UNIVERSIDAD EAFIT

Vigilada Mineducación

Movement in video classification using structured data: Workout videos application

Jonathan Damián Múnera Muñoz

Tesis de maestría

Asesor, docente

Marta Silvia Tabares

UNIVERSIDAD EAFIT

ESCUELA DE INGENIERÍAS

MAESTRÍA EN INGENIERÍA

May 2023

Movement in video classification using structured data: Workout videos application

Jonathan MÚNERA [a] Marta S. TABARES[a]
[a] *Grupo GIDITIC/Universidad EAFIT*

Abstract. Nowadays, several video movement classification methodologies are based on reading and processing each frame using image classification algorithms. However, it is rare to find approaches using angle distribution over time. This paper proposes video movement classification based on the exercise states calculated from each frame's angles. Different video classification approaches and their respective variables and models were analyzed to achieve this, using unstructured data: images. Besides, structure data as angles from critical joints Armpits, legs, elbows, hips, and torso inclination were calculated directly from workout videos, allowing the implementation of classification models such as the KNN and Decision Trees. The result shows these techniques can achieve similar accuracy, close to 95%, concerning Neural Networks algorithms, the primary model used in the previously mentioned approaches. Finally, it was possible to conclude that using structured data for movement classification models allows for lower performance costs and computing resources than using unstructured data without compromising the quality of the model.

Keywords. Machine Learning, computer vision, KNN, mediapipe, supervised classification, video classification, deep learning, tensorflow, neural network, workout, fitness, exercise, signal

1. Introduction

Fitness industry relies on videos to instruct their users how to perform better on their workouts, this also helps people train from home when schedules do not allow to go to the gym. This video classes or video workouts have multiple exercises to focus user efforts on multiple sections of their body, hence, a long video with different movement is created. This gives rise to the necessity of tagging these videos with each of their movements and identify the time where each one is performed, this is achieved manually by a person doing the job, which is inefficient and slow. Consequently, machine learning algorithms can help with this task, making it faster and easier. This task contains two different efforts to fulfill, the first one, identifying the changes in the exercises so each movement can be classified individualy, and the second one, movement classification. For the first task it will be used an algorithm called ruptures [1] which allows to detect the signal breaks on the movement thus identifying exercises changes. The second task, which is this paper's focus, represents a movement classification problem, which can be

approached by many means, the most common methods include using sequential neural networks [2] and recurrent neural networks [3] to capture the sequence of poses in each video frame, this is a great method when you have the computational resources and abundant training data. This research evaluates a niche of exercises that are trained manually which means there is not plenty of data samples, making the use of said methods harder. To achieve the main goal, then, the data is prepared in a different format; to process the poses in each video frame as mentioned before, the input data would be unstructured, images comming from the video; in this new proposed format of data it is calculated the angles in the body from each frame, this way the data gets a structured format allowing the use of a wide range of machine learning algorithms with high precision and lower computational cost.

2. Literature Review

The main focus of this paper is to present an effective way to make video training and classification extracting structured data from the videos instead of processing them as they come: a collection of images. For this reason this section is only reviewing literature based on video training and classification and doesn't express any concerns regarding to the change point in the signal.

In [4], it is proposed a way to identify actions performed by people. using fixed cameras on rooms to capture the movement and the person in frame as well as capturing a called ego-view to create better samples for the model. However, this is more focused on detecting an specific action rather that a set of movements, and this can be better segregated from one action to another based on the camera location and object detection. Multiple cameras are detecting multiple inputs at the same time, one first person view combined with at least one third person view to allow having a holistic view on the action. This is then combined with object detection, and camera placement to have a better understanding on what the person is doing. Finally, for the classification model it is used a **Convolutional 3D-ResNet model** as in [5], using unsctructured data as input for the model, processing the images in each frame. It is also used an athomic actions recognition allowing the model to predict better the final action based on those athomic ones, so the result would be a combination of these individual actions as well as what is being captured from the cameras. From this commented research it is posible to use its object recognition technic, based on the fact that some exercises include equipment so that may help better classification, however, this could be a future work, since the current scope is only focused on exercises with no equipment required.

