# Appendix - Learning Unified Spatio-temporal Representations for Efficient Compressed Video Understanding

Shristi Das Biswas, Efstathia Soufleri, Arani Roy, Kaushik Roy
Purdue University
{sdasbisw, roy173, kaushik}@purdue.edu; esoufler@alumni.purdue.edu

This appendix provides additional insights and implementation details supporting the main paper. We first describe our method's energy efficiency by analyzing inference costs in Sec. 1. We then present how our Spiking Temporal Modulator (STM) inherently denoises compressed signals by isolating meaningful motion cues in Sec. 2. Finally, we detail our full network architecture and the configuration of model variants used in experiments in Sec. 3.

## 1. Inference Cost Computation

This section evaluates the energy-efficiency of our method in terms of compute intensity per inference cycle. Spiking Neural Networks (SNNs), often referred to as the third generation of neural networks, are well known for being highly energy-efficient compared to traditional Artificial Neural Networks (ANNs) (non-spiking networks) [1]. For estimating the inference cost per cycle for different architectures, we try to highlight how computation in SNNs and ANNs primarily differ from each other. SNNs offer highly sparse, asynchronous event-driven ACcumulate (AC) operations over time. Hence, a synaptic compute operation is performed only when an input spike arrives. In contrast, ANNs perform expensive Multiply-and-ACcumulate (MAC) operations for computing dense Matrix-Vector Multiplication (MVM) functions, irrespective of the sparsity of inputs. We use the findings in [2] to specify that a MAC operation requires a total of $E_{MAC} = 4.6pJ$ of energy while an AC operation requires only $E_{AC} = 0.9pJ$ for a 32-bit floating-point computation in $45$nm CMOS technology. This makes an AC operation $5.1\times$ more energy-efficient than a corresponding MAC operation. Coupled with the number of floating point operations (FLOPs) performed by the network for a single inference including all temporal and spatial views used, we benchmark compute energy cost of existing approaches following the estimation method used in [1, 3, 4] on the popularly used $45$nm CMOS process node. Note that comparisons on a different technology node would also generate similar energy requirement trends between SNNs and ANNs: while absolute energy values may vary with hardware, FLOPs are hardware-agnostic, so the ratio of costs between MAC and AC operations is preserved across platforms.

It is worth mentioning that we neglect energy consumed by the memory or any peripheral circuitry and only consider the compute cost for MAC/ AC operations. First, we calculate the total number of synaptic operations performed in each layer. For SNNs, the number of FLOPs at a layer is obtained by multiplying the mean spiking rate at each timestep for that layer, the number of synaptic connections and the number of operating timesteps. The small input spiking activities obtained in different SNN layers are mainly because of the fact that P-frame streams are highly sparse in nature and the spiking neurons generate progressively sparser outputs as the network depth increases. This sparse firing rate is essential for exploiting efficient sparsity-based computations in the SNN layers. In contrast, ANNs execute dense matrix-vector multiplication operations without considering the sparsity of inputs. In other words, ANNs simply feed-forward the inputs at once, with a fixed total number of operations. This leads to high energy requirements (compared to SNNs) since operations are executed for both zero and non-zero input values, leading to unnecessary compute [1].

Given a number of neurons $M$, a number of synaptic connections $C$, and a mean firing activity $F$, the number of FLOPs at each timestep for a layer $l$ is calculated as $M_l \times C_l \times F_l$. In the case of ANNs, we have a $mean\_spiking\_rate = 1$ and $number\_of\_timesteps = 1$ for each layer. Hence, the total compute energy cost per inference cycle can be formalized as follows:

$$FLOPs_{ANN} = \sum_l M_l \times C_l$$

$$FLOPs_{SNN} = N \sum_l M_l \times C_l \times F_l$$

$$E_{Total} = FLOPs_{ANN} \times E_{MAC} + FLOPs_{SNN} \times E_{AC}$$

where $N$ is the number of timesteps and $E_{Total}$ denotes the total compute cost for a single inference. We utilize

