Features Understanding in 3D CNNs for Actions Recognition in Video

Kazi Ahmed Asif Fuad, Pierre-Etienne Martin, Romain Giot, Romain Bourqui, Jenny Benois-Pineau, Akka Zemmari

Features Understanding in 3D CNNs for Actions Recognition in Video

Kazi Ahmed Asif Fuad
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400
Talence, France
asif.ahmed.fuad@gmail.com

Pierre-Etienne Martin
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400
Talence, France
pierre-etienne.martin@u-bordeaux.fr

Romain Giot
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400
Talence, France
romain.giot@u-bordeaux.fr

Romain Bourqui
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400
Talence, France
romain.bourqui@u-bordeaux.fr

Jenny Benois-Pineau
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400
Talence, France
jenny.benois-pineau@u-bordeaux.fr

Akka Zemmari
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400
Talence, France
zemmari@u-bordeaux.fr

***Abstract*—Human Action Recognition is one of the key tasks in video understanding. Deep Convolutional Neural Networks (CNN) are often used for this purpose. Although they usually perform impressively, their decision interpretation remains challenging. We propose a novel visual CNN features understanding technique. Its objective is to find salient features that played a key role in decision making of the network. The technique only uses the features from the last convolutional layer before the fully connected layers of a trained model and builds an importance map of features. The map is propagated to the original frame thus highlighting the regions in them that contribute to the final decision. The method is fast as it does not require gradient computation as many state-of-the-art methods do. Proposed technique is applied to the Twin Spatio-Temporal 3D Convolutional Neural Network (TSTCNN), designed for Table Tennis Actions recognition. Features visualization is performed at the RGB and Optical flow branches of the network. Obtained results are compared to other visualization techniques both in terms of human understanding and similarity metrics. The metrics show that generated maps are similar to those obtained with known Grad-CAM method, e.g. Pearson Correlation Coefficient between the maps generated of RGB data for Grad-CAM and our method is 0.7 ± 0.05 and 0.72 ± 0.06 on Optical Flow data.**

***Index Terms*—Explainable Deep Learning, 3D convolutions, Action classification, Video indexing, Table Tennis**

I. Introduction

Human action recognition in video is one of the key problems in video analysis. This computer vision task involves identifying different actions in a sequence of images. It becomes challenging when actions have low inter-class variability. Although Convolutional Neural Networks are performing impressively in different action recognition datasets such as DeepMind Kinetics, UCF-101 or AVA [1], it is not quite explainable why they make the correct or incorrect classification decisions. Our main objective is to perform fine grained action recognition in table tennis with the aim of improving athletes' performances. From raw video of table tennis exercises, it is necessary to recognize actions properly before motion, posture and other performance indicators could be analyzed. The difficulties in indexing such video content reside in the low inter-class variability of actions [1]. The so-called 3D CNNs [2], [3] with Twin Spatio-Temporal architectures (TSTCNN) [1] classify actions quite efficiently, but the interpretation of their decision making still remains open. It is our concern in this paper. The explanation of decisions is specifically important for the target users of video classification, sport coaches in our case. Hence, this will help table tennis teachers to focus on particular strokes performed by students for post exercise analysis. It is needed to explain the decision making of the CNN, at the generalization step, as the user is interested why a particular image or video segment is assigned to a particular class by the CNN classifier.

The contribution of this paper is a new fast method of explanation of network decisions. It is generic and is applied to the RGB and Optical Flow (OF) branches of a TSTCNN architecture to highlight contributions of pixels both in video frames (RGB) and motion vectors (OF) into the final decision. Contrary to the popular visualization methods which are based on back-propagation with gradient computation, the proposed method uses only features value and global importance of features channels. It is compared with the state-of-the-art gradient-based techniques such as Vanilla Gradient-based Back-propagation [4], Guided Back-propagation and Grad-CAM [5].

The rest of the paper is organized as follows. Section 2 discusses related works on CNN features understanding techniques. Section 3 explains the proposed algorithm and Section 4 compares the results of different test cases between the proposed algorithm and existing algorithms. Finally, conclusion and discussions are drawn in Section 5.

