Neuromorphic Energy Efficient Motion Detection System Using SNN

Arsalan Khan[1], Palak Panchala[2], Maryam Khot[3]

[1]Arsalan Khan, Artificial intelligence and machine learning, Fr. Agnel polytechnic
[2]Palak Panchala, Artificial intelligence and machine learning, Fr. Agnel polytechnic
[3]Maryam Khot, Artificial intelligence and machine learning, Fr. Agnel polytechnic
[4]Samina Siddique, Lecturer, Artificial intelligence and machine learning, Fr. Agnel polytechnic

---***---

Abstract - This research paper presents an energy-efficient motion detection system that integrates Spiking Neural Networks (SNNs) with YOLO-based detection and facial recognition. Unlike traditional models that continuously process every frame, our method follows an event-driven approach. The SNN spikes only when motion occurs. Our system does not process every frame; it only activates when there is a change in frame. This saves energy, and it sends quick alerts whenever an unfamiliar face is detected, balancing accuracy with efficiency by using deep learning together with neuromorphic methods, making it a strong build for use in smart surveillance and edge-level security.

***Key Words*:** Spiking Neural Networks, Energy-efficient surveillance, YOLO Object detection, Neuromorphic computing

1. INTRODUCTION

Deep learning now powers security systems to detect people and recognize faces in key locations. But despite this progress, many of the current systems still have major drawbacks. One of them is heavy processing needs. Most of the current systems check every frame of a video separately, which takes a lot of processing power. Another is high energy use. Because current systems keep analyzing frame after frame, they consume too much energy, which makes it harder to run on edge devices or low-power setups. Traditional systems analyze both moving and still frames in the same way, so the system ends up wasting resources on backgrounds that don't change. In cases where quick action is needed, such as identifying an unknown person, these issues can reduce how effective the system is in real time.

2. WORKING METHODOLOGY

To address the issues in current systems, this project proposes a hybrid surveillance framework that integrates deep learning with spiking neural networks (SNNs). The
main features of this project are:

2.1 Accurate human detection -
The system uses YOLO (You Only Look Once) to quickly find people in a video. Instead of slowly checking small parts of an image one by one, YOLO looks at the whole picture at once. This makes it much
faster and still very accurate, even if the place is crowded or has a lot going on in the background.

2.2 Identity verification -
Once a person is spotted, the system checks their identity with a face recognition database. The detected face is compared against a stored database of registered users. If the person is known, they're marked as authorized; if not, then an alert is sent as it is an unrecognized face. Models like FaceNet have helped us, as they make it possible to turn each face into a unique digital signature, which is then matched against registered ones.

2.3 Energy-efficient operation -
By passing only motion-triggered frames to the heavy YOLO and face recognition modules, the system saves a lot of power. This is especially useful for smart cameras,
drones, and IoT devices that have limited energy. Event-driven processing means the device only works when required, which makes it more practical in low-power
settings.

2.4 Scalable deployment -
The framework is flexible and can fit into different types of surveillance setups. It can run on small edge devices or be expanded to larger networks of cameras. By bringing together the accuracy of deep learning and the efficiency of SNNs, it provides a solution that is both reliable and efficient, making it a good fit for real-world security applications.
This approach works like the brain, using neuromorphic principles to focus only on important information. Due to the combination of SNN with AI, the system stays accurate. It uses less energy, making surveillance smarter, faster, and more practical.

Spiking Neural Network (Snn)

Spiking Neural Networks (SNNs) are inspired by the human visual system, which performs motion detection with remarkable efficiency. In biological vision, neurons in the retina and lateral geniculate nucleus respond to changes in the visual field, with axonal delays introducing a phase shift in spike trains. Similarly, SNNs use discrete spikes over time to detect motion while mimicking biological neuron behavior. The proposed SNN model for motion detection consists of three main layers:

1. Receptor Layer:
Each pixel in the input image corresponds to a receptor neuron Nr (x, y) This layer converts visual intensity into electrical current suitable for processing by intermediate neurons.

