Politecnico di Torino

III Facoltà di Ingegneria
Master's Degree in Electronic Engineering

Master's Degree Thesis

Security of Event-Based Spiking Neural Networks: Attacks and Defense Methodologies

Supervisors

Prof. Maurizio Martina

Prof. Guido Masera

Prof. Muhammad Shafique

Project Ass. Alberto Marchisio

Candidate

Giacomo Pira

Acknowledgements

Now, at the end of my university career, I feel the duty to spend a few lines to express my gratitude to all the good and caring people who have been and still are close to me. First of all I have to thank Prof. Maurizio Martina, Prof. Guido Masera and Prof. Muhammad Shafique, for giving me the opportunity to work on an extremely interesting and innovative topic for my thesis. A special thanks goes to Alberto Marchisio, who was always available during these months, offering me help whenever I needed it. I cannot fail to mention my family, especially my parents and my sisters, who have supported me all my life: a simple thank you will never be enough to repay everything you have done for me. You raised me and made me who I am. Finally, how could I forget my friends, with whom I have shared some of the best moments of my life. I cannot quote you all, but you must know that as I write these lines I am addressing a thought of gratitude for each one of you. Being your friend is a blessing. I conclude by mentioning my dear friend Luca, who has been much more than a simple fellow student during all these years at Politecnico. After countless projects, papers and exams I can say for sure that without your help I would never have made it. Sharing my path with such wonderful people has been an honor and a priviledge. Once again, thank you. Life is more beautiful because of you.

Thank you all!!

Summary

Spiking Neural Networks (SNNs) are the third generation of Neural Networks and they are rising in popularity in the scientific community. Inspired by the functionality of natural Neural Networks such as the brain, they are energy efficient and biologically plausible due to their event-based computation. Nonetheless, they remain vulnerable to security threats, such as *adversarial attacks* (i.e. imperceptible perturbations added to the input whose purpose is to reduce the SNNs' accuracy). The purpose of this thesis is to investigate the effects of adversarial attacks on SNNs and their defense counter-measurements.

In our study, we focus on SNNs deployed for action and image recognition purposes. Such SNNs are usually fed through Dynamic Vision Sensors (DVS), which are neuromorphic versions of traditional frame-based cameras. A DVS is inspired by the human retina and works with events, similarly to the SNNs. They offer several advantages in terms of performances with respect to traditional cameras, and being event-based are easy to couple with SNNs implemented on neuromorphic systems. We thus analyze which are the main sources of noise that affect this kind of sensors, and which are the most common filters used against them. Among the several examples that we found in the literature, we focus on two: *the Background Activity Filter* and *the Mask Filter*. Moreover, we want to analyze whether the use of such common filters, developed not with the aim of countering attacks, but only to reduce noise, is also able to reduce the effectiveness of the attack and to restore the SNNs' accuracy to original levels. For this purpose, we develop a series of stealthy adversarial attacks, which are able to drastically reduce the SNNs' accuracy, and we analyze in detail which are the effect of those filters for classification applications, testing them on two different neuromorphic

datasets: the *IBM DVS128 Gesture* and the *NMNIST*.

Our experiments show that depending on the attack, in many cases the filters are able to restore the clean SNNs' performance, i.e., their accuracy at pre-attack levels. Not all the filters successfully work with every attack, and some work better than others. For example, the development of a filter-aware attack leads to a reduction of up to more than 20% of the SNNs' accuracy, when dealing with the *IBM DVS128 Gesture*, and to an even more significant 65% with the *NMNIST*, demonstrating once again the importance of developing effective defense systems.

Contents

List of Figures

List of Tables

Acronyms

BAF Background Activity Filter.

DNNs Deep Neural Networks.

DVS Dynamic Vision Sensor.

LIF Leaky Integrate and Fire.

MF Mask Filter.

NN Neural Network.

SNNs Spiking Neural Networks.

CHAPTER 1

Introduction

Spiking Neural Networks (SNNs), aim at providing an energy-efficient alternative to traditional Deep Neural Networks (DNNs), in an even wider spectrum of machine learning applications. They are already deployed in various fields, including autonomous driving [1], healthcare [2], and robotics [3]. SNNs implement an event-based communication system among neurons, which is inspired by the human brain. For this reason, they are considered biological plausible and they provide powerful tools for analysis of elementary processes in the brain, such as neural information processing, plasticity and learning [4]. In addition, they are a valid solution in resource-constrained embedded systems, given their power/energy efficiency and real-time classification performance [5]. Due to their reduced computational load and low latency, SNNs achieve better results than their traditional DNN counterparts [6]. The development of neuromorphic architectures, such as IBM TrueNorth [7] and Intel Loihi [8], allows to exploit SNNs' characteristics to the fullest, implementing an efficient asynchronous communication mechanism between neurons and the event-based propagation of the information through layers. SNNs are often associated with Dynamic Vision Sensor (DVS), innovative neuromorphic optical sensors, which are inspired by the human retina and allow for higher performance than traditional frame-based cameras [9]. Moreover, each pixel works independently from the others, generating asynchronous events every time a change in local brightness is recorded. Information is then encoded using Address Event Representation (AER), a communication protocol for transferring spikes between

bio-inspired chips. [10]. Being both event based, SNNs and DVS are easily coupled, in fact they can be directly connected, allowing the SNNs to process in real-time the images produced by the DVS.

1.1 Scientific Challenges

In recent years, the merits of SNNs in providing low-energy and low-latency learning systems have brought them to the attention of the scientific community, which has begun to study their properties and applications. While they have the above-discussed advantages over traditional DNNs, they suffer from similar safety hazards. In fact, as some recent studies have shown, SNNs are also vulnerable to some security threats, such as adversarial attacks, i.e. imperceptible perturbations that are added to the inputs to induce a misclassification. The nature of these attacks, already widely documented and studied in the case of DNNs, is a relatively new topic when it comes to SNNs, for which only a few examples are to be found in literature [11] [12] [13] [14]. The research, although in its early stages, has however shown that these attacks pose a real threat, especially if perpetuated against SNNs used in safety-critical applications. Moreover, to the best of our knowledge, no one has yet analyzed these types of threats in a DVS-feeded system, as we propose to do in this thesis.

In parallel with the attacks, some defense methodologies were also presented, such as adversarial learning algorithms [15] and loss/regularization functions [16]. Also in this case, the research is much more advanced for DNNs, while it has very few publications in the case of SNNs. Another method, that proved to be effective in countering adversarial attacks in the case of DNNs, is image preprocessing [17], which consists essentially in the use of noise filters to counteract the perturbations introduced by the attack. No one, as far as we know, has ever investigated this defense strategy in SNN-based systems feeded by DVS, which poses an interesting challenge, even more so considering that traditional filters cannot be used, as they are incompatible with the DVS output. For this reason, dedicated filters especially designed for DVS cameras, such as those presented in [18], must be used. Furthermore, it is also necessary to

understand that, unlike traditional sensors, the output of the DVS constitutes a flow of events, which in addition to containing spatial information, also has a temporal characteristics. Traditional attack methods must therefore be rethought, including this feature, if they want to be successful in undermining the SNNs'confidence. The same settings must be considered also for the defense methodologies. For this purpose, we have developed a series of attacks, which directly affect the flow of events produced by the DVS, and which constitutes the input to the SNN. Moreover we have tested them in the presence of filters, used as a defense mechanisms.