II. Related work

Deep Neural Networks (DNN) are commonly considered as black-boxes; indeed they are composed of stacked layers

978-1-7281-8750-1/20/$31.00 ©2020 IEEE

of individually very simple functions whose parameters correspond to weights that mainly depend on the training data. However, the global function that represents such a network is hardly understandable. Several works has been done to mitigate this issue by visually providing hints on the decision taken by the DNN [6]. Such visual application often relies on a view that provides features importance in the input space (*i.e* which feature positively voted for the final classification). Two main approaches are used: i) methods on the basis of back-propagation and gradient computation and ii) methods based on back-tracing feature values.

A. Methods with back-propagation and gradient computation

In the pioneering work by Simonyan et al. [4], the authors are interested in the so-called "class-saliency": what are the pixels which contribute the most into the decision of assigning an image to a particular class. A class score function $S_c(I)$ of a CNN was considered as a linear operation of convolution of an input image and a weight vector w. In reality the decision function of a CNN is highly non-linear. Hence the Taylor expansion is used and the derivatives (gradient) of the weights with regard to the argument -image I in the vicinity of I_0 were computed. Strong derivatives of weights indicate pixels which contribute to the decision the most. This allowed to build the so-called "class-saliency maps". The Grad-CAM method [5] is based on the same principle. It generates a heatmap that highlights features of interest by backgpropagating the gradient of the last layer until it reaches a convolution to compute the influence of the neurons on the prediction. The importance map is then upscaled to the initial image size in order to produce the heatmap. Zintgraf *et al.* [7] generates a heatmap that indicates the input pixels that voted against the predicted class and those that voted for. The method relies on difference analysis that modifies the input space in order to detect how the prediction changes if the feature is unknown. Despite the gradient is not computed as in [4], it is kind of "differential" approach.

B. Methods based on back-tracing feature values

The Guided Backpropagation uses the neuronal responses in high-level feature maps and propagates them back to the image thus finding pixels which contributed the most into the response of a single neuron [8]. When a single neuron is left non-zero in the high level feature map the resulting reconstructed image shows the part of the input image that is the most strongly activating this neuron. The authors of [8] work with fully convolutional neural networks which does not contain max pooling layers and thus the "tractability" of a neural response to the original pixel is possible. Layer-Wise Relevance Propagation (LRP) [9] also generates a heatmap of input features that supported the decision. The method relies on the concept of a relevance score per activation; the sum of all relevance scores of each layer is equal. Li *et al.* [10] generate salience relevant maps thanks to firstly LRP generated maps. This additional step allows to highlight part of image following human attention mechanisms by removing irrelevant parts highlighted by LRP. VisualBackProp [11] aims at visualizing

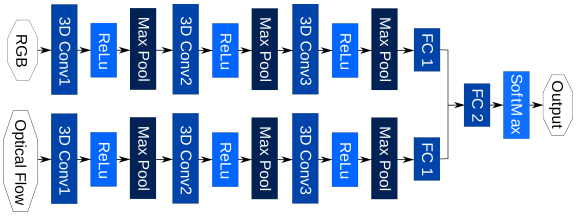

Fig. 1. Analyzed TSTCNN architecture.

the pixels at the origin of the decision in order to help to debug CNN in real time; the method can be used both during training and inference. The method is quite simple: the output of each ReLU layer is averaged, up-scaled to the resolution of the previous layer and multiplied by the previous layer. The operation is identically repeated until the input layer.

Our proposed approach relies on the feature maps of the last convolutional layer only following the philosophy that in Deep CNNs all feature layers except the fully connected (FC) layers are "feature extractors". Hence, we only use the last one, the most relevant features for the final decision to identify the salient regions in the video frames due to their back-propagation. The method is presented in the next section.

III. Proposed Method

The proposed method is generic and can be applied to features understanding in 2D or 3D spatio-temporal convolutional networks. We shortly present the architecture of this later type in the following section.

A. 3D spatio-temporal CNN for action recognition

The TSTCNN for action recognition has been already proposed and explained in [1]. We shortly review it. It is illustrated in figure 1. It is a twin network which consists of 2 individual branches of the same architecture. It comprises three 3D conv blocks consisting of 3D conv layers, ReLu and Max Pooling layers. The number of filters and hence features maps from input to output respectively is of 30, 60 and 80, followed by a fully connected layer of size 500. Each 3D conv layer uses $3 \times 3 \times 3$ space-time filters. The pooling operation is performed uniformly in space and time with a factor of two. The first branch, the upper one in figure 1 takes, as input data, chunks of cropped video frames using RGB values of pixels. The second branch takes as input the motion vectors $V = (v_x, v_y)$ computed using the"Beyond Pixel" [12], [13] estimator on the same cropped frames. Velocity fields v_x and v_y are considered as the input data channels. The two branches are fused through a final fully connected layer of size 21, representing the number of classes in the taxonomy proposed in [1], ending by a Softmax function for computation of a classification score.