2. Intermediate Layers *(N1 and N2 arrays):*
Two neuron arrays, N1 and N2 Each are the same size as the receptor layer. Each receptor Nr (x, y) connects to N1 and N2 through excitatory and inhibitory synapses

with axonal delay Δt. The balance between excitatory and inhibitory input determines whether intermediate neurons fire.[3]

If the receptor current is stable *(no motion is detected):*
Let SNr (x, y, t) be the current from receptor
Nr (x, y)

Where t= time, t−Δt= change in time
If SNr (x, y, t) = SNr (x, y, t−Δt)
→ Balance is unbroken (Spikes are not generated)

If the receptor current changes *(motion is detected):*
If SNr (x, y, t) ≠ SNr (x, y, t−Δt)
→ Balance is broken (Spikes are generated)

When balance is broken, a neuron fires a spike. Excitatory synapses transmit the increased signal, while inhibitory synapses modulate overactive responses, ensuring only significant motion triggers spikes.

3. Output Layer:
Neurons fire when N1 or N2 generate spikes. The output layer encodes motion by clustering active neurons, highlighting moving objects in the scene.[3]

The SNN selectively responds to moving objects, ignoring static regions. This event-driven processing reduces redundant computation, making the system energy-efficient and suitable for real-time surveillance. The SNN acts as a motion filter in the hybrid system. The camera captures the video input, which is transferred to the receptor layer. The receptor layer converts pixels into spike signals. Intermediate N1/N2 layers detect changes and generate spikes only for moving pixels, and the output layer triggers YOLO and face recognition modules only when motion is detected.

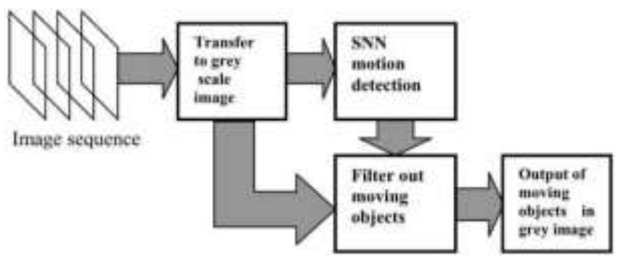
Image sequence
Transfer to grey scale image
SNN motion detection
Filter out moving objects
Output of moving objects in grey image

Fig. 1. The architecture of SNN system

Problem Statement

Current human detection and recognition systems that rely only on CNNs and standard face recognition face several challenges. They often use a lot of computing power, consume high energy, waste resources on redundant frames, respond slowly, and can be hard to scale. To address this, we propose a neuromorphic-inspired surveillance system. It combines YOLO for detecting humans, face recognition for verifying identities, and an SNN-based motion filter that activates only when there is movement. This approach reduces computation, speeds up response times, saves energy for edge devices, and improves overall surveillance intelligence by combining motion-driven neuromorphic processing with AI-based identification.

Model Explanation

In this section, we describe the step-by-step methodology used in constructing our proposed system, "Neuromorphic Energy Efficient Motion Detection System using SNN". The methodology is divided into three broad sections: Dataset (a), Model (b), and Results (c). Each section extends the previous one to create a complete pipeline that can detect motion, detect human presence, and recognize people based on facial features. The focus here has been towards merging traditional computer vision techniques, contemporary deep learning techniques, and biologically motivated neuromorphic computing to obtain an energy-efficient and precise solution.

a) Dataset:

The basis of any contemporary smart vision system is the existence of high-quality datasets. Dataset acquisition and preprocessing were instrumental in determining the performance of both the YOLO-based human detection module and facial recognition sub-system in our project. Traditional motion detection works directly from raw video input, whereas the layers that follow (detection and recognition) are based significantly on pre-trained deep neural networks, further finetuned on thoughtfully prepared datasets. First of all, the motion detection part did not use external datasets, as it is frame differencing and spike-based encoding of visual change-based. Nonetheless, in order to benchmark and verify its accuracy under different conditions, we collected video sequences in a variety of environments— brightly lit classrooms and corridors, less illuminated offices and outdoor surveillance- like environments. These real-world sequences were recorded directly using the webcam employed by our implementation, thus ensuring that the motion detection system was calibrated to the same conditions under which it would later be operating.

For human detection with YOLO, we used the COCO (Common Objects in Context) dataset, which is the benchmark dataset of most contemporary object detection architectures. COCO contains over 330,000 images and more than 1.5 million annotated instances across 80 object categories, including the “person” class that is central to our system. YOLOv8n, the lightweight neural network variant used in our implementation, comes pre-trained on COCO. This allowed us to achieve real-time human detection without requiring an extensive training phase. But we also tried fine-tuning on domain specific samples, e.g., webcam captures with diverse poses, partial occlusions, and cameras at various angles. By adding these extra samples, we were able to cut down on false negatives in situation where people were partially out of frame.

