Key Point-Based Driver Activity Recognition

Arpita Vats
Santa Clara University
Santa Clara, CA, USA
avats@scu.edu

David C. Anastasiu
Santa Clara University
Santa Clara, CA, USA
danastasiu@scu.edu

Abstract

We present a key point-based activity recognition framework, built upon pre-trained human pose estimation and facial feature detection models. Our method extracts complex static and movement-based features from key frames in videos, which are used to predict a sequence of key-frame activities. Finally, a merge procedure is employed to identify robust activity segments while ignoring outlier frame activity predictions. We analyze the different components of our framework via a wide array of experiments and draw conclusions with regards to the utility of the model and ways it can be improved. Results show our model is competitive, taking the 11th place out of 27 teams submitting to Track 3 of the 2022 AI City Challenge.

1. Introduction

Driver activity recognition is an important problem with real-world applicability. According to the National Highway Traffic Safety Administration, 9 people die in the United States every day due to distracted driving [1]. It is thus imperative that we use technology to reduce this statistic and, in time, eliminate it altogether. Towards that end, the AI City Challenge [9, 10] has introduced a new challenge track this year, posing the problem of driver activity recognition from video. Teams are provided with videos of 5 different drivers performing the 18 activities denoted in Table 1, in random order, for varying short intervals (typically 10 seconds) and with varying short breaks between activities. Each set of activities is performed twice, the second time with some facial occlusion, e.g., wearing a hat or sun glasses. The experiments are recorded from three vantage points, *dashboard*, *rear view mirror*, and *right window*, and teams are provided with synchronized videos and activity labels for the training set.

In this work we present a key point-based activity recognition framework, named KNDAR[1], that uses complex static and movement-based features extracted from key frames in the videos to classify activity segments.

[1] https://github.com/davidanastasiu/kndar

Table 1. Activity Classes

ID	Class	ID	Class
0	Normal forward driving	9	Adjust control panel
1	Drinking	10	Pick up from floor (driver)
2	Phone call (right)	11	Pick up from floor (passenger)
3	Phone call (left)	12	Talk to passenger at the right
4	Eating	13	Talk to passenger at backseat
5	Text (right)	14	Yawning
6	Text (left)	15	Hand on head
7	Hair / makeup	16	Singing with music
8	Reaching behind	17	Shaking or dancing with music
9	Adjust control panel	18	N/A

2. Related Works

Driver activity recognition is a relatively new area of study. However, several related domains, such as video anomaly detection and human pose estimation, have seen significant recent advances due to novel neural network architectures. Recent techniques combine Convolutional Neural Networks (CNNs) with more complex feature extractors, such as Long Short-Term Memory networks (LSTMs) and Autoencoders [7]. Xu *et al*. [13] suggested using a stacked denoising auto-encoder in order to extract feature representations for motion and appearance in videos. Our framework relies on pose estimation and facial feature detection models to construct its complex features, which we will discuss next.

2.1. Pose Estimation

In a recent study, Papandreou *et al*. [11] proposed a technique for detecting multiple people in the same frame using a 2-stage top-down 2-D pose estimation of each person. The method first predicts the bounding boxes likely to contain human beings, and then, for each bounding box, the model estimates the positions of 17 human body key points (12 body joints and 5 face landmarks). The models were trained using the COCO key points dataset as well as an ad-

