Deep Learning 3D CNN Applied to Low-Light Human Action Recognition Videos

Jeremy Styborski (Matric. G2202711K)
School of Computer Science and Engineering
Nanyang Technological University
Singapore, Singapore
styb0001@e.ntu.edu.sg

***Abstract*—In this paper, we analyze the R3D-18 3D ResNet applied to human action recognition videos in low light. Gamma correction is the primary method used for correcting low-light behavior and varying degrees of gamma correction strength are examined. After determining the optimal gamma correction strength, we subsequently analyze the effect of brightness and contrast augmentation, both before and after gamma correction. Finally, we analyze the performance across four types of frame sampling, including two novel sampling method proposed herein. Given the results of these surveys, we synthesize an optimal R3D-18 model and evaluate its test-set performance, achieving a maximum test accuracy of 63.11%.**

***Index Terms*—deep learning, 3D CNN, low light, gamma correction, frame sampling**

I. Introduction

The field of human action recognition (HAR) characterizes how we move through and interact with the world around us [18], [14], [21]. HAR is primarily concerned with classifying human motion such that each action, such as 'running', 'throwing', and 'grabbing' is identified as distinct from other actions. Due to its focus on human movement, research on HAR is highly relevant to industries such as healthcare [4] and virtual reality [17].

The actions associated with HAR are typically outwardly focused and emotive, often involving the entire human body. For example, actions such as 'breathing' or 'thinking' are not typically considered, in favor of more active verbs like 'running' or 'picking'. On one hand, HAR is closely related to other tasks that concern the entire body or scene, such as pose estimation or scene graph generation. On the other hand, HAR begins to mix with disciplines, such as sentiment estimation or sign language recognition, once facial features and fine motor control activities become relevant.

HAR is primarily linked to the field of machine vision, typically in the visual video medium. In this way, the HAR task highlights problems with existing computer vision methods that focus on static images and inspires better methods for processing spatiotemporal data. However, HAR is not limited to the visual video medium, with recent methods attempting HAR with other modalities, such as audio [16], infrared [19], and WiFi [6].

As research on video processing advances, additional challenges are added to the prototypical HAR task. Additional challenges might include vigorous camera movement, object localization, and future frame prediction. One more challenge is HAR in environments with adverse lighting conditions, such as low-light or with the sun in frame. Deep learning strategies can prove robust in these situations, but image processing and image correction has a much longer history. As such, image correction solutions from decades ago, such as gamma correction [1] and filtering [5], are still useful as a first step in processing adverse lighting images.

A. EE6222 Project Task and Dataset

HAR is the focus of the project given to the EE6222 Machine Vision class. Specifically, the task involves HAR on 10 distinct action classes from videos shot on a regular visible spectrum camera in low-light conditions. Video samples typically include only one human, and there is only one action performed per clip. The goal is to classify the action in each video despite the low-light conditions.

Data is presented as video clips shot in 240x320 resolution and three-channel RGB format. Each clip is recorded with 30fps for roughly 1-4 seconds (corresponding to roughly 30-120 frames). There is no audio included in the video files. The training set includes 750 samples (75 clips per class) and the validation set includes 320 samples (32 clips per class).

II. Related Work

As mentioned in the Introduction, research on image processing has been ongoing for decades. Many techniques were developed to address common problems such as adverse lighting conditions, sharpening/blur, and noise. Many of these techniques are covered in the EE6222 Machine Vision lecture notes [11]. Filtering [5] uses convolutional kernels (i.e., filters) to perform operations such as blurring, sharpening, and noise removal. Gamma correction [1] is a pixelwise operation that shifts the brightness of each pixel in an image by applying an exponential term. For the purposes of this work, both gamma correction and filtering were considered, though ultimately filtering techniques were not used.