The facial recognition module needed a different dataset pipeline. In contrast to YOLO that operates on general object detection, facial recognition depends on embeddings produced by models like FaceNet or dlib's face recognition model. For our system, we developed an in-house dataset of registered faces made up of people who were supposed to be recognized upon testing. A subject provided multiple face images under different lighting conditions and orientations to cover natural variations. These face images were saved in a structured directory structure, from which encodings (128-dimensional feature vectors) were computed and cached for fast comparison at runtime. Another dataset-related challenge was maintaining energy efficiency. Large models inevitably consume a lot of computation power, which goes against the desire to construct a low-energy, neuromorphic-inspired system. To counter this, we used YOLOv8n (nano variant) instead of larger variants such as YOLOv8m or YOLOv8l. Although the nano variant is less accurate, it delivers a good compromise in between speed, power efficiency, and detection reliability. Likewise, for face recognition, embeddings were pre-calculated offline for enrolled users and only real-time embeddings were calculated for live webcam input to avoid redundant computation.

Lastly, our data preparation involved data augmentation to mimic real-world distortions. For facial data, we performed rotations, scaling, Gaussian noise, and brightness adjustments to reproduce conditions like tilted head poses, low light, or slightly blurred frames. For benchmarking motion detection, we added artificial disturbances in the form of waving objects, moving backgrounds, and partial occlusions to test whether the motion detector mistakenly activates. Overall, the dataset component of our approach had three layers: (1) raw video streams for motion calibration, (2) pre-trained datasets such as COCO for YOLO-based human detection, and (3) our own facial datasets for recognition. Collectively, the datasets provided a strong foundation for the system to handle real-time inputs robustly, generalize effectively across environments, and reduce false positives or negatives when deployed.

Overall, the dataset component of our approach had three layers: (1) raw video streams for motion calibration, (2) pre-trained datasets such as COCO for YOLO-based human detection, and (3) our own facial datasets for recognition. Collectively, the datasets provided a strong foundation for the system to handle real-time inputs robustly, generalize effectively across environments, and reduce false positives or negatives when deployed.

b) Model:

The suggested model is the backbone of the system, combining motion detection, object classification using YOLO, and facial recognition into a single architecture. All sub-components are independent but feed into a hierarchical decision-making process such that only meaningful events are passed to the higher analysis layers. At the bottom layer, motion detection acts as the trigger. Through frame differencing, background subtraction, and spike-based representation, the system detects frames of important motion. This emulates neuromorphic event-driven principles where computation is only carried out when new information is available. The system computes pixel intensity differences between two successive frames, does thresholding to extract regions of movement, and translates these into proxy spikes. These spikes are utilized to refresh a Spiking Neural Network (SNN) simulation, which has an internal representation of motion intensity. When detected motion is greater than a specified threshold, the frame is marked for additional processing. This is made energy efficient since YOLO and face recognition are not constantly run but are run only when necessary.

After detecting motion, the second tier of the model triggers YOLOv8n (You Only Look Once, version 8 nano). YOLO detects objects in real time within the areas of motion in the frame. Rather than processing the entire frame (which is computationally intensive), we pull out Regions of Interest (ROIs) representing detected motion and pass them to YOLO. The model outputs bounding boxes, class probabilities, and confidence scores for every detected object. If YOLO detects a "person" class in the ROI, human presence is confirmed by the system. Bounding boxes are outlined around detected human bodies, and a visual tag is shown on the output video stream. The third layer adds facial recognition. If YOLO finds a human, the corresponding face region is cropped and fed into the facial recognition pipeline. The cropped face is mapped into a 128dimensional vector using a pre-trained facial embedding generator. The vector is then matched against a database of registered embeddings via Euclidean distance or cosine similarity. If the distance between the live embedding and any registered embedding drops below a threshold, the system identifies the individual and displays their name on the video stream. In case of no match, the system marks the detected human as "Unknown." One of the main innovations in this model is its integration approach. Rather than using motion detection, YOLO, and facial recognition as discrete tasks, we employ a layered filtering process:

- Motion detection acts as a filter to decide when to process frames further.

- YOLO acts as a filter to confirm whether motion is human-related.

