Group Members
1. Argho Das [ID: 21076891, Email: a86das@uwaterloo.ca]
2. Aman Sharma [ID: 21033571, Email: a74sharm@uwaterloo.ca]

Title:
Enhancing Human Action Recognition in Still Images: Comparative Analysis of Traditional Machine Learning and Deep Learning Methods with a focus on Custom CNN Architecture

Abstract:

An increasingly challenging task in the field of computer vision in recent times has been the automatic detection of human actions and activities. Furthermore, it is essential for many artificial intelligence applications, such as robots, video surveillance, computer gaming, and human-computer interactions. The increasing need for security and safety has led researchers to look into intelligent monitoring. The three primary elements of an action recognition system are feature extraction, action representation, and classification. Consequently, every stage is essential to attaining a high recognition rate. The action representation and feature extraction models feed raw data into a feature vector. The right feature extraction and selection would have a significant impact on the classification outcome. In order to develop the best human activity identification algorithm possible, we have selected seven human actions from a custom dataset. The activities are yawning, phoning, sitting, standing, walking, running and hand waving. This study evaluates the performance of our custom CNN model, with deep learning approaches, and conventional methods in identifying human activities from still photos.

Introduction:

Computer vision, a discipline within the realm of computer science, strives to emulate the intricate workings of the human visual system. Its purpose is to enable computers to perceive and analyze objects within images and videos, much like the way humans do. In the past, computer vision was limited in its capabilities, but recent advancements have expanded its potential. For instance, autonomous vehicles rely on computer vision to comprehend their surroundings [1]. Through an array of cameras capturing video from various perspectives, this technology processes the imagery in real time, identifying road boundaries, deciphering traffic signs, and recognizing other vehicles, objects, and pedestrians [2]. Consequently, self-driving cars navigate city streets and highways, expertly avoiding accidents as they safely transport passengers to their intended destinations. Moreover, computer vision is instrumental in facial recognition applications. By detecting distinct facial features within images, computer vision algorithms compare them to databases of individual profiles, allowing for the matching of faces to identities[3, 4]. Consumer devices employ facial recognition for owner verification, while social media platforms utilize it to identify and tag individuals. Law enforcement agencies also employ this technology to identify criminals captured in video footage, and it is frequently utilized in determining gender [5, 6, 7, 8]. Furthermore, computer vision plays a pivotal role in augmented and mixed reality, enabling computing devices like smartphones, tablets, and smart glasses to overlay virtual objects onto real-world visualizations. By employing computer vision, augmented reality devices can ascertain the precise placement of virtual objects within their display interfaces.

Related Studies:

Human action recognition is required for a variety of computer vision programs that require information about people's behavior, such as public safety surveillance, human–computer interaction, and robotics [9, 10, 11, 12, 13, 14]. Video-based human action recognition [15, 16, 17, 18], wearable sensor based human action recognition [19, 20, 21, 22, 23], and wireless sensor network-based human action recognition [24, 25] are all examples of human action recognition systems. The authors of [26] present a thorough analysis of human action recognition techniques and give a thorough overview of recent developments in the field. These developments include advancements in the recognition of human–object interactions, hand-designed action features in RGB and depth data, and action detection techniques, a hot topic in action recognition research right now. The authors of [27] provide a straightforward but efficient method for identifying human actions in video clips. Their method combines motion-based techniques with the benefits of human action identification in static photos by utilizing Motion History photos (MHI) and Motion Energy Images (MEI) variants. They used the MuHAVi data set for the action detection challenge. They used a leave-one-out cross validation process and were 98.5% accurate. Using the widely used Weizmann data sets, the authors obtain 100% accuracy for single-view action recognition. A human action recognition technique is provided in [28], where sequences of multiview key poses are used to teach actions, and pose representation is based on the contour points of the human silhouette. The author's method demonstrates applicability for online recognition and real-time scenarios by achieving state-of-the-art success rates without sacrificing recognition process speed. To the best of our knowledge, experimental results on the publicly available Weizmann, MuHAVi, and IXMAS datasets yield excellent and consistent success rates—the best rate on the MuHAVi Novel Actor test to date. A technique for identifying human activities using posture primitives is presented in [29]. The authors expand a Histogram of Oriented Gradient (HOG) based descriptor for position primitive recognition in order to better handle articulated stances and crowded backdrops. The authors use n-gram expressions to add the local temporal context for sequences. A basic comparison of histograms is the foundation for action recognition. The authors of [30] describe a unique approach to action recognition that combines the appearance invariance and adaptability of patch matching based algorithms with the effective description properties of local binary patterns. Due to its high level of efficiency, the final technique can be applied in real time for the simultaneous recovery of human actions at various lengths and beginning places. The authors of this work introduce a novel standard for broadcast sports video uncut motion recognition.