Although deep learning currently dominates research in HAR, multiple techniques were developed previously for visualizing and processing action in videos were developed. Motion Energy Image (MEI) [2] outputs a binary image classifying all pixels in which a non-negligible intensity change occurred across a video. Motion History Image [2] are similar

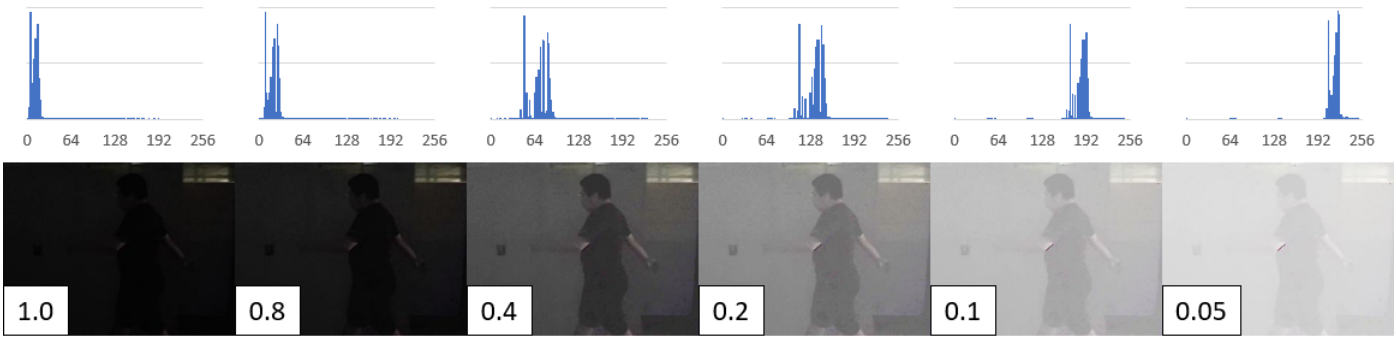

Fig. 1. Gamma correction applied to a sample frame from the low-light dataset. Gamma correction values are noted on the images. A pixel intensity histogram (from 0-255) is included above each image.

to MEI but incorporate a decay factor for pixel intensity changes across video frames such that the final image is not binary; instead, the output image varies in intensity, with intensity corresponding to recency. Optical Flow [10] tracks pixel movement within an image by searching for local pixels of similar intensity in subsequent video frames. Optical Flow outputs a 2D vector array corresponding to the direction and magnitude of each pixels movement between frames. Unfortunately, methods such as MEI, MHI, and Optical Flow are susceptible to pixel intensity changes due to non-movement factors, such as camera viewpoint and lighting changes.

HAR is a major focus for deep learning video processing with most modern methods using some form of convolutional frame processing. Many of the techniques covered here are addressed in the EE6222 lecture notes [25]. Initial methods simply used existing CNNs to processing video frames individually after which image embeddings or action predictions would be combined and processed further [12]. DB-LSTM [8] was developed to combine CNNs for individual video frames with LSTM networks to carry information across frames. The two-stream 2D CNN [20] was developed to process individual video frames in parallel to an Optical Flow of the video and then fuse class predictions. Beyond this, networks using CNNs with 3D convolutions [22] became popular, wherein a 3D kernel would slide along height, width, and depth dimensions. Based on the successful ResNet framework [9], a 3D convolution version of ResNet was developed [7] and gave state-of-the-art performance on video classification. Subsequent developments for 3D CNNs often focused on reducing the computational load of processing videos with large networks. I3D [3] reduced parameter count by constructing 3D kernels from 2D kernels. Based on the success of ResNeXt and grouped convolutions [15], grouped convolutions were developed for 3D CNNs [23] in order to reduce cross-channel parameter count.

III. Methodology

The deep learning approach to HAR in low-light conditions detailed in this paper utilized the 3D CNN ResNet model, R3D-18 [24], with weights pretrained on the Kinetics400 video dataset [13]. The final fully-connected (FC) layer of the R3D-18 model was replaced with a FC layer appropriately sized for the 10 classes of the project dataset. The model used a final softmax operation for class prediction and then evaluated loss using a multi-class cross-entropy function. Models were trained using a SGD optimizer with a batch size of 8, momentum coefficient of 0.9, and weight decay of 0.005. The SGD trainer used cosine annealing of the learning rate from a maximum of 1 to a minimum of 0.001. Each model was trained for 100 epochs.

In order to correct low-light image intensity, each video was processed with gamma correction. After gamma correction, training data augmentations like horizontal flipping and color jittering were applied. Color jittering scale factor limits were set to +/- 50%. These training data augmentations were not applied to the validation dataset.

