Fact sheet: CVPR 2021 ChaLearn Looking at People Large Scale Signer Independent Isolated SLR Challenge

I. Team details

- **Challenge Track** RGB
- Team leader name: Marek Hrúz
- Username on Codalab: hpoes
- Team leader affiliation: University of West Bohemia, Faculty of Applied Sciences, Department of Cybernetics; NTIS
- Team leader address: Technická 8, 301 00 Plzen, Czech Republic
- Team leader phone number: +420 377632555
- Team leader email: mhruz@ntis.zcu.cz
- Name of other team members (and affiliation): Ivan Gruber, Zdeněk Krňoul, Jakub Kanis
- Team website URL (if any): http://eyes.zcu.cz/

II. Contribution details

A. Title of the contribution

Our method is composed of several approaches combined in an ensemble scheme to perform isolated sing-gesture recognition. We combine modalities of video sample frames processed by a 3D ConvNet (I3D), with body-pose information in the form of joint locations processed by a Transformer, hand region images transformed into a semantic space, and linguistically defined locations of hands. Although the individual models perform sub-par (60% to 93% accuracy on validation data), the weighted ensemble results in 95.46% accuracy.

B. Introduction and Motivation

Our main motivation was to use a state-of-the-art model of gesture/action recognition and augment it with other approaches based on different modalities. We wanted to analyze the performance of an ensemble scheme when different models utilizing different data are used. As the state-of-the-art model for gesture/action recognition we use I3D [1] and finetune it from Kinetics400 dataset [2] to several representations of the challenge data. To incorporate the motion of hands we detect the body joints using OpenPose [3] and predict the sign class using a transformer model inspired by the Vision Transformer [4]. The information about the pose of the hands is added by our Visual Language Embedding (VLE) model. In this work, we present a proof-of-concept that transforms images of hands into a semantic space, where similar poses lie close to each other. We use concepts from other vision tasks, that show that deep neural networks trained for the classification of images fulfill the requirement of embedded space in the penultimate layer. We finetune a MobileNet [5] architecture pre-trained on ImageNet [6] to classify our mined dataset of hand images. To add information about the location of the hand, we developed an algorithm for computing linguistically defined locations of hands. We compute the location vectors from OpenPose detections and together with VLE input them into a Transformer to classify the sign. Lastly, we compute weights of these different models using Covariance Matrix Adaptation Evolution Strategy (CMA-ES) optimization algorithm [7].

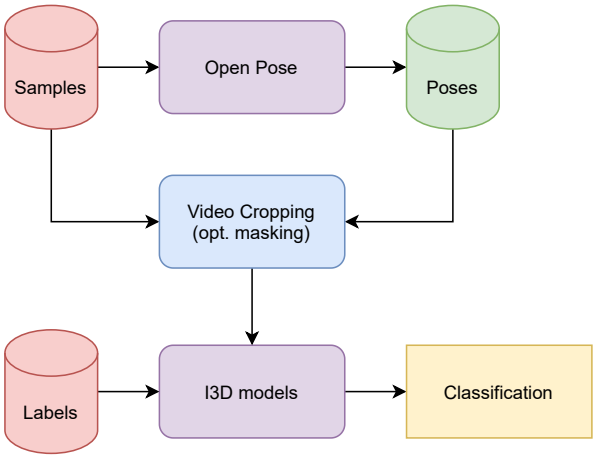

Fig. 1. The processing of samples by OpenPose. The poses are then used to crop the video recordings so that we remove unimportant parts of the image. The crop is per-video constant. Optionally, we apply a mask, so that only regions with body-parts are visible. These videos are used to train the different I3D models.

C. Representative image / workflow diagram of the method

In this section, we provide supportive images. Generally, red cylinders depict ”outside” data provided by the organizers; green cylinders are data produced by us during this challenge; purple rounded rectangles are 3rd party methods; blue rectangles are our own models/methods.

D. Detailed method description

In this section, we describe the pipeline and its individual parts. The first step of processing is running OpenPose [3] on all the video samples. We detect all bodies in the video and choose the largest one for further processing. We use the model BODY_25 (default model) to detect 8 body joints (face, neck, left/right shoulder, left/right elbow, and left/right wrist) and 21 joints per hand, meaning in total 50 body joint locations per frame. We also store the confidence of the detected joints.