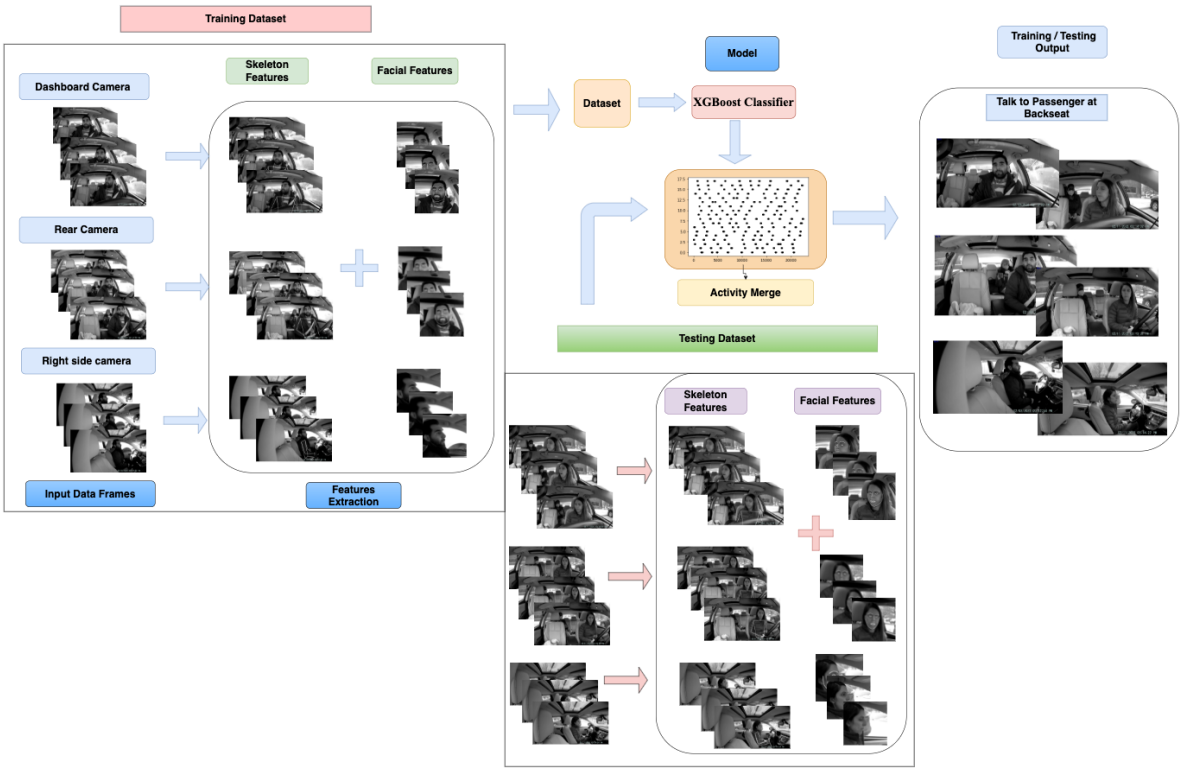


Figure 1. Activity segment prediction framework.

ditional in-house labeled dataset. The method is similar to that of Kendall *et al*., named PoseNet. For every person it identifies in the frame, it predicts a vector of 17 body key points together with a confidence score for each key point.

A similar multi-person 2D pose estimation model was recently proposed by Cao *et al*. [2]. Their method is however a bottom-up approach, which they argue is able to estimate high-quality body poses even as the number of persons in the frame increases, unlike some top-down approaches. McNally *et al*. [8], on the other hand, rely on a dense detection network to predict a set of key point objects and a set of pose objects. This network is designed to simultaneously detect both object types with minimal computational overhead using a single shared network head. Furthermore, they conclude that their model, named KAPAO, can be effectively applied to the problem of single-stage multi-person human pose estimation by detecting human pose objects directly. Moreover, fusing jointly detected key point objects improves the accuracy of the predicted human poses with minimal computational overhead

Yang *et al*. [14] introduced a Transformer-based model for human pose estimation, which they named *Transpose*. It uses an attention layer to capture the long range relationship in a pose effectively. The model is considered to be more lightweight and faster than any mainstream CNN architectures.