the above-described formulation to estimate the total compute energy required by different networks during inference. Note that we only consider convolution operations to compute the number of FLOPs, and neglect energy consumed by Batch-norm layers, or activations after each convolution layer. The results in the main manuscript suggest that our method achieves competitive performance and the least compute energy requirement compared to the heavier current state-of-the-art approaches using compressed videos [5–11], raw-domain videos [12–15] or both [16, 17]. Specifically, we reduce inference cost ($\sim$ 113$\times$ lower) and inference latency ($\sim$ 56$\times$ lower) compared to prior art. This is mainly attributed to our efficient hybrid temporal-spatial feature learning method which enables us to use a lightweight architecture for edge-compute without compromising on video recognition performance. Additionally, the STM pathway in our network contributes negligibly to the total compute energy cost, reducing our energy requirements even further.

## 2. Inherent robustness to noise with STMs

To model the compressed signals efficiently, we aim to isolate relevant motion corresponding to objects of interest in the scene, filtering out any spurious values or movement generated due to background clutter and camera noise. Spiking neurons such as the Leaky-Integrate and Fire [1, 18] are excellent candidates as they are capable of maintaining an internalized state called membrane potential $u_{mem}$, which decays over time at a rate controlled by the leak factor. The leak factor denotes how much of the membrane potential is retained for the next time step, i.e., the higher the leak factor, the slower the rate of decay. If the accumulated membrane potential of the neuron exceeds the threshold at any point, $(u_{mem} > v_{th})$, the neuron emits an output spike and resets its membrane potential.

Spurious motion in the P-frames due to sensor noise is usually generated at much lower rates than actions made by objects of interest in a scene. As such, if the time between input events is large, the membrane potential decays its value before it can reach the threshold. However, if these input events occur more frequently, they are able to overcome the decay and increase the membrane potential towards the threshold. Thus, the neuron generates output spikes if the recorded motion occurs at a frequency higher than a certain value, which is usually the case for actions made by foreground characters in contrast to redundant background motion. We visualize this phenomenon in Fig. 1. Leveraging this sparse spiking property of LIF neurons, our STM module enjoys the benefits of implicit denoising of the compressed signals with no extra overheads. This is in contrast to the dense processing mechanism in RNNs which does not directly lend itself to filtering out sensor and background noise. This contributes to our observed performance gains in noisy, compressed settings.

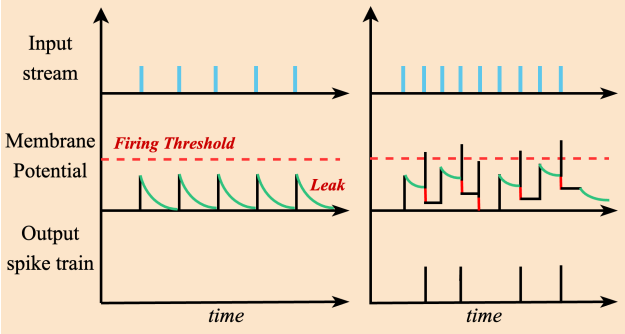


Figure 1. **Temporal sensitivity of our LIF neuron.** Left graph shows the neuron response to inputs at a given pixel occurring at lower rates; the right graph shows neuron response to inputs occurring at a higher rate. Once the inputs occur at an intensity higher than the learnt threshold, i.e, there is significant motion recorded at a pixel location, the neuron records 'spikes' and resets to a resting value.

## 3. Architecture Details and Network Variants

Our network is inspired by the U-Net [19] architecture. We use a temporal (STM), local spatial (learnable patch embedding) and global spatial encoder (unified transformer encoder), followed by a feature mixing module (Multi-modal Fusion Block) and a decoder (Residual-in-Residual Dense Block and linear classifier).

