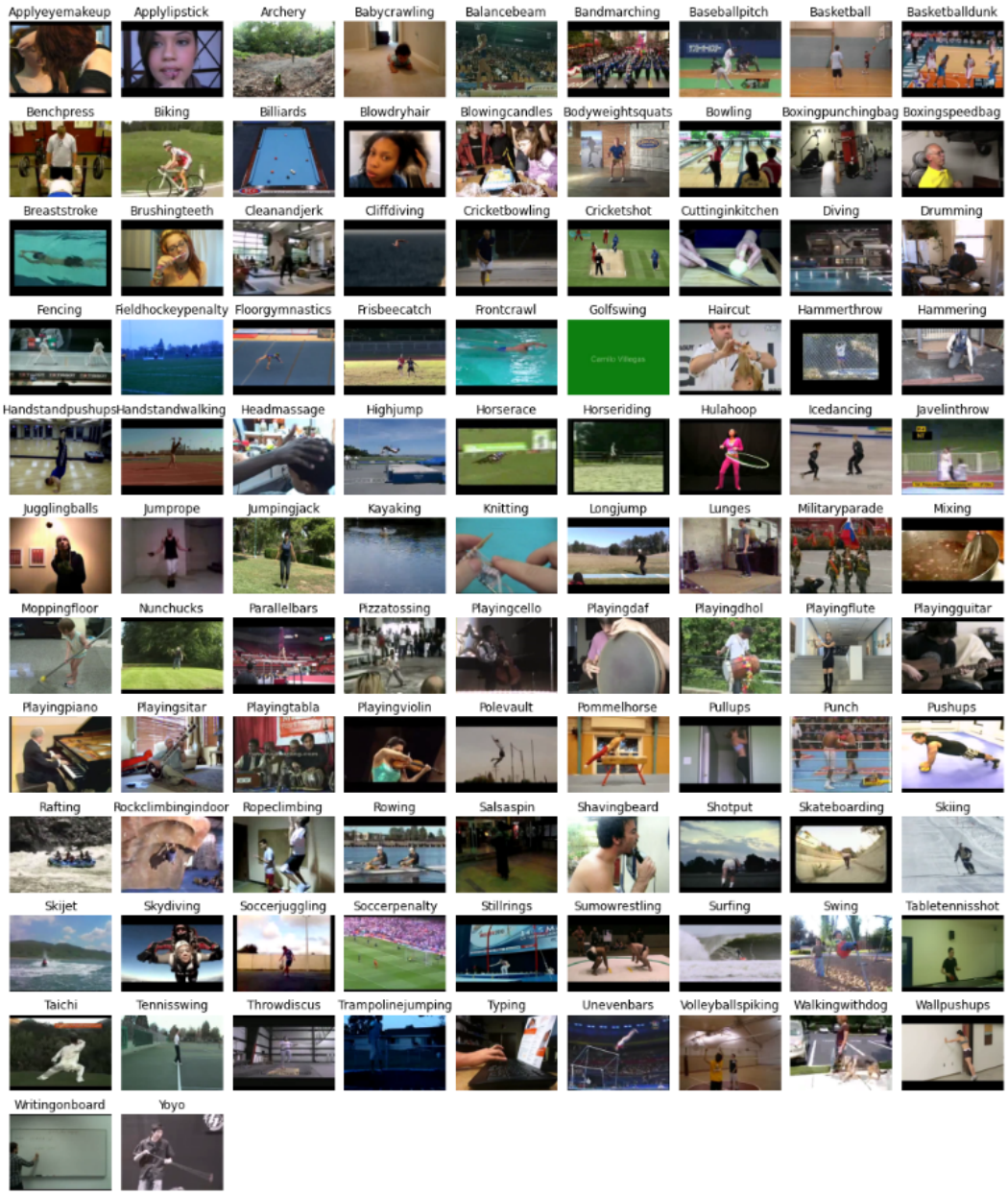


Figure 2: 101 human actions included in UCF101 shown in a sample frame.

each contains 16 consecutive frames. These clips are separated by a predetermined amount of time. We would prefer that there is no intersection among these clips. However, overlapping is acceptable if the total number of frames in the video is insufficient to create 5 videos (a video contains less than $16 \times 5 = 80$ frames).

We also used several augmentation techniques on the input to improve the generalization of our models. To do that, we made use of the well-known library *Albumentations*.[8] Converting all clips to the same dimensions and normalizing the frames are two common approaches for training, evaluation, and test set. The same enhancement should be applied for the evaluation and test set. This ensures that the test set and the evaluation set have the same distribution. For the training set, we operate some additional methods such as horizontal and vertical flipping, and adjusting the brightness and contrast of the images with a given probability p.

(a) 8 sample frames of a ParallelBar video (UCF101), where the actual action is performed only at 2 last frames

(b) Frames extracted from a HandstandWalking class (UCF101), which having the human pose similar to some first frames of Figure 3a

Figure 3

(a) 8 sample frames of a Basketball video (UCF101), in which contains camera motion.

(b) 7 frames of a random hit video (HMDB51), in which there are a scene transition between frame 2 and 3

Figure 4

3 Methods

In the first half of the section, subsection 3.1, we will describe five different models, why we chose them, what are their advantages and their disadvantage starting from the simplest model (Late Fusion) from the most complicated one (Non-local Neural Networks). Furthermore, some techniques to monitor what 3D-related models have learned also describe in the last subsection 3.2.

3.1 Models

3.1.1 Late Fusion - The Baseline Model

The Late Fusion model[3] is one of the basic techniques for solving problems that require handling a combination of input from individual sensors (images) like video classification. As mentioned before, we first extract a certain number of frames from a given video to generate the input for the model; the model will process these frames through a shared 2D CNN before concatenating and feeding them to a fusion layer to get the answer 5. The first steps are quite apparent as we want to fully manipulate our raw input to get the best answer, but the step that combines the input distinguishes late fusion and other fusion models such as early fusion and slow fusion.

Because action is basically a sequence of motion in a certain period of time, the approach of this model perfectly fits the fundamental thoughts of humans when we attempt to identify the action. The fact that having a quite simple architecture can benefit the model training time, so it can show a reasonable solution in a short period of time.