1.2 Our Novel Contributions

In this study, we aim at addressing the aforementioned research challenges, providing the following novel contributions:

- We introduce the problem, providing an analysis of the main actors involved. **(see Chap. 2 - 3)**;
- We present two types of noise filters for DVS, the Background Activity Filter (BAF) and the Mask Filter (MF), explaining their functionality and implementation. We also verify their impact on the clean input, i.e. the original dataset without external perturbations. **(see Chap. 4)**;
- We are the first, to the best of our knowledge, to analyze the impact of normally and uniformly distributed noise injection on a SNN systems feeded by a DVS. We also investigate the effect of filters on this noisy dataset **(see Chap. 5)**;
- We generate an attack based on the calculation of gradients, and we verify the impact of the Background Activity Filter in three different threat models, varying their position within the system and assuming a different level of knowledge on the part of the adversary. **(see Chap. 6 - 7)**;
- We design a new series of attacks, following an innovative approach, and we verify their effectiveness in the presence of two different types of filters, the Background

Activity Filter and the Mask Filter. **(see Chap. 8 - 9)**. As far as we are aware, no other work carries out a similar analysis.

– We develop a novel attack aware of the presence of filters and verify its resistance to the above-discussed defense methods.**(see Chap. 10)**.

CHAPTER 2

Background

2.1 Spiking Neural Networks

Spiking Neural Networks (SNN) are the third generation of neural networks [19]. They are inspired by the natural neural systems and for this reason offer a more truthful representation of the human brain. Unlike traditional artificial neural networks, SNNs also include the concept of time in their model: they actually encode information in the form of impulses that propagate from one neuron to another, just like in biological nervous systems. Each neuron is characterized by a membrane potential, an electrical quantity linked to the neuron's electrical potential, and by a threshold that, only if exceeded, allows the neuron to produce an impulse and to propagate information. Neurons, therefore, receive a series of impulses over a certain period of time, which can increase or decrease their membrane potential, depending on the weight of every synapses, and allow them to fire an impulse in turn. There exist several methods to encode information in the form of spike train, among them the most common are rate coding and time coding [20]. The first simply transforms the continuous numbers into more or less frequent pulses depending on the original value, while the second represents the values via the latency between spikes or the time-to-first-spike after stimulation. SNNs can also be classified according to the model they adopt to describe the neuron. The most common are the Hodgkin-Huxley Model [21], very accurate but also computationally expensive, and the Leaky Integrate and Fire Model [22], which is

simpler and not biologically plausible, but allows to process a larger number of neurons. For this reason it is the most used and it is already implemented in actual neuromorphic chips.

2.1.1 Leaky Integrate and Fire Model (LIF)

The most used spiking neuron model is the leaky integrate and fire model. As the name suggests the LIF neuron integrates the incoming spikes, appropriately weighted, and then it fires an output impulse when the membrane potential overcomes its threshold. It is called 'leaky' because it also models a leakage that makes the potential decrease continuously. After firing, the potential is reset to a lower value. The mechanism just explained is summarized in the following equation:

$$\tau_m \frac{dV_m}{dt} = -V_m + I(t), \tag{2.1}$$

where V_m is the membrane potential and τ_m is the time constant for the membrane potential leakage.

2.1.2 Why SNN?

Artificial Neural Networks (ANNs) are now the de facto standard when it comes to Machine Learning. Several articles have been written about it, there are different training methods that have achieved excellent results and they are already reach a good level of production. In this context, where are the SNNs located and why can they be considered a valid alternative? SNNs are used in the same fields as other ANNs, for example image recognition or object tracking. Moreover, given their intrinsic similarity to biological networks, they can offer a good starting model for understanding some aspects of the brain that are still not clear. SNNs are relatively new and for this reason the research is still in its early stages, offering some challenging but also very interesting topics. In any case, in addition to the aforementioned similarity with biological networks, which could lead to interesting discoveries also in the medical and physiological field, the SNNs have also other key advantages over the ANNs, in fact they are faster and have a much lower power consumption thanks to their discrete

nature. Today, one of the most common problems that engineers have to face is the reduction of power consumption, which is becoming even more important if we consider the diffusion of networks in mobile devices. For this reason SNNs can provide a valid alternative, even if they are more difficult to train with backpropagation, due to the non-differentiability of their spiking function. Furthermore, as demonstrated by Marchisio et al. in [12] and [23] they are more resistant to possible attacks, which is the ultimate purpose of this thesis.

2.1.3 SlayerPytorch

In this work all the networks were trained and tested using *SlayerPytorch*: a PyTorch port of the original SLAYER framework [24]. SLAYER stands for Spike LAYer Error Reassignment and it offers a training method that resembles the traditional back-propagation algorithm and that is able to distribute the error among all the layers that composed the network. In addition, SLAYER also takes care of redistributing the errors back in time because a spiking neuron's current state depends on its previous states (and therefore, on the previous states of its input neurons). Thanks to this method it is possible to obtain excellent results in image recognition, comparable to those obtained by ANNs trained with back-propagation. SLAYER allows training both fully connected and convolutional neural network (CNN) architectures.

2.2 Adversarial Attacks

Adversarial machine learning is a machine learning technique that attempts to fool models by supplying deceptive inputs. The most common reason is to cause a malfunction in a machine learning model. They were first described by *C. Szegedy et al.* in 2014 [25] and have been the focus of many studies and articles ever since. They found out that it is possible to "cause the network to misclassify an image by applying a certain imperceptible perturbation, which is found by maximizing the network's prediction error. In addition, the specific nature of these perturbations is not a random artifact of learning: the same perturbation can cause a different network, that was trained on a different subset of the dataset, to misclassify the same input" [25]. With the spread of neural networks in new areas, many of which characterized by an extreme attention to security and safety, the study of adversarial attack has increasingly attracted the attention of the scientific community. For example, it is easy to imagine what risks could be posed to people's safety if such an attack were directed against a network mounted on a self-driving car. The purpose of an attack of this kind is to undermine a neural network by introducing a perturbation that is imperceptible to the human eye, but which for some reason is not indifferent to the network and leads to a significant decrease in its performance. Often this attacks are directed against networks trained in image recognition. In this case, the purpose of the attack would be to significantly reduce the accuracy achieved by the neural network. An attacker could insert a tiny and imperceptible perturbation, not noticeable to the naked eye, but which for some reason causes the accuracy of the net to drop well below the one obtained originally. In the same way, the attacker could, with an appropriate attack, make sure that some objects belonging to a certain class are always recognized as members of another specific class. In the latter case, we refer to targeted adversarial attacks. Another distinction that can be made to classify this type of problem is to consider the attacker's knowledge of the system under attack. If the knowledge is null, except for the interface with the outside, that is, if there is no knowledge on the internal structure of the network, we refer to Black Box Attacks, as the network is modelled as a black box. On the

contrary, if the attacker has knowledge of the internal structure, for example of which layers make up the network and how they are structured, then we refer to White Box Attacks. It is clear that a White Box Attack is more likely to be effective, because it can take advantage of a deeper understanding of the model under attack and therefore can exploit some known weakness. A Black Box Attack, on the other hand, although more difficult to implement, it is much more general, as it does not depend on any network parameter.

