Biologically Inspired vs. Attention-Driven Architectures: Comparative Evaluation on Blood Cell Image Classification

Arina Ponomareva†, Margarita Shnaider‡

***Abstract*—This project presents a comparative study between two neural network architectures - Spiking Convolutional Neural Networks (SCNNs) and Attention-Enhanced Convolutional Neural Networks (AE-CNNs) - applied to the task of multiclass blood cell classification. The SCNN combines conventional convolutional layers with spiking neuron models to capture both spatial and temporal patterns, while the AE-CNN integrates a lightweight channel attention mechanism within a standard convolutional framework to enhance feature selection.**

Both models were implemented using modern deep learning libraries and trained on the BloodMNIST dataset. The evaluation focuses on accuracy, macro F1-score, confusion matrices, and ROC curves to assess classification performance across eight cell types. Despite architectural differences, both models achieved high classification accuracy and balanced performance across classes.

This comparison provides insights into the respective strengths and trade-offs of biologically inspired and attention-based models for static biomedical image analysis. It also highlights the practicality of deploying either approach depending on the application constraints and future interest in hybrid designs.

***Index Terms*—Spiking neural networks, convolutional neural networks, attention mechanisms, biomedical image classification, BloodMNIST, SE blocks, deep learning, class imbalance**

I. Introduction

With the growing demand for increasingly powerful AI models, there is an urgent need for novel solutions that not only meet performance expectations but also address the rising concerns over computational cost and energy consumption. One promising direction lies in biologically inspired models, particularly Spiking Neural Networks (SNNs) [1]. While SNNs have already demonstrated excellent energy efficiency, their accuracy still remains a challenge.

In our work, we prioritize classification accuracy over strict biological plausibility. We adopt a simple convolutional architecture where each convolutional layer is followed by a Leaky Integrate-and-Fire (LIF) spiking neuron. This design efficiently captures spatial and temporal patterns while leveraging gradient-based training through the `snnTorch` library [2], a PyTorch package. Historically, the lack of suitable platforms for training SNNs posed significant challenges [3]. However, recent advances in tools like `snnTorch` have streamlined spiking neural network training, making it more practical for real-world applications, a key advantage leveraged in our work.

Beyond exploring the capabilities of SNNs, we also compare them with a modern deep learning approach: an attention-enhanced convolutional neural network. Originally proposed to improve computational efficiency, AE-CNNs enhance conventional CNNs by incorporating attention mechanisms, which have shown to boost accuracy while maintaining relatively low computational cost [4]. This comparison allows us to evaluate the trade-offs between biologically inspired and attention-based models in terms of both performance and efficiency.

We evaluated our models, a Spiking Convolutional Neural Network and AE-CNN, on the BloodMNIST dataset. Despite the dataset's challenges, including non-trivial blood cell patterns, limited sample size, and class imbalance, our models achieved competitive performance: 95% accuracy for SCNN and 93% for AE-CNN.

The rest of this report is organised as follows. Firstly, in Section II we describe the theoretical foundations of the models we explore. Descriptions of the network architectures, datasets, and data preprocessing techniques used in our study are provided in Sections III and IV, respectively. The experimental results, including model performance comparisons and key findings, are discussed in Sections V and VI. Finally, we conclude the report with remarks and future directions in Section VII.

II. Related Work

Spiking Neural Networks. Being a brain-inspired computing model, it uses spatio-temporal dynamics to mimic biological neuronal functionality. There exists a wide range of spiking neuron models, spanning from biologically realistic to computationally efficient implementations. Despite this variety, all models share the same core principle: processing information through binary spikes across discrete time steps, formally represented as sums of Dirac delta functions.

Since spiking neural networks require spike sequences for information transmission, input data encoding is a crucial preprocessing step. The two primary encoding schemes are rate coding and temporal schemes [5]. Notably, some advanced models can directly process raw input data without explicit encoding [6].