However, the Late Fusion model finds it challenging to compare the low-level motion between frames[9] .i.e the model is not capable to utilize small shifts between consecutive frames, which can be a unique characteristic of an action, to recognize the action.

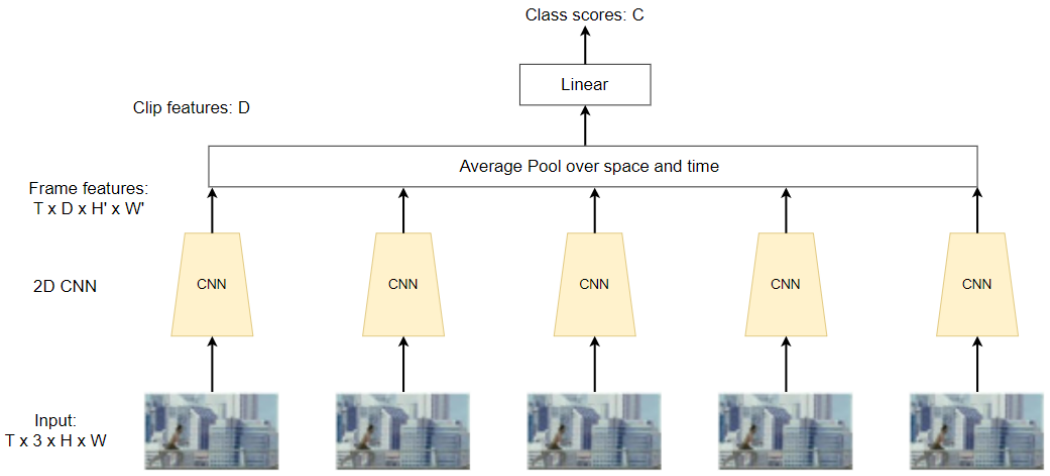


Figure 5: Late Fusion architecture

3.1.2 Vanilla 3D Convolutional Networks (C3D)

The previous model had succeeded in implementing 2D convolutional kernels, so now we should move on to an application of 3D kernels, which is C3D.[5] Concretely, it is true to consider C3D as 'the VGG of 3D CNNs' (Figure 6) when looking at its architecture; this model uses 3x3x3 as convolution kernel size and 2x2x2 as pooling size,[9] and the model has its inputs stacked at the beginning which is similar to Early Fusion, an alternative version of Late Fusion

The VGG net had shown its reliability in many image-related problems in the past, so its variant with 3D convolution could have noticeable performance in solving our problems. Alternatively, the increase of the dimension of the convolution leads to significant growth of the parameters, so it will take more resources and time to train the model.

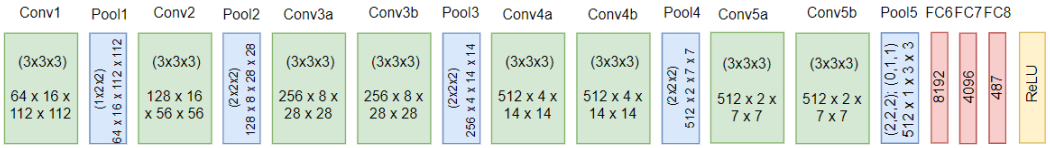


Figure 6: C3D architecture

3.1.3 Long-term Recurrent Convolutional Networks (LRCN)

When working with data in the form of sequences like video which is a sequence of images, it would be a miss if we do not consider RNN and its implementation as candidates for solving video-involved problems. The first few steps of LRCN are similar to that of Late Fusion which is respectively extracting frames and extracting features from them by a 2D or 3D CNN; however, the input in the next step will be fed to an RNN, usually, an LSTM, to generate the output strings (Figure 7). In other words, the model uses CNN as an encoder and RNN as a decoder.

By implementing RNN, the model not only have a global temporal structure among all motions but also have a local temporal structure among motions in a short clip.[9] Because of that, this model shows a greater ability to use the factor of time or the order of motions to improve the classification of actions in comparison with the previous model. Moreover, the RNN layer is ideally considered to solve the problems pertaining to variable-length inputs and outputs, so it is expected to have a better prediction. On the other hand, the RNN layer can also result in a long training time since developers can not manipulate parallel techniques in training a model in RNN layers to diminish its learning duration.

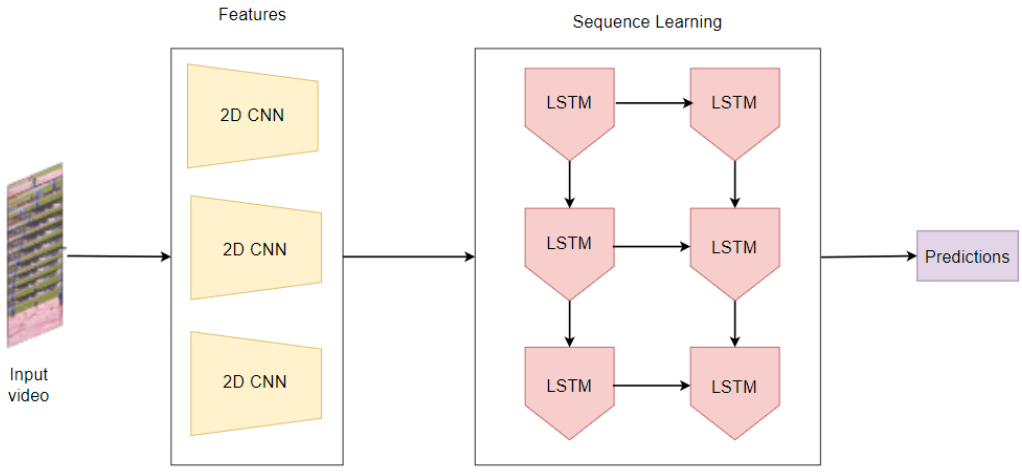


Figure 7: LRCN architecture

3.1.4 Inflated 3D ConvNet (I3D)

The idea behind the I3D model is also inspired by using models that have had a great performance in the past like the C3D model. However, in this model, we do not only can reuse the architectures of some popular models regarding image problems but also can reuse all of its pre-train parameters to initialize the model; in order to retain the result of the 2D convolution in this '3D version', it is recommended to add a new dimension to the convolution by a certain number then divide the value by the same number[9] (Figure 8).