2.2.1 Attack Against SNNs

As discussed above, adversarial attacks have been a rather popular topic in the scientific community in recent years. Nevertheless, most of the examples found in the literature refer to attacks conducted against traditional ANNs, while there are few references to SNNs, a symptom of the fact that this topic of study is relatively new. Some of them take the samples to attack and convert them to a normal, non-neuromorphic dataset. That is, they do not use spikes, but convert the image into traditional frames and work with a traditional DNN. After, they convert back the adversary input and feed it to an SNN. For example this is done by *S. Sharmin et al.* in [26]. However, this type of attack is not very effective, as it is a mere adaptation of pre-existing attacks, meant to be used with DNN. Others follow a better approach, which allows you to implement a type of attack that also makes use of a sort of back-propagation to achieve excellent results [27] (remember that back-propagation is not directly applicable to SNN due to non-differentiable nature of the spike function). A further example is [28] , where the authors introduced a cross-layer attack that threatens the SNNs integrity. In their work they used an hardware Trojan to launch effective fault-injection attacks.
Another problem that may rise when attacking a SNN regards its inputs. In fact, one of the most intriguing challenges is that the inputs to a SNN, being impulsive in nature, also include a time factor. The images that are processed by the network are therefore 'videos' in all respects and must be considered as such. For this reason, in making our attack we were inspired by the, albeit few, examples of adversarial attacks against videos. In particular in [29], the authors demonstrate that a successful attack can be

perpetuated also acting on a a reduced set of frames, exploiting what they call ‘sparsity’, namely the fact that the adversarial perturbations are temporally sparse. This leads to several advantages, both in terms of computation cost, which is greatly reduced, and imperceptibility, which, on the contrary, increases. In addition to ‘sparsity’, they exploit also ‘propagation’, which is the ability of perturbations to be transferred to the frames adjacent to the one to which they are applied, via their temporal interactions. The combination of these two characteristics allows to create efficient attacks, which register a high fooling rate, even when the perturbation is added to only one frame.

2.3 Dynamic Vision Sensors

The dynamic vision sensor (DVS) is the first camera based on events. It was first introduced by Patrick Lichtsteiner and al. in [9]. These sensors are inspired by the biological retina, and therefore they are also know as neuromorphic cameras or silicon retinas. Unlike traditional cameras that use a shutter to capture an entire image within a regular interval of time, DVS cameras works in an event based way. Every pixel of a neuromorphic camera represents an entity of its own, completely independent from the others. Whenever a pixel detects a local variation in brightness, that is higher than a prefixed threshold, it produces an event. This leads to a considerable reduction of redundant operations, as opposed to what happens with traditional cameras where each pixel is processed in any case, even if it has not registered any variation compared to before.

Modern event cameras have microsecond temporal resolution (10 μs), 120 dB dynamic range, and less under/overexposure and motion blur than frame cameras [30]. In addition, their response is stable under a wide range of illumination, since each pixel only reacts to the variation it registers itself. A traditional camera, to reach the same temporal resolution of 10 μs, should take 100k frames per second, which would require a very large computational effort and a memory capable of holding and processing all this data. DVS, instead, being event-based are able to reach this high speed, while maintaining a low throughput.

2.3.1 Address Event Representation AER

AER is a way of representing information in the form of events. As mentioned, every time a pixel registers a variation in brightness that exceeds a certain threshold, this produces an event. This way of communicating through asynchronous impulses is reminiscent of that of the human brain, a system to which this kind of sensor refers. The impulses that propagate in the brain have a binary nature: they are binary digital event that occur in analog time. In the case of sensors, on the other hand, the concept of event can be generalized and expanded, to allow the level of information contained

in any event produced by the sensor to be increased. In particular, the spikes are represented using their addresses, either of source or destination. In this representation each event includes at least 4 pieces of information:

- x, the x-coordinate of the pixel (who generated the event),
- y, the y-coordinate of the pixel,
- t, the timestamp of the event,
- p, the polarity of the event (increase or decrease of brightness).

It can also contain some extra notion such as edge orientation, object category, etc. Using AER, chips can communicate ensembles of neural information in a sparse, data-driven manner, resembling the brain activity. In this way, while maintaining the impulsive nature - the event exists or does not exist - it is possible to have a lot of useful information to clarify who generated the event, when it was produced and what nature it was. All while maintaining a compact and easily usable representation.

2.3.2 Noise affecting DVS

The DVS, as all kind of sensors, are affected by noise. There are two main types of noise affecting Dynamic Vision Sensors [18]:

- Background Activity Noise (BA),
- Hot Pixel Noise (HP).

The Background Activity noise is due to thermal noise and junction leakage current and it causes an output even if there is no change in the light intensity, while the Hot Pixel Noise consists in a pixel that continuously outputs events, since it can not reset properly. It affects all those pixels that, due to transistor mismatch, have low temporal contrast thresholds and thus high spontaneous noise activity. Both of them lead to a reduction of the image quality and also of performance since they produce false events that occupy bandwidth and consume power.

CHAPTER 3

Networks and Datasets

All experiments are performed on two different datasets and consequently on two different neural networks. Both networks have been developed, trained and tested in the SLAYERPytorch environment.

3.1 IBM DVS128 Gesture Dataset

The DVS128 Gesture dataset comprises 1,342 instances of a set of 11 hand and arm gestures, grouped in 122 trials collected from 29 subjects under 3 different lighting conditions. During each trial one subject stood against a stationary background and performed all 11 gestures sequentially under the same lighting condition. The gestures include hand waving (both arms), large straight arm rotations (both arms, clockwise and counter-clockwise), forearm rolling (forward and backward), air guitar, air drums, and an "Other" gesture invented by the subject. The 3 lighting conditions are combinations of natural light, fluorescent light, and LED light, which were selected to control the effect of shadows and fluorescent light flicker on the DVS128. Each gesture lasts about 6 seconds. To evaluate classifier performance, 23 subjects are designated as the training set, and the remaining 6 subjects are reserved for out-of-sample validation [31].

In order to use the dataset with our networks, it is necessary to proceed with a preprocessing: first, we divide the different gestures performed by every subject into the

respective classes, then we shorten the duration of the single gesture from 6 s to 1.5 s. As suggested in the paper, the first 23 subjects are used for the *Training* set, and the remaining 6 subjects for the *Tesingt* set. This results in 1078 training samples and 264 testing samples.

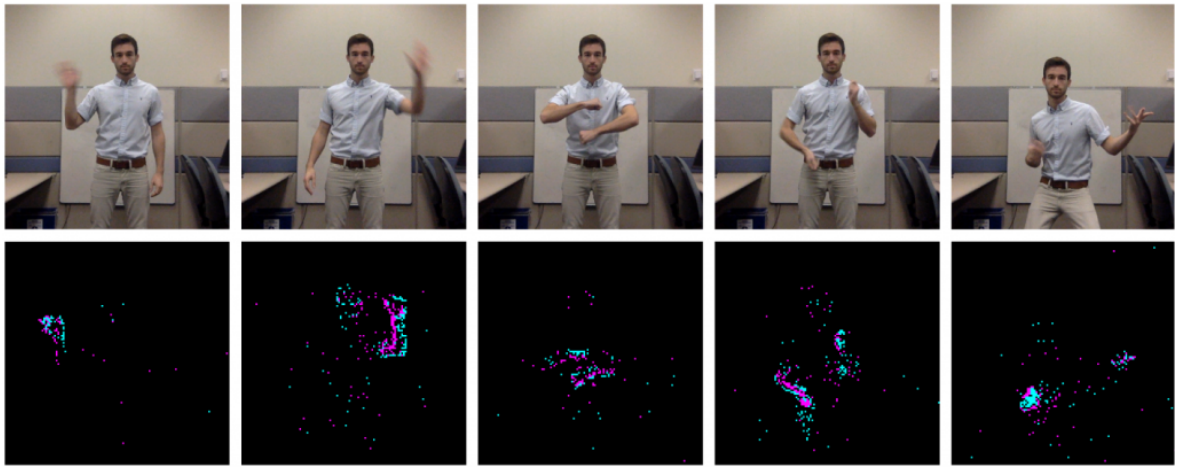

Figure 3.1: DVS128 Gestures: the figure shows 5 different gestures (right hand wave, left hand wave, arm roll, air drums and air guitar), recorded with a traditional (up) and a DVS camera (bottom). DVS pixels emit separate ON (magenta) and OFF (cyan) events to encode an increase or decrease in pixel illumination. The background is automatically eliminated.