2.2. Facial Features

Recently, there has been impressive progress on the problem of detecting facial expressions, which has become available to the general public via open source and user-friendly libraries. Cheong *et al*. [4] proposed Py-Feat, a python open source toolbox that provides support for detecting, processing, analyzing, and visualizing facial expression data. Similarly, Zhou *et al*. [15] proposed a facial recognition algorithm based on semantic features (including eyes and mouth), which are further evaluated using tensors subspace analysis. Silva *et al*. [5] proposed the idea of facial feature extraction and emotion recognition in real time using edge computing and image correlation optical flow techniques, which calculate the local motion vectors of facial features. Kim *et al*. [6] explores a similar approach for face detection using edge detection and a 3D-CNN classification model that extracts spatial and temporal features.

3. Methodology

Figure 1 depicts our framework. The main idea of our method is that driver activity can be determined from the movement of key points on their body. As such, we can use a pre-trained pose estimation model to identify key points

Figure 2. Human pose estimation for a frame of a video, showing detected key points and segments depicting major bones in the human body. Best viewed in color.

Figure 3. Extracting facial features for a given pose.

(e.g., nose, right ear, left wrist), and derive features from these key points. The features we designed include angles between key points (e.g., angle of the segments created by nose, left ear, and left eye), distances between key points (e.g., left wrist to left hip), position information (distance from the center of the driver's bounding box to corner of the image), shifts between some of the current key points and the respective ones in the last key frame, and shifts between certain angles and the respective angles in the last key frame. Finally, a merge algorithm is needed identify, based on the individual frame activity predictions, segments of time when the driver is engaged in the same activity and to predict the activity of each segment. In the following, we will describe the methods we employed for each component in our framework.

3.1. Driver identification

Our framework relies on a pre-trained human pose estimation model to detect humans in the video and their associated poses. While any such model may be used, we employed the KAPAO model by McNally *et al*. [8]. Given an image extracted from the video, the model identifies key points of the pose of any humans in the image, as shown in Figure 2. Noting that persons in the front seat of the car will appear larger in the image, and relying on the fact that drivers will appear on the right side for U.S.-based cars, our method then identifies the pose of the driver as the one with a larger area and a bounding box center on the right-half of the image. At times, the pose estimation model fails to identify the driver and only identifies the passenger. Our method filters our such spurious detections and the framework only extracts features from frames in which the driver has been detected.

3.2. Feature extraction

We designed our method to extract both static features, based on the current key frame at time t, and motion features, based on differences between the current and a past key frame at time $t-k$. This strategy not only allows faster processing, but also enhances our motion features as the driver's movements will be more pronounced between key frames. However, as we will see in Section 4, the meta-parameter k must be carefully chosen to ensure the captured motion features are meaningful for the frame activity classification problem.

Our method extracts a variety of feature types that were carefully designed as indicators of both the driver's position and movement. Feature values of each type are also normalized during extraction to ensure feature values are roughly in the range $[-1, 1]$. Table 2 lists all the features our method currently captures, which are extracted as follows.

- **Angles.** We measure the cosine of angles formed by certain key points in the human pose estimation. The angle between the driver's right wrist, right elbow, and right shoulder, for example, will be very different during normal driving vs. when they are talking on the phone and holding the phone with their right hand next to their right ear. The cosine of an angle already has a range between $[0, 1]$ and does not need to be normalized.

- **Distances.** We measure the Euclidean distance between key points, such as the right wrist to the right ear, and normalize distances by the width of the image, w, i.e., $d_i = \frac{d_i}{w}$.

- **Positions.** We measure the driver bounding box center position relative to the upper left corner of the image (the origin) and its relative width and height. Positions are also normalized by the width of the image, w.

- **Position shifts.** We measure the distance between key point positions in the current and last key frames. Given that position shifts are expected to be much smaller than positions, they are normalized by $0.25 * w$.

- **Angle shifts.** We measure the absolute difference between certain angles in the current and last key frames.