One advantage of having pre-train weights as the initialization is that the model has the ability to display substantially superior results in comparison with other I3D that do not use pre-train parameters. In addition, the image models should treat the dimensions of the images equally, but when the input is time considering like videos, it is not optimal to have symmetric receptive fields. When the fields grow too quick relative to time, it may break early detections, and if the fields grow too slow, the model may fail to capture the field dynamically. Therefore, I3D prefers to use non-symmetric kernels or strides to avoid these problems.

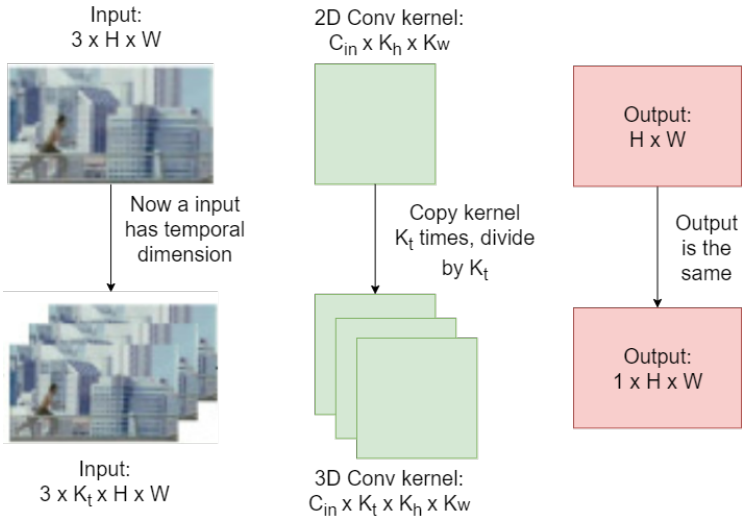


Figure 8: From inflated 2D to 3D networks (I3D)

3.1.5 Non-local Neural Networks - Modeling Long-term Temporal Video Structure

The non-local blocks in this network are actually the application of the attention mechanism. In the figure, we can easily point out the similarity in the architecture between self-attention and non-local blocks: there are still queries, keys, and values and residual connections (Figure 9).

The attention technique is considered to actively capture the important information from the given data and have a residual connection to ensure long-term memory to enhance the overall performance of the model. The non-local network is simply constructed by inserting the non-local block into existed 3D networks like C3D or I3D[9] , so it can take both pros and cons of the flow of these architectures.

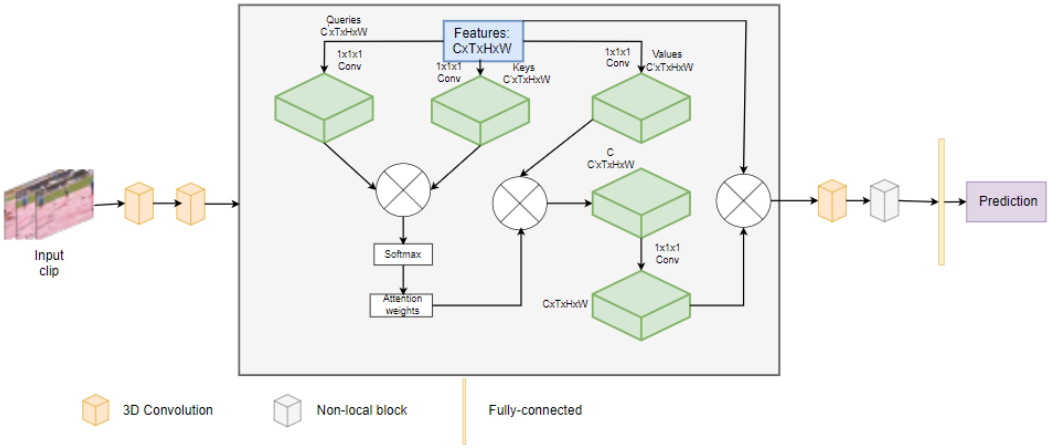


Figure 9: Non-local Neural Networks

3.2 What 3D-related models have learned?

Neural networks are often accused of being too vague and obscure, compared to a black box, only able to view the inputs and outputs without the knowledge its internal components. To find out what our models view and interpret, we have applied some techniques to visualize what features were extracted and how important it is to the classification.

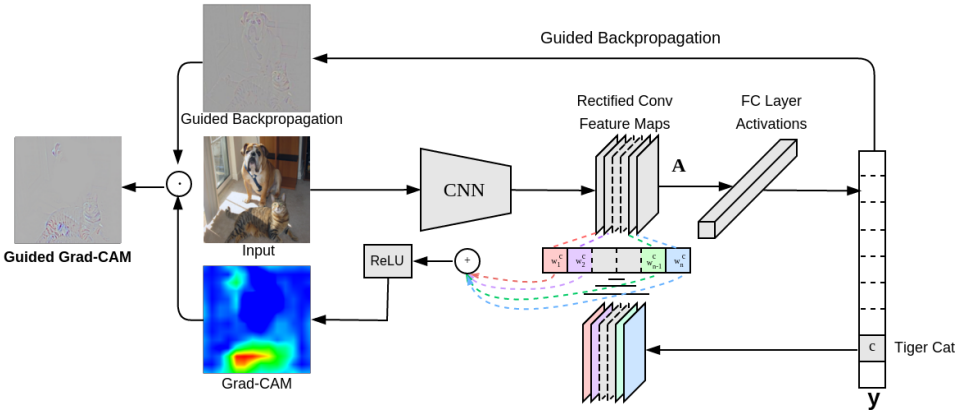


Figure 10: Grad-CAM: Why did you say that? by Ramprasaath R. Selvaraju et al[10]

First of all, the Activation Maps,[11] also known as the Feature Maps, capture the result of applying the filters to input, such as the input images or other feature maps. By visualizing, we will know what features of the input were detected and preserved for the next layer. The closer the layer to the output the map show more general features that are captured.

Next, Gradient-weighted Class Activation Mapping[12] , also known as Grad-CAM, shows the important areas for the score of the classifier to depend on. It is calculated by taking the activation map on the forward pass, the gradient of the same layer with respect to the classified label and multiplying those together, and then running through a ReLU activation to zero out the negative areas. The Grad-CAM, in theory, shows the best result when used for the last convolutional layer. For the result, it should show the areas that contribute most the class as the brightest color, darkening down for the rest.