The STM includes three conv-LIF layers with a 'soft reset' strategy [20, 21]. Accumulated analog membrane potential maps from the STM branches and the I-frame streams are used as context inputs to separate local spatial encoders comprising a learnable convolution and flatten layer. This decomposes the multi-modal features into a $N$ non-overlapping patches, where $N = 16$. The global spatial encoder (unified transformer) adds spatiotemporal positional encoding [11] to maximize context in the multi-modal feature maps and follows the Vision transformer (ViT) [13] architecture as its base (depth= 12 and nheads= 12). Tokens from the transformer encoder are then reshaped back to their original dimensions using the pytorch token folding operation and modulated with Conv. Folding modules comprising a $3 \times 3$ overlapping kernel convolution layer with ReLU activation. Fusion with the initial feature maps happens with the Conv. Folding outputs after passing through a learnable convolution connection in the skip branch. These consolidated features are then passed through a Conv. Fusion block consisting overlapping $3 \times 3$ kernel convolution layers with LeakyReLU activation [22]. The output filter size for all these conv. layers are set to 64. We do element wise addition on these multi-modal features and feed them to the residual in residual dense block (RRDB) [23] before the final classifier. For the 'Ours-L'

variant reported in the main paper, the output filter size is set to 96 and the unified transformer is increased to 16, to enhance network complexity and depth for learning more challenging datasets.

## References


[1] C. Lee, A. K. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks," in *European Conference on Computer Vision*, pp. 366–382, Springer, 2020. 1, 2

[2] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pp. 10–14, IEEE, 2014. 1

[3] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014. 1

[4] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017. 1

[5] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6026–6035, 2018. 2

[6] Z. Wang, Q. She, and A. Smolic, "Team-net: Multi-modal learning for video action recognition with partial decoding," *arXiv preprint arXiv:2110.08814*, 2021.

[7] H. Terao, W. Noguchi, H. Iizuka, and M. Yamamoto, "Efficient compressed video action recognition via late fusion with a single network," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.

[8] Z. Shou, X. Lin, Y. Kalantidis, L. Sevilla-Lara, M. Rohrbach, S.-F. Chang, and Z. Yan, "Dmc-net: Generating discriminative motion cues for fast compressed video action recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1268–1277, 2019.

[9] S. F. dos Santos, N. Sebe, and J. Almeida, "Cv-c3d: action recognition on compressed videos with convolutional 3d networks," in *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 24–30, IEEE, 2019.

[10] M.-C. Wu and C.-T. Chiu, "Multi-teacher knowledge distillation for compressed video action recognition based on deep learning," *Journal of systems architecture*, vol. 103, p. 101695, 2020.

[11] J. Chen and C. M. Ho, "Mm-vit: Multi-modal video transformer for compressed video action recognition," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1910–1921, 2022. 2

[12] Y. Zhang, X. Li, C. Liu, B. Shuai, Y. Zhu, B. Brattoli, H. Chen, I. Marsic, and J. Tighe, "Vidtr: Video transformer without convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 13577–13587, 2021. 2

[13] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6836–6846, 2021. 2

[14] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

[15] Y. Chen, D. Chen, R. Liu, S. Zhou, W. Xue, and W. Peng, "Align before adapt: Leveraging entity-to-region alignments for generalizable video action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18688–18698, 2024. 2

[16] B. Battash, H. Barad, H. Tang, and A. Bleiweiss, "Mimic the raw domain: Accelerating action recognition in the compressed domain," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 684–685, 2020. 2

[17] Y. Liu, J. Cao, W. Bai, B. Li, and W. Hu, "Learning from the raw domain: Cross modality distillation for compressed video action recognition," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023. 2

[18] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016. 2

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015. 2

[20] E. Ledinauskas, J. Ruseckas, A. Juršėnas, and G. Buračas, "Training deep spiking neural networks," *arXiv preprint arXiv:2006.04436*, 2020. 2

[21] B. Han, G. Srinivasan, and K. Roy, "Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13558–13567, 2020. 2

[22] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015. 2

[23] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, pp. 0–0, 2018. 2