Dataset:

The UCF101 [31], ILSVRC [32], and HACS [33] datasets are only a few of the many datasets available for Human Action. However, those datasets were not used in this study because UCF101 and ILSVRC had activities that are not commonly performed by people, while HACS had mostly videos and images that are very similar. As a result, a custom dataset was constructed that included seven tasks that are carried out on a daily basis. The activities are yawning, phoning, sitting, standing, walking, running and hand waving. A total of 1974 images were taken. Hand waving 293 images, phoning 306 images, running 183 images, sitting 259 images, standing 320 images, walking 311 images and yawning 302 images were taken. After that the dataset was augmented in order to find a more optimal result. We take two images and combine them linearly using the tensors of those images. Finally Cutout is a convolutional neural network regularization technique that includes eliminating contiguous portions of input images, essentially augmenting the dataset with partially occluded copies of existing samples. Then all the images were converted into 300*300 dimensions in order to avoid overfitting. After that all the images were augmented to 11 angles. After augmentation the dataset contained 23736 images, where hand waving had 3516, phoning had 3672, running had 2196, sitting had 3156, standing had 3840, walking had 3732 and yawning had 3624 images. In the next step the entire dataset was split into an 80-20 ratio for dividing the training and testing dataset. In the training dataset hand waving had 2808, phoning had 2940, running had 1752, sitting had 2532, standing had 3072, walking had 2988 and yawning had 2904 images. In the testing dataset hand waving had 708, phoning had 732, running had 444, sitting had 624, standing had 768, walking had 744 and yawning had 720 images. The dataset is available at: https://rb.gy/n4bjj4

Execution Plan:

Our primary focus was to conduct a thorough analysis of image classification on our novel dataset using a step-by-step execution plan. The first phase involved the development and evaluation of a Custom Convolutional Neural Network (CNN) model. In the next phase, we aimed to contrast the performance of our Custom CNN model with that of other widely used, pre-built machine learning models. The traditional models to be evaluated included Naive Bayes, Support Vector Machine (SVM), K-nearest Neighbor (KNN), Random Forest, Gaussian Naive Bayes, and Decision Tree. Following the evaluation of traditional models, we evaluated the performance of deep learning models like ResNet, AlexNet, VGG16, and DenseNet on our dataset. The goal is to discern whether the pre-trained models can effectively generalize and perform competitively on our dataset. We also evaluate the performance of untrained deep learning models too.

The Custom CNN Model:

The CNN architecture presented here is tailored for image classification tasks, encompassing distinct segments for convolutional, dense, and output layers. The architecture encompasses six convolutional layers, each employing 3x3 kernels, coupled with Rectified Linear Unit (ReLU) activation functions and max-pooling operations. This strategic combination enables the model to progressively reduce spatial dimensions while increasing the depth of feature maps, allowing for the capture of detailed patterns in the input images. Notably, the convolutional layers are structured to provide a hierarchical representation of image features, with the spatial dimensions decreasing as the model delves deeper into the network. Following the convolutional layers, a dropout layer is introduced to prevent overfitting, further enhancing the model's generalization capability.