Finally, Guided Backpropagation[13] shows which pixel of the input image actually matters to the output of the model. It is done by guiding our neurons to be activated or not for the backpropagation, hence the name. During the backward pass with respect to the classified label, we do not backpass negative gradients or where the negative activations are in the forward pass. The pixels with gray color do not contribute to the output, however of the 3 colors Red, Green, and Blue, each pixel of each color represent that the pixel is needed for the output corresponding to the color channels.

4 Experiments

This section will describe how we applied theories from previous sections into practice. Starting from training strategy in subsection 4.1 where we describe how to prepare the data, what hyperparameters we chose for each model and which techniques we have tried to improve our result and the last subsection 4.2, we will present our results in both quantitative and qualitative matters.

4.1 Training Strategy

4.1.1 Dataset Preparation

For each dataset, each dataset is divided according to the first train-test split (already attached to each dataset). In particular, HMDB51 is divided into 70% training set and 30% test set and UCF101 dataset is divided into 80% training set and 20% test set on each class.

In order to tune hyperparameters, try out different strategies for each dataset, we further split the dividied training set into training set and validation set with the proportion of 4/1 and tune the trained models on the validation set.

Furthermore, after trials and errors, we decided to use three different extracted frames techniques to train different models. In particular:

(1) Uniform-five-frame one clip: In one raw video, we cut five frames uniformly and stack them to make 4D tensor. This approach we used for Late Fusion and LRCN models as these model do not required a high correlated clips compared to C3D, I3D, and non-local models.

(2) One 16-frame clip: In one raw video, we sample a clip with 16 frames randomly. This approach of course suitable for C3D, I3D, and non-local models as they can detect the movement of each frame. However it is clearly not suitable for Late Fusion and LRCN models.

(3) Uniform five 16-frame clips: Similar to the previous technique, instead of one short clip, we sample five clips uniformly from the video and each clip act as a input for the model (so now we have the number of instances as five times as total number of video instances for training). We believe that would reduce overfitting of C3D, I3D, and non-local models

Last but not least, in order to increase efficiency for our training process, we extracted frames from the video and save them on the disk then read these frames from the data loader instead of reading the video directly from the disk when feeding data to the model.

4.1.2 Data Augmentation

We also experimented different augmentations for each model. We applied the same augmentation for all frames in one clip as the frames in one clip should be correlated to each other. For more details:

(1) The combination of Resize, Normalisation: The resize size for each model is different as we used different pretrained models for each models, for normalise part, we used default mean and standard deviation (for ImageNet).

(2) The combination of RandomResizeCrop, HorizontalFlip, VerticalFlip, ColorJitter, Normalisation: As we saw overfitting can occur in the previous settings so we deciced to take a further step to change the space and color of the clip itself.

(3) The combination of Resize, HorizontalFlip, VerticalFlip, ColorJitter, Normalisation: We saw that the previous settings sometimes led to worse results so to balance between two previous settings we chose Resize instead of RandomResizeCrop due to the fact that sometime RandomResizeCrop can crop mislead images.

4.1.3 Cost Function

For the cost function, we used Cross Entropy Loss[14] for multiclass as our problems are classifications with 101 and 51 classes of two datasets. The following is the equation of the loss function for multiclass $\gamma = \{1, ..., C\}$:

$$L_{CE} : \{0, 1\}^C * [0, 1]^C \to \mathbb{R}$$

$$L_{CE}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \ln(\hat{y}_i)$$

where $\hat{y} = (\hat{y}_1, ..., \hat{y}_C)$ is the class distribution, i.e. $\hat{y}_1 + ... + \hat{y}_C = 1$, returned by the model.

4.1.4 Evaluation Metrics

We chose accuracy is our evaluation metric for the problems because of its popularity for the classification problem, whose equation is equal to the total number of correct classification divided by the total number of instances. Confusion matrix is also chose as it enables us to see which classes are misclassifed with which classes.

4.1.5 Optimizers and Schedulers

For the optimizer, we used Adam optimizer[15] because of its popularity and it can get the learned parameters of the model converge quickly during training. For the scheduler, we utilised exponential learning rate[16] which decrease the learning rate by a constant number after each epoch as the model tends to need smaller learning rate when it is going to reach the minimum points.

4.1.6 Training

We utilised Pytorch[17] as the main tool for us to train and evaluate our models. Besides, Weights & Biases[18] is also the platform for us to log and save the checkpoints. In this subsection, we will describe which strategy hyperparameters we used for training and evaluating and which strategies we used to train for each model.

For each model, we chose hyperparameters based on trials and errors to get better training results, better validation scores. There is no equations and calculations that we used to choose the suitable set of hyperparameters. For more information about each parameter we used, please visit our Weight and Bias project at the end of the report.

On the training set, we augmented the dataset by using different three strategies described in subsection 4.1.2, but for evaluation set and test set, we only used the first technique in that section.

Because of our relative small dataset, our models are almost pretrained. For example, Resnet-152[19] backbone training on ImageNet for LRCN and Late Fusion models, as well as

Not only training on one dataset, to utilise the most out of all two datasets, we also tried to train two dataset simultaneously by feeding the training model two different batches of data and changing the last fully connected layer correspondingly.

4.2 Results

In this section, we will present the results of each model for each dataset and make comparison, give some reason why some models works better than some models for each dataset. The section is divided into two subsections which are quantitative results where tables, plots, and confusion matrix are presented and qualitative results where we wanted to see what the models have learned by

using the techniques described in subsection 3.2. For more information, we also saved logging and checkpoints in Weight and Bias Project which could found in section 6.