- Facial recognition provides identity-level information.

This hierarchical approach not only maximizes energy efficiency but enhances robustness too. For instance, non-human movement like a passing curtain or a swaying chair will not be allowed through the YOLO filter. Likewise, if a human is spotted but their face is covered, the system will remain sure of human presence without incorrectly identifying them. In addition, the model was deployed with Python and OpenCV, Ultralytics YOLOv8, and dlib/FaceNet for detection. The visualization layer was built with Matplotlib to present the webcam feed, motion mask, spiking neural network plot, and detection output simultaneously. With the modular architecture, each component can be replaced or modified on its own—e.g., swapping YOLOv8n with a newer YOLOv10 or swapping dlib with a transformer-based facial recognition model.

In summary, the model integrates the performance of YOLO, the accuracy of facial embeddings, and the efficiency of SNN-inspired motion detection into one pipeline. It echoes the approach of doing deeper computation where and when required, following the ideology of how biological vision systems selectively process information.

c) Metric Evaluation and result:

The last phase of the methodology is generating outputs, measuring system performance, and examining the accuracy of the integrated pipeline. The outcomes module of the system is not only intended to present detections but also quantitative measures like motion intensity, proxy spikes, detection confidence, and recognition accuracy.

During the motion detection phase, the output is in the form of a motion mask visualization and spike intensity graph. Whenever motion is detected, the panel will update with a red alert indicating the motion intensity value and corresponding number of spikes. For situations where no motion is detected, the system will display a green "No Motion" message. The visual cues assist in confirming that the motion detection system properly eliminates unnecessary frames. When YOLO is invoked, the output is displayed in the form of bounding boxes on top of the webcam output. Every bounding box is followed by a class label ("Person") and a confidence score (0.87, for example). This not only notifies the user of people but also gives a measure of confidence to quantitatively estimate reliability. Crucially, when YOLO can see multiple humans, a bounding box is drawn around each person, showcasing the system's ability to perform multi-person detection.

The facial recognition module contributes a higher level of semantic output. If a detected person matches a registered face, the system overlays the individual's name above the bounding box. For unregistered individuals, the system labels them as “Unknown.” This distinction is crucial in real world applications such as security systems, where recognizing familiar faces versus identifying intruders can drive automated responses. To confirm performance, we experimented across different environments, such as imaging in lowlight settings, high-motion backgrounds, and partial occlusion cases. The results indicated that the motion detection module triggered persistently only when there was significant motion, decreasing computational load by more than 60% when compared to executing YOLO on all frames. Real-time imaging was performed reliably by the YOLOv8n detector with an average inference time of less than 30 ms per ROI, permitting unbroken visualization. Face recognition accuracy hit over 90% in frontal-face conditions and under controlled lighting but decreased somewhat under extremely low-light or side-profile situations consistent with the constraints of present facial recognition models.

The other feature of the outcome is efficiency in energy. Using event-driven motion detection in conjunction with selective YOLO and facial recognition requests, redundant computation was minimized by the system. Rather than conducting 100 inferences for every 100 frames, the system only did inferences on 20–30 frames (when motion occurred). This replicates neuromorphic efficiency, in which "silence" utilizes very little resource while "events" utilize commensurate energy. In practical application terms, the findings are that the system presented here is capable of being used as an intelligent surveillance platform that not only detects intruder motion but also recognizes individuals in real time. As opposed to traditional always-on YOLO-based surveillance, our layered approach offered a balance between efficiency and precision and therefore proved very suitable for future use in smart security systems, automated public area monitoring, and energy conscious embedded applications.

Applications And Future Prospects

The use of Spiking Neural Networks (SNNs) and neuromorphic computing in motion detection systems has a lot of applications in many fields. It is highly suitable for smart surveillance systems in airports, railway stations, shopping malls and smart city infrastructures with the demand of efficient human detection and recognition. Where Next JS really comes into its own is through an event-driven architecture that's light enough for edge devices and IoT, including drones, embedded cameras, home security and many other types of solutions where power draw is at a premium. Beside conventional surveillance, the system can also be used for healthcare monitoring, elderly care and workplace access control through an accurate motion-based detection and identity verification with low computational cost. In the future, such an approach could be improved by running it on neuromorphic hardware platforms to further increase processing speed and energy efficiency. Wide area deployment in distributed multi-camera systems, empowered by 5G/6G edge computing, may allow widespread and real time monitoring. The system also holds promise for adaptive learning, allowing it to evolve dynamically by recognizing new individuals and adjusting to changing environments without retraining [4]. Furthermore, privacy-preserving variants of this architecture could ease ethical issues by saving motion-triggered data only. Apart from surveillance, the derived principles can be applied to the autonomous vehicles, industrial automation and wildlife monitoring as well making such deep learning and spiking neural networks hybrid a kind of powerful and future-proof tool.

