# *Supplementary Material*

## 1 SPIKES ACTIVITY PER NEURON FOR EACH LAYER

In this section, we observe the spikes activity per neuron in each layer when using the baseline and the compressed network. The goal is to analyze the effect of the compression on each neuron's spikes activity during the learning phase. For this experiment, we use two convolutional-pooling pairs and SVM to classify the Caltech face/motorbike dataset (400 train images). The first convolutional layer contains 32 neurons, and the second 64 neurons. For the compressed network, we record the following rates: 79% for the first layer and 98% for the second layer. Moreover, we train each layer ten epochs and report the recorded accumulated spikes activity in Figure S1.

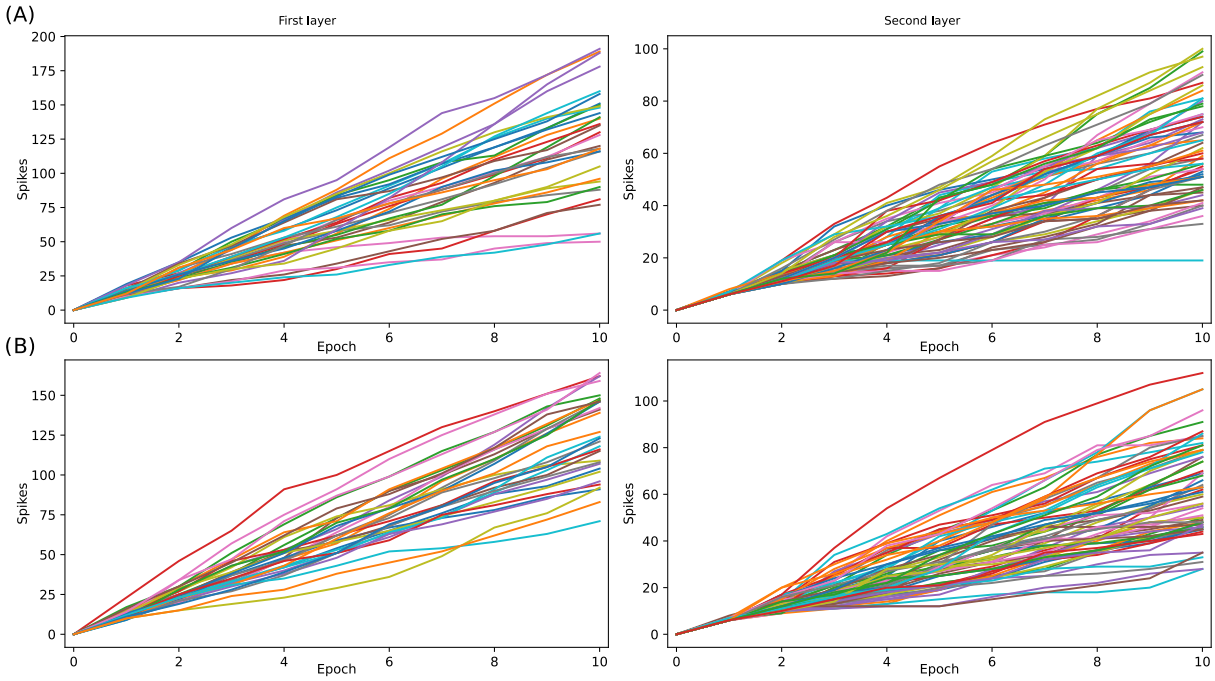



**Figure S1.** Accumulated spikes activity per neuron for the first and second layer, for (A) the baseline and (B) the compressed network.