B. Features understanding method

The core of the method we propose relies in the back-tracing of "strong" features from the last feature-layer (conv layer). It "explains" the Network decisions at the generalization step.

At the generalization step, the chunk of input video frames to classify is forward-propagated through the trained network. In a CNN the conv layers act as features extractors and the last FC-layers - as classifiers. The upper conv layers extract low level features [14] from input data, while deeper ones extract higher level semantic features. Hence, we extract features from the last conv layer. The features are taken after the activation layer and just before the fully connected layers. The overview of our proposed algorithm is given in Figure 2. Here the method is illustrated for one video frame in RGB data without loosing generality.

By pushing the input data (chunks of video frames of size $(W \times H \times T)$=$(120 \times 120 \times 100)$, through the convolution and polling layers of the network, the input video frames become "feature frames". Their number is reduced by pooling in temporal dimension by a factor of 2. Accordingly to the chosen number of filters and pooling factors in spatial and temporal dimensions [1] before fully connected layers we get $F = 80$ feature maps containing each $N = 12$ feature frames of size 15×15. The proposed method is applied to each feature frame and the resulting N importance maps $M'_n, n = 1, ...N$ are back-projected on all input frames by trilinear interpolation in space and time.

The second step generates a binary map for each channel of this feature map in order to give an importance value for each feature channel-by-channel. To detect the strongest features, we suppose that the features values in features maps follow Gaussian distributions. Obviously, the mean is positive as we take the features after a commonly used ReLu non-linearity that transforms negative values to 0.

In the last convolutional layers features are "expressive" and only few of them are important. Following the Gaussian distribution hypothesis we are interested in the right queue of the distribution corresponding to "rare" and strong features. Hence we threshold the features maps $x_{i,c}$ accordingly to the K-sigma rule. c mean μ_c and standard deviation σ_c are calculated for each channel. Then a binary map per channel is built which marks the strong features as in eq. 1:

$$b_c(R(x_{i,c})) = \begin{cases} 1 & \text{if } x_{i,c} \geq \mu_c + K * \sigma_c \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Hence after thresholding, in each channel we have marked the strongest features (see the illustration of binary features maps in the figure 2 in the bottom part).

Next, not all features channels are globally significant for decision making. The number of convolutional filters in each layer is often chosen on the basis of preliminary experiments, full optimization of the network hyper parameters being too heavy. We propose to weight channel contributions accordingly their importance. The importance is measured by the mean value $w_f = \mu_f$ of all the features in a channel after the ReLu layer over the temporal dimension. In our case, according to the size of the features before the fully connected layer, the channel importance matrix is composed a $F = 80$ vectors of size $N = 12$.

Then, we compute the importance map M as a linear combination of all channel binary maps b_c using the channel weights μ_c and normalize it to $[0; 1]$ by using Min-Max feature scaling. Finally, the normalized importance map M' is up-scaled to the original image/video frame dimension $W \times H$ by a linear interpolation giving $M'_{W,H}$. Further, the importance map $M'_{W,H}$ is superimposed on the original image as a heat-map to visualize the spatial information which has contributed the most into strong features of the last conv layer. Therefore, we visualize the importance map without calculating the gradient.

IV. Experiments and results

All experiments have been conducted using the Twin Spatio-Temporal CNN - TSTCNN - model trained on `TTStroke-21` presented in [1], and shortly described in Section III-A. The dataset, and the training process are presented in Section IV-A, our visualization results are compared with common methods in Section IV-B and computation times of all methods are compared in Section IV-D.

A. Experimental Protocol

In this work we used `TTStroke-21` dataset composed of 129 videos recorded at 120 frames per second. It was annotated by experts of table tennis. The beginning, the end and the type of stroke performed were labeled. It contains a total of 1387 crowd-sourced annotations for 20 stroke types and a rejection class. After filtering, a total of 1164 samples are considered with 1058 strokes and 106 negative samples. Each sample represents a video segment of length ranging from 99 frames (0.82s) to 276 frames (2.30s) with mean length of 174 ± 43.14 frames (1.46 ± 0.36 sec). Those samples are then split in 3 sets: "Train set" with 808 samples, "Validation set" with 230 samples and "Test set" with 116 samples.

