# Efficient Two-stream Action Recognition on FPGA

Jia-Ming Lin[1], Kuan-Ting Lai[2], Bin-Ray Wu[1] and Ming-Syan Chen[1]
[1]National Taiwan University
[2]National Taipei University of Technology
ktlai@ntut.edu.tw, {jmlin, brwu, mschen}@arbor.ee.ntu.edu.tw

## Abstract

*Action recognition is an important research field that has many applications in surveillance, video search, autonomous vehicles, etc. However, current state-of-the-art action classifiers are still not widely adopted in embedded applications yet. The major reason is that action recognition needs to process both spatial and temporal streaming data to precisely identify actions, which is compute-intensive and power hungry. To solve this issue, researchers start using FPGA to run action recognition models with minimum power. In this paper, we propose a new hardware architecture of action recognition on FPGA. Our model is based on the popular two-stream neural network. By optimizing the optical flow and convolution operations in the temporal domain, our method can achieve similar accuracy with one order of magnitude less operations than other C3D baseline models. We have implemented our model on Xilinx Ultrascale+ ZCU102 and released the source code.*

## 1. Introduction

Human action recognition has a wide range of applications including intelligent video surveillance [1, 29], video storage and retrieval [20, 26], intelligent human–machine interfaces [14, 15]. However, unlike other deep learning applications, such as image classification and object detection, which have been adopted in many industry applications. The development of action recognition is still behind and cannot meet the industry standard. This is because that human action is complex and diverse. Moreover, compared to images that have only spatial data, videos have both spatial and temporal data and require several order of magnitude more computation resources than images. Therefore, dedicated hardware is necessary to accelerate the inference time. One solution is Field-Programmable Gate Array (FPGA), which can achieve real-time performance while consuming less energy than GPU [13]. By leveraging the efficiency and flexibility of FPGA, researchers can fulfill real-time action recognition with minimum cost.

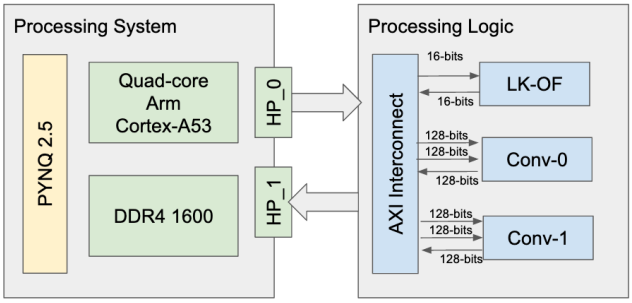


Figure 1. Block diagram of the proposed two-stream architecture for action recognition on FPGA.

There are several existing FPGA solutions for action recognition. Fan et al. [5] proposed to employ the 3D CNN [19] for video analysis. However, 3D CNN (C3D) requires a large number of operations. Later studies focused on improving the efficiency of 3D convolutions [4, 11].

In this paper, we proposed a new efficient FPGA model for action recognition, which is based on the two-stream model. Our method uses much less operations while maintaining similar accuracy to other C3D-based counterparts. The major advantages of our method are:

- Utilizing only 2D CNN operations, which have much less operations than 3D CNN.
- Processing both spatial and temporal streams parallelly. Temporal data can increase overall accuracy.
- Reducing the high latency of traditional optical flow by using Lucas-Kanade algorithm with slightly accuracy drop ($< 2\%$).

Fig. 1 illustrates our architecture. Our model has been implemented and tested on Xilinx ZCU102 platform. The source code has been published on GitHub[1]. Experiment results on UCF101 [18] show that our model has around 10 times less operations than other C3D-based FPGA systems while keeping similar accuracy.

[1]https://github.com/NetDBFPGA/ecv2021_demo

## 2. Related Work

In this section, we briefly review related works of action recognition and corresponding implementations on edge devices.

**Action recognition.** There are two main research directions in CNN-based action recognition: two-stream models [8, 22, 21, 6] and 3D CNNs [7, 19, 17]. The first approach processes RGB frames (spatial stream) and pre-computed optical flow (temporal stream) using two 2D CNNs independently, Late fusion is then applied to obtain final predictions. The second approach jointly learns spatiotemporal features from RGB frames by using 3D CNN. However, both approaches are still not widely deployed on edge devices. The two-stream based models require accurate optical flow as input, which is time-consuming and becomes the bottleneck. On the other hand, the 3D convolutions also have large computation cost and are too heavy to be deployed on edge devices.