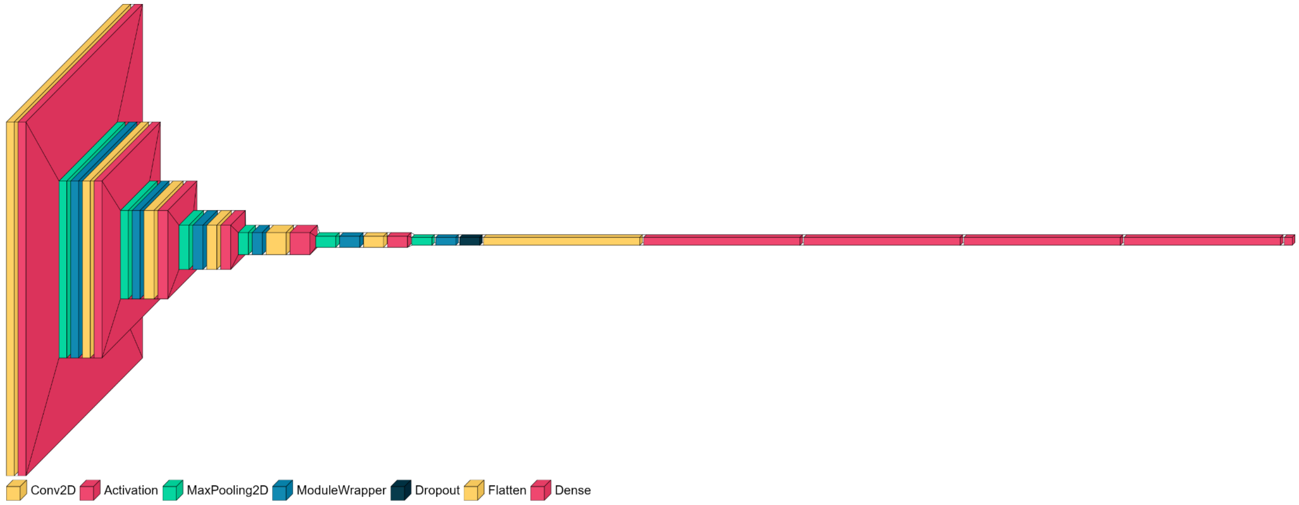
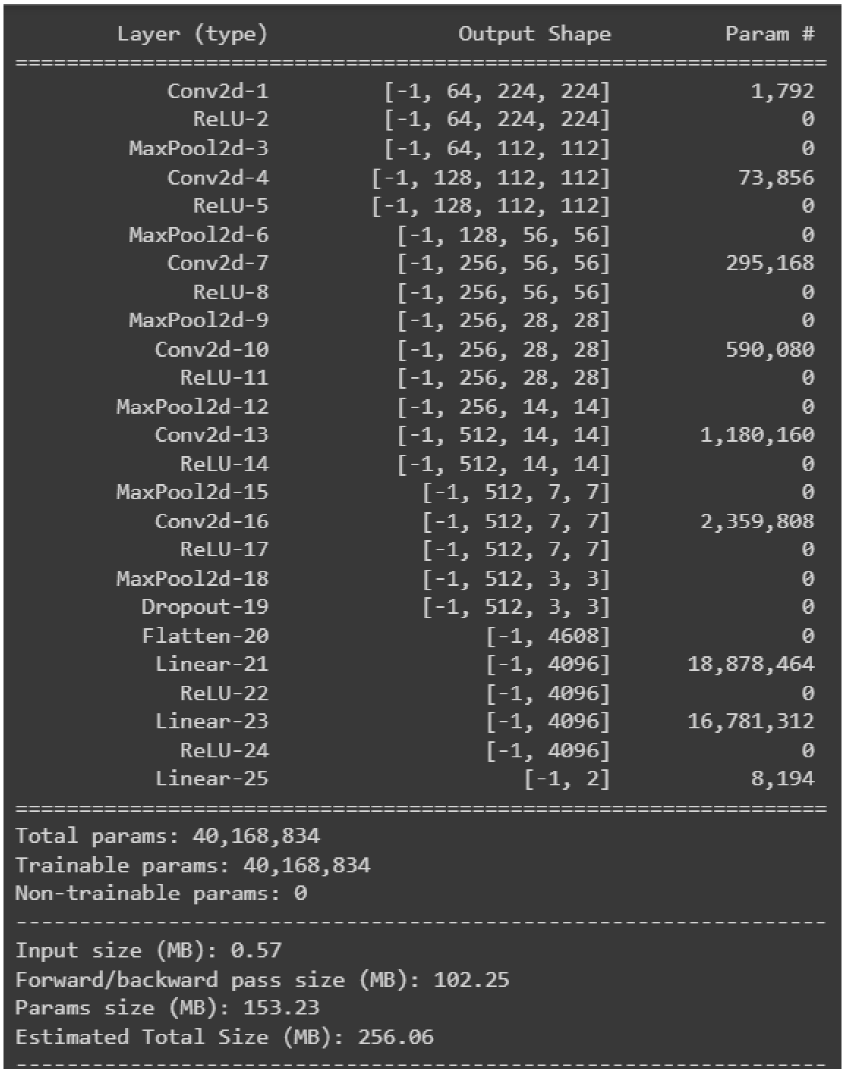