IV. Experiments

In order to explore the effect of various hyperparameters on the model, a total of five experiments were conducted. Each experiment was conducted by varying the parameter of interest with all other hyperparameters held constant. The experimental hyperparameters are listed below.

1) Gamma Correction Exponent
2) Frame Sampling Method
3) Color Jitter Before/After Gamma Correction
4) Number of Frames Sampled
5) Number of Frozen Layers

Gamma correction values of 0.1 and 0.2 were used throughout most experiments. The color jittering operation was typically conducted after gamma correction. Unless otherwise stated, 20 frames were sampled from each video clip using a Sequential Full sampling method (explained in detail in the Experiments section). Additionally, unless otherwise stated, all model weights were fully trainable.

A. Gamma Correction

Gamma correction rescales pixel intensity value. A gamma correction exponent value less than one will increase pixel intensity, corresponding to a brighter output image. A gamma

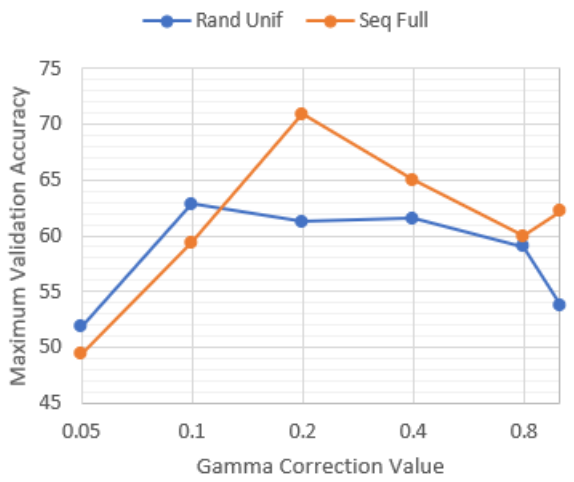

Fig. 2. Maximum validation accuracy across gamma correction values 0.05 to 1.0. Two sampling methods Random Uniform and Sequential Full.

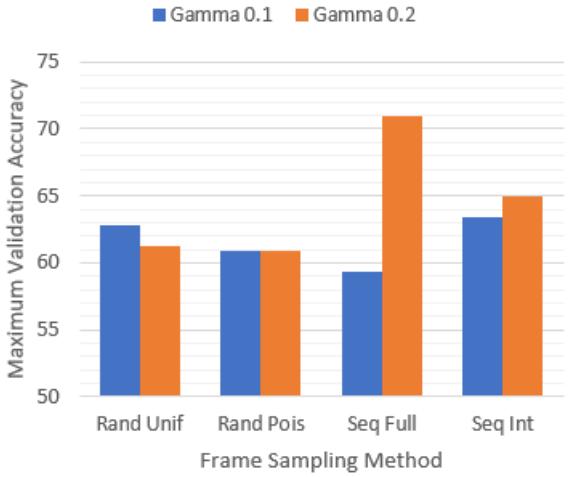

Fig. 3. Maximum validation accuracy across four sampling methods for gamma correction values of 0.1 and 0.2.

correction exponent value greater than one will decrease pixel intensity. Figure 1 shows the outputs of a dark image with gamma correction applied for various values of gamma. A smaller gamma value causes a larger brightness shift.

In this experiment, multiple gamma correction values between 0.05 and 1.0 were examined. To better understand the effect of gamma correction under multiple sampling methods, we analyzed the gamma correction under Random Uniform (RU) sampling and Sequential Full (SF) sampling, both of which are discussed further in the Frame Sampling Method subsection.

Figure 2 shows the results of the survey on gamma correction values. Looking at the difference between the two sampling schemes, we find that SF sampling generally allows greater validation accuracy. SF sampling is consistently better at higher gamma correction values, though the greatest increase occurs at $\gamma = 0.2$. Focusing on the Sequential Full sampling curve alone, we see a exponential drop (linear on the log scale) in validation accuracy when increasing or decreasing gamma away from 0.2. This suggests that the low-light image features are more naturally extracted once gamma corrected by $\gamma = 0.2$. The curve for Random Uniform sampling is relatively flat, bumping up slightly at $\gamma = 0.1$. The flatness of the Random Uniform curve across multiple gamma values suggests that random sampling may generate a noisy and inconsistent input. This noisy input could make the model more robust to other inputs, acting as a sort of regularizer, but it also drags down overall performance. Regardless, random sampling does not seem as useful if the videos are too dark or too bright, as shown by lower performance at high and low gamma values.