The article *Piano Skills Assessment* [6] uses an 3DCNN to classify little video clips, which means its goal is not only to detect an action or specific instance in time, but to classify input video data. Its experiment consists in assess a pianist skill level based on their performance using aural and video input. Aural does not concern to this paper, so the video input is the one being analyzed. **3D convolutional neural networks** can be specially useful for the purpose of video classification, so instead of classifying frame per frame, its input can be directly short video clips that allow to adquire more information for the model. However, work in this paper are trying to detect movement and poses on a video, so that methodology helps for an instance in which the comparison between this CNN and the method proposed here is being conducted.

Hierarchical video frame sequence representation [7], presents an interesting way of processing videos, this is done by using each frame as a node and their respective relations between these nodes as edges for them. This relation is weightened by each frame similarity vector, which means that similar shots or frames, would be closely connected between them. This hierarchical approach helps understand videos as a whole action, so it could be an interesting approach on exercise video classification if same movements are strongly connected to each other.

Yoga pose detection is one of the many applications in image and video recognition when it comes to workouts, and the used technic in this cases get closer to the one in this paper. As implemented by Gupta and Jangid in [8], they are using **Human Pose Estimation** in their work, helping them extract structured data from the videos. They proceed to calculate the angles using the inverse cosine function based on the dot product as shown in (1) which have been copied exactly as it appears in their paper. This equation allows to calculate the angles in 2 dimensions making it clear that the person in the video is standing in front of the camera and ignoring depth in **z axis**.

$$\theta = \cos^{-1}\left(\frac{X_{PQ} * X_{QR} + Y_{PQ} * Y_{QR}}{\sqrt{X_{PQ^2} + Y_{PQ^2}}\sqrt{X_{QR}^2 + Y_{QR}^2}}\right) \tag{1}$$

Where,

$$X_{PQ} = x_2 - x_1$$
$$Y_{PQ} = y_2 - y_1$$
$$X_{QR} = x_3 - x_2$$
$$Y_{QR} = y_3 - y_2$$

It is important to notice that this ([8]) article is focused on pose detection or classification which means that they only need to classify the pose on one time, which reduces the complexity of the movement. Yoga has this particularity in which their poses are very static, so classifying them can be based on the angles for each frame, if they change, the pose change. For the purpose of this paper, the movement or change in the angles doesn't necesarily means change in the exercise, on the contrary, movement allows to identify the exercise, which adds a layer of complexity for this model.

3. Methodology

3.1. Proposed Method

Movement in video classification has being mostly approached by processing frames as images and then finding a relationship between them so the video is understood as a collection of images in sequence or with a correlation between them. This is highly useful when we are labeling what is happening on the videos or to identify an action in it. It gets harder when it is needed to determine the specific action between a set of

multiple ones, for example, classifying a person doing exercise is a relatively easy task, but to label which exercise are they doing is more complex. This paper proposes this exercise labeling based on the angles distribution on each frame in time, different set of angles would represent different exercises and, since exercises are repetitive movements, it is understandable that the dataset will have a recognizable pattern among the angles distribution, allowing to detect peaks and valleys in the signal. This extreme points are exercise states and since all of the exercises would have a range of angles distribution, these states are helpful to differentiate them apart. After that having multiple up and down states combining them creates more data samples, this is, all up states mixed with all down states, obtaining not only more records on the dataset but better training data. These record combination (up and down states) are used to train multiple ML models, allowing to classify the movement in video not only based on the image, but rather on the actual movement.

3.2. Angles calculation

For this task the use of the mediapipe library [9] is implemented, this library allows users to detect 32 body landmarks as shown on figure number 1. With these landmarks it is possible to calculate the angles [10] made up of vectors created with the limbs that comforms important joints for the exercises, these joints -as shown in figure 2 - being armpits, knees, hip and elbows.

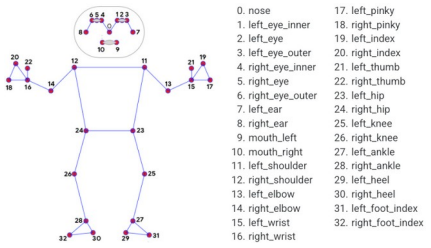


Figure 1. Mediapipe body Landmarks

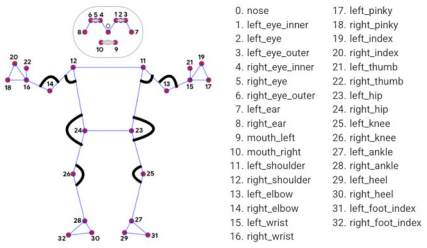


Figure 2. Calculated angles

Since the landmarks are in a 3D space, angles between the joints are calculated on the vectors created by the limbs. For example, the left arm vector is created between the points (in a 3D space) **11** and **12** in figure 1, so calculating the left armipit angle would imply calculate this vector as well as the vector created by landmarks **11** and **23**. These vectors are created using the mutual joint (in this case **11**) as the starting point, so vectors are created as shown in equation (2). Where **11** corresponds to the left shoulder landmark, **12** to the left elbow landmark and **23** to the left hip landmark, all of them with their respective 3D coordinates.