Figure 1: Proposed CNN Model Architecture



```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 64, 224, 224]           1,792
              ReLU-2        [-1, 64, 224, 224]               0
         MaxPool2d-3        [-1, 64, 112, 112]               0
            Conv2d-4       [-1, 128, 112, 112]          73,856
              ReLU-5       [-1, 128, 112, 112]               0
         MaxPool2d-6         [-1, 128, 56, 56]               0
            Conv2d-7         [-1, 256, 56, 56]         295,168
              ReLU-8         [-1, 256, 56, 56]               0
         MaxPool2d-9         [-1, 256, 28, 28]               0
           Conv2d-10         [-1, 256, 28, 28]         590,080
             ReLU-11         [-1, 256, 28, 28]               0
        MaxPool2d-12         [-1, 256, 14, 14]               0
           Conv2d-13         [-1, 512, 14, 14]       1,180,160
             ReLU-14         [-1, 512, 14, 14]               0
        MaxPool2d-15           [-1, 512, 7, 7]               0
           Conv2d-16           [-1, 512, 7, 7]       2,359,808
             ReLU-17           [-1, 512, 7, 7]               0
        MaxPool2d-18           [-1, 512, 3, 3]               0
          Dropout-19           [-1, 512, 3, 3]               0
          Flatten-20                [-1, 4608]               0
           Linear-21                [-1, 4096]      18,878,464
             ReLU-22                [-1, 4096]               0
           Linear-23                [-1, 4096]      16,781,312
             ReLU-24                [-1, 4096]               0
           Linear-25                   [-1, 2]           8,194
================================================================
Total params: 40,168,834
Trainable params: 40,168,834
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 102.25
Params size (MB): 153.23
Estimated Total Size (MB): 256.06
----------------------------------------------------------------
```

Figure 2: The Custom CNN Model Architecture

The convolutional section initiates with the first layer, Conv2d-1, processing the input images to generate 64 feature maps with a spatial dimension of 224x224. The subsequent application of the ReLU activation function introduces non-linearity, and max-pooling reduces the spatial dimensions by half through MaxPool2d-3. Convolutional Layer 2 follows a similar pattern, doubling the number of feature maps to 128 and maintaining spatial dimensions. Convolutional Layers 3 to 5 continue this progression, successively increasing feature map depth while decreasing spatial dimensions, effectively extracting hierarchical features from the input data. Convolutional Layer 6 represents a critical point in the architecture, further increasing the depth of feature maps to 512 with a spatial dimension of 7x7. This layer serves as a key feature extractor before transitioning to the fully connected layers.

The transition from the convolutional layers to the fully connected layers involves a Flatten layer, reshaping the 3D tensor into a 1D tensor suitable for processing by densely connected layers. The flattened representation is then passed through three fully connected layers, each comprising 4096 neurons and ReLU activation functions. These fully connected layers serve as powerful classifiers, capturing high-level representations of the features extracted by the preceding convolutional layers. The final layer, Linear-25, is the output layer with two neurons, facilitating binary classification. The model's ability to capture complex representations is underscored by the substantial number of trainable parameters, totaling 40,168,834.

The use of dropout after the last convolutional layer is a prudent choice to mitigate overfitting. Dropout introduces a regularization technique by randomly dropping neurons during training, preventing the network from relying too heavily on specific neurons and improving its generalization to unseen data. This inclusion enhances the model's robustness and ensures that it does not memorize the training data but learns meaningful features.

Results:

The table below presents a comprehensive comparison of accuracy metrics, including Precision, F1 Score, and overall Accuracy, for traditional machine learning models (SVM, Gaussian NB, Decision Tree, KNN, Random Forest) and a custom Convolutional Neural Network (CNN) model. Starting with Support Vector Machine (SVM), the model exhibits relatively low precision, F1 Score, and accuracy, all hovering around 0.12. Similarly, Gaussian Naive Bayes (Gaussian NB) demonstrates low precision and F1 Score, with a slightly improved accuracy of 0.10. Decision Tree performs better, with precision and F1 Score at 0.23, and accuracy at 0.22, indicating a modest level of correctness in its predictions. Moving to K-Nearest Neighbors (KNN), the model shows improvement with higher precision (0.32) but lower F1 Score (0.29) and accuracy (0.27). Random Forest, while offering a reasonable precision of 0.29, falls short in terms of F1 Score (0.27) and accuracy (0.26). In contrast, the custom CNN model outperforms the traditional models across all metrics. The CNN model achieves a precision of 0.35, an F1 Score of 0.33, and an accuracy of 0.35. These results suggest that the CNN model has a higher level of correctness in positive predictions, better balance between precision and recall, and an overall improved accuracy compared to the traditional machine learning models evaluated in this study.