We can observe from Figure S1 that the spikes activity in all the figures is increasing during training. Moreover, if we compare the activity in both cases (baseline and compressed), we see that the spikes activity is different. In the first layer, we have values between 50 and 190 for the baseline and between 72 and 162 for the compressed network due to the compression. However, it is interesting to see that the minimum value increases in the compressed network due to the threshold adaptation mechanism used in the network. For the second layer, we got spikes values between 20 and 100 for the baseline and between 28 and 110. Therefore, for the second layer, we have a slight increase in the minimum spikes activity due to threshold adaptation and an increase in the maximum spikes activity on the compressed network, which is the opposite of what we had in the first layer. We suspect this is due to the synaptic reinforcement mechanism used with the pruning process.

## 2 NEURONS THRESHOLD EVOLUTION ON THE CSNN SIMULATOR

Due to the importance of the threshold adaptation mechanism used in our network, we analyze the evolution of the neuron's threshold during training in this section. We keep the same network architecture with the same compressed rate for this experiment and use the same dataset. In Figure S2, we can see the activity of the two layers for both networks.

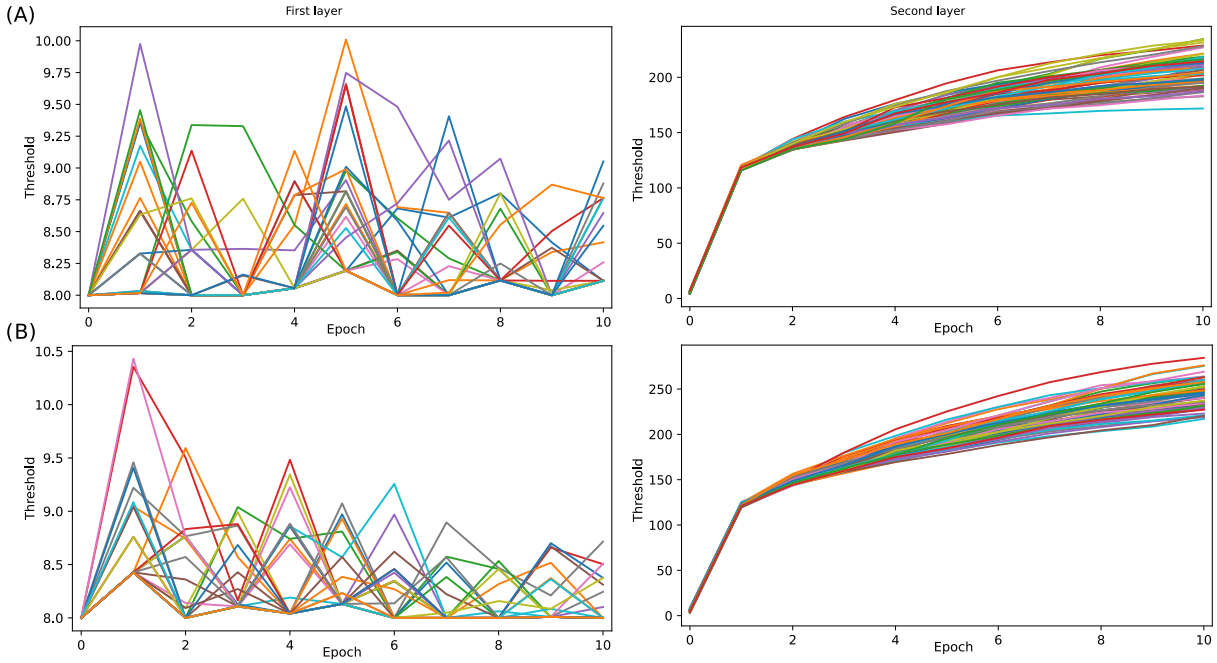



**Figure S2.** Neurons threshold evolution of the first and second layer during training, for (A) the baseline and (B) the compressed network.