Next, we prepare data for training the I3D models (Fig. 1). We crop the RGB video frames on a per-video basis. We

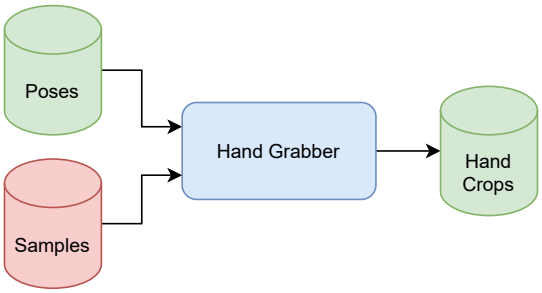

Fig. 2. Extraction of hand images (crops) from samples. The grabber is configurable. In this work we extract square regions enclosing all hand joints and resize them to 70×70 pixels. Only hands with mean detection confidence higher than 0.4 are considered.

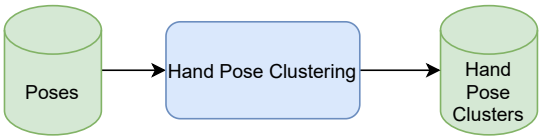

Fig. 3. Hand pose clustering. The clustering is based on the computation of weighted pose distance. First, we find per-sign clusters and then we cluster these across all the signs.

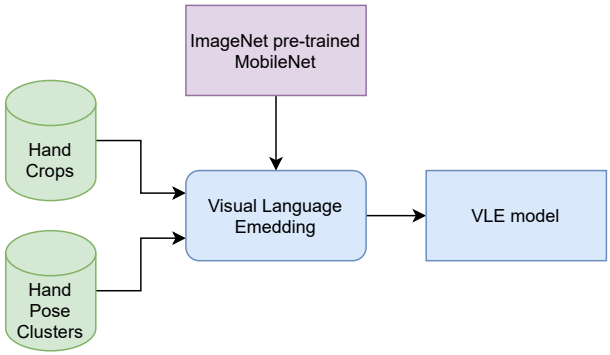

Fig. 4. We train the VLE model from the hand crops. Each crop has an assigned cluster (class) from the prior processing. The VLE is a MobileNet pre-trained on ImageNet and fine-tuned on our hand pose clusters.

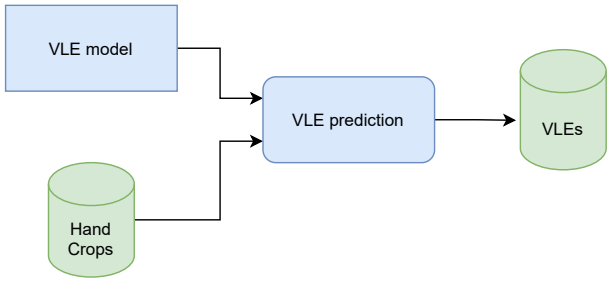

Fig. 5. To obtain the VLE representation of test data, we have to obtain the hand crops from test video samples using the process depicted in Fig. 2. Then the model produces VLEs that are a semantic representation of the hand images.

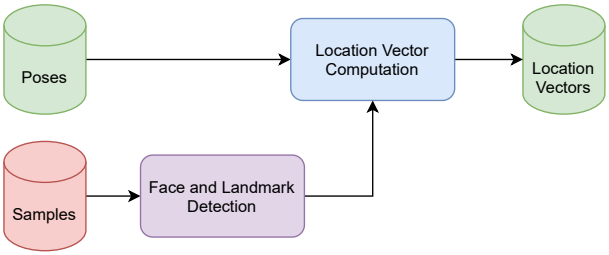

Fig. 6. The location vectors represent the location of individual hands in individual frames relative to other body-parts. We selected 15 locations depicted in Fig. 7. Since many locations represent facial landmarks, we compute them using the dlib library.

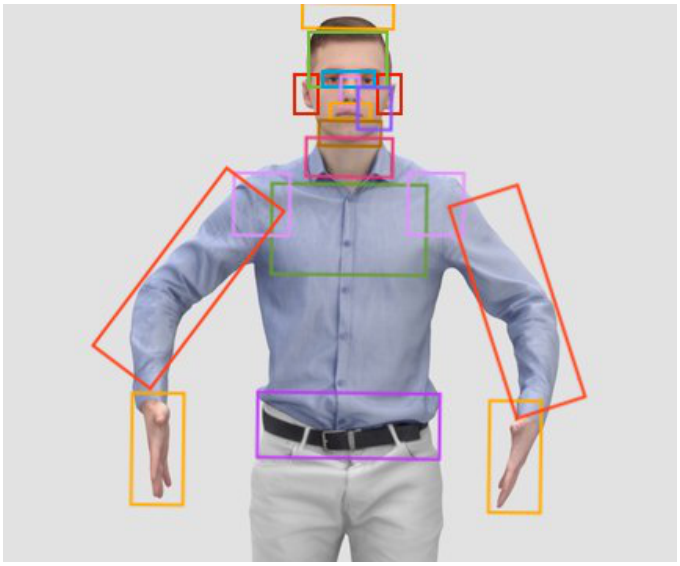

Fig. 7. Regions representing different body-locations, from which the location vectors are computed.