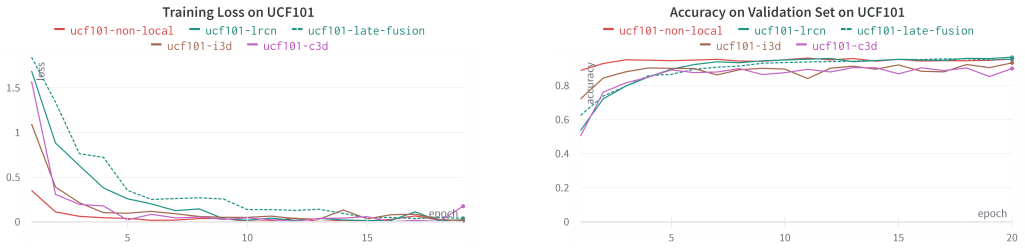
4.2.1 Quantitative results

Models	Data Processing	Backbone / Pretrained	UCF101 val acc	UCF101 test acc	HMDB51 val acc	HMDB51 test acc
Training on dataset itself						
Late Fusion	5 frames	Resnet-152	95.44	74.19	61.33	41.62
C3D	16 frames clip	Sport-1M[3]	89.99	60.55	47.68	15.98
LRCN	5 frames	Resnet-152	96.61	74.50	65.94	43.87
I3D	16 frames clip	Resnet-50	93.27	66.73	59.59	36.47
Non-local	16 frames clip	Kinetics 400	**95.48**	**83.01**	**67.74**	**53.35**
Training on two datasets						
Late Fusion	5 frames	Resnet-152	92.49	72.85	63.78	37.63
C3D	16 frames clip	Sport-1M	89.18	60.50	56.03	32.34
Non-local	16 frames clip	Kinetics 400	**95.01**	**84.11**	**68.80**	**50.92**

Table 1: The results for each model for each dataset with two training strategies, on the dataset itself and two datasets

We summarize results of each models for each model, which preprocessing process we used in table 1. For each model, by trying different strategies, we chose the best settings which gave the highest accuracy on validation and test set. For more information about what we have settings, please visit the Weigh and Bias websites on the end of the report in the section 6.

For training on just the dataset itself, there is no suprise that non-local networks give the best results when training on only one dataset in both UCF101 and HMDB51 with the accuracy 83.01% and 54.34% on the test set. But surprisingly, LRCN and Late Fusion, with 74% on UCF101 test set and 42 - 43 % on HMDB51 test set gives better results than C3D and I3D with only around 65% and 30% on the two datasets. Theoretically, C3D and I3D should be better than the two other models but we believed that due to the fact that we used Resnet-152 pretrained for LRCN and Late Fusion models compared to Resnet-50 backbones and pretrained Sport-1M on I3D and HMDB51. As we all know, Resnet-152 pretrained maybe produce better spatial representation for both LRCN and Late Fusion. Furthermore, only 16% of HMDB51 test set could be guessed corrected by C3D models, we believed because the HMDB51 has much more sense-transition than UCF101 dataset as well as Sport-1M dataset maybe not generalised well for HMDB51 dataset.



(a) The figure shows the training loss of five models in each epoch on UCF101.

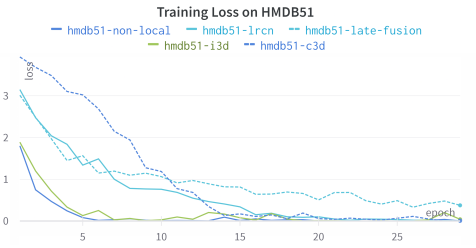
(b) The figure shows the accuracy on the validation set in each epoch UCF101.

Figure 11: Comparison of training loss and validation accuracy of UCF101 dataset

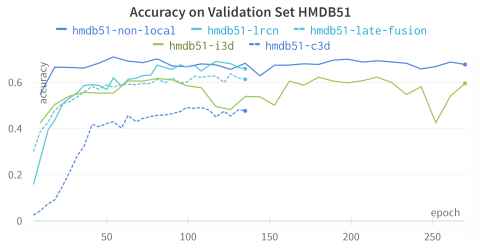
The figure 11 shows that all five models in UCF101 dataset converges very fast, which is just around the seventh epoch. However, on HMDB51 dataset, based on figure 12, all models converges around 12th epoch and as you can see on the right figure, the accuracy lines of models are inconsistent, unlike on UCF101 dataset. We believed it is due to the nature of HMDB51 dataset itself.

For training on two datasets simultaneously, the Non-local model showed its stability by just having a small change in accuracy, yet it still had the best performance in the test set overall. Though training

Non-local or Late Fusion on more diverse datasets failed to enhance the accuracy of the HMDB51 test set, the C3D model that had been trained on two datasets surprisingly could double that number. On the UCF101 data set, the Non-local model can slightly boost its performance after training on different datasets while the accuracy rest of the models decreased by about 2%.



(a) The figure shows the training loss of five models in each epoch on HMDB51.

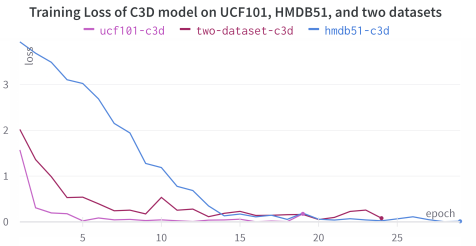


(b) The figure shows the accuracy on the validation set in each epoch HMDB51.

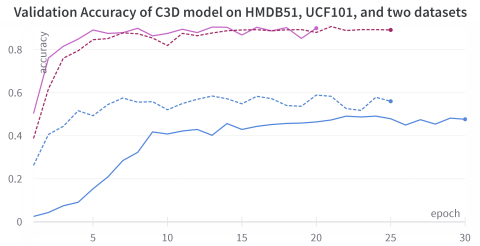
Figure 12: Comparison of training loss and validation accuracy of HMDB51 dataset

On the other hand, the table also demonstrated that collecting more data could not solve the overfitting problem in the HMDB51 dataset except by using the C3D model, yet it was able to bring the accuracy of the validation set and test set of the UCF101 dataset closer to each other.

From figure 17, we can conclude that the Non-local model predicted quite accurately on the UCF101 dataset despite the fact that has more labels which mean there is a higher chance of existing similar actions in this dataset. However, on the HMDB51 dataset, which only has half of the number of labels compared to that of CF101, we can clearly see that the model confused between similar actions like 'cartwheel' and 'handstand' or 'eat', 'chew', and 'smoke'. The confusion matrix in figure 18 can inform us about the problem in the dataset itself, or could be a sign for us to carefully re-examine our models or how we preprocessed our input data.



(a) The figure shows the three models' training loss of C3D model when training on HMDB51, UCF101, and two datasets.