Table I

ACCURACY COMPARISON OF TRADITIONAL MODELS WITH OUR CUSTOM CNN MODEL

	Precision	F1 Score	Accuracy
SVM	0.12	0.12	0.12
Gaussian NB	0.07	0.07	0.10
Decision Tree	0.23	0.23	0.22
KNN	0.32	0.29	0.27
Random Forest	0.29	0.27	0.26
Custom CNN Model	**0.35**	**0.33**	**0.35**

In the table below, AlexNet, exhibits a precision of 0.31, an F1 Score of 0.29, and an accuracy of 0.33. DenseNet performs well with a higher precision of 0.43, but its F1 Score (0.34) and accuracy (0.33) are slightly lower. ResNet50 shows balanced performance with a precision of 0.37, an F1 Score of 0.33, and an accuracy of 0.33. VGG16, while having a lower precision of 0.25, demonstrates an improved F1 Score (0.24) and accuracy (0.30) compared to some other models. In comparison, the custom CNN model achieves a precision of 0.35, an F1 Score of 0.33, and an accuracy of 0.35. These results suggest that the custom CNN model performs competitively with, or surpasses, the untrained deep learning models in terms of precision, F1 Score, and overall accuracy. It's noteworthy that the custom CNN model exhibits a well-balanced performance across the three metrics, indicating its effectiveness in making accurate predictions while considering both false positives and false negatives.

Table II
ACCURACY COMPARISON OF UNTRAINED DEEP LEARNING MODELS WITH OUR CUSTOM CNN MODEL

	Precision	F1 Score	Accuracy
AlexNet	0.31	0.29	0.33
DenseNet	**0.43**	**0.34**	0.33
ResNet50	0.37	0.33	0.33
VGG16	0.25	0.24	0.30
Custom CNN Model	0.35	0.33	**0.35**

In the provided table below, AlexNet, demonstrates a precision of 0.52, an F1 Score of 0.50, and an accuracy of 0.49. DenseNet exhibits notably high performance with a precision of 0.85, an F1 Score of 0.84, and an accuracy of 0.84. ResNet50 showcases balanced accuracy metrics with a precision of 0.79, an F1 Score of 0.79, and an accuracy of 0.79. VGG16, similar to DenseNet, achieves high precision (0.85), a strong F1 Score (0.84), and an accuracy of 0.84.

Table III
ACCURACY OF PRE TRAINED DEEP LEARNING MODELS

	Precision	F1 Score	Accuracy
AlexNet	0.52	0.50	0.49
DenseNet	**0.85**	**0.84**	**0.84**
ResNet50	0.79	0.79	0.79
VGG16	**0.85**	**0.84**	**0.84**

These results underscore the effectiveness of pre-trained deep learning models, particularly DenseNet and VGG16, in achieving accurate predictions across the specified metrics. These models, having been pre-trained on large and diverse datasets (ImageNet), demonstrate a high degree of transferability to the task at hand, showcasing the advantages of leveraging pre-trained architectures for image classification.

Result Evaluation:

In this research a total of five participants were used and all of them performed the seven actions. We chose these particular seven actions because these are the most frequent actions performed by humans. Traditional models like Support Vector Machine, Gaussian Naive Bayes, Decision Tree, K-Nearest Neighbors, and Random Forest revealed limitations in capturing intricate image patterns, with Decision Tree and KNN showing moderate accuracy. However, their overall effectiveness was surpassed by the custom CNN model, which demonstrated superior precision (0.35), F1 Score (0.33), and accuracy (0.35). Untrained deep learning models, encompassing AlexNet, DenseNet, ResNet50, and VGG16, exhibited varying degrees of success, with DenseNet and VGG16 leading in accuracy. Nevertheless, these models fell short of the custom CNN model's performance, emphasizing the importance of customization and fine-tuning for optimal image classification. Pre-trained deep learning models, specifically AlexNet, DenseNet, ResNet50, and VGG16, displayed strong overall performance with DenseNet and VGG16 achieving notable precision (0.85), F1 Score (0.84), and accuracy (0.84). Despite this, the custom CNN model demonstrated competitive results, underscoring the significance of model customization even against well-performing pre-trained models.