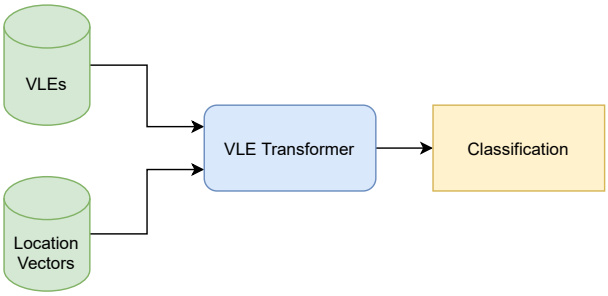

Fig. 8. The VLE-transformer takes as input a concatenation of the VLEs and Location Vectors. It has a form of Vision Transformer.

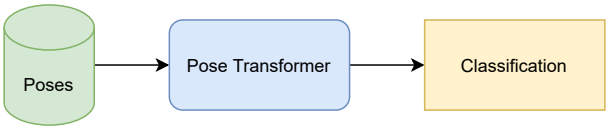

Fig. 9. The Pose-transformer takes as input a fraction of the poses from OpenPose. It has a form of Vision Transformer.

use the body joints detected by the OpenPose. Firstly we get a scale of the body in the sample according to Euclidean distance of the shoulder body joints in the first frame. Next, we crop all remaining frames of the sample relative to the scale (four times), centered in the x-axis on the neck joint and y-axis is below the neck joint by 30% of the shoulder distance. Finally, the crops are resized to the size of 256×256.

Furthermore, we prepare a masked version of these cropped videos. We start by preparing a binary mask by rendering the detected hand skeleton and face region. We repeat a 3×3 dilatation on the per-pixel hand skeleton binary mask, for metacarpal bones by a factor of 4, for reaming hand bones by a factor of 2, and for neck/hand bone by a factor of 20. These masks are then used on the original videos to produce the masked versions in which only the important body parts are visible.

We also represent the videos in terms of key-frames. We detect a constant number of frames per video, where there is minimal motion. In our experiments, we have chosen 16 key-frames per video. To obtain the key-frames we first compute the velocity vectors of the detected joints of the whole body as a simple difference in x and y axis. Next, we compute the magnitude of these vectors and sum up all the joint velocities. The key-frames are then the $N = 16$ minimal locations in the velocity signal. To suppress the detection of nearby key-frames we use a non-minimum suppression with a window of ± 3 frames.

1) Visual Language Embedding: VLEs are used as the input for the VLE-Transformer. The premise of the system is to train a deep neural network that will transform an input image into a semantic vector space. This space should have a property that similar hand poses are close to each other. To train such a model we first needed to obtain images of hands in the same pose (i.e. configuration of fingers). The input hand images are obtained using the algorithm depicted in Fig. 2. We use two consecutive algorithms. The first one finds representative hand poses for each sign. We consider only the dominant hand (left hand in the mirrored video samples, even for left- or both-handed signers). We set a parameter of how many representative hand poses per video sample we want to detect at most (in our case 5). We sort the detected OpenPose hand joints by the mean confidence. We consider only hand-poses with a minimum confidence of 0.6. We apply a non-maximum suppression of ± 5 frames to suppress the detection of the same hand-pose from the same video sample. For each other video sample, we find all hand-poses that have a distance smaller than 0.42. The distance measure is defined as:

$$D(p_1, p_2) = \min_{\mathbf{A}} \sum \|\mathbf{A}p_1 - p_2\|, \tag{1}$$

where p_1 and p_2 are the two hand-poses represented as a set of 21 2D vectors. The matrix $\mathbf{A}$ is a similarity transform (i.e. restricted to scale, rotation and translation). The sum is performed over the 21 joints. The result is in the metric of pixels. Since p_1 and p_2 can stem from images of different resolution a normalization needs to be performed. In this step of the solution, we normalize to the length of the shoulder. This normalization is not perfect and hence the function is not symmetric and thus is not a real distance. But for the purpose of our solution, it is sufficient. This algorithm produces a detection of per-sign representative hand-poses. Next, we want to cluster these representatives, because one sign can be composed of more representative hand-poses. We employ an agglomerative clustering with the distance function from Eq. 1.