Leaky Integrate-and-Fire Model. The LIF model remains one of the most widely adopted spiking neuron models in practice, striking an optimal balance between biological plausibility and computational efficiency. The model takes the sum of weighted inputs and then it operates on principles analogous to RC circuits, where input currents are temporally

integrated until the membrane potential U exceeds a threshold θ, triggering a spike [5].

The dynamics of a LIF neuron in discrete time are described by some variations of the following equation [1]:

$$U[t] = \beta U[t-1] + WX[t] - \theta S[t-1] \tag{1}$$

where:

- $U[t]$ represents a membrane potential at time step t,
- $\beta \in [0, 1)$ is a membrane leakage factor,
- $WX[t]$ is a weight matrix multiplied by input at time t,
- $\theta > 0$ is a firing threshold,
- $S[t] \in \{0, 1\}$ is a binary spiking function (1 when $U[t] > \theta$).

Due to its similarity with artificial neuron models, the LIF model has been efficiently implemented in the `snnTorch` library, which provides flexible parameter control, enabling to fine-tune its dynamics for various applications [2].

Attention-Enhanced Convolutional Neural Networks. While biologically inspired models such as Spiking Neural Networks have shown promise in energy-efficient computing, modern convolutional neural networks have advanced significantly through the use of attention mechanisms. A key development in this direction is the Squeeze-and-Excitation (SE) block introduced by Hu et al. [7], which enables channel-wise attention by learning to reweight feature maps based on global context. This lightweight module has been successfully integrated into standard CNN backbones, improving performance on classification benchmarks with minimal computational overhead.

In the field of medical imaging, attention-based architectures have also demonstrated practical advantages. Attention U-Net [8] and Attention Gated Networks [9] apply spatial and channel attention mechanisms to guide networks toward salient regions, improving both segmentation and classification accuracy in challenging biomedical datasets. Such models also contribute to better interpretability, a desirable feature in clinical settings.

Despite these advances, systematic comparisons between attention-enhanced CNNs and biologically plausible models such as SNNs remain limited. Most evaluations treat these two directions in isolation. In this work, we aim to fill this gap by comparing an Attention-Enhanced CNN, based on SE-style bottleneck attention, to a Spiking CNN architecture on the same biomedical dataset. This controlled comparison offers insight into the trade-offs between biological plausibility, interpretability, and predictive performance in real-world classification tasks.

III. Processing Pipeline

A. Spiking Convolutional Neural Network

The aim of our approach is to combine the spatial feature extraction capabilities of convolutional neural networks with the temporal processing dynamics of spiking neurons. The pipeline begins with image preprocessing, where input images are normalized to the range [0, 1] and converted into PyTorch tensors. No explicit temporal encoding is applied; instead, static images are repeatedly processed across a fixed number of time steps using the inherent temporal dynamics of LIF neurons.

The convolutional layers, incorporating ReLU activation and max-pooling that reduces spatial dimensions by a factor of two, extract spatial patterns from static input images. Subsequently, the LIF spiking layers process these features across multiple discrete time steps (10 in our case), accumulating information over time. Despite the static nature of the inputs, the spiking neurons temporally integrate information by accumulating activations based on their intrinsic leak, threshold, and reset dynamics. The resulting feature maps are flattened and passed to a fully connected spiking layer.

At each time step, the same image is presented to the network, allowing neurons to accumulate membrane potential and fire over time. Classification is performed using rate coding, where spike counts are accumulated over fixed number of time steps, and the class with the highest average firing rate is selected.

B. Attention-Enhanced Convolutional Neural Network

The processing pipeline for our Attention-Enhanced Convolutional Neural Network (AE-CNN) consists of five main stages: input preprocessing, data augmentation, feature extraction via convolutional layers, attention-based feature refinement, and classification through a fully connected head.

The pipeline begins with raw RGB images resized to 64×64 pixels and normalized to the $[0, 1]$ range. During training, data augmentation techniques such as flipping, rotation, and zooming are applied to improve generalization and mitigate class imbalance.