In order to alleviate the inefficiency of optical flow computation, several recent works attempt to speed up the process by using specifically-designed CNN models. Hidden two-stream [27] and ActionFlowNet [12] use encoder-decoder network to compute optical flow. Those networks can be jointly trained with the action recognition model. But the encoder-decoder architecture also has expensive computation cost.

Another line of research proposed to utilize difference between consecutive frames (RGBDiff [21]) or consecutive feature maps [24]. The advantage of RGBDiff lies in that only subtractions are involved for computing temporal information, which can be computed on the fly. However it lacks the important information of motion directions and has lower accuracy. For more deep action recognition algorithms, a comprehensive review can be found in [28].

**Action recognition on edge devices.** Recently, researchers have designed new hardware architecture for action recognition on edge devices [5, 4, 25, 11]. Fan et al. [5, 4] implemented C3D model on FPGA, and designed specific data layout and memory hierarchy to optimize the throughput. Sun et al. [11] exploited the sparsity of 3D CNN, and reduced computational cost by skipping zero values in 3D convolution kernel. Nevertheless, the computation cost of 3D CNN is still too high. Thus, several works have started to investigate new methods to model temporal relationship between frames, such as the Temporal Shift Module (TSM) [9] proposed by Lin et al. TSM is an efficient model that needs zero multiply-accumulate (MAC) operation to process temporal features. However, experiments have showed that temporal stream can increase the accuracy. For example, an experiment made by TSM authors showed that $\text{TSM}_{\text{RGB+Flow}}$ is higher than single $\text{TSM}_{\text{RGB}}$ by $5.4\%$ on something-to-something dataset.

## 3. Efficient Two-Stream Action Recognition

The bottleneck of the two-stream model is the computation of optical flow [28]. In this section, we first present our design of optical flow accelerator, and introduce the 2D ConvNet accelerator and the concurrent CNN inferences architecture. Finally, we present the details of our prototype implementation on Xilinx Ultrascale+ ZCU102.

### 3.1. Temporal Stream Accelerator

Our accelerator architecture of temporal stream is based on Lucas-Kanade algorithm [10, 2], which is used to compute dense optical flow and denoted as LK-OF.

**LK-OF.** There are four stages in our accelerator, as shown in Fig. 2. The details of each stage are elaborated below:

- **Stage 1**: Smoothing the input images by a noise reduction filter of size $5 \times 5$, and complexity of processing one pixel is $5^2$.
- **Stage 2**: Computing spatial derivative with a filter of size $5 \times 5$. The temporal derivative is the difference between two input images. Complexity of this stage is $5^2 + 1$ to output one pixel.
- **Stage 3**: Lucas-Kanade algorithm assumes that motion vectors are all the same in a $N \times N$ window, so the problem can be translated to the least square problem. This stage computes the coefficients of the $2 \times 2$ matrix. We can reduce the complexity of $O(N^2)$ to $O(N)$ by caching intermediate results from previous window. The complexity of this stage is $4N$.
- **Stage 4**: Computing one $(u, v)$ pair by $2 \times 2$ matrix-vector multiplication and scaling to integers in $[0, 255]$ range. The complexity of this stage is $6$.

The above stages are implemented in a pipeline manner, so the storage complexity is negligible, and the total computational complexity is

$$(5^2 + 5^2 + 1 + 4N + 6) \times X_{img} \times Y_{img}. \quad (1)$$

In addition to LK-OF, we have studied another optical flow algorithm TV-L1 [16]. Table 1 compares the accuracy and latency of TV-L1 and LK-OF. We implemented the LK-OF algorithm on FPGA at 200MHz clock rate and the latency is 2ms with $6.24\%$ accuracy gain against single spatial stream on UCF101. We did not have enough time to implement TV-L1, so we run the model on Nvidia GTX1080. TV-L1 has 85ms latency on GPU. Therefore, our LK-OF hardware module can achieve at least 40 times speedup. Although TV-L1 has slightly higher accuracy gain ($8.5\%$) on UCF101, the slow inference speed makes it hard to be applied to real-time applications on embedded devices.