In the second algorithm, we want to join these clusters from different signs to obtain the definite hand-pose clusters. We find representative hand-poses for each per-sign cluster. It is the hand-pose that has a minimal sum of distances to all the other hand-poses. During experimentation with the clustering we modified the distance measure (Eq. 1) so that the similarity transform is found only on palm joints:

$$\mathbf{A}^\star = \min_{\mathbf{A}} \sum_k \|\mathbf{A}p_1 - p_2\| \tag{2}$$

where k represents the palm joints (MCPs, CMC, and wrist). And the distance is weighted:

$$D(p_1, p_2) = \sum_i \omega_i \|\mathbf{A}^\star p_1 - p_2\|, \tag{3}$$

where ω_i is weighting different finger joints. Precisely fingertips have the weight of 3.0, DIPs have the weight of 2.0, PIPs have weights of 1.5. The rest of the weights are 1.0. Finally, they are normalized to a sum of 1. This algorithm leaves us with 52 final hand-pose clusters. When observing the data, we found some errors that were a result of the imperfections in the distance measure. The main problem is that we are limited to 2D distance computation and perspective plays a huge role (larger than we anticipated). Hence, we corrected the errors manually and ended up with 39 hand clusters. Unfortunately, the clusters were heavily imbalanced.

The last stage was the training of the deep neural network. We performed experiments with ResNet-18 [8], MobileNet [5] and a custom model. We tested randomly initialized models and models pre-trained on ImageNet. From these experiments, we selected MobileNet pre-trained on ImageNet [6] to serve as the VLE extractor. We used SGD for optimization with a learning rate of 0.001 and momentum of 0.9. We augment the training data using color jitter, horizontal flip (to accommodate for the right hand), per-pixel Gaussian noise, grid distortion, motion blur, random brightness, and contrast transform, RGB shift, rotation of max ± 10 degrees, random crop and resize. The categorical cross-entropy was used as the optimization criterion. After 40 iterations of finetuning we obtain a 95% training accuracy and 65% validation accuracy. The validation data were from two left-out signers. This shows the high sensitivity to the hand shape, and perspective transformations of the hands (since signers perform the signs with different hand orientations). The VLE is the 1280 dimensional vector from the penultimate layer of

MobileNet. The algorithms are depicted in Figures 3, 4 and 5.

2) Location Vectors: We wanted to incorporate knowledge from the field of sign language linguistics, namely the location of the performed sign. We define 15 locations: Neutral space (fallback), Above the head, Upper part of the face, Eyes, Nose, Mouth, Lower part of the face (chin), Cheeks, Ears, Neck, Shoulders, Chest, Waist, Arm, Wrist (of the other hand). The algorithm computes the location of both hands for every frame. First, it detects whether the hand is in a "pointing" gesture by computing the extension of the index finger. If so, the fingertip of the index finger is used for relative location computation. If not, then a mean location computed from all hand joints is considered. The hand location is then compared to the defined regions (see Fig. 7) and the closeness to the region center is computed. A soft-max vector representing the relative closeness to each region is constructed for each hand and concatenated into a 30-dimensional vector (see Fig. 6).

3) I3D: In our final ensemble, we utilize 13 I3D models [1] in total. Their implementation is based on https://github.com/IBM/action-recognition-pytorch. Before the training, each video was cropped (with optional masking) based on detected poses, see Fig. 1. Furthermore, 16 frames with a size 256×256 pixels were selected per video. The selection of these frames is based on two different methods. The first method is a pseudo-random choice from the original repository (denoted as random). The second method is based on our key-frames (denoted as key-frames), see subsection above. All I3D models can be divided into three groups.

The first group of four models was trained before the start of the competition test phase. The validation set for these models were signers 40, 41, and 42. The rest of the signers were in the training set. The models in the first groups were trained during 50 epochs using SGD optimizer with starting learning rate $lr = 0.01$ and cosine learning schedule. These models were fine-tuned using the whole training set after the start of the competition test phase during 20 additional epochs using SGD with starting learning rate $lr = 0.001$ and cosine learning rate schedule.