The core of the AE-CNN consists of stacked convolutional layers that extract hierarchical spatial features from the input images. These are followed by a bottleneck-style channel attention module, inspired by the Squeeze-and-Excitation mechanism, which recalibrates feature maps by weighting the most informative channels.

Finally, global average pooling reduces the spatial dimensions, and the result is passed to a dense classification head with dropout for regularization. The full architecture is optimized end-to-end using supervised learning on the BloodMNIST dataset.

Figure 1 provides a visual summary of the AE-CNN pipeline and how the different components interact.

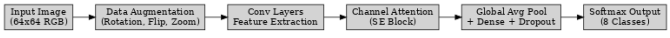

Fig. 1: Processing pipeline for the AE-CNN model. Input images undergo augmentation and feature extraction via convolution and attention blocks, followed by classification.

IV. Signals and Features

A. SCNN Model

The input data for spiking convolutional neural network consists of static RGB images with a fixed dimension. Each

image is normalized to the range $[0, 1]$ and converted to a PyTorch tensor format. Since the model architecture does not require time-varying signals, there was no additional signal preprocessing.

Feature extraction is performed internally by the convolutional layers. After two stages of convolution and max-pooling, spatial features are flattened and passed to the spiking layer. Temporal dynamics is handled by the LIF neuron model, which integrates input current over time and emits spikes based on learned thresholds and decay parameters. On the Figure 2 shown how the model extracts both spatial and temporal features through convolutional and spiking layers, demonstrated on a single test image. For both models we used the standard pre-defined splits from the MedMNIST PyTorch API.

B. AE-CNN Model

The input to the AE-CNN model consists of static RGB images from the BloodMNIST dataset. Each image was resized to 64×64 pixels and normalized to the $[0, 1]$ range. The model was implemented in TensorFlow and does not rely on time-varying or temporal signals.

To improve model generalization and mitigate overfitting, we applied offline data augmentation techniques to the training set. These included random flips, rotations, and zooms. Augmented images were added to the training dataset, effectively increasing its size and variability while preserving label integrity.

Feature extraction is performed through three convolutional blocks, each followed by max-pooling. These blocks capture progressively abstract spatial patterns such as cell borders and textures. A bottleneck attention module, inspired by the Squeeze-and-Excitation (SE) block, was added after the third convolution to allow the model to recalibrate feature importance at the channel level. This mechanism uses global average pooling to generate per-channel weights, which are then used to modulate the convolutional feature maps.

After the attention layer, a final convolution and global average pooling compress the features before they are passed to a fully connected head. The classification head consists of one dense layer with dropout for regularization, followed by a softmax layer outputting class probabilities for the eight blood cell types.

We used the same dataset split as for the SCNN: 70% for training (including augmented samples), 15% for validation, and 15% for testing.

V. Learning Framework

A. SCNN Architecture

We trained a spiking neural network using surrogate gradient descent on rate-coded spike counts. The model combines convolutional feature extraction with LIF spiking neurons to perform image classification. The network architecture is summarized in Table 1.