$$LeftArmVector = (LE_x - LS_x, LE_y - LS_y, LE_z - LS_z)$$

$$LeftTorsoVector = (LH_x - LS_x, LH_y - LS_y, LH_z - LS_z) \quad (2)$$

Where,

$$LE = Le\,f\,tElbow$$
$$LS = Le\,f\,tShoulder$$
$$LH = Le\,f\,tHip$$

This process is repeated for every joint of interest shown in figure 2 so we can obtain the necessary angle distribution for the following sections. Once the vectors needed for every joint angle are calculated it is then calculated each angle. This is done based on the dot product which states what is shown in equation (3) for the vectors **a** and **b**. So clearing for the angle it is obtained a good equation to calculate the angles between these limb vectors, obtaining the eight needed angles for this work.

$$a \cdot b = |a||b| \cos\theta$$
$$\theta = \arccos \frac{a \cdot b}{|a||b|} \tag{3}$$

This angles creates a distribution in time for the movement. For better understanding, this movement is transformed into a single variable created based on the sum of the average of the angles on each side of the body as explained on equation (4). This aggregation is created over each processed frame of the video, so a vector is obtained.

$$\frac{LAA_i + RAA_i}{2} + \frac{LEA_i + REA_i}{2} + \frac{LHA_i + RHA_i}{2} + \frac{LKA_i + RKA_i}{2} \tag{4}$$

Where,

$$LAA = le\,f\,tArmpitAngles$$
$$RAA = rightArmpitAngles$$
$$LEA = le\,f\,tElbowAngles$$
$$REA = rightElbowAngles$$
$$LHA = le\,f\,tHipAngles$$
$$RHA = rightHipAngles$$
$$LKA = le\,f\,tKneeAngles$$
$$RKA = rightKneeAngles$$

This new vector expresses a wave of the body movement as shown on figure 3, where x axis is time in seconds and y axis is the angles aggregation. In this figure we can see

how some segments of the signal show a periodical pattern, this is due to the person in video, the trainer, doing repetitive movements, in this case exercises, so we can start to separate these signals and analyze each movement individually. This particular example consist of a 20 minute workout doing multiple exercises.

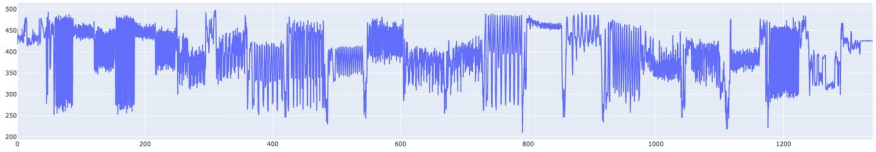


Figure 3. Workout Motion Wave

3.3. Movement Seggregation

Figure 3 shows the wave of a compilation of multiple exercises, and visually can be differentiated because of the changes in the signal intensity and frequency, so for the next step it is needed to detect this changes automatically and to separate the exercises using a ruptures [1] algorithm for off-line change point detection. Off-line methodology is used based on the fact that all of the frames have already been processed. Online change point detection could be a possible next implementation for reducing processing time detecting changes on the signal at the same time the frames are being processed. This algorithm helps users obtain the points where the signal switch from one pattern to another as shown in figure 4, some exercises need to be manually separated from each other, but this is minimal compared to the number of exercises performed in this workout. This exercise segregation allows to start the classification process.