The second group of four models was trained after the start of the competition test phase. The validation set for these models was competition development data. The models were trained during 80 epochs using SGD optimizer with starting learning rate $lr = 0.01$ and cosine learning schedule.

The third group of five models was trained under 5-fold cross-validation protocol using a competition training set only. Each fold was selected manually with respect to different signers. The models were trained during 50 epochs using SGD optimizer with starting learning rate $lr = 0.01$ and cosine learning schedule again.

All the I3D models were pretrained on the Kinectics400 dataset [2]. Data were normalized to the ImageNet mean and standard deviation. We used batch size $bs = 10$ for all the experiments. Moreover, group center crop is used during the training. A comparison of the models can be found in Table I. The models with the highest validation recognition rate were selected for the final evaluation.

TABLE I

DETAILS OF THE MODELS. THE RECOGNITION RATE IS CALCULATED ON THE COMPETITION DEVELOPMENT SET.

Model	**Data**	**Frames Slc.**	**Rec. Rate**
I3D-Crop	crops	random	0.9232
I3D-Key	crops	key-frames	0.9090
I3D-Mask	masked	random	0.9178
I3D-Key_mask	masked	key-frames	0.9063
I3D-Crop_new	crops	random	0.9292
I3D-Key_new	crops	key-frames	0.9149
I3D-Mask_new	masked	random	0.9187
I3D-Key_mask_new	masked	key-frames	0.9035
I3D-Cros-1	crops	random	0.9029
I3D-Cros-2	crops	random	0.9117
I3D-Cros-3	crops	random	0.9095
I3D-Cros-4	crops	random	0.8993
I3D-Cros-5	crops	random	0.8952
Pose-transformer	Body-Pose	whole video	0.8660
VLE-transformer-1	VLE-Locations	key-frames	0.6034
VLE-transformer-2	VLE-Locations	key-frames	0.6048
VLE-transformer-3	VLE-Locations	key-frames	0.6517

4) Transformer model: Other than I3D models, we also used four transformer models in our final ensemble. To be more specific, we utilized vision transformer architecture [4]. The difference from the original architecture lies in the embedding layer that prepares the input data. We utilized feed-forward multi-layer perceptron, which transforms the input vectors of the given parametrization into the transformer input dimension. In our experiments, we used two different input parametrizations: Body-Pose, and a combination of VLE and Location vectors.

Body-Pose parametrization is based on selected 2D keypoints provided by OpenPose. These skeleton data are further pre-processed per frame for each video sample. The whole dataset is normalized to have a uniform distance of the shoulders of signing persons. This distance is calculated from the first frame of each sample. Furthermore, the skeleton data are centered according to the position of the neck. The final pre-processing step corrects hand poses for low confidence images (confidence < 0.3). These hand poses are replaced with poses from surrounding frames.

During the training of the model, we utilized the following augmentations:

- random drop of the first 10-15 frames from beginning and the end of the video;
- random selection of even/odd frames;
- random horizontal flip of the data (simulation of left/right handed signing) ;
- Gaussian noise addition to wrist locations and hand-pose scale.

In our final ensemble we used one model based on the body-pose parametrization (Pose-transformer) with the following parameters: max length of the sequence: 120, size of the input vector: 84, N-stages: 2, transformer size: 1024, size of feed-forward layer: 2048, number of heads: 2.

VLE and Location Vectors are concatenated into a single

TABLE II

ENSEMBLE WEIGHTS FOR THE SPECIFIC MODELS.

Model	Weight
I3D-Crop	0.04762968
I3D-Key	0.05915348
I3D-Mask	-0.02789492
I3D-Key_mask	0.12705846
I3D-Crop_new	0.13529561
I3D-Key_new	0.04292918
I3D-Mask_new	0.07635797
I3D-Key_mask_new	0.03582622
I3D-Cros-1	0.05322100
I3D-Cros-2	0.12860186
I3D-Cros-3	0.02714533
I3D-Cros-4	-0.05312429
I3D-Cros-5	-0.04703171
Pose-transformer	0.17253467
VLE-transformer-1	0.08502941
VLE-transformer-2	0.11312029
VLE-transformer-3	0.02414777

vector with a size of 2590. Only key-frames are selected from the whole sequence, which forms the input sequence for the embedding layer. We did not use any additional augmentations during the training phase.