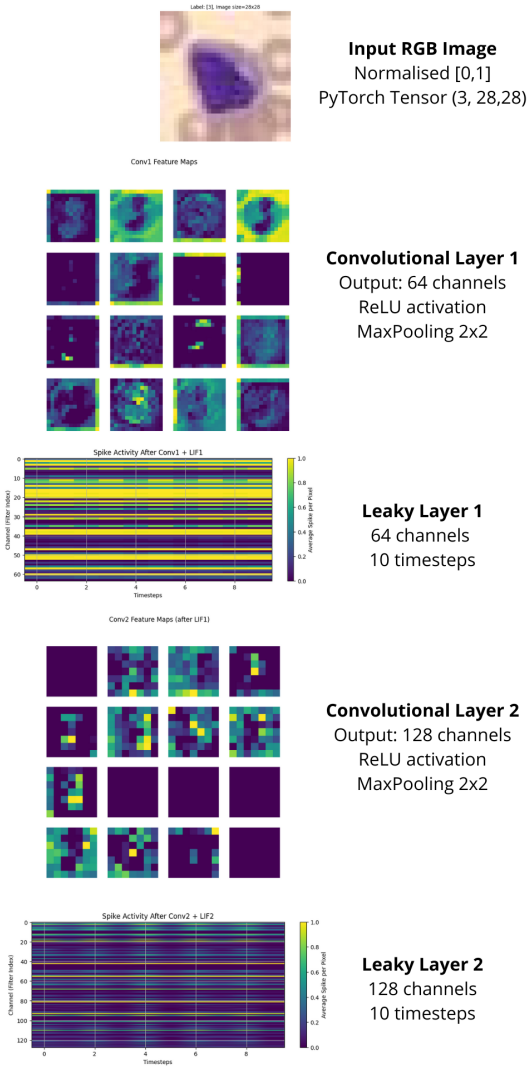

Fig. 2: Feature extraction and temporal processing pipeline for the SCNN. The input image, with dimension (3, 28, 28), where 3 is a number of channels (RGB) and (28, 28) is an image size, is first passed through the first convolutional layer (Convolutional Layer 1), producing 64 spatial feature maps (16 shown). These are then processed over 10 timesteps by the first Leaky Integrate-and-Fire (Leaky Layer 1) layer, which integrates information temporally across channels. The resulting membrane states are passed to a second convolutional layer (Convolutional Layer 1), followed by another LIF layer (Leaky Layer 2) that accumulates spiking activity.

Block	Description
Conv Block 1	2D convolution (64 filters, kernel size 3, stride 1, padding 1) followed by ReLU activation and MaxPooling (kernel size 2); output passed to LIF spiking layer
LIF Layer 1	Leaky Integrate-and-Fire (LIF) neurons with learnable decay β and threshold; captures temporal dynamics of Conv Block 1 output
Conv Block 2	2D convolution (128 filters, kernel size 3, stride 1, padding 1) followed by ReLU and MaxPooling (kernel size 2); output passed to LIF spiking layer
LIF Layer 2	LIF neurons processing temporal evolution of second convolutional output
Flatten	Converts 3D feature maps to 1D vector
Spiking FC Layer	Fully connected LIF layer with learnable threshold and β
Output	Classification into 8 classes using rate coding over time steps

TABLE 1: SCNN network architecture

The network processes inputs over 10 discrete timesteps. Surrogate gradients, implemented via a fast sigmoid function, enable backpropagation through the spike non-linearity.

Although placing a ReLU activation before a LIF layer is not common in biologically motivated spiking models (LIF adds non-linearity instead), it was a beneficial choice in our case. ReLU ensures that only non-negative activations are passed to the spiking layer, which can stabilize membrane potential accumulation and promotes consistent spiking behavior. This helps to avoiding vanishing gradients often caused by negative or highly variable inputs. Moreover, the combination of ReLU-induced static sparsity and the temporal sparsity of LIF neurons may act as a form of temporal filtering, helping the model suppress noise and emphasize important features [10], [11].

Training was performed using the Adam optimizer with the loss function `SF.ce_count_loss()`, which computes the cross-entropy loss based on spike count distributions over the temporal window. Figure 3 illustrates the spiking activity of the output layer over 10 discrete timesteps. The class with the highest number of spikes was selected as the predicted label, which matched the true label of the input image.

Early stopping with patience of 10 epochs was applied during a maximum of 50 training epochs. The model was trained on the training split, validated on the validation split, and tested on the test split.

B. AE-CNN Architecture

We trained an attention-enhanced convolutional neural network using the TensorFlow/Keras framework. The model learns spatial features through convolutional layers and applies a lightweight channel attention mechanism (SE block) to adaptively reweight feature maps.