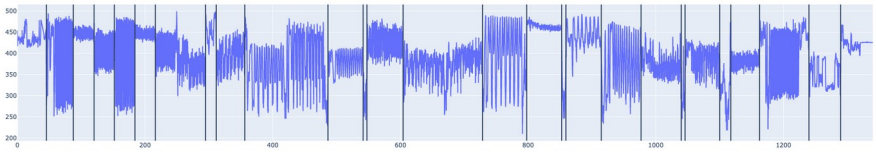


Figure 4. Ruptures algorithm applied over the workout signal

3.4. Exercise States

After the change point identification its needed to detect the exercises states, this is done with the help of another library for the peak and valley detection, this is the peakdetect [11] library and, with some tuning, allows to detect the maximun and minumun values for each segment, as shown on figure 5. This signal represents the first exercise being done by the trainer, in this case, corresponds from the second 58 to the 87, the first random

signal received, second 0 to 57, is the introduction to the workout, so it is ignored. States are used for training data due to the fact that exercises in this case are made up of two main positions, up and down, so these allows to identify the exercise with fewer data than using the whole movement pattern.

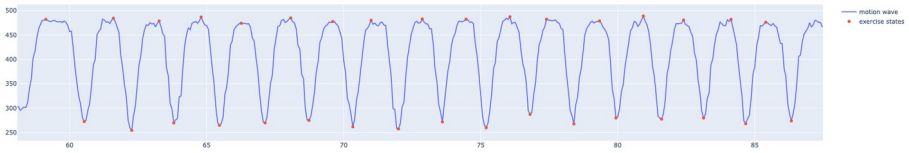


Figure 5. Exercise States

The next step after obtained these states is to create model training data labeling this exercise based on previous knowledge and expertise, this is done via manual tagging or either obtaining video metadata to correctly label this signal. This training dataset is the input for the machine learning algorithms to be tested. The dataset is not created based on the angles aggregation, but on the individual angles present in each state of the exercise, this means that for every state -either up or down- we obtain the angle distribution at that time, creating 17 dimensional vectors -being each of the 8 joints times both states- and adding one last variable being the person's body inclination. This last metric is calculated based on a new artificial vector created on the center of the torso, this vector represents the torso position over the 3 dimensions -x,y,z- so its respective coordinates are obtained and evaluated, if **y** coordinate is greater than the other two, the person is standing based on a simple definition stipulated, if not, they are lying and is a boolean feature added to the dataset.

3.5. Exercise Classification

3.5.1. Creating more samples in dataset

The input dataset from peak detection will contain as much samples as repetitions being done by the trainer, this is, if the trainer did 10 squat repetitions, 15 push-ups and 10 burpees, the dataset will contain the angle distribution for state up and down for each repetition of each exercise, so it would be 10 samples with label squat, 15 with label push-ups and 10 with label burpees. For this particular example, the dataset contains around 15 samples of each exercise and a total of 5 exercises, which is too few samples for a proper model training. Artificial data is created based on those samples we already have, since the dataset contains up and down states, it is combined every down state of an exercise with every up state of the same exercise, obtaining, from 15 repetitions, 225 records to train the model per exercise. So the final training dataset contains the columns shown in table 1 and the label would be only the exercie name for each case.

3.6. Model training

Before the last fix on the dataset -cross join every down state with every up state- the validation accuracy of every model tested didn't get over 83%, which re-afirm the fact that good models train over good datasets.

Table 1. Training variables

0	backAnglesPeaks
1	elbowAnglesPeaks
2	armpitAnglesPeaks
3	legsAnglesPeaks
4	backAngles2Peaks
5	elbowAngles2Peaks
6	armpitAngles2Peaks
7	legsAngles2Peaks
8	slopePeaks
9	backAnglesValleys
10	elbowAnglesValleys
11	armpitAnglesValleys
12	legsAnglesValleys
13	backAngles2Valleys
14	elbowAngles2Valleys
15	armpitAngles2Valleys
16	legsAngles2Valleys
17	slopeValleys

First model tested was a **KNN classifier** [12]. This was due to the first conception of having an 8 dimensional array of angles converted into 16 dimensions when joining both states -up and down-, so having the dataset across multiple dimensions gave the idea of classifying with an algorithm suitable for this task, keeping in mind that exercises would fall around the same space when distributed over the 17 dimensions, here is where KNN looked useful. For this particular training the following parameters in table 2 where used.

Table 2. Parameters used for the KNN algorithm

Parameter	**Value**
algorithm	'auto'
leaf_size	15
metric	'manhattan'
n_neighbors	1
weights	'uniform'

The manhattan parameter for the distance metric was the most impactful one, making the model improve significatively, and the fact that using only one neighbor to compare produced the best results means that the classes are very well separated from each other, which is good in this cases. Using a 70-30 data split for training and validation KNN gives results above the 98% of accuracy testing it multiple times and performing cross_validation, these are very good numbers in terms of a machine learning model.