Based on all the findings, the result evaluation indicates that the custom CNN model stands out as a robust and versatile solution for binary image classification. Its competitive performance, when compared to traditional and untrained deep learning models, underscores the significance of customization and domain-specific adaptation in achieving optimal outcomes. These results also tell us that given enough computational resources and training time, if our custom CNN model is to be trained on big datasets like ImageNet, it will also perform the image classification task similar to these State of the art models.

Conclusion

In this research the performance of all the traditional methods, deep learning methods and CNN was tested on the task of human action recognition in still images. A custom dataset was created by the author for this research. The dataset consisted of images of 7 human activities, which are yawning, phoning, sitting, standing, walking, running and hand waving. 1974 images were taken in total and all of the images were augmented. A total of five participants were employed in this research, and each of them completed all seven actions. These seven activities were chosen because they are the most common behaviors performed by humans. Because of underfitting, conventional process models produced poor performance. The same issue arose with deep learning techniques. Due to underfitting, they also had very low accuracy. Finally, in order to get a better result for the custom dataset, a custom CNN model was introduced, and the model produced promising results. In conclusion, the developed CNN model architecture stands as a testament to the careful consideration given to balancing depth, non-linearity, and regularization. The sequential arrangement of layers facilitates the extraction of hierarchical features, and the significant number of trainable parameters empowers the model to learn intricate representations.

Reference:

1. M. Daily, S. Medasani, R. Behringer and M. Trivedi, "Self-Driving Cars," in Computer, vol. 50, no. 12, pp. 18-23, December 2017, doi: 10.1109/MC.2017.4451204.
2. R. Kulkarni, S. Dhavalikar and S. Bangar, "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697819.
3. Leo, Marco; Carcagnì, Pierluigi; Mazzeo, Pier L.; Spagnolo, Paolo; Cazzato, Dario; Distante, Cosimo. 2020. "Analysis of Facial Information for Healthcare Applications: A Survey on Computer Vision-Based Approaches" Information 11, no. 3: 128. https://doi.org/10.3390/info11030128
4. David Ahmedt-Aristizabal, Clinton Fookes, Kien Nguyen, Simon Denman, Sridha Sridharan, Sasha Dionisio, Deep facial analysis: A new phase I epilepsy evaluation using computer vision, Epilepsy & Behavior, Volume 82, 2018, Pages 17-24, ISSN 1525-5050, https://doi.org/10.1016/j.yebeh.2018.02.010
5. Ng C.B., Tay Y.H., Goi BM. (2012) Recognizing Human Gender in Computer Vision: A Survey. In: Anthony P., Ishizuka M., Lukose D. (eds) PRICAI 2012: Trends in Artificial Intelligence. PRICAI 2012. Lecture Notes in Computer Science, vol 7458. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32695-0_31
6. Mäkinen, E., Raisamo, R.: An experimental comparison of gender classification methods. Pattern Recognition Letters 29(10), 1544–1556 (2008)
7. Lian, H.C., Lu, B.L.: Multi-view gender classification using local binary patterns and support vector machines. In: Advances in Neural Networks-ISNN 2006, pp. 202–209 (2006)
8. Benabdelkader, C., Griffin, P.: A Local Region-based Approach to Gender Classification From Face Images. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, CVPR Workshops, p. 52 (2005)
9. C. Chen, R. Jafari, and N. Kehtarnavaz, "Action recognition from depth sequences using depth motion maps-based local binary patterns," in Proc. IEEE Win. Conf. Appl. Comput. Vis., 2015, pp. 1092–1099.
10. J. Yu and J. Sun, "Multiactivity 3-D human pose tracking in incorporated motion model with transition bridges," IEEE Trans. Syst., Man, Cybern., Syst., to be published.
11. W. Chi, J. Wang, and M. Q.-H. Meng, "A gait recognition method for human following in service robots," IEEE Trans. Syst., Man, Cybern., Syst., to be published.
12. G. Liang, X. Lan, J. Wang, J. Wang, and N. Zheng, "A limb-based graphical model for human pose estimation," IEEE Trans. Syst., Man, Cybern., Syst., vol. 48, no. 7, pp. 1080–1092, Jul. 2018.
13. Y. Guo, D. Tao, W. Liu, and J. Cheng, "Multiview Cauchy estimator feature embedding for depth and inertial sensor-based human action recognition," IEEE Trans. Syst., Man, Cybern., Syst., vol. 47, no. 4, pp. 617–627, Apr. 2017.
14. S. Zhang, C. Gao, F. Chen, S. Luo, and N. Sang, "Group sparse-based mid-level representation for action recognition," IEEE Trans. Syst., Man, Cybern., Syst., vol. 47, no. 4, pp. 660–672, Apr. 2017.
15. H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense

trajectories,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2011, pp. 3169–3176.
16. H. Wang and C. Schmid, “Action recognition with improved trajectories,” in Proc. IEEE Int. Conf. Comput. Vis., 2013, pp. 3551–3558.
17. X. Peng, C. Zou, Y. Qiao, and Q. Peng, “Action recognition with stacked Fisher vectors,” in Proc. Eur. Conf. Comput. Vis., 2014, pp. 581–595.
18. X. Peng, L. Wang, X. Wang, and Y. Qiao, “Bag of visual words and fusion methods for action recognition,” Comput. Vis. Image Understand., vol. 150, pp. 109–125, Sep. 2016.
19. H. Xu, J. Liu, H. Hu, and Y. Zhang, “Wearable sensor-based human activity recognition method with multi-features extracted from Hilbert–Huang transform,” Sensors, vol. 16, no. 12, pp. 1–26, 2016.
20. H. Ponce, M. D. L. Martínez-Villaseñor, and L. Miralles-Pechuán, “A novel wearable sensor-based human activity recognition approach using artificial hydrocarbon networks,” Sensors, vol. 16, no. 7, pp. 1–28, 2016.
21. J. Qi, P. Yang, M. Hanneghan, S. Tang, and B. Zhou, “A hybrid hierarchical framework for gym physical activity recognition and measurement using wearable sensors,” IEEE Internet Things J., to be published.
22. J. Qi, P. Yang, M. Hanneghan, and S. Tang, “Multiple density maps information fusion for effectively assessing intensity pattern of lifelogging physical activity,” Neurocomputing, vol. 220, pp. 199–209, Jan. 2017.
23. P. Yang et al., “Lifelogging data validation model for Internet of Things enabled personalized healthcare,” IEEE Trans. Syst., Man, Cybern., Syst., vol. 48, no. 1, pp. 50–64, Jan. 2018.
24. J. Sriwan and W. Suntiamorntut, “Human activity monitoring system based on WSNs,” in Proc. Int. Joint Conf. Comput. Sci. Softw. Eng., 2015, pp. 247–250.
25. G. Chetty and M. White, “Body sensor networks for human activity recognition,” in Proc. Int. Conf. Signal Process. Integr. Netw., 2016, pp. 660–665.
26. Zhang, Hong-Bo & Zhang, Yi-Xiang & Zhong, Bineng & Lei, Qing & Yang, Lijie & Du, Ji-Xiang & Chen, Duan-Sheng. (2019). A Comprehensive Survey of Vision-Based Human Action Recognition Methods. Sensors. 19. 1005. 10.3390/s19051005.
27. A. Eweiwi, S. Cheema, C. Thurau and C. Bauckhage, "Temporal key poses for human action recognition," 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, 2011, pp. 1310-1317, doi: 10.1109/ICCVW.2011.6130403.
28. Alexandros Andre Chaaraoui, Pau Climent-Pérez, Francisco Flórez-Revuelta, Silhouette-based human action recognition using sequences of key poses, Pattern Recognition Letters, Volume 34, Issue 15, 2013, Pages 1799-1807, ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2013.01.021.
29. C. Thurau and V. Hlavac, "Pose primitive based human action recognition in videos or still images," 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, 2008, pp. 1-8, doi: 10.1109/CVPR.2008.4587721.
30. L. Yeffet and L. Wolf, "Local Trinary Patterns for human action recognition," 2009 IEEE 12th International Conference on Computer Vision, Kyoto, 2009, pp. 492-497, doi: 10.1109/ICCV.2009.5459201.
31. UCF101 - Action Recognition Data Set (2013) URL

https://www.crcv.ucf.edu/data/UCF101.php

32. ILSVRC Dataset (2017) URL https://www.image-net.org/challenges/LSVRC/

33. HACS Dataset (2019) URL http://hacs.csail.mit.edu