3.1.1 Network

In our experiments, we consider the 7-layer SNN as described in [24], with two convolutional layer and two fully-connected layers, plus 3 pooling layers used to reduce the data dimensions. We implement it on a ML-workstation with two Nvidia GeForce RTX 2080 Ti GPUs, using the PyTorch framework [32] and train it for 625 epochs with the SLAYER backpropagation method [24], using a batch size of 4 and learning rate equal to 0.01. After the training phase, the test accuracy of the network reaches 92.04%.

Layer	Output shape
Input	(128,128,2)
Pool1	(32,32,2)
Conv1	(32,32,16)
Pool2	(16,16,16)
Conv2	(16,16,32)
Pool3	(8,8,32)
Dense1	512
Dense2	11

Table 3.1: Structure of the convolutional SNN used with the DVS Gesture

3.2 NMNIST

The MNIST is one of the most popular machine learning datasets, its made of images of hand-written digits. It is one of the standard benchmark used to evaluate the performance of a neural network, when it comes to image recognition. Since we are working with a SNN, we use the NMNIST dataset, which is the neuromorphic version of MNIST. The NMNIST dataset consists of MNIST images converted into a spiking dataset using a Dynamic Vision Sensor (DVS), mimicking the movements of the human eye (i.e. saccadic movements). The NMNIST dataset was captured by mounting the ATIS sensor on a motorized pan-tilt unit and having the sensor move while it views MNIST examples on an LCD monitor [33] The resulting dataset is made up of 34 x 34 images, with a duration of 350 ms, containing both ON and OFF spikes. The original dataset is made up of 70,000 images (60000 of training and 10000 of tests), but we adopt a smaller version made of 1000 training and 100 testing samples.

3.2.1 Network

The network in this case is different from before: we employ a multilayer perceptron (MLP) with two fully-connected layers [24], trained for 350 epochs with the SLAYER

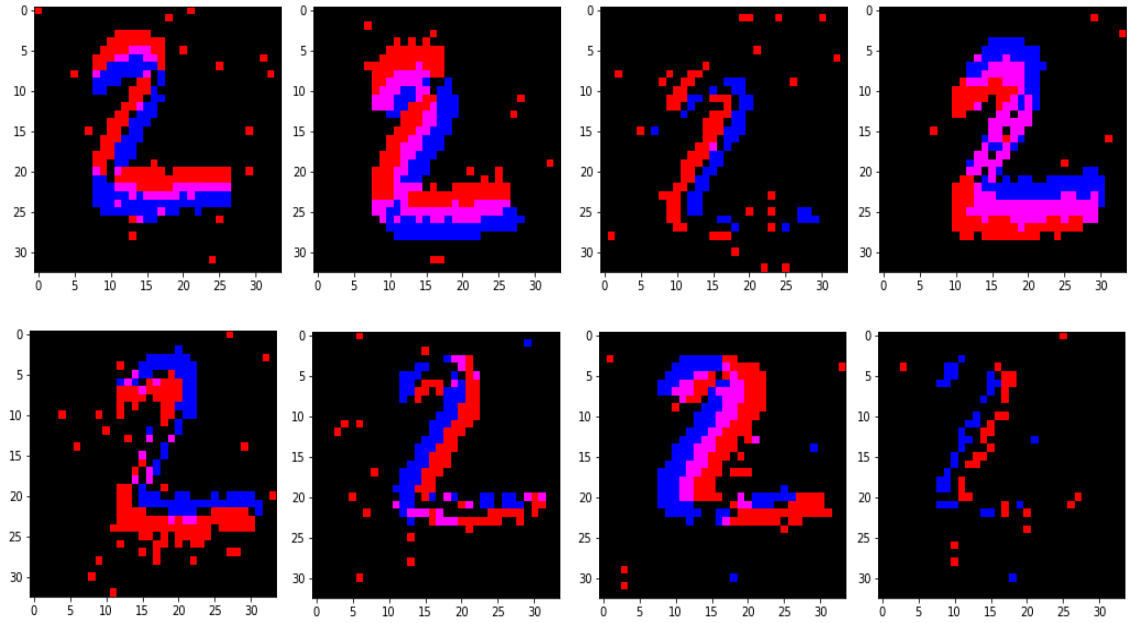

Figure 3.2: A sample from NMNIST spannig over 350 ms. It is possible to notice ON events (magenta) and OFF events (cyan). Whenever they are superimposed the image is pink.

backpropagation method [24], using a batch size of 4 and learning rate equal to 0.01. We reach a test accuracy on clean inputs of 95%.

Layer	**Output shape**
Input	34x34x2
Dense	512
Dense	10

Table 3.2: Structure of the Multilayer Perceptron network used with the NMNIST

CHAPTER 4

Noise Filters for Dynamic Vision Sensors

Several examples of anti noise filters for DVS are present in literature. Among those we focus on two different types, which were specifically designed to limit the above discussed kind of noise (see Sec. 2.3.2).

4.0.1 Background Activity filters (BAFs)

Given the nature of the noises that afflict the DVS cameras, some filters have been developed that exploit one of the characteristics of this type of sensors: the spatio-temporal correlation. It has been noticed that the events detected by a neuromorphic camera show a high spatio-temporal correlation between them. This is easily explained given the nature of this type of sensor: whenever a DVS camera detects a movement or a change in lighting, it is likely that a neighbourhood of pixels will be active at the same time, generating events. Therefore, real events shows a higher spatio-temporal correlation than noise related events. This feature can be used to reduce the noise level of the images. Events are associated with a spatio-temporal neighbourhood, within which the correlation between them is calculated: if it is lower than a certain threshold the events are likely due to noise and thus are filtered out, otherwise they are kept. By exploiting this mechanism it is possible to develop filters that are able to distinguish real events from noise related ones with high probability, without losing information and

improving the quality of the images. In practice, Background Activity filters (BAFs) record the timestamp of an event and compare it with that of the last recorded event in the spatio-temporal neighbourhood it belongs to. If the delta time between the two is lower than a certain threshold T then the new event is kept, otherwise it is filtered.

4.0.2 Mask filters (MFs)

Mask filters (MFs) are used to deal with Hot Pixel Noise or to eliminate uninformative or distractor pixels. They work with another property of the DVS images: *pixel activity*, that is the number of events produced by each pixel. As mentioned before, DVS pixels only produce an event when they register a variation on brightness. Therefore, they do not tend to constantly produce new events, indeed it is more likely that they remain silent for most of the time. Hence, to spot and filter Hot Pixel Noise, it is possible to elaborate a statistics of how many time a pixel produces an event and use it to identify those pixels that show an out-of-normal value. MFs realize this approach following a two-step procedure. First, they analyze the images to detect if there are any pixels whose activity has a higher event rate than a configurable threshold and save this information into a mask from which the name of the filter is derived. Then, they use the mask to block the output of these pixels.