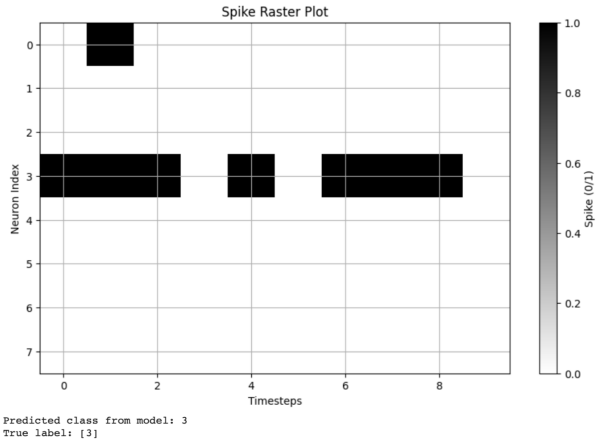

Fig. 3: Spiking activity in the output layer over 10 timesteps for a test sample input. The predicted class corresponds to the one with the highest spike count.

The classification head consists of a global average pooling layer followed by a dense layer (128 units), a dropout layer (rate 0.4), and a softmax layer for 8-class output.

The model was trained on the training subset of the BloodMNIST dataset, using a batch size of 32 for a maximum of 30 epochs. Early stopping was applied with a patience of 5 epochs, restoring the best model weights based on validation loss. The loss function used was sparse categorical cross-entropy, and the optimizer was Adam with a learning rate of 1×10^{-3}. The dataset was split into 70% training, 15% validation, and 15% test sets.

The use of the SE attention module aimed to enhance the model's ability to prioritize informative feature channels, especially in the presence of class imbalance and subtle visual differences between cell types. Dropout was included to prevent overfitting due to the relatively small dataset size.

Block	Description
Conv Block 1	2D Conv (32 filters, 3x3, ReLU, padding='same') + MaxPooling 2x2
Conv Block 2	2D Conv (64 filters, 3x3, ReLU, padding='same') + MaxPooling 2x2
Conv Block 3	2D Conv (128 filters, 3x3, ReLU, padding='same') + MaxPooling 2x2
Attention	Squeeze-and-Excitation (Global Avg Pool, Dense 32, Dense 128)
Conv Final	2D Conv (256 filters, 3x3, ReLU, padding='same')
Global Pool	Global Average Pooling (spatial, 1D)
Dense + Dropout	Dense(128) + Dropout (rate=0.4)
Output	Dense(8), Softmax activation

TABLE 2: AE-CNN network architecture

TABLE 3: Selected hyperparameters for AE-CNN

Hyperparameter	Value
Batch size	32
Learning rate	1×10^{-3}
Dropout rate	0.4
Dense units	128
Image size	64×64

A limited hyperparameter search was performed using KerasTuner to explore dropout rates, dense layer sizes, and learning rates. The selected configuration provided the best validation accuracy and was used for final training.

VI. Results

A. CSNN Model

A hyperparameter search was conducted to optimize training parameters. The selected values are summarized in Table 4.

Hyperparameter	Value
Batch size	64
Learning rate	1×10^{-4}
Timesteps	10

TABLE 4: Selected hyperparameters after optimization for SCNN

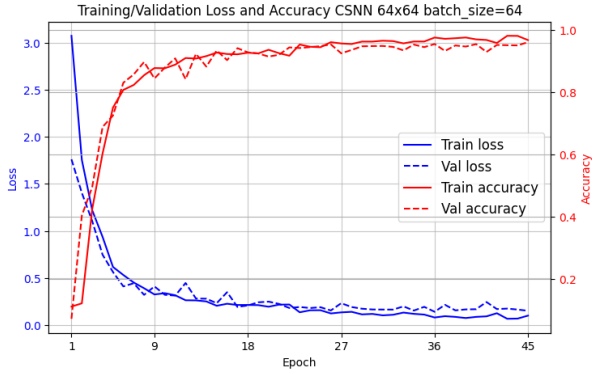

Fig. 4: Training and validation accuracy/loss for SCNN on 64×64 images.