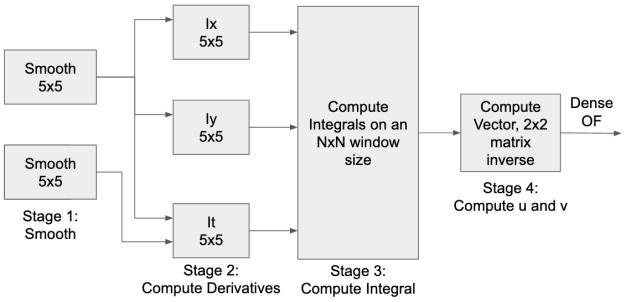


Figure 2. Block diagram of our optical flow architecture.

| OF Algorithm | Accuracy | Latency(ms) |
|---|---|---|
| TV-L1 | +8.5% | 85 |
| LK-OF | +6.24% | 2 |

Table 1. Accuracy gain on UCF101 by adding temporal streams into single spatial stream. LK-OF has shorter latency than TV-L1. The backbone model is ResNet18 pretrained on ImageNet.

### 3.2. Architecture of 2D Convolution Accelerator

We consider a flexible design for convolutional neural network accelerator [23, 3], which can be easily configured to any filter size and stride with negligible resource. Therefore, our architecture can easily process various data and scale up to concurrent inferences of multiple streams. The loop optimization techniques for 2D-convolutions in our architecture are elaborated below:

**Tiling.** For large feature maps that can not be fully loaded into on-chip memory, we cache small portion of the data in on-chip memory by tiling the loop along row, column and channel directions of the input and output feature maps.

**Unrolling.** To speed up the 2D-convolution in CNN, we choose to parallelize the computations of output channels since they are independent with each other. To output one pixel in a channel, the corresponding data in all the input channels are element-wise multiplied by filter weights and then summed together. We choose to parallelize a portion of multiplications along input channel for saving resource consumption.

In our accelerator design, the feature map and weight data are quantized to 8-bit unsigned integer while accumulator and partial sum are represented by 32-bit unsigned integer.

### 3.3. Two-Stream Action Recognition System

We have implemented a prototype system on FPGA. Fig. 3 demonstrates a screenshot of our output, in which the left part is the RGB frame and the right part is the computed optical flow. The throughput is around 10 to 15 FPS and the predictions are showed in the bottom, where the class text is yellow if the prediction is correct. The implementation details of system implementation are elaborated below:

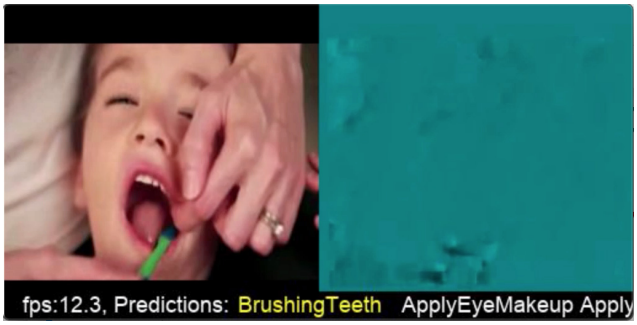


Figure 3. A detection result of our two-stream model on FPGA. The extracted optical flow is shown at right side.

**Hardware.** We implemented the system of our two-stream action recognition model on Xilinx Ultrascale+ ZCU102 by using Vivado HLS 2019.1. As shown in Fig. 1, there are one module of LK-OF accelerator and two independent modules of 2D-Convolution accelerator. All the modules are connected by AXI interconnection. The working frequency is 200 MHz. Feature maps and weights are stored in DRAM and accessed through two high performance ports. Table 2 shows the resource utilization of the system.

| Resources | LK-OF | CNN Accel | Total |
|---|---|---|---|
| DSP | 97 (4%) | 1293 (50%) | 1390 (54%) |
| BRAM | 46 (2%) | 426 (11%) | 472 (13%) |
| LUT | 9.2K (3%) | 218K (78%) | 227.8 (81%) |

Table 2. FPGA resource utilization of our two-stream action recognition model.

**Software.** We have implemented the whole two-stream action recognition system on Xilinx PYNQ framework. For an input video, the optical flow frame is computed by feeding every two consecutive frames into our module, and then stacking the most recent five optical flow frames in DRAM. We also implemented an online action recognition process, in which we define a time window on the temporal dimension of input video and sample one frame $I$ from each window for spatial stream inference. The stacked optical flow data are then sent into temporal stream module. The spatial and temporal stream are processed simultaneously by the corresponding convolution accelerators. To process longer temporal features, we perform average pooling for the most recently 8 outputs of each stream. Finally we fuse the outputs of the two modules.