(b) The figure shows the accuracy on the validation set of C3D model when training on HMDB51, UCF101, and two datasets.

Figure 13: Comparison of training on one dataset and two datasets, noted that: in the figure 13b, the purple and blue lines are of UCF101 and HDMB51 correspondingly, the dash lines belongs to the results of traning two datasets then evaluate on each dataset.

4.2.2 Qualitative results

In this section, we will report some analysis of what the model has learnt and provide some clear explanation for the figures. We chose the I3D architecture with the best trained weights to predict on a clip of 16 consecutive frames. In each figure, we illustrate the first frame of input clip, with some approaches for analyzing the model's knowledge. From each dataset, UCF101 and HMDB51, we choose a clip from the train split to make prediction.

In Figure 14, the input clip is labeled as 'ApplyLipstick' belonging to the UCF101 dataset. We could see that one of the video frames extracted in the left of the figure (14a). We already plotted the average of activation maps of the first convolution layer of the model in sub-figure 14b. As illustrated, the

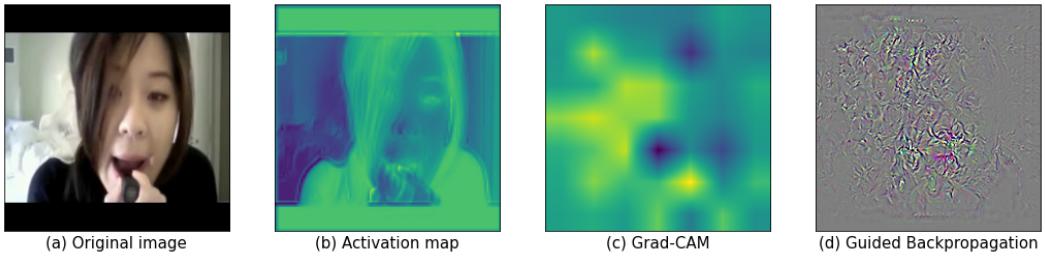


Figure 14: Qualitative analysis on an example of a 'ApplyLipstick' video of UCF101 dataset

layer perceives the image by extracting the features such as eyes, hair, shoulder, and most importantly lips. These brighter regions of the mean activation represent the part that is activated. Figure 14c displays the Grad-CAM map for one of the last convolutional layers in our I3D model. The meaning of the brighter region: what lets our model classify the label. The emphasized region (brighter) covers the region of the human on the left and the lips area. As expected, the area around the lips is what gravitates the model toward, but surprisingly, the hair area also contributes toward the output as well as some noise on the left side. This might be because of our datasets, people who apply lipstick also have their bangs and fringe recognized by the model, which might be the same story for 'ApplyMakeUp'. Figure 14d illustrates the Guided Backpropagation technique. From the image, the woman's lips as well as the side bang are spotlighted. It means that the score of our output is highly influenced by these parts. The reason explaining this situation is probably the same as before.

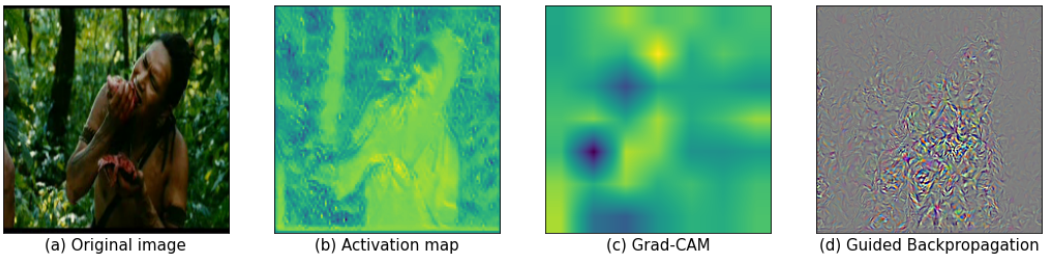


Figure 15: Qualitative analysis on an example of 'Eat' video of HMDB51 dataset

Similarly, figure 15 visualizes some analysis on the test input of the HMDB51 dataset. The first figure is the origin capturing a man eating something with the correct label 'Eat'. The second sub-figure(15b]) is an average of activation maps of the first convolution layer. Most of the character's parts are marked with brighter pixels in the activation map. This shows that the model has found the character of the action, however, some background trees are detected as well. In the Grad-CAM visualization, as shown in figure 15c, we could see that the emphasized regions (brighter) cover almost the entire right hand. We can imply that the model's prediction on the given input is mostly based on the feeding hand but there is still some noise, such as the top and bottom left of the picture. We could improve the model's performance by reducing the highlights of the tree behind. When plotting the Guided Backpropagation, the result is clearly shown in figure 15d. The character's arms are clearly more noticeable than the head and shoulders. This demonstrates that the pixels in the hand region have a stronger impact on predicting the target label.

5 Conclusion & Possible Extensions

So far, we have introduced and solved our classification problem with different techniques in computer vision and deep learning models on two benchmark datasets UCF101 and HMDB51. The best model, in general, is the Non-local neural networks model which performs most effectively with both training on the individual dataset and two datasets.

It is the first time we have done a deep learning project on such a big scale so it taught us many lessons. Firstly, we need to handle a big dataset with limited resources. Secondly, we need to fully

understand our problem, which is the input and which is the output. Finally, data preprocessing and model selection can have a tremendous effect on the result of our solution.

The most challenging step when we solved the problem is when we needed to decide how to store our datasets and which models, and what techniques we should do to analyze would use in order to divide the workload between each team member correctly. And finally, we decided to choose the aforementioned techniques and models in the previous sections.

Moreover, if we have more time on this project, we definitely research some new deep learning models such as ViViT[20] - the state-of-the-art model- which applies Transformer to video understanding problems, try out different new techniques in data preprocessing such as RandAugment[21] as well as applying self-supervised and semi-supervised learning for these two datasets.

6 Contributions

In this section, we will list all of works each member has done.