- **Facial features.** The KAPAO pose estimation model detects both body and some basic face key points (eyes, nose, and ears). In order to capture whether the driver may be singing or talking, our method optionally uses the `face_recognition` Python package[2] to extract additional face key points and measures the distance from the bottom of the upper lip to the top of the lower lip. Figure 3 shows an example of detecting additional facial key points using the `face_recognition` package. We tested our model's performance both with and without this feature.

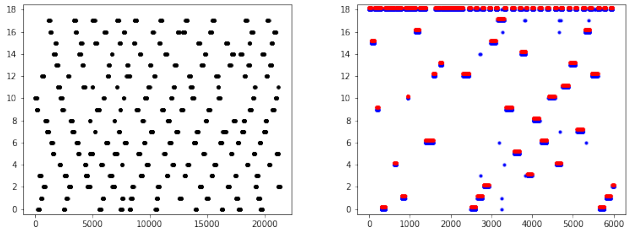
Our input data consists of three separate views of the driver (*dashboard*, *rear view*, and *right side*). Our method can extract features from either of the views or from all views at the same time. In the *all* view case, our method sequentially extracts features from each view of a given driving scenario and merges the constructed feature vectors by concatenating feature frame vectors for the same frame in each of the views. A frame may not produce a feature vector for a certain view if the pose estimation model fails to detect the driver. In that case, the frame will not be included in the *all* view dataset.

3.3. Frame activity classification

Our framework uses the features extracted from frames in the training set videos, along with their associated labels, which are provided by the organizers, to train a classification model that is able to predict the driver's activity in teh given frame. Our hypothesis is that the rich set of features we extract from the frames is sufficient to accurately predict driver instantaneous activity. Towards this goal, we trained models using both an `LSTM`-based neural network [12] and the popular `XGBoost` classifier [3]. Given the relative small number of extracted features per frame, the LSTM model performed poorly for this task and it will not be included in our results in Section 4.

XGBoost (eXtreme Gradient Boosting) [3] is a scalable implementation of gradient boosting, which uses gradient descent to identify additional shallow models (in this case shallow decision trees) that may lead to a smaller prediction error. The target for the next shallow model to be iteratively added to the ensemble is chosen based on the gradient of the error with respect to the prediction, which gives the method its name. The method stops learning when additional shallow models do not reduce the error or it has reached the maximum number of iterations. Predictions are made using a weighted vote by each of the models in the ensemble.

[2] https://github.com/ageitgey/face_recognition



(a) (b)

Figure 4. (a) Activity segments for some videos in the training set. Class IDs 0–17 are depicted vertically. (b) Class labels in a subset of the training set; red labels are predicted labels while blue ones are the ground truth, slightly offset vertically for visibility. Class ID 18 (N/A) is also shown, denoting times between activity segments. Best viewed in color.

3.4. Segment activity classification

Figure 4 (a) depicts a visualization of the activities in the training set. As can be seen, the training set contains a variety of activities which are in general repeated for roughly 10 seconds, and there is generally a gap of a few seconds between activities. Activities are fairly evenly distributed across the dataset and most times separated by at least a gap of 5 seconds. As shown in Figure 4 (b), our model is able to predict the frame-wise training set activities fairly well, yet some outliers do appear from time to time. Having been told the test set follows a similar pattern as the training set, we devised the `Merge` algorithm to identify activity segments by merging sequences of the same activity of sufficient length after first removing outliers.

Algorithm 1 depicts our `Merge` algorithm. It takes as input the sorted list of class predictions for all frames of a video and provides as output a list of time segments and the associated activity class label for each. It works by combining the individual per-frame predictions of the activity classification model into segments in which the driver is assumed to be doing the same activity. It first looks for classes with a much higher percentage of predictions than expected. $S_{\cap x}$ on Line 2 represents the elements of the list S with value x, which are replaced with $\varnothing$ (representing the N/A class, or class ID 18) if their relative count is higher than *maxp*. Then, short empty gaps connecting segments of the same class are filled in (Line 5). Finally, maximal consecutive class segments of length of at least *minlen* are added to the result set R (Line 10), which is returned on Line 15.