Some of the most popular machine learning algorithms used due to their reliability are tree classification models, these being **Decision Tree** [13], and the ensemble models **Extra Tree** [14] and **Random Forest** [15], all of these algorithms comming from the **scikit-learn** [16,17] library. All of these tree based algorithms got similar accuracy, with a little more variation between iterations, varying from 93% to 98% of accuracy, practic-

ing the same methods of evaluation as with the **KNN classifier**. Parameters for each of these algorithms are found in table 3.

Table 3. Tree based algorithm parameters

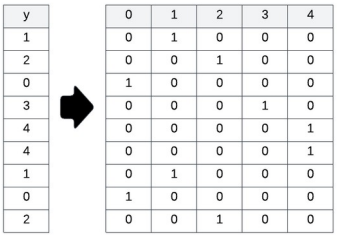
Algorithm	Parameter	Value
decissionTreeClassifier	criterion	'entropy'
	max_depth	6
	max_leaf_nodes	58
	min_samples_split	2
extraTreeClassifier	max_leaf_nodes	33
	min_samples_split	2
	n_estimators	90
RandomForestClassifier	max_depth	5
	min_samples_leaf	1
	min_samples_split	2
	n_estimators	300

The last algorithm tested is the **Sequential Neural Network (SNN)** [2], using directly the **Keras** [18] library from **Tensorflow** [19]. The model is constructed as shown in table 4 and is using dense layers for training, and dropout and normalization layers to avoid overfitting.

Table 4. SNN Parameters

Model: "sequential"		
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	9728
batch_normalization (BatchNormalization)	(None, 512)	2048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
batch_normalization_1 (BatchNormalization)	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
batch_normalization_2 (BatchNormalization)	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8256
batch_normalization_3 (BatchNormalization)	(None, 64)	256
dense_4 (Dense)	(None, 32)	2080
dense_5 (Dense)	(None, 5)	165
Total params: 188,293		
Trainable params: 186,373		
Non-trainable params: 1,920		

For model compilation the used values are shown on table 5. Since **categorical_crossentropy** is used it is needed to transform the labels vector **y**, so first of all convert the label vector to numbers -0 to 4 being 5 different classes/exercises-, and then using the **one hot encoding** methodology [20] vector **y** should look like figure 6, where



y
1
2
0
3
4
4
1
0
2

0	1	2	3	4
0	1	0	0	0
0	0	1	0	0
1	0	0	0	0
0	0	0	1	0
0	0	0	0	1
0	0	0	0	1
0	1	0	0	0
1	0	0	0	0
0	0	1	0	0

Figure 6. One hot encoding y label

Table 5. Compilation parameters for the SNN

Parameter	Value
loss	'categorical_crossentropy'
optimizer	'nadam'
metrics	['categorical_accuracy']

each column represents each class an its respective boolean indicates whether or not the label corresponds to this class.

The training for this model was made using 20% data for testing, and 80% for training, iterating over 100 epochs, the results can be seen on figure number 7, where it can be seen that the training as well as validation get to very high values, ensuring that the model is not overfitting the data.

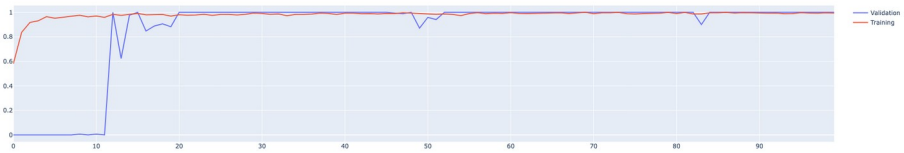


Figure 7. SNN accuracy over the epochs

Table 6 shows the final comparison between the tested models and shows how high the accuracy reach. This shows that with the implemented methodology in this paper, several models can be implemented achieving very goo results and we can classify movemnt in videos not only with unestructed data and convolutional neural networks, but with any vector supported model.

4. Discusion and Conclusion

Compared to [4], it can be appreciated that their methods present very good results for the use of Neural Network configurations, They use the precision as their main metric so we can use the average precision as well for a better comparison, as shown in table 7, the average precision of our model looks better once it reaches the stabilization point

Table 6. Model Accuracy Comparison

Model	Min_accuracy	Max_accuracy
KNN	98.2%	100%
Decission Tree	65.8%	93%
Random Forest	86.4%	100%
Extra Tree	95.2%	100%
SNN	75.8%	100%

after the 20 first training epochs of the model. For comparison purposes, we added the full average precision as well so we can have the full view.