Figure 4 shows the evolution of training and validation accuracy and loss over epochs for the SCNN. The training process required a total of 29.29 minutes, with an average epoch time of **39.05 seconds**. Memory profiling indicated an average of 116.52 MB allocated and 2908.00 MB reserved per epoch. The final model size is 1.2 MB. SCNN achieved a test accuracy of **95.23%** on inputs of size 64×64. When trained on lower-resolution inputs (28×28), the test accuracy dropped to **90%**, but the average epoch time decreased significantly to **13 seconds**, highlighting a trade-off between computational cost and performance.

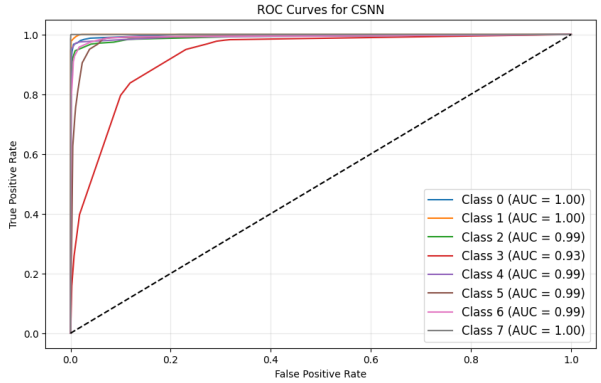

Fig. 5: ROC curves for each class of the SCNN model. While most classes achieve near-perfect AUC values (0.99-1.00), class 3 stands out with a lower AUC of 0.93 due to underrepresentation in the dataset.

In addition, the ROC curve shown in Figure 5 illustrates the per-class discriminative performance of the SCNN. Most classes achieved near-perfect separability, with AUC values of 0.99 or 1.00. However, class 3 exhibited a lower AUC of **0.93**, indicating that most misclassifications occurred in this category. The confusion matrix in Figure 6 shows that class 3 is sometimes confused with classes 5 and 6.

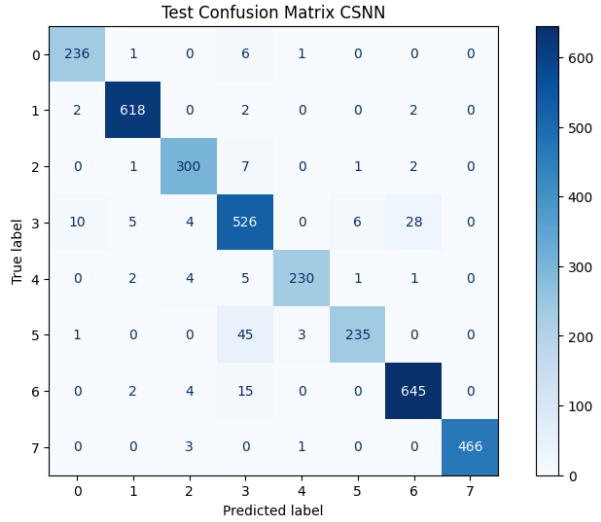

Fig. 6: Confusion matrix of the SCNN on the test set.

Despite this, the overall performance remained strong, with a macro-averaged **F1 score of 0.95**, reflecting the model's balanced behavior across all classes.

The SCNN's training energy was estimated by counting total spikes per epoch in a nanoJoule unit [12]. Energy ranged from 3.57 to 4.51 J/epoch (average: 3.89 J), totaling 147.95 J. Consumption decreased over time as the network stabilized, producing fewer spikes for efficient feature encoding.

At the output layer, the network exhibited highly sparse spiking activity, averaging just 0.02 spikes per neuron per sample (for the classification layer). This means that, on

average, each output neuron fired just once every 50 inputs, which is an exceptionally low rate yet sufficient for accurate classification.

B. AE-CNN Model

Figure 7 shows the evolution of training and validation accuracy and loss for the AE-CNN model over 30 epochs. Training was performed on inputs resized to 64×64 using early stopping with a patience of 5 epochs. The average epoch duration was approximately **180 seconds**, with a total training time of around 90 minutes. The final model size is 5.00 MB.