Interestingly, the neuron's threshold dynamic in the first layer differs from the second in both use cases because the first layer is exposed to the inputs directly after preprocessing. In contrast, the second layer receives data processed by the first convolutional and pooling layer. In the simulation, there is a $th_{min}$ value, which represents the minimum possible threshold value, and we set it to 8 for the first layer and 1 for the second layer. Comparing the first layer in both cases, we see that the threshold values increase during training. In the baseline, the maximum value is ten, and in the compressed one, it is a little bit higher. Furthermore, after epoch 4, we can see that in the compressed network, the thresholds are lower and stable between 8 and 8.7. Yet, in the baseline, the values kept changing the same way. In the second layer, we can see that the neuron's threshold is much higher after ten epochs than in the first layer. In the baseline, the values are between 170 and 238. In the compressed one, the values are between 220 and 280. Therefore, the increase in the neuron's threshold value is due to the compression and the synaptic reinforcement applied, which affects the threshold adaptation mechanism and causes it to increase the threshold to preserve the required activity.

## 3 WEIGHTS DISTRIBUTION IN CSNN SIMULATOR AND SPINNAKER

In this section, a network of two fully-connected layers of 400 and 1600 neurons is trained with the CSNN simulator. After that, we upload the trained weights on the SpiNNaker board. Figure S3 and Table S1 show the weight distribution for the two use cases in both layers.

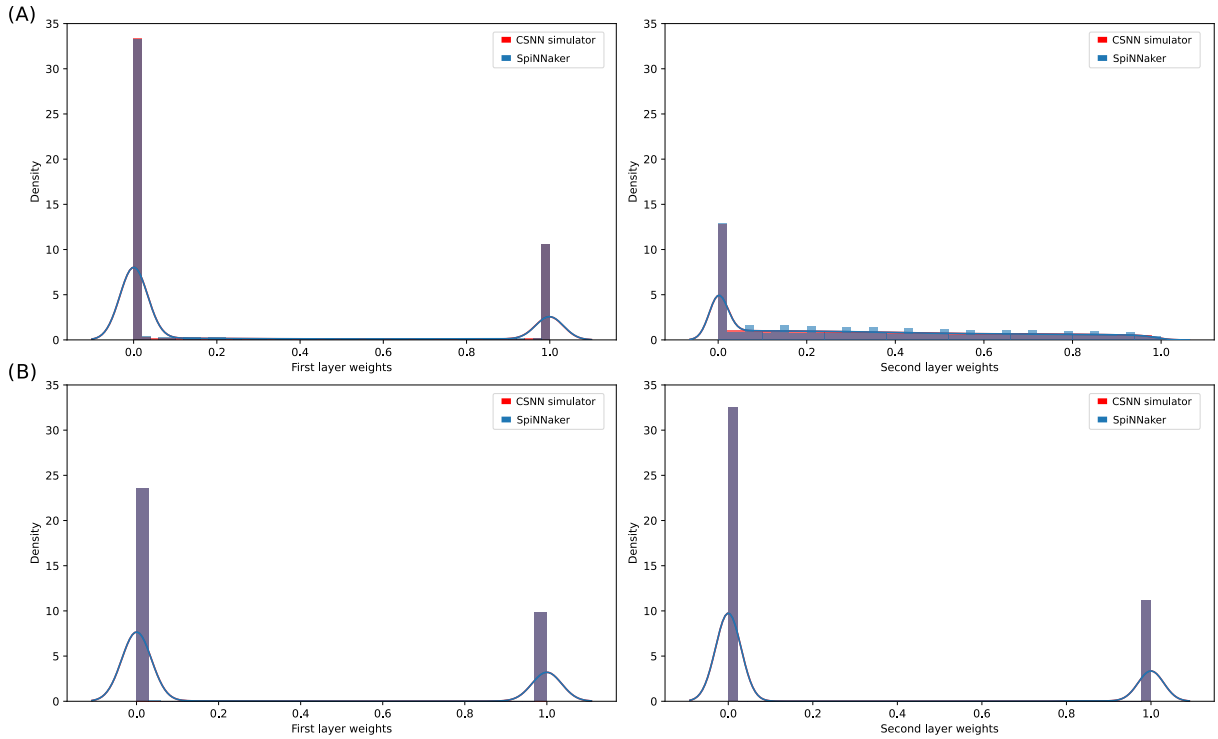



**Figure S3.** Trained weights distribution in CSNN simulator and SpiNNaker for (A) the baseline and (B) the compressed network.