4. Experimental Evaluation

In this section, we will present a series of experiments we conducted to train and evaluate our framework. As seen from Table 3, our model achieved an F1 score of 0.2528 on the competition test set A2, which earned us the $11th$ place

Algorithm 1 The Merge algorithm for segment activity classification.

Input: S, maxgap, minlen, maxp
Output: R
Require: $|S| \geq$ maxgap

$R \leftarrow \varnothing$
for all $x \in S$, s.t. $|S_{\cap x}|/|S| \geq$ maxp **do**
  $x \leftarrow \varnothing \; \forall x \in S_{\cap x}$
end for
for all i, j s.t. $S_{i-1} = S_{j+1} \wedge S_k = \varnothing \; \forall k = i \ldots j$ **do**
  if $j - i <$ maxgap **then**
    $S_k = S_{i-1} \; \forall k = i \ldots j$
  end if
end for
for all s, e s.t. $S_{i-1} = S_i \; \forall i = s+1 \ldots e \wedge S_{s-1} \neq S_s \wedge S_e \neq S_{e+1}$ **do**
  if $e - s >$ minlen **then**
    $R \leftarrow R \cup (s, e, S_s)$
  end if
end for
return R

out of 27 teams that submitted results to the Track 3 public leader board. As we present our experimental evaluation, we will also discuss potential pitfalls of our method and ways that it may be improved. We executed all our experiments on a system running Ubuntu Linux version 20.04 and equipped with a 12-core Intel(R) Core(TM) i9-7920X CPU @ 2.90GHz, 128 GB RAM, and 4 NVIDIA RTX 3090 24G GPUs. However, while portions of our method take advantage of multi-threaded CPU-based processing, our method only uses one GPU.

4.1. Feature Extraction

Our feature extraction method has a number of meta-parameters, including which camera view should be used to extract features, how many frames should be skipped between each key frame being captured, and whether or not facial features should be extracted. We made the later choice optional as it adds considerable time to the preprocessing stage of the framework, due to the `face_detection` package is CPU-bound and, in general, single-threaded. While features could be extracted from most videos at a rate of 120 fps on average on our system, the same processing would slow to 15-20 fps when enabling facial feature extraction.

We executed a series of experiments over all combinations of the following parameters: we extracted features from from the dashboard (*dash*), rear view mirror (*rear*), side window (*side*), or all camera views (*all*), taking key frames every $k \in \{3, 5, 10\}$ frames of each video, and extracted features both with and without facial features. In all, we executed experiments using 24 different feature sets.

4.2. Frame activity classification

For each experiment detailed in Section 4.1, we split the training dataset into a training set and a validation set using a 80/20% split and trained a large number of XGBoost models, optimizing for the best validation F1 score. The XGBoost classifier has several meta-parameters, including number of epochs, the learning rate α of the gradient descent algorithm, the maximum number of estimators (decision tree weak learners), and the maximum depth of each decision tree. Due to lack of time, we used the default learning rate and set the number of epochs at a high value (500) while enabling early stopping, and focused primarily on tuning the maximum number of estimators and maximum depth parameters. We tested values between 100 and 1000 in increments of 100 for the first and between 4 and 14 in increments of 2 for the later. We skipped some experiments when it was clear results would not improve upon already existing models, e.g., models with number of estimators above 300 for $k = 10$.

Figure 5 (a) shows the effectiveness results of our frame activity classification model. Overall, we found the *all* view was superior to single-camera views at capturing the driver's action. Moreover, capturing key frames every 3 frames of the video outperformed the other key frame choices. Also, while the top-4 validation F1 scores among our tested models belong to models built using the added facial features, this type of model only comprised 41% of the top-100 performing models, achieving an average F1 score of 0.9548, while models without the feature had an average F1 score of 0.9662.