Due to the relatively high performance of $\gamma = 0.1$ and $\gamma = 0.2$ on Random Uniform sampling and Sequential Full sampling respectively, both gamma values were selected for further analysis.

B. *Frame Sampling Method*

The lecture notes from EE6222 Machine Vision discuss two main methods of frame sampling, RU sampling and Sequential Interval (SI) sampling [25]. In RU sampling, n frames are uniformly sampled (without replacement) from each clip as input to the model. In SI sampling, n frames are extracted at a constant frame interval from each clip (e.g., extract every 10th frame when extracting 10 frames from a 90 frame clip).

In addition to the sampling methods mentioned above, two alternate sampling methods were devised and implemented: Random Poisson (RP) sampling and SF sampling.

RP sampling is a variant of RU sampling based on the Poisson distribution and the idea of an 'average rate'. RP sampling may be thought of as a hybrid of random uniform sampling and sequential interval sampling. In RP sampling, the probability of a frame being extracted depends on the distance to the last frame sampled. It is unlikely that two adjacent frames will be extracted together, but it is also unlikely that zero frames will be extracted over a long interval. The extracted frames resemble a sequential interval sampling method with some random noise applied to the interval. RP sampling is implemented in Python with numpy.random.Poisson() and then heuristic methods are used to ensure that the desired number of frames are sampled.

SF sampling is a minor variant of SI sampling. While SI sampling guarantees a constant interval, SF sampling will vary the discrete interval to guarantee that the start and end frames of a clip are included. For example, if 10 frames are to be extracted from a 75 frame clip, then the sampling interval would be 75 / (10 - 1) = 8.33 frames. SI sampling would simply use a constant interval of 8 frames. SF sampling maintains an interval of 8.33 frames and then chooses the nearest whole number. Therefore SF sampling guarantees that the entire clip range is included.

Figure 3 shows the maximum validation accuracy for the four sampling methods at $\gamma = 0.1$ and $\gamma = 0.2$. Broadly

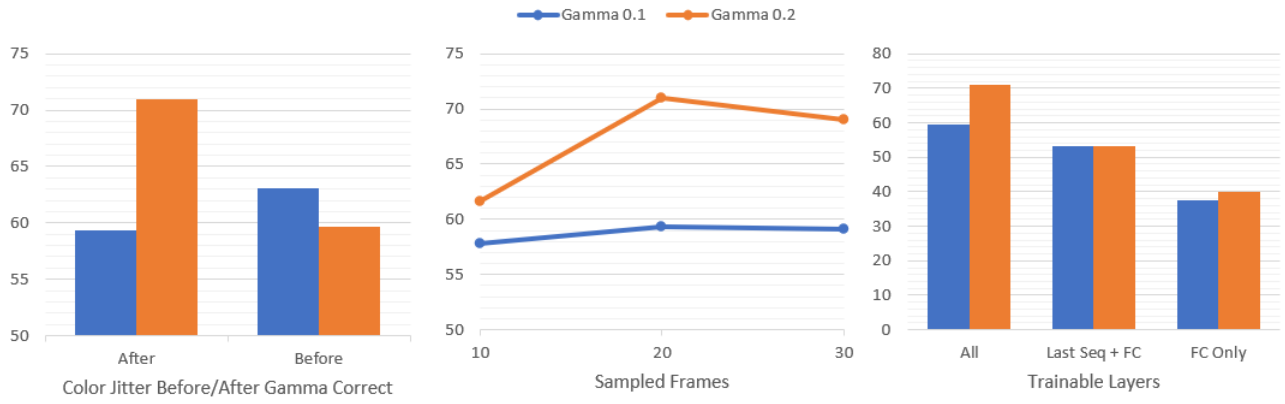

Fig. 4. Maximum validation accuracy for $\gamma = 0.1$ and $\gamma = 0.2$ with SF sampling. Left: Effect of ColorJitter before or after gamma correction. Middle: Effect of number of sampled frames. Right: Effect of number of trainable layers

speaking, sequential sampling methods gives the best performance, with the highest scores occurring for SI and SF sampling at $\gamma = 0.1$ and $\gamma = 0.2$ respectively. Notably, SF sampling gives the best validation accuracy for $\gamma = 0.2$, over 5% higher than the next best method. Oddly however, SF sampling gives the lowest accuracy for $\gamma = 0.1$. This suggests that model performance is highly sensitive to gamma value and that sampling method may further enhance or diminish performance. Consistent with the results of the gamma survey experiment, random sampling methods give moderate and consistent results across gamma values.