**Table S1.** Weights distribution in CSNN simulator and SpiNNaker

| | | Baseline | | | Compressed | | |
|---|---|---|---|---|---|---|---|
| | Weights | 0 | $0 < w < 1$ | 1 | 0 | $0 < w < 1$ | 1 |
| CSNN simulator | L1 | 202729 | 59606 | 51265 | 209143 | 13902 | 90555 |
| | L2 | 148253 | 491325 | 422 | 463459 | 15010 | 161531 |
| SpiNNaker | L1 | 208611 | 38676 | 66313 | 217177 | 5632 | 90791 |
| | L2 | 154828 | 483965 | 1207 | 472103 | 5912 | 161985 |

According to Table S1 in the first layer of the baseline, a considerable amount of the weights are zero, and the rest are distributed between zero and one. In the compressed network, %76.68 of weights from the middle interval ($0 < w < 1$) are pushed close to either zero or one, as we can see in Figure S3. For the second layer of the baseline, most of the weights are between zero and one. Moreover, the weights are near zero and one for the compressed network, which is the expected result after compression. For the SpiNNaker board, we extract the weights from the board after uploading them to see if they are unchanged. As shown in Figure S3, the extracted weights are in the same distribution. However, since ARM968 has no specific hardware for floating points, SpiNNaker supports up to 32-bit fixed-point precision. Therefore, the stored weights in SpiNNaker are slightly different, as shown in Figure S3.

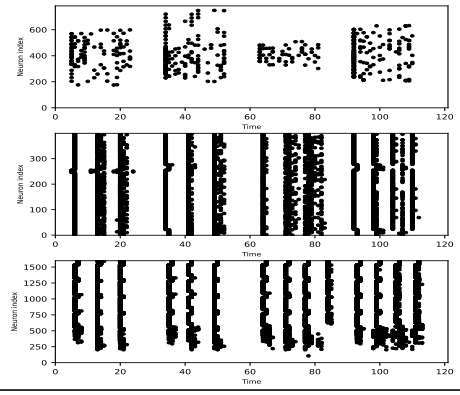
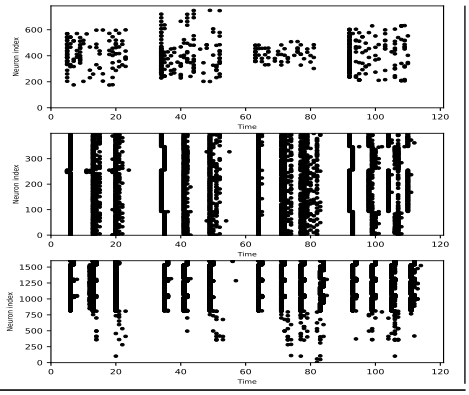
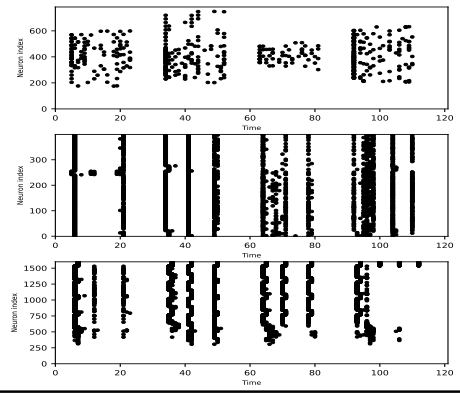
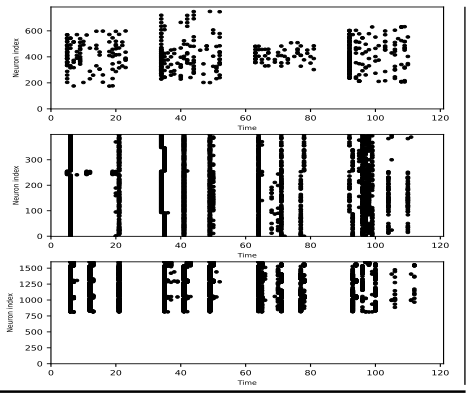
## 4 REDUCING THE SPIKES ACTIVITY IN SPINNAKER