One explanation of the mixed results with facial features may be the quality the face detection model we chose to use. We observed that, in many cases when the driver was not looking toward the camera (e.g., while driving normally but being viewed from the side window camera), or while they were reaching down for something, the face detection algorithm failed to detect any faces. This scenario would lead to assuming a distance between the upper and lower lips of 0, which is equivalent to the driver's mouth being closed (whether in reality it was or not). Employing a better model for mouth movement prediction may significantly improve this result, as several of the target classes have to do with the driver talking or singing.

4.3. Segment activity classification

Our `Merge` algorithm for segment activity classification has three additional parameters, namely maximum gap size to be filled (*maxgap*), minimum segment length (*minlen*), and maximum class percentage (*maxp*). We set the *maxp* parameter to an overly-cautious 0.15 value, which is more than twice higher than the average class percentage within

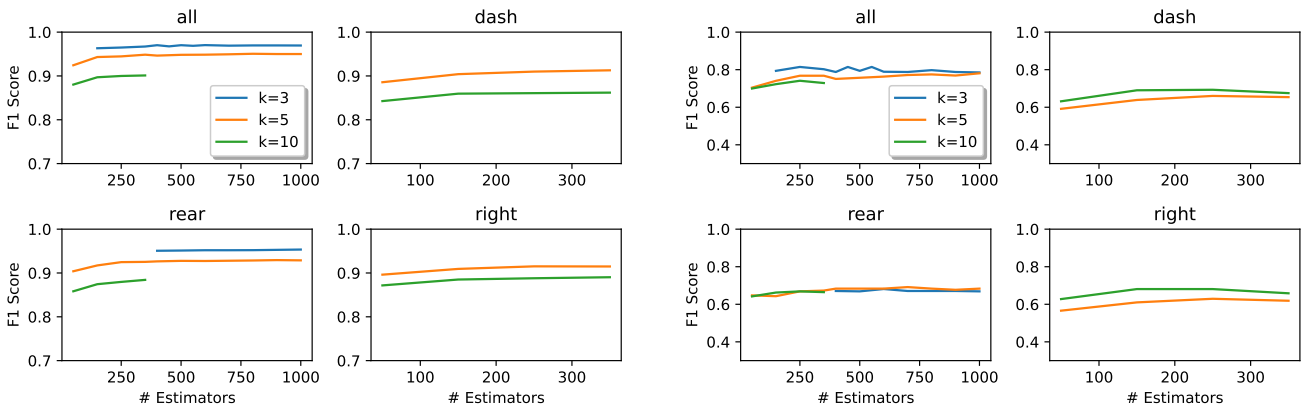


Figure 5. Effectiveness of our driver activity classification model given key frames extracted every k frames and using video from the dashboard (*dash*), rear view mirror (*rear*), side window (*side*), or all camera views (*all*). Best viewed in color. (a) Validation set effectiveness of our XGBoost key frame-based activity classification model. (b) Validation set effectiveness of our overall segment activity classification model.

the training set, and tuned the *maxgap* and *minlen* parameters using the validation set. We tested *maxgap* in the range $[0-90]$ in increments of 10, i.e., 0–3 seconds, and *minlen* in the range $[250, 650]$ in increments of 50. Finally, we chose among several of the top-performing models to predict activity segments on the test set and submit results to Track 3 of the AI City Challenge.

Surprisingly, while models built using $k = 3$ seem to always outperform those built using $k = 5$ in the frame activity classification task, the effectiveness of the models is quite similar for the segment activity classification task, both on the validation and the test sets. Figure 5 (b) shows a narrow gap between the performance of the $k = 3$ and $k = 5$ models for the segment activity classification task. Moreover, the top-2 most effective test set models were "all-5-f-600-8" and "all-5-f-500-8", i.e., built using all camera views, key frames every 5 frames, with added facial features, maximum XGBoost tree depth of 8, and the number of estimators set to 600 and 500, respectively.