**Scalability.** Regarding the DRAM used in this work, the theoretical peak bandwidth is 12.8 GB/s (single memory port). The largest required bandwidth for our two 2D-

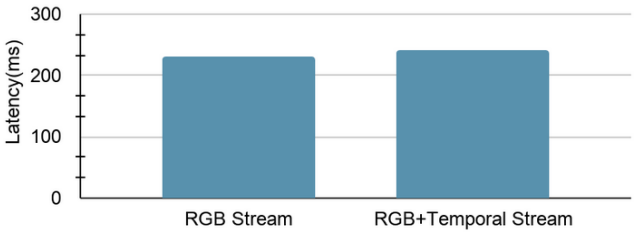


Figure 4. Our two-stream model has been optimized for the DRAM bandwidth and the latency increase is less than $5\%$.

ConvNet accelerators is $128 \times 4 \times 0.2 = 102.4$ Kb/s $= 12.8$ GB/s, which exactly matches the theoretical peak of the DRAM on-board. We have also conducted an experiment to verify and the workload of a customized CNN architecture with 5 layers of standard convolution, then each one accelerator is assigned with one inference task. As shown in Fig. 4, our two-stream architecture has been optimized for the on-board DRAM bandwidth and has a little higher latency than single RGB stream ($< 5\%$).

# 4. Experiment Results

## 4.1. Experiment Setup

**Datasets.** We conducted experiments on the most popular action recognition dataset, UCF101, which contains 101 action classes and $13,320$ video clips. We adopt the original three training/testing splits for evaluation. The Lucas-Kanade optical flow is computed from the OpenCV toolbox by setting iteration number to one and integral window size to $11 \times 11$. To reduce the storage requirements, we rescale the horizontal and vertical components of the flow linearly to integers in $[0, 255]$.

**Training & Testing.** We choose the ResNet18 as our backbone for each stream, and the models are both pre-trained on ImageNet. We employ the common training techniques including regularization, data augmentation, partial batch normalization, corner cropping and scale-jittering to avoid overfitting. With regard to testing, we make a small change of setting in TSN [21]. We sample 8 snippets, one central cropping and stack 5 optical flow frames for the input of temporal stream. We then use average pooling to aggregate the prediction from snippets and fuse the predictions from both streams by using the weights $(\text{spatial} : \text{temporal}) = (1 : 1.5)$.

## 4.2. Comparison of Two-stream Methods

Table 3 shows the comparison with previous two-stream methods in terms of accuracy gained by temporal stream and model complexity. For fairness, the comparison is on video level and uses protocol adopted by TSN [21]. But as mentioned in section 4.1, the input data for a temporal stream inference is 5-stacked optical flow frames and 8 time windows for a video. From the equation 1, we can calculate the complexity of optical flow computation for one video, and the total complexity is obtained by adding the CNN model complexity. Note that, RGBDiff uses the difference between RGB frames and does not have information of motion directions.

| Architecture | Acc. Gain | GOPs | Backbone |
|---|---|---|---|
| TV-L1 [21] | 8.5% | - | ResNet50 |
| Y. Zhu [27] | 10% | 356 | VGG16 |
| RGBDiff [21] | 4.6% | 0.007 | ResNet50 |
| PAN [24] | 5.7% | 2.8 | BN-Inception |
| Ours | 6.24% | 0.52 | ResNet18 |

Table 3. Comparing with other two-stream models on UCF101.

## 4.3. Comparison of Hardware Accelerated Action Recognition

Table 4 shows the comparison of our model with previous hardware accelerators of action recognition. Our work is based on 2D convolution, and has less computational complexity than others C3D based models while having comparable accuracy.

| Architecture | Accuracy | GOPs | Size (MB) | Backbone |
|---|---|---|---|---|
| F-C3D [5] | 79% | 76 | 321 | C3D |
| F-E3D [4] | 85% | 12.2 | 8.6 | E3DNet |
| Sun et al. [11] | 88% | 26.13 | 126 | (2+1)D |
| Ours | 86% | 4.12 | 22.3 | ResNet18 |

Table 4. Comparison of hardware accelerated action recognition.

# 5. Conclusion and Future Work

In this paper, we propose a new hardware design for action recognition different on FPGA. Our implementation is based on two-stream network with 2D-convolutional filters, which need much less computations than 3D CNN based models. We further optimize the hardware architecture to shorten the latency of temporal stream and perform efficient two-stream computation on FPGA. Experiment results show that our model can achieve up to 40 times speedup with slightly accuracy drop ($< 2\%$).