Due to the outstanding performance of SF sampling for $\gamma = 0.2$, SF sampling was chosen for use in subsequent experiments.

C. ColorJitter Before/After Gamma Correction

ColorJitter is a native PyTorch augmentation method that randomly modifies the brightness and contrast of an image with a scalar factor drawn uniformly from the range between user-provided minimum and maximum values. Analyzing the code for ColorJitter, it can be seen that the brightness and contrast modifications are linear operations. As mentioned in the Methodology section, the brightness and contrast scalar factors were limited between +/- 50%. As gamma correction is an exponential operation, the exact output of an augmented image depends on the order of the gamma correction and ColorJitter operations. In this experiment, we analyze the effect of swapping the order of these two image operations.

The leftmost chart in figure 4 shows the effect of ColorJitter before or after gamma correction for $\gamma = 0.1$ and $\gamma = 0.2$ and SF sampling. Similar to previous studies, we see counteracting behaviors between $\gamma = 0.1$ and $\gamma = 0.2$. Accuracy is highest for $\gamma = 0.2$ with ColorJitter after gamma correction. When restricting our view to $\gamma = 0.1$, then ColorJitter is best used before gamma correction. The explanation for this phenomenon is lies in the brightness of the gamma corrected images. A gamma correction of $\gamma = 0.1$ results in a much brighter image compared to $\gamma = 0.2$. Since the ColorJitter brightness and contrast changes scale with pixel intensity, a brighter input image will have a larger variation in ColorJitter output, which seems detrimental to training. Furthermore, a ColorJitter operation that increases brightness for an already-bright image may wash out some information contained in the image, setting multiple pixel values to the maximum value. It appears that ColorJitter is more useful on images with more natural intensity values, such as those after a gamma correction of $\gamma = 0.2$.

D. Number of Frames Sampled

The amount of compute time required to train each model scales linearly with the number of frames sampled from each video due to the increased number of forward propagation and backpropagation calculations. However, the additional information captured from each video for higher sampling rates may be beneficial for model training. Therefore, we explored the marginal benefit due to sampling additional frames from each video, at rates of 10, 20, and 30 frames per video clip.

The middle chart in figure 4 shows the effect on number of frames sampled for $\gamma = 0.1$ and $\gamma = 0.2$ and SF sampling. In all cases, $\gamma = 0.2$ gave superior performance. Interestingly, for both cases, the maximum accuracy was achieved for 20 frames sampled, followed by 30, and finally 10. Counter to intuition, adding more frames to the training and validation input does not always seem to result in better performance. It seems that 20 frames is an optimal value that contains enough video clip information for correct classification without adding too many frames that act as noise or redundancy. This is also good news for model training, as it encourages quicker training times.

E. Number of Frozen Layers

Researchers will often restrict the number of trainable layers in a pretrained model in order to preserve early model layers that are typically more suited for base-level image feature extraction. Occasionally, training is restricted as far as only training the final classification layer. However, these layer-freezing techniques are usually employed in cases where

Iter	Max Val Acc	Test Acc	Difference
0	70.94	63.11	7.83
1	68.44	58.00	10.44
2	64.06	58.22	5.84
3	71.56	62.22	9.34

Fig. 5. Validation and test accuracies for four iterations of the final model.

the finetuning dataset contains base-level features (e.g., color gradients) that are largely similar. This is not the case here. The R3D-18 model was imported with weights pretrained on the Kinetics400 dataset, which consists of videos shot in normal lighting conditions. In adapting this pretrained model to the low-light project dataset, the content of the videos is largely similar, but the pixel intensity is much lower. This implies that the pretrained base-level feature extraction layers may not function for features relevant in low-light conditions. To test this hypothesis, we conducted an experiment to vary the number of frozen layers in the model.