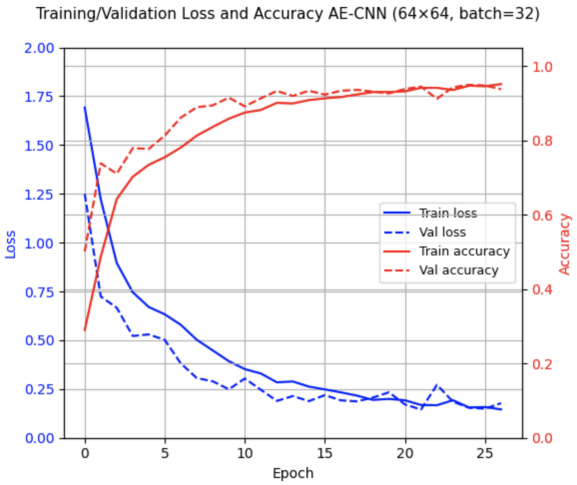

Fig. 7: Training and validation accuracy/loss for AE-CNN on 64x64 images. The model converges smoothly with consistent performance on the validation set.

The model achieved a test accuracy of **94.3%** and a macro-averaged F1 score of **0.935**, indicating robust performance across all eight blood cell classes. Class imbalance was addressed using class weights and offline data augmentation, which helped improve generalization and reduce overfitting.

Figure 8 presents the confusion matrix, showing that the majority of classes were correctly classified. Some minor confusion occurred between visually similar classes, but performance remained consistent overall.

In addition, the ROC curves shown in Figure 9 illustrate the per-class discriminative performance of the AE-CNN. The model achieved AUC = 1.00 for seven out of eight classes, and AUC = 0.99 for class 3, which was underrepresented in the dataset. These results demonstrate high separability and reliability of the network's predictions.

The addition of a channel attention mechanism enabled the AE-CNN to achieve better class separation compared to the CSNN baseline. However, contrary to expectations, the SCNN demonstrated superior classification accuracy (95.23%) compared to the AE-CNN (94.3%) on the BloodCellMNIST dataset. AE-CNN was expected to be faster due to dense computation and attention mechanisms, in practice it was not faster than the SCNN.

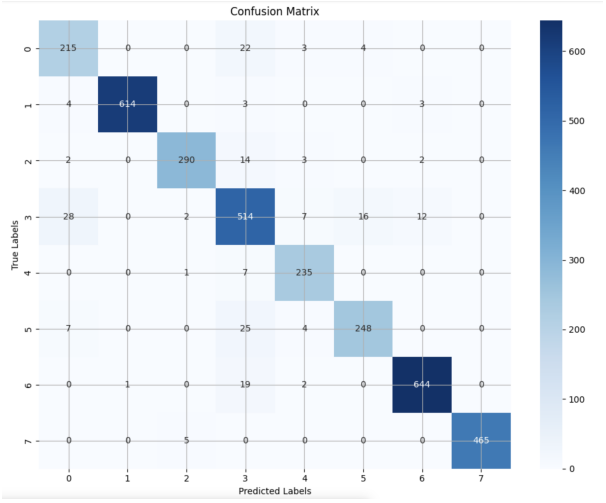

Fig. 8: Confusion matrix for AE-CNN on the BloodMNIST test set. Most classes are predicted with high accuracy. Minor confusion is observed between visually similar cell types.

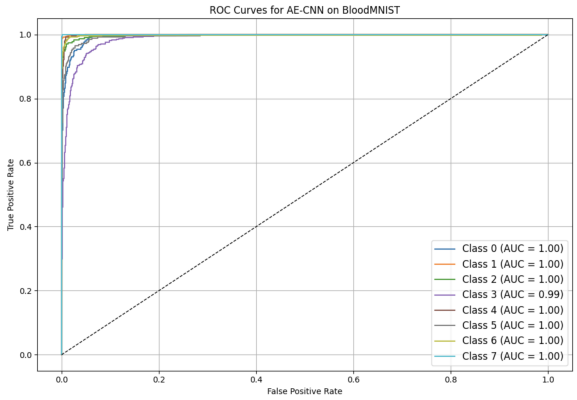

Fig. 9: Per-class ROC curves for AE-CNN on the BloodMNIST test set. The model achieved AUC = 1.00 for seven classes and AUC = 0.99 for class 3.