Table 7. Model Accuracy Comparison vs Hierarchical

Model	Average_Precision	Average_loss
SNN	98.9%	0.054
SNN without stabilization	82.6%	0.37
Hierarchical + average graph pooling	84.1%	4.01
Hierarchical + self-attention graph pooling	84.5%	3.98

The Hierarchical model uses image video classification with their own methodology to achieve their goal, however, in this paper it is used bodymarks recognition so it is only logical to compare with the Yoga pose estimation research [8]. They achieve the best results using a SVM, in this paper we did not tested that model, however they are also using RandomForest algorithm, which allow us to have a very good validation start line. Table 8 shows their results against the ones obtained in this paper. It is included what they achieved with SVM as well as our best performing model, KNN, which allow us to compare better both results.

Table 8. Model Accuracy Comparison vs Yoga. This paper's results are expressed as average, while yoga research results are as reported on their results

Model	Average_Accuracy
KNN	99.6%
Our RandomForest	96.8%
Yoga RandomForest	96.47%
Yoga SVM	97.64%

It is seen that classifying videos can be well performed with structured data based on relevant information extracted from the video, unlike ”normal” video classification in which we process every frame as a picture, processing every frame as a collection of structured data can be helpful to implement not only neural networks -which is the main algorithm for image/video processing- but classical classification algorithms. It is proved once more that achieving a well trained and useful model can only be done by preprocesing the training dataset, making cleanse and adding the right features, even restructuring the whole dataset. With these methodology, we were able to perform faster and accurate video classification to improve the speed of content delivery. Next steps will be to test this trained models over new exercises and collect more data for the existing ones, as well as testing to another trainers, and normalize and standarize the body lenght for more consistent data. In general terms this work turned out to be a success given the fact

that we can classify the videos using a wide variety of classification algorithms available, making it easier and faster for users that are not familiariced with neural networks. Recurrent neural networks [3] are very helpful for classification in videos due to their ability to preserve information over the layers, however, they were not necessary in this case because of the way the data was prepared, making a mix with the multiple states of the exercise repetitions.

Referencias

[1] C. Truong. L. Oudre, N. Vayatis. Selective review of offline change point detection methods. signal processing, 167:107299, 2020.

[2] Ludovic Denoyer and Patrick Gallinari. Deep sequential neural network. *CoRR*, abs/1410.0510, 2014.

[3] Jürgen Schmidhuber Sepp Hochreiter. Long short-term memory, 1997.

[4] Nishant Rai, Haofeng Chen, Jingwei Ji, Rishi Desai, Kazuki Kozuka, Shun Ishizaka, Ehsan Adeli, and Juan Niebles. Home action genome: Cooperative compositional action understanding. 2021.

[5] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[6] Brendan Morris Paritosh Parmar, Jaiden Reddy. Piano skills assessment. 2021.

[7] Nishant Rai, Haofeng Chen, Jingwei Ji, Rishi Desai, Kazuki Kozuka, Shun Ishizaka, Ehsan Adeli, and Juan Carlos Niebles. Hierarchical video frame sequence representation with deep convolutional graph network. 2019.

[8] Ayush Gupta and Ashok Jangid. Yoga pose detection and validation. In *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, pages 319–324, 2021.

[9] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*, 2019.

[10] Stephen Andrilli and David Hecker. Chapter 1 - vectors and matrices. In Stephen Andrilli and David Hecker, editors, *Elementary Linear Algebra (Fifth Edition)*, pages 1–83. Academic Press, Boston, fifth edition edition, 2016.

[11] avhn. Simple peak detection library for python, 2022.

[12] Padraig Cunningham and Sarah Delany. k-nearest neighbour classifiers. *Mult Classif Syst*, 54, 04 2007.

[13] Johannes Fürnkranz. *Decision Tree*, pages 263–267. Springer US, Boston, MA, 2010.

[14] Jaak Simm, Ildefons Magrans de Abril, and Masashi Sugiyama. *Tree-Based Ensemble Multi-Task Learning Method for Classification and Regression*, 2014.

[15] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[16] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Ga ̈el Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] Franc ̧ois Chollet et al. Keras, 2015.

[19] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xi-

aoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[20] John T Hancock and Taghi M Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):28, April 2020.