Training step was performed over 2000 epochs. The optimization algorithm used was Stochastic Gradient Descent with Nesterov Momentum. The momentum coefficient was set to 0.5 and decreased to 0.1 and 0.05 respectively at epoch 1000 and 1500, with a learning rate of 0.001, Cross-entropy loss, weight decay of 0.005 and a batch-size of 10 are other hyper parameters of the model. The input of the model is of size $(W \times H \times T)$=$(120 \times 120 \times 100)$ representing the width, the height and the temporal dimension (duration) of our cropped region of interest (ROI), from original frames of size (320), with 3 channels for RGB and 2 channels for OF. The RGB values are normalized by theoretical maximum (255 in our case of 8-bit coding of each color channel). The OF data are normalized using the "Normal" method [13] described in eq. 2 where v and v^N represent respectively one component of the OF $\mathbf{V}$ and its normalization, and μ and σ are standing for the mean and the standard deviation of the distribution of maximum OF magnitude values over the whole `TTStroke-21` dataset.

$$v' = \frac{v}{\mu + 3 \times \sigma}$$
$$v^N(i,j) = \begin{cases} v'(i,j) & \text{if } |v'(i,j)| < 1 \\ SIGN(v'(i,j)) & \text{otherwise.} \end{cases} \quad (2)$$

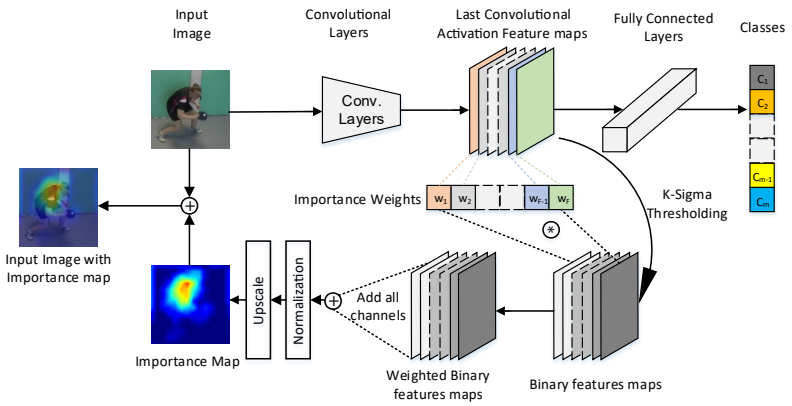

Fig. 2. Overview of proposed visualization algorithm. Features are extracted from the last convolutional layer after the activation function (upper part). Binary features maps are generated. Importance weights are calculated. Importance map is computed as a normalized linear combination with channel weights and visualized as a heat-map on the original image

The training samples are randomly augmented at each epoch using a random translation in x and y direction respectively in the range of $\pm 0.1 * W$ and $\pm 0.1 * H$, a random rotation in the range of $\pm 10°$, and a random homothecy in the range of 1 ± 0.1. Temporal augmentation is also done by selecting the T successive frames of the sample following a normal distribution, around the temporal center of the segmented video [1]. Without data augmentation, the T frames from the RGB and OF are centrally extracted in the temporal dimension. Model with the best accuracy on the validation set has been saved and used in our experiment in features explanation. This model attained 91.3% and 87.9% of accuracy respectively on the validation set and on the test set.

B. Visual Analysis

Visualization was initiated with PyTorch Code from [15]. The author developed the code for different visualization algorithms considering Single Branch CNNs in 2D. We extended it for Multi Branches CNNs and for 3D application. Our video results are available online[1], the code will be made available too.

To keep the comparison uniform in the Figures 3 and 4, all the importance maps are scaled between 0 and 1 using Min-Max Feature scaling and are visualized as heatmap over the input frame using *"jet"* color scale represented on the right of the Figures. Even if the data processed are in 3D, we only show one frame from the 100 frames of the video input for better visualization. In Figure 3 the OF are visualized by converting $V = (v_x, v_y)$ into an image in the color domain HSL where the Hue represents the angle created by v_x and v_y, the Saturation is set to 1 and the Lightness represents the amplitude of the motion.