VII. Concluding Remarks

These results demonstrate that, although SCNNs present implementation challenges, once properly developed and optimized, they can achieve superior classification performance on complex image datasets.

The SCNN model uses a mix of convolutional layers and spiking neurons to process both spatial and time-based patterns. While this helps create more efficient, over-time-sensitive features, it still mostly relies on simple rate-based encoding for static data, which limits the advantages of precise spike timing. As a result, its performance is only slightly better than traditional neural networks.

However, this method proves that spiking neural networks can still work well with static datasets using modern tools like `snnTorch`. By blending standard convolutional networks with spiking neurons, the model gains some benefits, such as better noise resistance and improved handling of imbalanced data.

Future work should address the class imbalance problem more thoroughly. Our initial attempt using class weight balancing had limited effect, partly because the role and application of weights in spiking neural networks differ from conventional networks and require deeper investigation. Trying more advanced learning methods and testing different encoding techniques may unlock further performance improvements of the SCNN model.

The primary challenges stemmed from the diversity of spiking neural network methodologies, including a wide range of encoding schemes and learning rules, each requiring significant effort to implement effectively. Additionally, managing tensor dimensions through the network proved complex and required careful tracking to ensure correctness. These implementation difficulties underscore the field's need for both more user-friendly development tools and standardized methodologies in SNN research.

The AE-CNN architecture demonstrated that integrating a lightweight channel attention mechanism into a traditional convolutional network can substantially boost classification performance, even on small biomedical datasets. The model achieved strong accuracy and F1 scores while maintaining a compact size and fast training time. Unlike the SCNN, the AE-CNN operates entirely on static inputs, without temporal encoding, and converged more easily due to its architectural simplicity.

Together, these two approaches illustrate distinct strengths: SNNs offer biologically inspired robustness and temporal modeling, while attention-based CNNs leverage modern deep learning techniques for efficient and scalable learning. Future work could explore hybrid models that combine attention mechanisms with spiking dynamics, or compare their energy efficiency and deployment feasibility in real-world applications.

REFERENCES

[1] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," 2023.

[2] J. K. Eshraghian, "SNNtorch documentation," 2023. Last access: 2025-06-02.

[3] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," 2018.

[4] N. Kapila, J. Glattki, and T. Rathi, "Cnntention: Can cnns do better with attention?," *arXiv preprint arXiv:2412.11657*, 2024.

[5] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Brain Sciences*, vol. 12, no. 7, 2022.

[6] X. Wang, Y. Rong, Z. Wu, L. Zhu, B. Jiang, J. Tang, and Y. Tian, "Sstformer: Bridging spiking neural network and memory support transformer for frame-event based recognition," 2025.

[7] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] O. Oktay, J. Schlemper, L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *arXiv preprint arXiv:1804.03999*, 2018.

[9] J. Schlemper, O. Oktay, M. Schaap, M. Heinrich, B. Kainz, B. Glocker, and D. Rueckert, "Attention gated networks: Learning to leverage salient regions in medical imaging," *Medical Image Analysis*, vol. 53, pp. 197–207, 2019.

[10] Y. Huang, X. Lin, H. Ren, H. Fu, Y. Zhou, Z. Liu, B. Pan, and B. Cheng, "Clif: Complementary leaky integrate-and-fire neuron for spiking neural networks," 2025.

[11] A. Stanojevic, S. Wozniak, G. Bellec, G. Cherubini, A. Pantazi, and W. Gerstner, "An exact mapping from relu networks to spiking neural networks," 2022.

[12] Z. Yan, Z. Bai, and W.-F. Wong, "Reconsidering the energy efficiency of spiking neural networks," 2024.