4.1 Filters Implementation

4.1.1 BAF

The first filter we present is a *BAF*, which, like those introduced in 4.0.1 , is inspired by the filter described in [18]. It is characterized by only two parameters s and t that define the dimensions of the spatio-temporal neighborhood. The larger s is, the larger the neighborhood will be and therefore the filter gets slower. On the other hand, the parameter t does not affect the speed, but the selectiveness of the filter (like s anyway). To have a more selective filter, it is sufficient to decrease s and t, while increasing them widens the filter mesh and this makes possible to keep a higher number of events. The filter is associated with a matrix M, whose dimension corresponds to the one of the sample to be filtered (i.e. for the IBM DVS128 Gesture the samples came from a 128x128 camera, so M is a 128x128 matrix). Every location of M is thus associated to a pixel of the camera. When the events starts being processed, their timestamp is written in all the locations belonging to its spatio-temporal neighborhood, except for its own. Then the filter compares the timestamp t to the one written in its location, that correspond to the last event occurred in the neighbourhood. If their difference is higher than t, then the event is likely noise related and it is eliminated, otherwise it is kept. This explanation allows to understand why only s influence directly the speed of the filter: the bigger s is, the more writing operations are required, while whatever t is, there will be always only one comparison. The detailed functioning of the filter is summarized in the following Alg. 1 :

Algorithm 1 : Background Activity Filter

Being E a list of events of the form (x, y, p, t)
Being (x_e, y_e, p_e, t_e) the x-coordinate, the y-coordinate, the polarity and the timestamp of the event e respectively
Being M a 128×128 matrix
Being s and t the spatial and temporal filter's parameters
Initialize M to zero
Order E from the oldest to the newest event
for e in E **do**
    for i in $(x_e - s, x_e + s)$ **do**
        for j in $(y_e - s, y_e + s)$ **do**
            if not $(i == x_e$ and $j == y_e)$ **then**
                $M[i][j] = t_e$
            end if
        end for
    end for
    if $t_e - M[x_e][y_e] > t$ **then**
        Remove e from E
    end if
end for

Analysis on clean input

IBM DVS128 Gesture

Before analyzing the impact of the filter on the corrupted datasets, we have applied it to the original one, in order to examine its impact on the SNN and to have a yardstick to later verify the results it can obtain on the adversarial examples.
After applying the filter we notice how the accuracy of the net shows a slight decrement. This behaviour is more marked when $s = 1$ and $t = 1$, in fact in this case a value of

69.70% is recorded, lower than the original by 22.34%. Considering the other filters, the decrease in performance is limited to a few percentage points, with the $s = 4$ and $t = 10$ filter obtaining the best result, maintaining a precision of 89.02%.

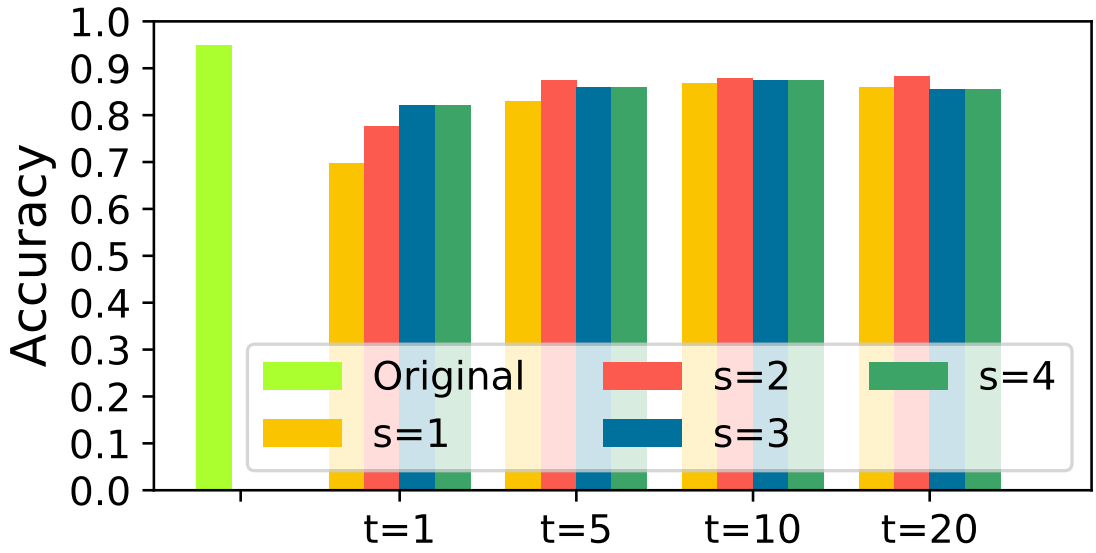

Figure 4.1: DVS128 Gesture: BAF impact on the clean input for different s and t values.

NMNIST

The filter response on this dataset is much more uniform than in the above discussed case. Everywhere the accuracy remains between 92% and 93%, and it even reaches 94% when $s = 4$ and $t = 1$. We can say then the response of the *Background Activity Filters* on the NMNIST dataset is better than on DVS128 Gesture, because even if in both cases adding the filter leads to a decrease in the accuracy of the SNN, in the first case this remains limited for all the possible configurations of s and t, while in the latter the parameters have a larger influence and the impact on the performance is greater.

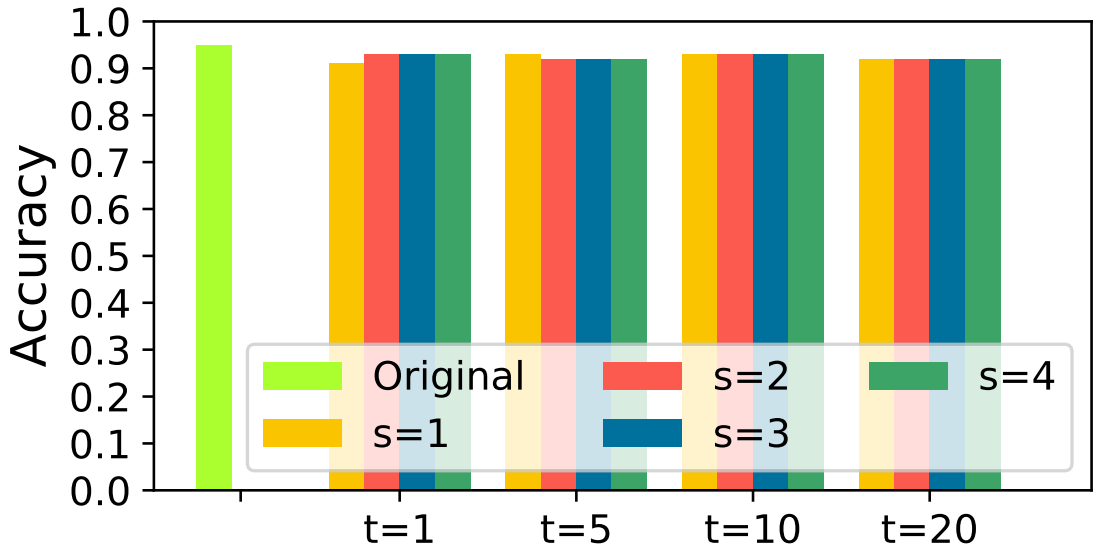

Figure 4.2: NMNIST: BAF impact on the clean input for different s and t values.

4.1.2 MF

The second filter that we propose is a Mask Filter (see Section 4.0.2). It has been developed in the Pytorch environment and is derived from the one presented in [18]. The filter, as explained before, is based on the concept of *pixel activity*, that is the number of events produced by every pixel during the observation time. The filter has only one parameter T, which represents the *activity* threshold of every pixel and comes with a NxN matrix M (i.e. N is the size of the samples), with a cell for each pixel, that act as a mask. First, it calculates the activity of every pixel and compares them with the threshold T, supplied as the only parameter to the filter. Whenever a pixel overcomes the threshold, its corresponding cell in M is marked. The pixels that have been marked are masked and filtered out. The detailed procedure is reported in Algorithm 2:

Algorithm 2 : Mask Filter

Being E a list of events of the form (x, y, p, t),
Being (x_e, y_e, p_e, t_e) the x-coordinate, the y-coordinate, the polarity and the timestamp of the event e respectively,
Being M a $N \times N$ matrix, where N is the size of the frames,
Being *activity* a $N \times N$ matrix, representing the number of event produced by each pixel,
Being T, the temporal threshold passed to the filter as a parameter,
Initialize to *activty* and M to zero,
for x in range(N) **do**
  for y in range(N) **do**
    for e in E **do**
      if $(x, y) == (x_e, y_e)$ **then**
        $activity[x][y] += 1$
      end if
    end for
    if $activity[x][y] > T$ **then**
      $M[x][y] = 1$
    end if
  end for
end for
for e in E **do**
  if $M[x_e][y_e] == 1$ **then**
    Remove e from E
  end if
end for

Analysis on clean inputs

IBM DVS128 Gesture

As previously done for the *BAF*, even with the *Mask Filter* it is useful to proceed with an analysis on the clean input, in order to have some figures to better evaluate the results that will later be obtained on the attack. As can be seen from Fig. 4.3, the precision remains above 90% for all $T > 75$. When $T \geq 200$ the value reaches the original 92.04%. In any case, even when $T = 50$ the precision is equal to 87.50% with a decrease of only 4.54% compared to the original. The curve begins to drop faster when T decreases below a certain threshold, even if for $T = 25$ it is still 71.21%, that it may be still acceptable. The accuracy continues to drop as T decreases, even if for $T = 10$ it remains at 41%, a value just below half of the original. To get really low values, it is necessary to further decrease T, in fact when $T = 5$ the SNN is unable to predict with a precision higher than 14.39%.