As we saw in the manuscript, the SpiNNaker board generates many spikes, which may affect the classification task. Therefore, there are different possibilities to reduce it. This section discusses two of them: the use of inhibition and neuron parameter tuning.

## 4.1 Adding inhibitory neurons

Inhibition is essential for the STDP learning rule, which helps the network prevent neurons from learning similar features. In Table S2, we can see the effect of using the inhibition on the baseline and the compressed network when presenting four digits from MNIST.

**Table S2.** Spikes activity per layer with/without inhibition

| | Baseline | Compressed |
|---|---|---|
| Without inhibitory neurons |  |  |
| With inhibitory neurons |  |  |

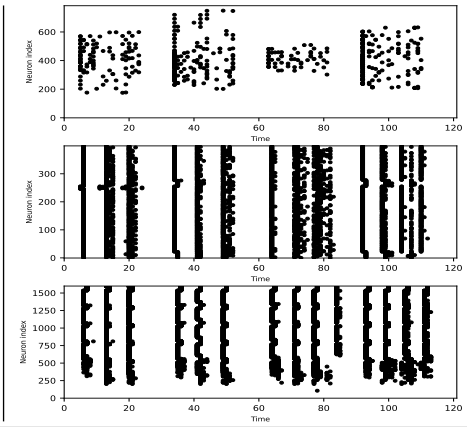
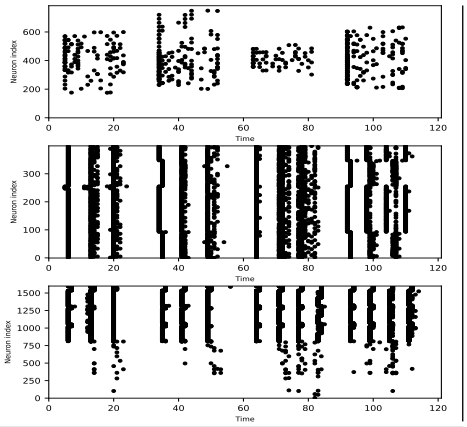
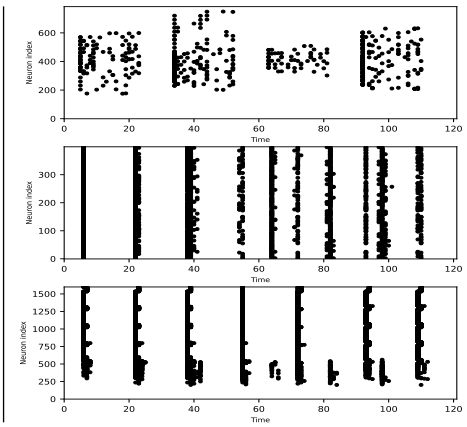
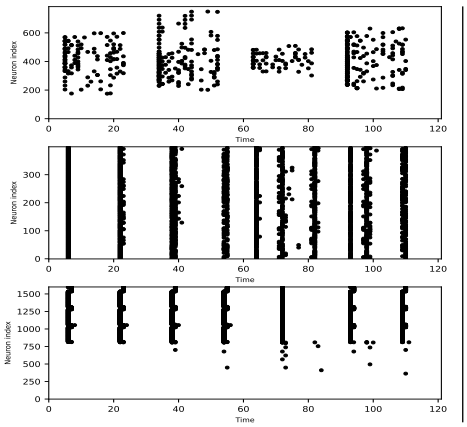
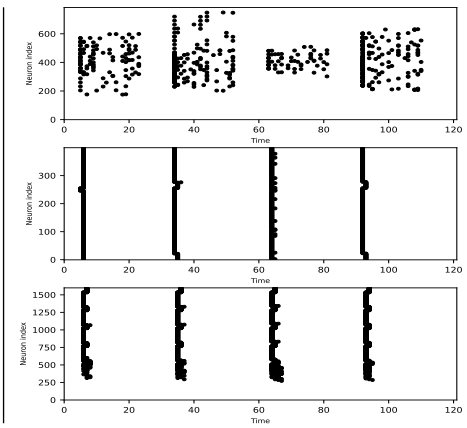
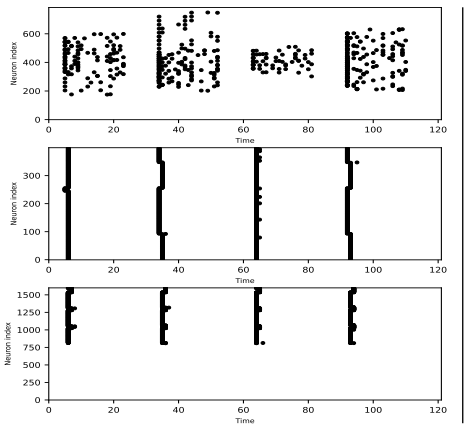
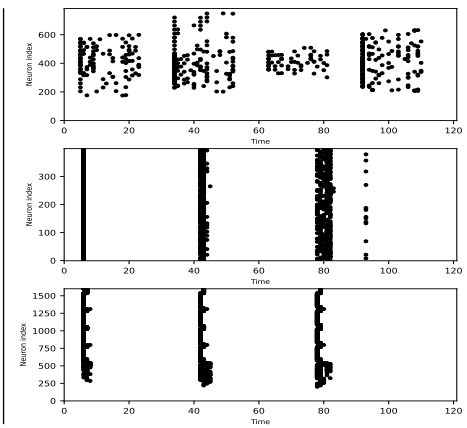
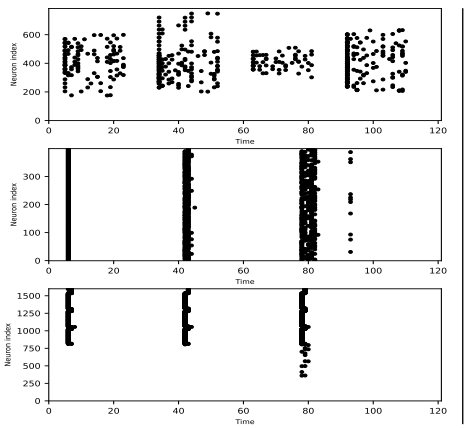
We can see from Table S2 that we have fewer spikes when using the inhibition. Furthermore, half of the neurons in the second layer are not firing when using compression.

## 4.2 Neural parameters tuning

We can prevent the neurons from generating more spikes by tuning the neural parameters. In this section, we test different values of the refractory period, as shown in Table S3. In our experiment, we present four samples, each for 20ms, and the resting time is 10ms. Therefore, the refractory period $\tau_{refrac}$ in our experiment equals 5ms, 15ms, 25ms, and 35ms.

As shown in Table S3, by increasing $\tau_{refrac}$ we have fewer spikes, and still, the neurons are triggered with the corresponding inputs. However, as we can see in the same table, higher values of $\tau_{refrac}$ (35 ms) affect the network responses, and we don't have any spikes for the last input. As we mentioned, other parameters ($v_{th}$, $v_{rest}$, $v_{reset}$) have also an effect on the neuron activity. In our future work, we will explore other parameters to find the best configuration for the neuron parameters.

**Table S3.** Spikes activity per layer with different refractory period (5ms and 15ms)

| Refractory period | Baseline | Compressed |
|---|---|---|
| 5ms |  |  |
| 15ms |  |  |
| 25ms |  |  |
| 35ms |  |  |