4.4. Feature importance

After we identified the best performing model, once the challenge was closed, we were interested in identifying which of its 171 features was most important to achieve good overall predictions. Figure 6 plots the importance of all the model features, but it only shows the feature labels on the x axis for the top-10 most important features. These are, in order, features 19, 22, and 30 (see Table 2 for reference) from the *dashboard* view, capturing relative driver position, face rotation, and position of the left elbow, features 2, 5, 19, and 21 from the *rearview mirror* view, capturing two joint angles and two relative positions of the driver, and features 6, 18, and 21 from the *right side* camera view, capturing the right shoulder and two relative positions of the driver.

Interestingly, features from the right side camera view tend to have higher importance in the prediction. Moreover, angles and distances have higher importance than other feature types, while position shifts, angle differences, and facial features all have bellow average importance towards the accurate prediction of the in-frame driver activity. More research and experiments are needed to improve the utility of the movement-specific features, which may improve the overall segment activity classification performance. One possible line of research which we did not have time to investigate would be extracting features based on a longer history of movements, rather than just movements between the current and the last key frames.

5. Conclusion

In this work, we presented a framework for driver activity prediction that uses off-the-shelf pre-trained human pose estimation and facial feature detection models to predict time sequences in which the driver is performing one of a set of pre-defined actions. Our model first extracts a series of both static and movement-based features from key frames in the videos, and then uses a classification model to identify the potential action being performed in each key frame. Finally, it merges sequences of action predictions into robust action segments while at the same time removing outliers. We have tested our model under a wide array of meta-parameter choices and found it to be competitive in Track 3 of the 2022 AI City Challenge, achieving $11th$ place on the public leader board out of 27 teams submitting

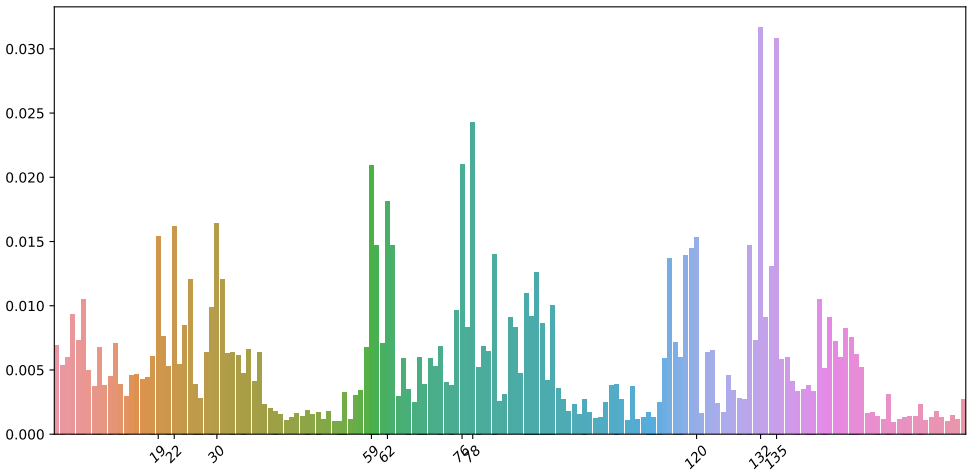


Figure 6. Feature importance for our best achieving model, "all-5-f-600-8".

results.