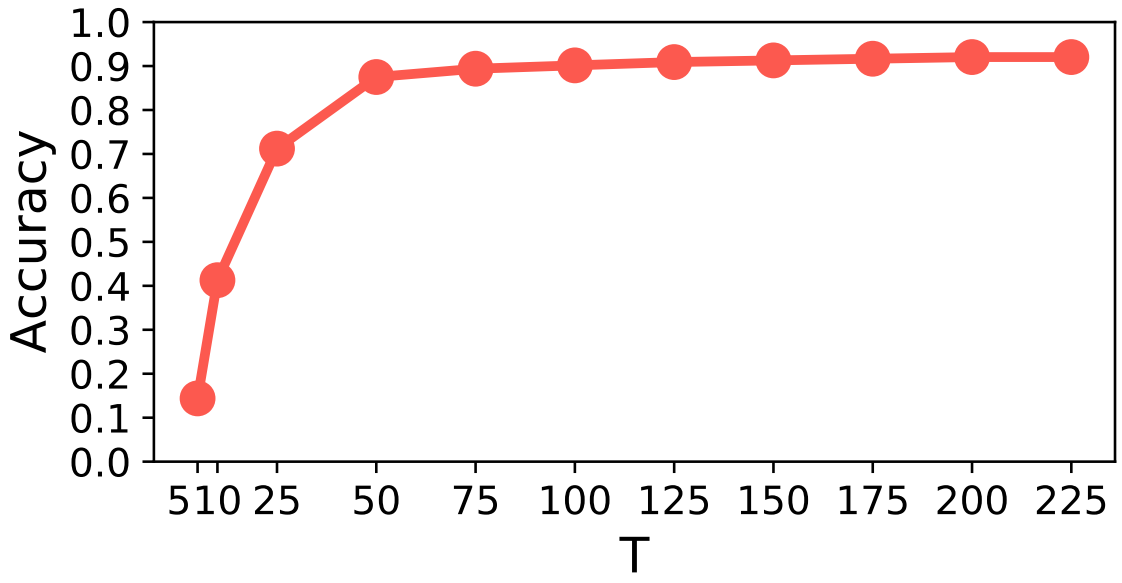

Figure 4.3: DVS128Gesture: MF impact on the clean input for different T values.

NMNIST

The same procedure was also applied to the second dataset, achieving similar results as can be seen by comparing the two graphs, which show a similar trend: the intervals of the T parameter are different in the two cases, because the duration of the samples is not the same , but both graphs have a plateau that tends towards original accuracy and begins to fall for lower T values. Looking at the Fig. 4.2, it is possible to notice how the precision remains constant and equal to the original one for a large portion of the values considered for the T parameter, in particular when $T > 25$. For lower values of T, a slight decrease in the accuracy begins to appear, which however remains contained: in fact the precision is still equal to 89% when $T = 20$ and drops to 76% for $T = 15$. The SNN is still able to classify with an accuracy of 48% when $T = 10$, which is a value approximately half of the original, while it does not exceed 17% when $T = 5$.

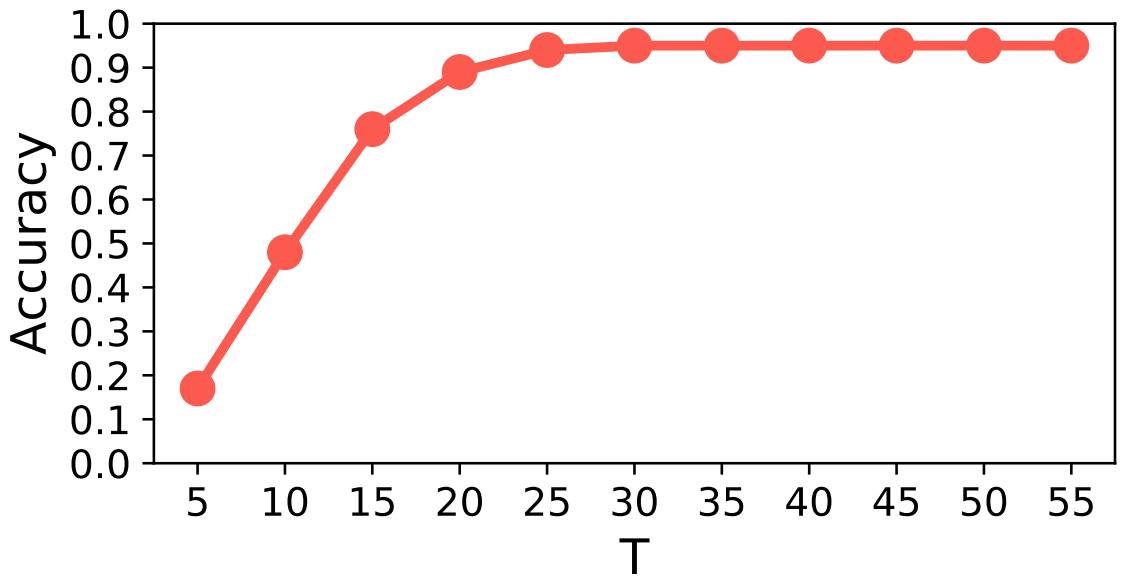

Figure 4.4: NMNIST: MF impact on the clean input for different T values.

CHAPTER 5

Noise Influence Evaluation

Before proceeding with the analysis of the attacks and of the possible countermeasures we decide to conduct a preliminary analysis of the effect of random noise on the SNN. For this reason, we perturb the IBM DVS128Gesture dataset by injecting uniform and normally-distributed random noise, then we measure the classification accuracy achieved by the SNN. These noises have been multiplied by a factor called noise magnitude before being added to the samples. We therefore observe how the noise magnitude value affects the accuracy of the SNN. After this, we apply different BAFs and MFs, described respectively in Sec. 4.0.1 and in Sec. 4.0.2, to mitigate the effect of the perturbations, varying their parameters (i.e., BAF: s and t, MF: T). Then, we analyze the effect of the filters on this noisy dataset and notice a marked recovery in performance. On the other hand, when the effect of the noise is limited or null, for low values of the noise magnitude, the filter causes a small reduction in the original accuracy. The noise following either the normal or the uniform distribution is not applied to all the pixels, but only to about 2% of them. To do this we create a sparse matrix that acts as a mask. This mask is then used to select only a portion of pixels, to which apply the noise. If, on the contrary, we applied the noise to all the pixels we would not be able to appreciate the effect of the BAF. In fact, regardless of their magnitude, the filters recognize all the pixels with a value other than zero as events and would therefore find themself filtering a sample full of them. In particular, the BAF eliminates the space-time isolated events, which however in this case would no longer

exist. From these considerations it is clear that it is necessary to hit only a fraction of the pixels to appreciate the work of this kind of filter.