Three levels of layer-freezing were tested. 'All' implies that all weights (33M parameters) were trainable. 'Last Seq + FC' trained the final sequential block and the FC classifier layer (25M parameters). Finally, 'FC Only' trained only the final FC classifier layer (5130 parameters). The rightmost chart in figure 4 shows the results of the frozen layers experiment with three degrees of freezing tested. Across gamma values, the model's validation accuracy decreases as the number of trainable layers decreases. The best performance occurs when all layers are trainable. This agrees with our hypothesis.

F. Test Set Performance

Given the results of the above experiments, a final model was created for test set evaluation. The final model used $\gamma = 0.2$, SF sampling, ColorJitter after gamma correction, 20 sampled frames, and trained all layers. A total of four models were trained using this hyperparameter configuration. Figure 5 displays the validation and test performance of all four models. The variation in validation accuracy is due to randomness in data augmentation and batch sampling. For all models, test accuracy is consistently below validation accuracy. Assuming no significant drift between validation and testing datasets, then the difference is likely due hyperparameter overfitting on the validation dataset. Although the validation data was never used directly during model training, repeated tests on validation data likely biased the hyperparameter values. Still, a final average test accuracy of roughly 61% is not terrible.

V. Conclusion

In this paper, we reviewed the challenges associated with video action recognition in low-light conditions. Methods for correcting low-light images and videos were discussed and implemented. We introduced a ResNet3D model and evaluated its performance across a wide variety of hyperparameter settings, including two novel frame sampling methods. Collecting the results from these hyperparameter surveys, we synthesized four variants of the optimal model and evaluated their test set performance, achieving a maximum test accuracy of 63.11%.

References

[1] Appendix: Gamma tutorial. https://www.w3.org/TR/PNG-GammaAppendix.html. Accessed: 2022-10-14.

[2] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:257–267, 2001.

[3] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[4] M. Chan, E. Campo, and D. Esteve. Assessment of activity of elderly people using a home monitoring system. *International Journal of Rehabilitation Research*, 28:69–76, 2005.

[5] R. Chandel and G. Gupta. Image filtering algorithms and techniques: A review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3:198–202, 2013.

[6] Y. Fang, F. Xiao, B. Sheng, L. Sha, and L. Sun. Cross-scene passive human activity recognition using commodity wifi. *Frontiers of Computer Science*, 16, 2022.

[7] K. Hara, H. Kataoka, and Y. Satoh. Learning spatio-temporal features with 3d residual networks for action recognition, 2017.

[8] J.-Y. He, X. Wu, Z.-Q. Cheng, Z. Yuan, and Y.-G. Jiang. Db-lstm: Densely-connected bi-directional lstm for human action recognition. *Neurocomputing*, 444, 2020.

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

[10] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[11] X. Jiang. Image correction lecture notes in ee6222 machine vision, September 2022.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[13] 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, 2017.

[14] Y. Kong and Y. Fu. Human action recognition and prediction: A survey, 2018.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84–90, 2017.

[16] D. Liang and E. Thomaz. Audio-based activities of daily living (adl) recognition with large-scale acoustic embeddings from online videos. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(1), mar 2019.

[17] A. I. Maqueda, C. R. del Blanco, F. Jaureguizar, and N. García. Human-action recognition module for the new generation of augmented reality applications. In *2015 International Symposium on Consumer Electronics (ISCE)*, pages 1–2, 2015.

[18] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28:976–990, 2010.

[19] A. Shah, R. Ghosh, and A. Akula. A spatio-temporal deep learning approach for human action recognition in infrared videos. *SPIE Proceedings*, 10751, 2018.

[20] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos, 2014.

[21] Z. Sun, Q. Ke, H. Rahmani, M. Bennamoun, G. Wang, and J. Liu. Human action recognition from various data modalities: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2022.

[22] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[23] D. Tran, H. Wang, L. Torresani, and M. Feiszli. Video classification with channel-separated convolutional networks, 2019.

[24] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition, 2017.

[25] Y. Xu. Video processing lecture notes in ee6222 machine vision, October 2022.