Figure 3 presents Table Tennis stroke Defensive Backhand block with different visualization algorithms for both RGB and OF data input. From visual observation, we can see that Vanilla Gradient-Based Back-propagation(VaGrBp) [4], Guided Back-propagation(GuBp) [8] and Guided Grad-CAM(GuGrc) [5] visualizations suffer from "discretization effect". On the other hand, continuous and smooth visualization has been obtained in Grad-CAM(GrC) [5] and Our Algorithm. If we observe the RGB data and it's visualizations, in the Figure 3. a-f, Vanilla Gradient-Based Back-propagation has focused on all over the body and table but Guided approaches focused mostly on the upper body and the Ball. In contrast, Grad-CAM and Our Algorithm focused on the left leg and the hands as they are most important features that are changing over the time. Regarding OF (see Figure 3.g-k), our algorithm highlights the ball mainly (remember that we select only the most prominent and "rare" features in the last conv layer, see Section III-B), as most algorithms but Vanilla Gradient-Based and Guided BP give quite a noisy picture of "important" motion vectors on the body.

We have also conducted experiments to check if the choice of the features from the last conv layer was justified, see motivation for this in Section III-B. Figure 4 illustrates a typical visualization (on one stroke) using features of different conv layers of the analyzed 3D CNN.

Furthermore, our method uses features thresholding with K-sigma rule 1. Hence we use the popular 2-sigma and 3-sigma rules supposing Gaussian distribution of features, i.e. $K = 2$ and $K = 3$. What comes out of these experiments, is that the features from the third conv layer are the most coherent with the motion of the person and the ball (the third row) and that the 3-sigma rule is more interesting as allows to filter-out contrasted features on static objects and concentrate on changing details captured due to 3D convolutions. The objective visual evaluation of explanation requires a large user study which is in the perspective of our work. Nevertheless, we can quantify the similarity of resulting visualization maps obtained by our method and by the baseline methods. This comparison is presented in the following section.

[1]https://github.com/asifahmedfuad/feature_understanding_method

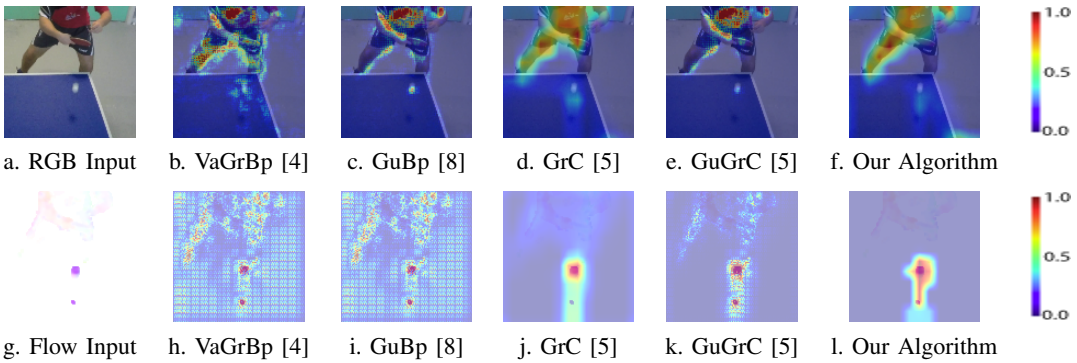

Fig. 3. Different Visualization algorithm outputs of our model for the class: Defensive Backhand block. First row shows the visualization for RGB input data and second row does the same for Flow input data.

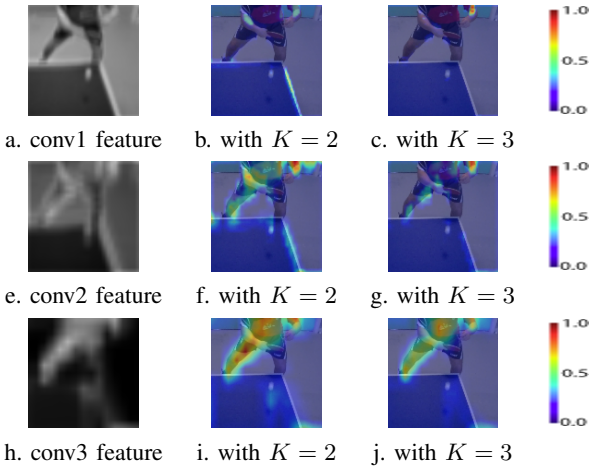

Fig. 4. Visualization with features from different conv layers and Influence of K - value in thresholding. At top row, result with first conv layer features. At middle row, second conv layer features and at the third row, results with third conv layer features

C. Metric-based Comparison of methods

To compare our method with the baselines we use a common strategy of bench-marking of algorithms for prediction of visual attention. Here we will compute the so-called "Similarity" and Pearson correlation coefficient metrics [16]. We perform the comparison on the normalized importance map M re-scaled to the resolution of input frames, see Section III.