1. Programming, Coding
    - Frames Extraction:
      Nguyen Quang Duc - 20204876 (100%)
    - Create dataset on Kaggle, Google Drive:
      Tran Hoang Quoc Anh - 20200044 (50%), Luu Trong Nghia - 20204888 (50%)
    - Exploratory Data Analysis (EDA):
      Le Hong Duc - 20204874 (100%)
    - Training, Evaluation loop, Wandb Logging, Checkpoints Saving:
      Nguyen Quang Duc - 20204876 (100%)
    - Late Fusion and LRCN models:
      Luu Trong Nghia - 20204888 (60%) Le Hong Duc - 20204874 (40%)
    - C3D, I3D model:
      Le Hong Duc - 20204874 (70%), Tran Hoang Quoc Anh - 20200044 (30%)
    - Non-local model:
      La Dai Lam - 20204918 (100%)
    - What 3D model have learned:
      Tran Hoang Quoc Anh - 20200044 (70%), Le Hong Duc - 20204874 (30%)
    - Deployment:
      La Dai Lam - 20204918 (70%), Nguyen Quang Duc - 20204876 (30%)
    - Testing Code:
      Luu Trong Nghia - 20204888 (50%), Tran Hoang Quoc Anh - 20200044 (50%)
2. Report Writing, Presentation, Readme file
    - Report Writing:
        - Le Hong Duc - 20204874 (25%)
        - La Dai Lam - 20204918 (25%)
        - Nguyen Quang Duc - 20204876 (20%)
        - Luu Trong Nghia - 20204888 (15%)
        - Tran Hoang Quoc Anh - 20200044 (15%)
    - Presentation:
        - Luu Trong Nghia - 20204888 (20%)
        - Le Hong Duc - 20204874 (20%)
        - Tran Hoang Quoc Anh - 20200044 (20%)
        - La Dai Lam - 20204918 (20%)
        - Nguyen Quang Duc - 20204876 (20%)
    - Readme file:
        - Nguyen Quang Duc - 20204876 (70%)
        - La Dai Lam - 20204918 (30%)

Github repository & Weights and Bias logging, and checkpoints

All the codes, notebooks, loggings, and checkpoints written for the project with detailed instructions can be found in the following Github respostory and Weights and Bias project:

`https://github.com/QuangDuc-HUST/DL.20221.04.ActionRecognition`

`https://wandb.ai/dandl/dl_action_recognition`

Deployment

We also deployed the web deployment of our work using FastAPI[22] , please read and follow carefully the instruction to start and use our UI tool. The figure 16 shows an example of our application.

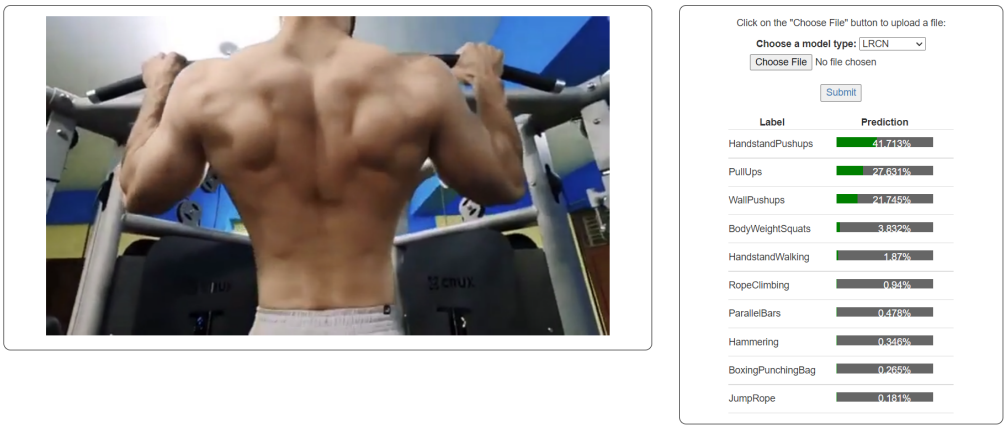


Figure 16: Prediction of Pull Up video downloaded from Youtube and model LRCN in web application

References

[1] Soomro K, Zamir AR, Shah M. UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:12120402. 2012.

[2] Kuehne H, Jhuang H, Garrote E, Poggio T, Serre T. HMDB: a large video database for human motion recognition. In: 2011 International conference on computer vision. IEEE; 2011. p. 2556-63.

[3] Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L. Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition; 2014. p. 1725-32.

[4] Donahue J, Anne Hendricks L, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K, et al. Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2015. p. 2625-34.

[5] Tran D, Bourdev L, Fergus R, Torresani L, Paluri M. Learning Spatiotemporal Features with 3D Convolutional Networks. ArXiv e-prints. arXiv preprint arXiv:14120767. 2014.

[6] Carreira J, Zisserman A. Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 6299-308.

[7] Wang X, Girshick R, Gupta A, He K. Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018. p. 7794-803.

[8] Albumentations: fast and flexible image augmentations; 2022. Available from: `https://albumentations.ai/`.

[9] CS231n - Lecture 12: Video Understanding; 2022. Available from: http://cs231n.stanford.edu/slides/2022/lecture_12_ruohan.pdf.

[10] Ramprasaath R Selvaraju RVMCDP Abhishek Das, Batra D. Grad-CAM: Why did you say that?;. Available from: https://doi.org/10.48550/arXiv.1611.07450.

[11] How to Visualize Filters and Feature Maps in Convolutional Neural Networks; 2019. Available from: https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/.

[12] Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 618-26.

[13] Deep Learning: Guided BackPropagation; 2020. Available from: https://leslietj.github.io/2020/07/22/Deep-Learning-Guided-BackPropagation/.

[14] A Gentle Introduction to Cross-Entropy for Machine Learning; 2020. Available from: https://machinelearningmastery.com/cross-entropy-for-machine-learning/.

[15] Gentle Introduction to the Adam Optimization Algorithm for Deep Learning; 2021. Available from: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[16] ExponentialLR; 2022. Available from: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ExponentialLR.html.

[17] Pytorch; 2022. Available from: https://pytorch.org/.

[18] Weights & Biases – Developer tools for ML; 2022. Available from: https://wandb.ai/site.

[19] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770-8.

[20] Anurag Arnab GHCSMLCS Mostafa Dehghani. ViViT: A Video Vision Transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2021. p. 6836-46.

[21] Cubuk ED, Zoph B, Shlens J, Le Q. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In: Advances in Neural Information Processing Systems; 2020. p. 18613-24.