5.1 Results and Comments

BAF

Fig. 5.1 reports more in detail the results of the experiment, conducted on the IBM DVS128 Gesture Dataset, filtering with different BAFs. The graph shows the accuracy of the network as the noise magnitude varies. First of all, we can see how the impact of the normally distributed noise is generally higher than the uniform one, even if they show a similar trend. As indicated by pointer ① in Fig. 5.1, the filter slightly reduces the accuracy of the SNN when no noise is applied (i.e., *noise magnitude* = 0), as we have already observed when we have analysed the impact of the filter on the clean input (see Fig. 4.1). The effect of the filter starts being appreciated when the noise magnitude increases. Pointer ② highlights the difference between the performances in the two cases, with and without filter, when normal noise with noise magnitude equal to 0.55 is applied. As can be seen,the recovery of performance is considerable, with an improvement of 64% with the filter with $s = 1$ and $t = 5$. The results are even better with the uniformly-distributed noise. In fact, as indicated by pointer ③, the same filter with $s = 1$ and $t = 1$ is able to reduce the classification accuracy loss to a very small amount, even when the perturbations with large magnitude of 0.85 and 1 are applied (accuracy equal to 85% and 74% respectively).

MF

We conduct the same procedure as before, but this time using the MF. The results are reported in Fig. 5.2. As we can see from the figure, the filter shows two different behaviour, depending on the nature of the perturbation. When Normal noise is injected, the MFs with $T \geq 100$ seem to work even better than the BAFs in the presence of large perturbations. Indeed, the perturbations with magnitude of 1.0 are filtered out relatively well by the MFs with large T, while, for the same noise magnitude, both the

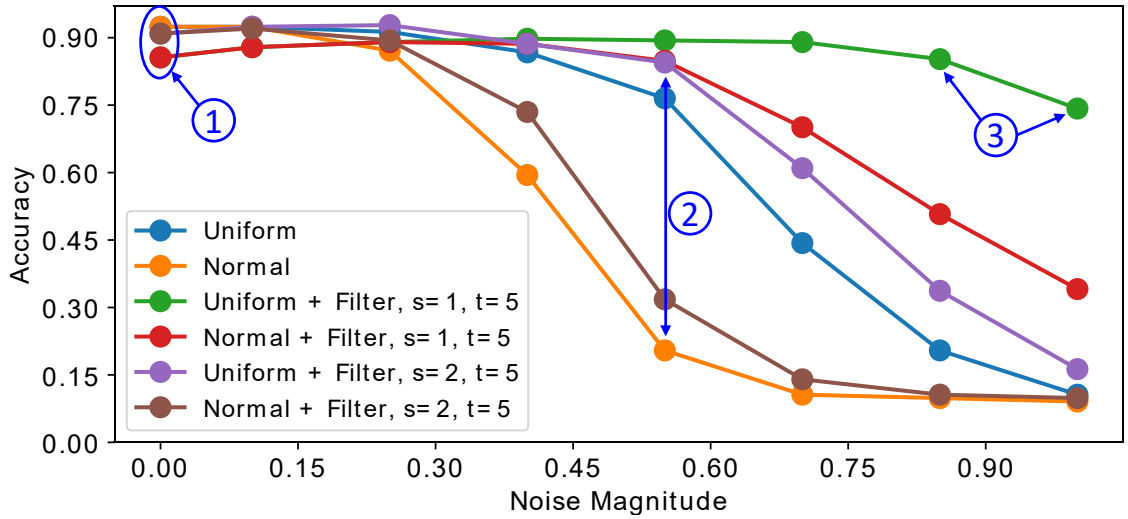

Figure 5.1: Analyzing the impact of applying the normal and uniform noise to the DVS128Gesture dataset when different *BAFs* are applied.

MFs with $T \leq 50$ achieve an accuracy of only 33-34%. On the contrary, in the case of Uniform noise, the filter not only is not able to recover performance, but these even deteriorate when $T \leq 50$. On the other hand, the performances remain in line with those obtained on the noisy and unfiltered dataset when $T \geq 100$.

This preliminary experiment demonstrates that the noise filters for DVS can potentially restore a large portion of accuracy that would have been dropped due to the perturbations. Therefore, this motivates us to employ such filters as defense methods against adversarial attacks.

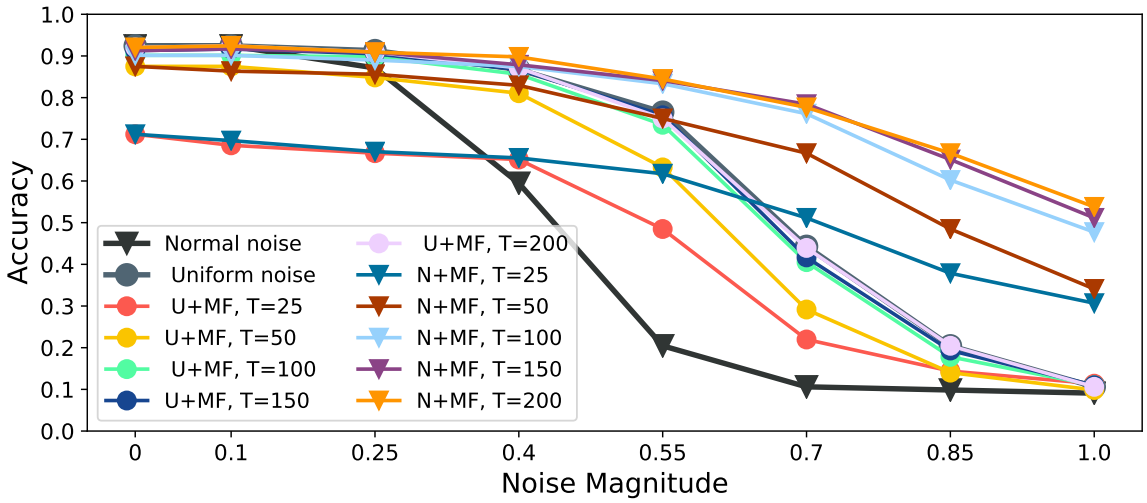

Figure 5.2: Analyzing the impact of applying the normal and uniform noise to the DVS128Gesture dataset when different *MFs* are applied.

CHAPTER 6

Gradient Based Attack

The first attack we present is the *Gradient Based Attack*. It is a modified version of the attack described in [29]. As the name suggests, the attack involves the computation of the gradients, used to minimize a loss function. In our case, we chose *the cross entropy loss function* to minimize the output probability of the network. Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. The SNN's output probability vector is an array that indicates the probability of a given class to be associated with the input. To extract the probability we took the number of total spikes for each class and passed this data to a *Softmax function*. Then we compute the cross entropy on the probability associated with the input label and try to minimize it throug the gradient calculation. The aim is to significantly reduce the output probability of the correct class, in favour of the others and, consequently to reduce the overall accuracy of the SNN.

6.1 Adversary Threat Models

In our experiments, we have considered three different threat models in the system setting, shown in Fig. 6.1. They essentially differ in the presence or absence of the noise filter and its position. In all three scenarios, the given adversarial attack algorithm perturbs the frames of events generated from the DVS camera, with the aim of fooling the SNN. In the first scenario, threat model Ⓐ, the attacker has access to the input

of the SNN and directly perturbs the frames of events coming from the DVS camera. The second model, threat model Ⓑ, includes also the noise filter, inserted in parallel to the adversarial perturbation. This situation corresponds to an attacker unaware of the filter presence, which could be a weak assumption. Therefore, we developed a further model: the threat model Ⓒ. This starts from the opposite assumption: an attacker aware of the presence of the DVS noise filter. To model this scenario, we created a slightly modified version of the attack in which the perturbed sample is first processed by the filter and then by the SNN, in other words the filter is seen as a preprocessing step of the SNN, and therefore is embedded in the attack loop. The SNN's outputs are then used to calculate the loss function and update the attack as before.