The *Similarity (SIM)* [16] is a popular metric to perform a simple comparison between importance maps. Given two importance maps P_1 and P_2, Similarity Metric is:

$$SIM(P_1, P_2) = \sum_i min(P_{1i}, P_{2i}) \tag{3}$$

iterating over the discrete pixel i where $\sum_i P_{1i} = \sum_i P_{2i} = 1$. Two completely overlapping importance maps will result in maximal Similarity of 1 and 0 when there is no overlapping which is good for partial matches [16].

The *Pearson's Correlation Coefficient (PCC)* [16] measures how two maps are correlated or depend on each other: it is close to 1 when two variables are perfectly correlated and 0 when they are not at all. For two importance maps P_1 and P_2, PCC is:

$$PCC(P_1, P_2) = \frac{\sigma(P_1, P_2)}{\sigma(P_1) \times \sigma(P_2)} \tag{4}$$

where $\sigma(P_1, P_2)$ is the covariance of P_1 and P_2.

TABLE I
COMPARISON OF VANILLA GRADIENT-BASED BACK-PROPAGATION [4] (VAGRBP) AND OF OUR METHOD (OURS) WITH GUIDED BACK-PROPAGATION [8] (GUBP), GRAD-CAM [5] (GRC), GUIDED GRAD-CAM [5] (GUGRC)

Methods	RGB		Flow	
	Similarity	PCC	Similarity	PCC
	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
VaGrBp vs GuBp	$.77 \pm .01$	$.75 \pm .02$	$.99 \pm .01$	$.99 \pm .01$
VaGrBp vs GrC	$.69 \pm .04$	$.61 \pm .08$	$.66 \pm .02$	$.54 \pm .06$
VaGrBp vs GuGrC	$.73 \pm .02$	$.70 \pm .03$	$.79 \pm .02$	$.80 \pm .03$
Ours vs VaGrBp	$.70 \pm .03$	$.63 \pm .05$	$.70 \pm .01$	$.61 \pm .02$
Ours vs GuBp	$.70 \pm .03$	$.64 \pm .05$	$.70 \pm .01$	$.61 \pm .02$
Ours vs GrC	$.70 \pm .05$	$.63 \pm .10$	$.72 \pm .06$	$.69 \pm .12$
Ours vs GuGrC	$.68 \pm .03$	$.63 \pm .05$	$.77 \pm .02$	$.71 \pm .02$

There has been many sanity checks done on visualization algorithms. In the paper [17], Adebayo *et al.* suggest Vanilla Gradient-Based Back-propagation (VaGrBp) is more effective than other visualization algorithms, as a result, it is taken as reference for comparison. In Table I, complete metric evaluation of the test set of `TTStroke-21` dataset is provided. The test set is composed of 118 instances over 21 classes. We calculated both mean and standard deviation to characterize stability on different instances. In our observation on the test cases, Similarity and PCC metrics are coherent for all the samples and algorithms for both RGB and OF data.

Comparing all state-of-the-art methods between them, we state that Vanilla Gradient-based Back-propagation (VaGrBp) and Guided Back-propagation (GuBp) have highest Similarity and PCC. These results are obvious as both algorithms rely on almost the same principle of using first conv layer gradients except on how they treat the gradients. Vanilla plots both positive and negative gradients whereas Guided Back-propagation plots only positive gradients. For Guided

TABLE II
COMPUTATION TIME FOR THE DIFFERENT VISUALIZATION TECHNIQUES

Visualization techniques	Time (in sec)
	$\mu \pm \sigma$
Guided Grad-CAM [5]	11.3691 ± 0.6154
Guided Back-propagation [8]	7.7530 ± 0.4282
Vanilla Back-propagation [4]	5.1233 ± 0.2309
Grad-CAM [5]	4.9033 ± 0.2131
Proposed method*	$\mathbf{2.9100 \pm 0.1322}$

Grad-CAM, Grad-CAM output is multiplied with Guided Back-propagation output. Hence, Vanilla Gradient-based Back-propagation has higher Similarity and PCC with Guided Grad-CAM. In contrast to Vanilla Gradient-based Back-propagation [4] and Guided Back-propagation, Grad-CAM uses last conv or deep conv layer gradients. Grad-CAM has SIM and PCC 69% and 61% respectively for RGB data but for OF data, it has SIM 66% and CC 54% which is nearly 30% and 40% less than compared to Guided Back Propagation.