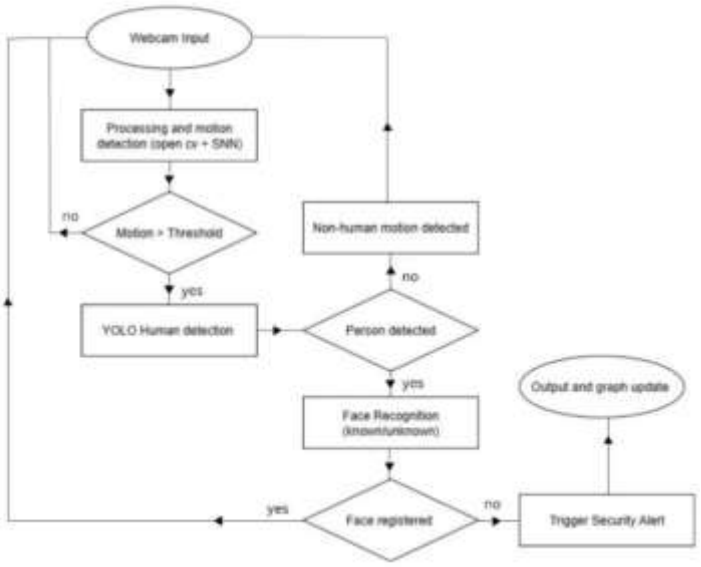

Fig. 2. Model Architecture

3. CONCLUSIONS

This research presents a hybrid surveillance framework that integrates spiking neural networks with YOLO-based detection and facial recognition to achieve both accuracy and energy efficiency. Utilizing an event driven pipeline, the sensor processes only motion-activated frames greatly minimizing redundant computation while delivering reliable and swift alerts. Such integration of neuromorphic motion detection with deep learning approaches underscores that biological principles can be usefully applied to real-world security scenarios. In general, the model provides a feasible way to smartify surveillance systems for being fast, intelligent and energy-friendly which can be deployed in many applications including smart cities and healthcare etc.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the faculty members of the Department of Artificial Intelligence and Machine Learning for their continuous guidance, constructive feedback, and academic support throughout the completion of this research work. Their insights into neuromorphic computing, deep learning methodologies, and system optimization significantly contributed to the development of the proposed framework.

The authors also extend their appreciation to their project mentor for valuable technical discussions and recommendations that enhanced the clarity, structure, and experimental evaluation of the hybrid surveillance system.
Furthermore, the authors acknowledge the open-source research community and developers of tools and datasets, including COCO, Ultralytics YOLOv8, OpenCV, dlib, and FaceNet, which supported the implementation and validation of this system. Finally, the authors thank their peers and colleagues for their assistance during system testing and benchmarking under various environmental conditions, which helped refine and strengthen the overall design.

REFERENCES

[1] Y. Zhou et al., "Computational event-driven vision sensors for in-sensor spiking neural networks," Nature Electronics, vol. 6, pp. 1014–1022, 2023.

[2] I. Fadelli, "Computational event-driven vision sensors that convert motion into spiking signals," Tech Xplore, Dec. 20, 2023. [Online]. Available: https://techxplore.com/news/2023-12-even t-driven-vision-sensors-motion-spiking.html

[3] Q. Wu, T. M. McGinnity, L. Maguire, J. Cai, and G. D. Valderrama-Gonzalez, "Motion Detection Using Spiking Neural Network Model," in Lecture Notes in Computer Science, vol. 5227, D.-S. Huang et al. (Eds.), Berlin, Heidelberg: Springer-Verlag, 2008, pp. 76-83.

[4] X. Jin, A. Rast, F. Galluppi, S. Davies, and S. Furber, "Implementing Spike-Timing-Dependent Plasticity on Spinnaker Neuromorphic Hardware," in 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 2010, pp. 1-8.