References

[1] National Highway Traffic Safety Administration. Traffic safety facts research note: Distracted driving 2019 (dot hs 813 111), Apr 2021. Accessed on 04/14/2022. 1

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(1):172–186, jan 2021. 2

[3] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. 4

[4] Jin Hyun Cheong, Tiankang Xie, Sophie Byrne, and Luke J. Chang. Py-feat: Python facial expression analysis toolbox, 2021. 2

[5] L.C. De Silva and Suen Chun Hui. Real-time facial feature extraction and emotion recognition. In *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint*, volume 3, pages 1310–1314 vol.3, 2003. 2

[6] Dong-Keon Kim and Kwang-Su Kim. Generalized facial manipulation detection with edge region feature extraction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2828–2838, January 2022. 2

[7] B. Ravi Kiran, Dilip Mathew Thomas, and Ranjith Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, 4(2), 2018. 1

[8] William McNally, Kanav Vats, Alexander Wong, and John McPhee. Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation, 2021. 2, 3

[9] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E. Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. The 5th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021. 1

[10] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Liang Zheng, Anuj Sharma, Rama Chellappa, and Pranamesh Chakraborty. The 4th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, page 2665–2674, June 2020. 1

[11] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286, 2018. 1

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press. 4

[13] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. In Xianghua Xie, Mark W. Jones, and Gary K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 8.1–8.12. BMVA Press, September 2015. 1

[14] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Keypoint localization via transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11802–11812, 2021. 2

[15] Huiyu Zhou, Yuan Yuan, and Abdul H. Sadka. Application of semantic features in face recognition. *Pattern Recognition*, 41(10):3251–3256, 2008. 2

Table 2. Extracted Features

FID	Feature
	Angles
0	nose, left lower corner, image left upper corner
1	left wrist, left lower corner, image left upper corner
2	right wrist, left lower corner, image left upper corner
3	left elbow (left shoulder, left elbow, left wrist)
4	left shoulder (left elbow, left shoulder, left hip)
5	right elbow (right shoulder, right elbow, right wrist)
6	right shoulder (right elbow, right shoulder, right hip)
7	left eye, nose, right eye
8	nose, left shoulder, right shoulder
9	nose, right shoulder, left shoulder
10	nose, left ear, left eye
11	nose, right ear, right eye
12	left ear, right hip, right shoulder
13	right ear, left hip, left shoulder
14	left shoulder, right hip, right shoulder
15	right shoulder, left hip, left shoulder
	Distances
16	nose to lower left corner
17	left wrist to lower left corner
18	right wrist to lower left corner
19	nose to upper left corner
20	left wrist to upper left corner
21	right wrist to upper left corner
22	nose to left shoulder
23	nose to right shoulder
24	nose to left ear
25	nose to right ear
26	left eye to right eye
27	left ear to right ear
28	left ear to left wrist
29	right ear to right wrist
30	left elbow to left hip
31	right elbow to right hip
32	left wrist to left hip
33	right wrist to right hip
	Positions
34	relative bounding box width
35	relative bounding box height
36	relative position of bounding box center x
37	relative position of bounding box center x
	Position shifts
38	nose
39	left eye
40	right eye
41	left ear
42	right ear
43	left shoulder
44	right shoulder
45	left elbow
46	right elbow
47	left wrist
48	right wrist
	Angle shifts
49	nose, left lower corner, left upper corner (of the image)
50	left wrist, left lower corner, left upper corner (of the image)
51	right wrist, left lower corner, left upper corner (of the image)
52	left elbow (left shoulder, left elbow, left wrist)
53	left shoulder (left elbow, left shoulder, left hip)
54	right elbow (right shoulder, right elbow, right wrist)
55	right shoulder (right elbow, right shoulder, right hip)
	Facial features
56	distance between upper and lower lip

Table 3. Top-15 of the AIC 2022 Track 3 Leader Board

Rank	Team Name	F1 score
1	VTCC-UTVM	0.3492
2	Stargazer	0.3295
3	CybercoreAI	0.3248
4	OPPilot	0.3154
5	SIS Lab	0.2921
6	BUPT-MCPRL2	0.2905
7	Winter is Coming	0.2902
8	HSNB	0.2849
9	VCA	0.271
10	Tahakom	0.2706
11	**SCU_Anastasiu**	**0.2558**
12	union	0.2301
13	Starwar	0.216
14	Aespa wInter	0.144
15	SEEE-HUST	0.1348