[22] FastAPI; 2022. Available from: https://fastapi.tiangolo.com/.

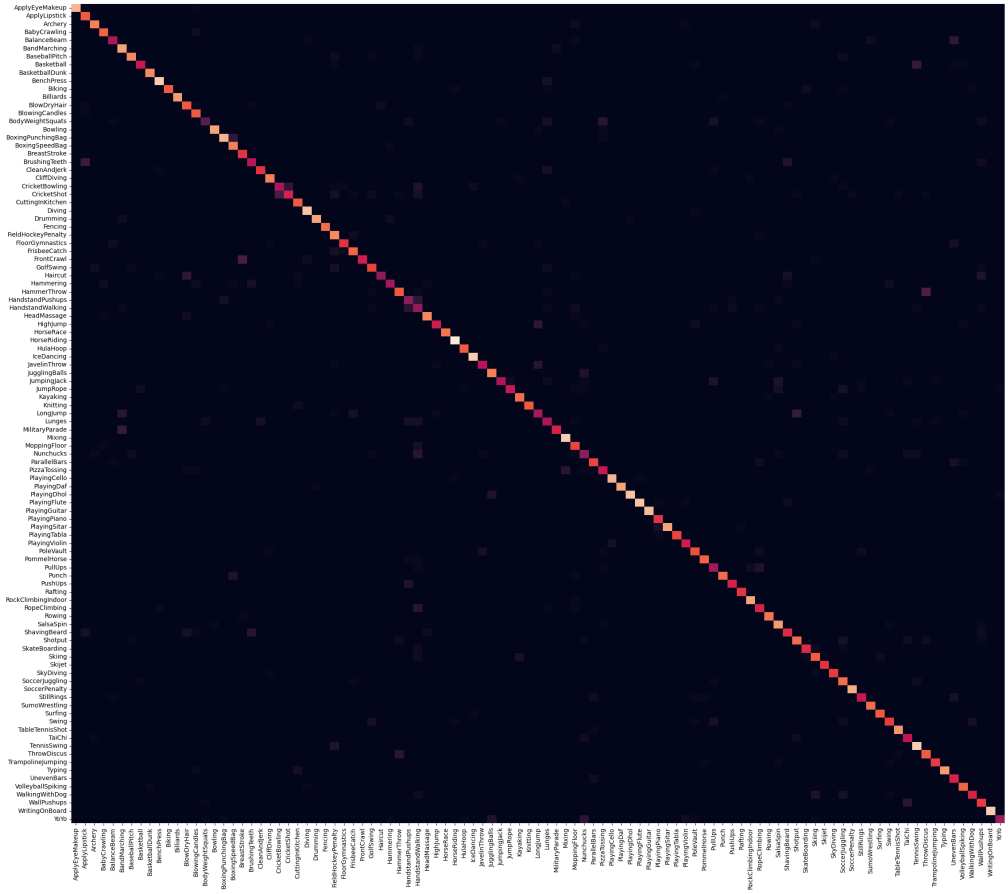


Figure 17: Confusion matrix of UCF101 with non-local models training on two datasets

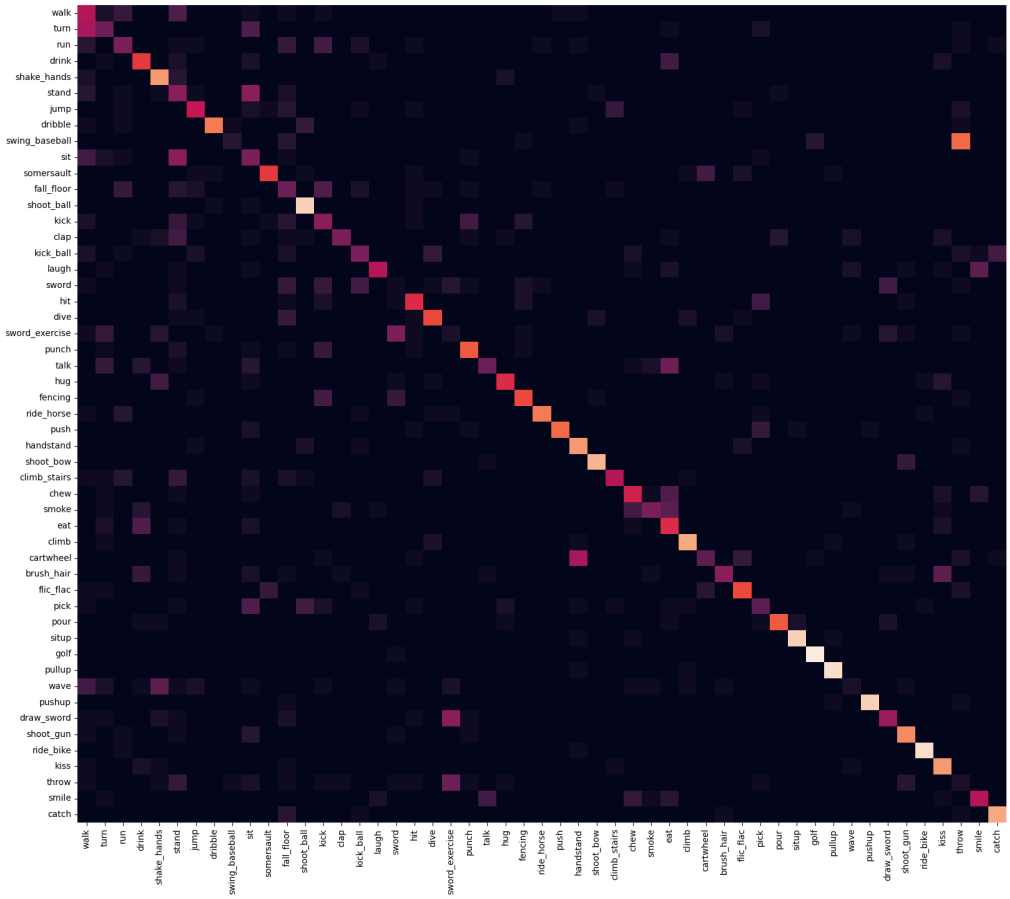


Figure 18: Confusion matrix of HMDB51 with non-local models training on two datasets