In our final ensemble we used three models based on the VLE-Locations parametrization (VLE-transformer) with the following parameters:

1) max length of the sequence: 120, size of the input vector: 2590, N-stages: 6, transformer size: 1024, size of feed-forward layer: 2048, number of heads: 2;
2) max length of the sequence: 120, size of the input vector: 2590, N-stages: 2, transformer size: 512, size of feed-forward layer: 2048, number of heads: 2;
3) identical as 2) but with additional learn-able positional encoding.

All the transformer models were trained for 100 epochs using SGD optimizer with starting learning rate $lr = 0.1$ and learning rate exponential shift $ex = 0.95$. The models with the highest validation recognition rate were selected for the final evaluation. A comparison of the models can be found in Table I.

5) Final ensemble: The final ensemble is composed of 13 I3D models, 1 Pose transformer model, and 3 VLE transformer models. To compute weights of these models, we utilized CMA-ES optimization algorithm. The input into the algorithm are predicted soft-max values for development set, whereas algorithm should maximize the development set recognition rate. Weight for each model can be found in the Table II-D.5.

E. Challenge results and final remarks

TABLE III

LEADERBOARD: RESULTS OBTAINED BY THE PROPOSED APPROACH.

Phase	Track	Rank position	Rec. Rate
Development	x	x	x
Test	RGB	12	0.9546

Given the nature of the competition, our method is based on combining many models that perform well on the development data. The work needs to be polished and when the testing phase will be open for experimenting we can perform additional ablation studies. We plan to focus on individual components of the system and also the possibility of training them dependently on each other.

III. ADDITIONAL METHOD DETAILS

Please reply if your challenge entry considered (or not) the following strategies and provide a brief explanation.

- **Did you use any kind of depth information (directly, such as RGBD data, or indirectly such as 3D pose estimation trained on RGBD data), either if during training or testing stage?** ( ) Yes, (X) No

- **Did you use pre-trained models?** (X) Yes, ( ) No
  All I3D models were pre-trained on the Kinetics400 dataset. The MobileNet for VLE was pre-trained on ImageNet.
- **Did you use external data?** ( ) Yes, (X) No

- **Did you use other regularization strategies/terms?** ( ) Yes, (X) No

- **Did you use handcrafted features?** (X) Yes, ( ) No
  The location vectors are handcrafted.
- **Did you use any face / hand / body detection, alignment or segmentation strategy?** (X) Yes, ( ) No

  The location vectors use information about facial landmarks (e.g. eyes, mouth, nose, ears, ...) and whole body pose. The pipeline is heavily dependent on OpenPose joint locations estimation.
- **Did you use any pose estimation method?** (X) Yes, ( ) No
  We use OpenPose [1].
- **Did you use any fusion strategy of modalities?** (X) Yes, ( ) No

  We fuse (a) sign recognition from video frames (I3D), (b) sign recognition from pose change (Transformer), (c) sign recognition from hand locations and appearances (VLE+Transformer).
- **Did you use ensemble models?** (X) Yes, ( ) No
  We used the ensemble of the following models: Crop, Key, Mask, Key_mask, Crop_new, Key_new, Mask_new, Key_mask_new, Cros-1, Cros-2, Cros-3, Cros-4, Cros-5, Pose-transformer, VLE-transformer-1, VLE-transformer-2, and VLE-transformer-3. The optimal weights of the ensemble were obtained by CMA-ES (Covariance Matrix Adaptation Evolution Strategy) optimization algorithm. CMA-ES is a stochastic optimizer for robust non-linear non-convex derivative- and function-value-free numerical optimization. As an

[1] https://github.com/CMU-Perceptual-Computing-Lab/openpose

starting optimization point we used equally weighted ensemble.

- **Did you use any spatio-temporal feature extraction strategy?** ( ) Yes, (X) No

- **Did you explicitly classify any attribute (e.g. gender)?** ( ) Yes, (X) No

- **Did you use any bias mitigation technique (e.g. rebalancing training data)?**
  ( ) Yes, (X) No

IV. Code repository

Code repository: `https://github.com/mhruz/ChaLearn-SLR`

References

[1] C.-F. Chen, R. Panda, K. Ramakrishnan, R. Feris, J. Cohn, A. Oliva, and Q. Fan, "Deep analysis of cnn-based spatio-temporal representations for action recognition," *arXiv preprint arXiv:2010.11757*, 2020.

[2] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *CoRR*, vol. abs/1705.06950, 2017.

[3] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[7] H.-G. Beyer and B. Sendhoff, "Covariance matrix adaptation revisited – the cmsa evolution strategy –," in *Parallel Problem Solving from Nature – PPSN X*, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 123–132.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.