For future work, we will investigate end-to-end trainable two-stream models and run experiments on latest action datasets. Additionally, we can extend two-stream model to triple-stream by adding audio data for detecting more diverse actions.

## References


[1] J.K. Aggarwal and M.S. Ryoo. Human activity analysis. *ACM Computing Surveys*, 43(3):1–43, apr 2011. 1

[2] D. Bagni, P. Kannan, and S. Neuendorffer. Demystifying the lucas-kanadeoptical flow algorithm with vivado hls. *Tech. note XAPP1300*, 2017. 2

[3] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam. Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622, 2014. 3

[4] H. Fan, C. Luo, C. Zeng, M. Ferianc, Z. Que, S. Liu, X. Niu, and W. Luk. F-e3d: Fpga-based acceleration of an efficient 3d convolutional neural network for human action recognition. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, volume 2160-052X, pages 1–8, 2019. 1, 2, 4

[5] H. Fan, X. Niu, Q. Liu, and W. Luk. F-c3d: Fpga-based 3-dimensional convolutional neural network. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, 2017. 1, 2, 4

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016. 2

[7] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. 2

[8] S. Karen and Z. Andrew. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 568–576, Cambridge, MA, USA, 2014. MIT Press. 2

[9] J. Lin, C. Gan, and S. Han. Tsm: Temporal shift module for efficient video understanding. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7082–7092, 2019. 2

[10] Bruce David Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, USA, 1985. AAI8601180. 2

[11] S. Mengshu, Z. Pu, G. Mehmet, P. Massoud, L. Miriam, and L. Xue. 3d cnn acceleration on fpga using hardware-aware pruning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020. 1, 2, 4

[12] J. Y.-H. Ng, J. Choi, J. Neumann, and L. S. Davis. Actionflownet: Learning motion representation for action recognition. *CoRR*, abs/1612.03052, 2016. 2

[13] E. Nurvitadhi, S. Subhaschandra, G. Boudoukh, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. O. G. Hock, Y. Tat Liew, Krishnan Srivatsan, and Duncan Moss. Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '17*. ACM Press, 2017. 1

[14] G. Th. Papadopoulos, A. Axenopoulos, and P. Daras. Real-time skeleton-tracking-based human action recognition using kinect data. In *MultiMedia Modeling*, pages 473–483. Springer International Publishing, 2014. 1

[15] L. Lo Presti and M. La Cascia. 3d skeleton-based human action classification: A survey. *Pattern Recognition*, 53:130–147, may 2016. 1

[16] JS Pérez, M.-L. Enric, and G. Facciolo. Tv-l1 optical flow estimation. *Image Processing On Line*, 2013:137–150, 2013. 2

[17] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5534–5542, 2017. 2

[18] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012. 1

[19] 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. 1, 2

[20] J. C. van Gemert, M. Jain, E. Gati, and C. G. M. Snoek. APT: Action localization proposals from dense trajectories. In *Procedings of the British Machine Vision Conference 2015*. British Machine Vision Association, 2015. 1

[21] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks for action recognition in videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(11):2740–2755, 2019. 2, 4

[22] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with deeply transferred motion vector cnns. *IEEE Transactions on Image Processing*, 27(5):2326–2339, 2018. 2

[23] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, page 161–170, New York, NY, USA, 2015. Association for Computing Machinery. 3

[24] C. Zhang, Y. Zou, G. Chen, and L. Gan. Pan: Towards fast action recognition via learning persistence of appearance, 2020. 2, 4

[25] C.-L. Zhang, X.-X. Liu, and J. Wu. Towards real-time action recognition on mobile devices using deep models. *ArXiv*, abs/1906.07052, 2019. 2

[26] H. Zhu, R. Vial, and S. Lu. TORNADO: A spatio-temporal convolutional regression network for video action proposal. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, oct 2017. 1

[27] Y. Zhu, Z. Lan, N. Shawn, and A. Hauptmann. Hidden two-stream convolutional networks for action recognition. In *Computer Vision – ACCV 2018*, pages 363–378. Springer International Publishing, 2019. 2, 4

[28] Y. Zhu, X. Li, C. Liu, M. Zolfaghari, Y. Xiong, J. Tighe C. Wu, Z. Zhang, R. Manmatha, and M. Li. A comprehensive study of deep video action recognition, 2020. 2

[29] M. Ziaeefard and R. Bergevin. Semantic human activity recognition: A literature review. *Pattern Recognition*, 48(8):2329–2345, aug 2015. 1