Comparing our algorithm with state-of-the-art methods, it has 70% Similarity and 63% PCC with Vanilla Gradient-Based propagation (VaGrBp) and Guided Back-propagation (GuBp) for RGB data. Compared to Grad-CAM(GrC), our algorithm has 5% similarity and 15% PCC more with respect to Vanilla Gradient-Based (VaGrBp) Visualization for OF data. With Grad-CAM, our algorithm yields the most similar balanced results: 70% similarity for RGB and and 72% for Optical Flow data. At the end, with Guided Grad-Cam, our algorithm has 77% similarity and 71% Correlation on OF data. This is explained by the use of the features of the same (last) conv layer.

D. Computational Analysis

All the algorithms have been implemented using Python based PyTorch and Numpy libraries for the visualization of our developed CNN model. The same functions were used for calculating similar functionality of different algorithms to make computational analysis uniform. Average time for each instance visualization of different algorithms is given in Table II. Model training was performed on GPU while visualization was implemented on CPU only. Computation time for each method was measured in Intel(R) Xeon(R) Gold 5118 CPU @2.3GHz and Intel(R) Core(TM) i9 9900 CPU @3.1GHz. In both CPUs, similar results were obtained. In the Table II, computation time is provided only for Intel(R) Xeon(R) Gold 5118 CPU @2.3GHz. Over all, average computation time was calculated on 118 instances of test set each having 100 image frames. From the Table II, it is clear that our algorithm is the fastest among all the visualizations since it does not contain a time-consuming gradient back-propagation. Our algorithm is almost two times faster than Vanilla Gradient-Based visualization and Grad-CAM.

V. CONCLUSION

We have proposed a new method for explanation of visual features of CNNs in classification tasks. It is generic and can be applied both to 2D and 3D CNNs. It is based on selection of important features in last conv layer, on the use of channel importance and back-projection of the feature importance maps to the original frames. We have shown that the method gives comprehensive results both on RGB input and on optical flow in a twin 3D conv net for table tennis stroke classification. We also compared it to the known features visualization methods with the help of classical similarity metrics used for saliency/importance maps. The method gives very much similar results in terms of Similarity and Pearson Correlation coefficient with regards to the Vanilla Gradient Back-propagation and the result is also the most similar to Grad-CAM. It remains faster than all considered baseline methods. In future work we intend to compare it with available Gaze Fixation density maps of observers in video action recognition tasks. We also intend to extend it using more information from the whole network architectures.

REFERENCES

[1] P. Martin, J. Benois-Pineau, R. Péteri, and J. Morlier, "Fine grained sport action recognition with siamese spatio-temporal convolutional neural networks," in *Multimedia Tools and Applications*, 2020.

[2] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1510–1517, 2018.

[3] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," *CoRR*, vol. abs/1705.07750, 2017.

[4] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *ICLR*, 2014.

[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[6] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[7] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," *International Conference on Learning Representations*, 2017.

[8] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," in *ICLR*, 2015.

[9] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: an overview," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 193–209.

[10] H. Li, Y. Tian, K. Mueller, and X. Chen, "Beyond saliency: understanding convolutional neural networks from saliency prediction on layer-wise relevance propagation," *Image and Vision Computing*, vol. 83, pp. 70–86, 2019.

[11] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, and K. Zieba, "Visualbackprop: efficient visualization of cnns," *arXiv preprint arXiv:1611.05418*, 2016.

[12] C. Liu, "Beyond pixels: Exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 5 2009.

[13] P. Martin, J. Benois-Pineau, R. Péteri, and J. Morlier, "Optimal choice of motion estimation methods for fine-grained action classification with 3d convolutional networks," in *ICIP 2019*. IEEE, 2019, pp. 554–558.

[14] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 4898–4906.

[15] U. Ozbulak, "Pytorch cnn visualizations," https://github.com/utkuozbulak/pytorch-cnn-visualizations, 2019.

[16] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand, "What do different evaluation metrics tell us about saliency models?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 3, pp. 740–757, 2019.

[17] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," pp. 9525–9536, 2018.