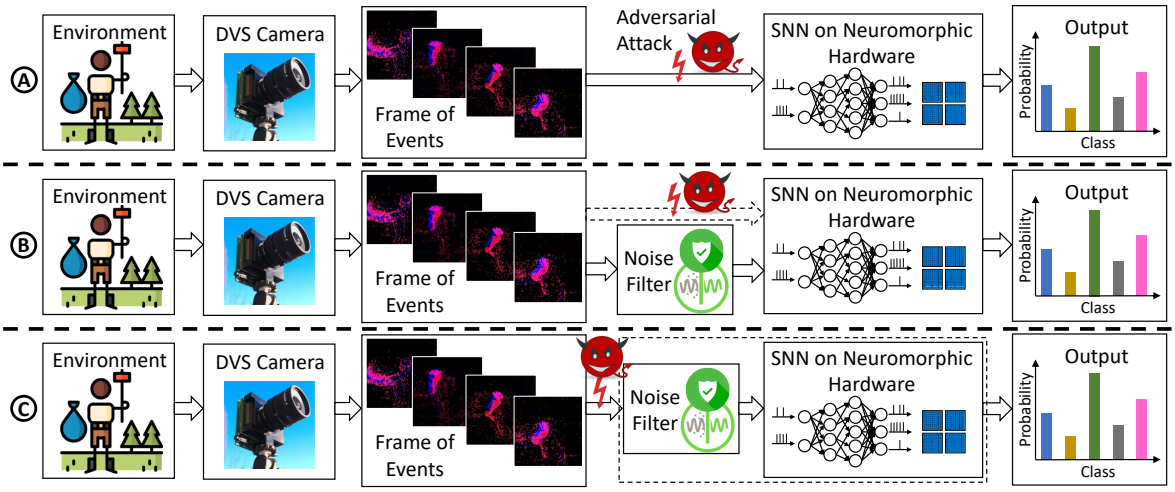

Figure 6.1: Adversarial threat models considered in this work. (a) The adversary introduces adversarial perturbations to the frames of events which are at the input of the SNN. (b) The noise filter is inserted as a defense to secure the SNNs against adversarial perturbations, while the adversary is unaware of the filter. (c) The adversary is aware of the presence of the noise filter, and sees it as a preprocessing step of the SNN.

6.2 Implementation

The procedure for generating the *Gradient Based Attack* is summarized in Alg. 3. The attack does not hit all the frames of events, but is able to focus only on some, appropriately selected through the mask M (see line 7). This characteristc in the original paper [29] was called $Sparsity$. The perturbation is added to the samples and it is progressively updated by the algorithm. To do so, the intermediate adversarial example is fed to the SNN and the output probability is extracted. This is then processed by the loss function and used to calculate the gradients, updating the perturbation. This process goes on for a given number of iterations, selectable by the user, before drawing the next sample to be perturbed from the dataset D.

Algorithm 3 : The SNN Adversarial Attack Methodology.

1: Being M a mask able to select only certain frames
2: Being D a dataset composed of DVS images
3: Being P a perturbation to be added to the images
4: Being *prob* the output probability of a certain class
5: **for** d in D **do**
6: **for** i in $max\_iteration$ **do**
7: Add P to d only on the frames selected by M
8: Calculate the prevision on the perturbed input
9: Extract *prob* for the actual class of d
10: Update the loss value as $loss = -log(1 - prob)$
11: Calculate the gradients and update P
12: **end for**
13: **end for**

CHAPTER 7

Gradient Based Attack: Results

This chapter collects and comments on all the results obtained with the three threat models described in Chap. 6. As stated before (see Chap. 3), the experiments were conducted on two different neuromorphic datasets, IBM DVS128 Gesture and NMNIST, using two different SNNs.

7.1 IBM DVS128Gesture

7.1.1 SNN Robustness under Attack Without the Noise Filter

In threat model Ⓐ (see sec. 6.1) the adverasy introduces a perturbation to the input of the SNN. This results in a significant drop of performances, while the impact of the perturbation remains relatively small and difficult to notice. When no filtering is applied to the input of the network, the *Gradient Based Attack* proves to be an efficient Adversarial Attack, able to reduce the network performances but also to maintain a good level of stealthiness. The accuracy dropped from 92.04% to 15.15% (see pointer ④ in Fig. 7.1), already after the first iteration. Further iterations of the algorithm do not appear to reduce the accuracy to a greater extent.

7.1.2 SNN Robustness under Attack by Noise Filter-Unaware Adversary

We have analyzed the robustness of the network also in threat model Ⓑ, that is, when the attacker is not aware of the presence of the filter, which is therefore placed in parallel to the attack. The experiments show the effectiveness of the filter as a means of defense, in fact the results are much better than those obtained for threat model Ⓐ. If we analyse the data shown in the figure, obtained by varying both the parameters of the filter (s, t), we can see that for most of the cases the accuracy remains almost constant and around 90% , as indicated by pointer ⑥. Among the considered values, $t = 10$ produces the highest accuracy for every s, peaking at 91.67% with $s = 3$ and $s = 4$ (see pointer ⑦). However, if the filter's mesh are widen too much, the filter becomes less effective as we can see when $t = 500$. In fact, in the latter case, the accuracy does not exceed 48% (see pointer ⑧). Generally, the parameter s does not influence much the accuracy, as we can see by fixing the other parameter t. This is not true when $t = 1$: in this case the accuracy varies in a wider range than before, passing from 62.5% when $s = 1$ to 83% when $s = 4$ (see pointer ⑤). In all the other cases, the difference is almost not noticeable. Notice, though, that the higher s is, the slower the filter is to process all the data.

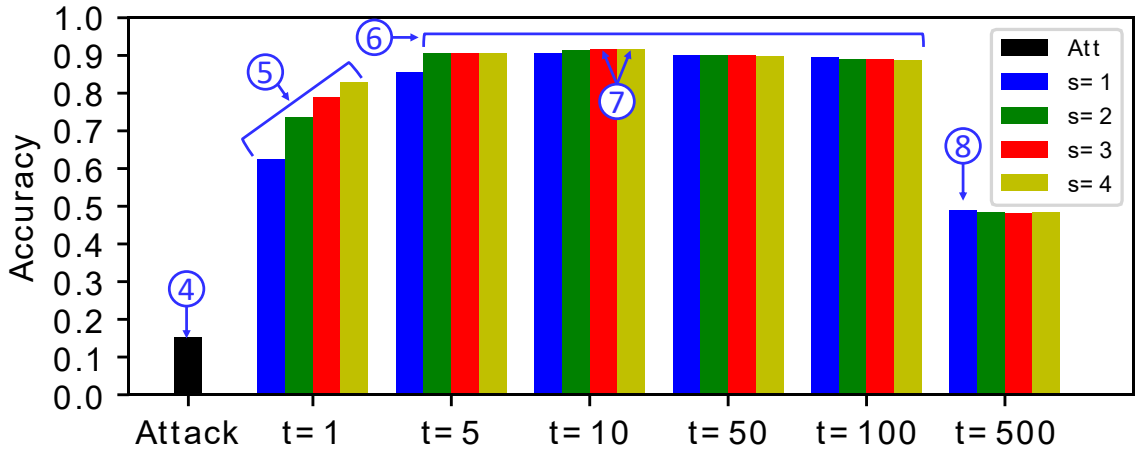

Figure 7.1: DVS128 Gesture: SNN robustness under the adversarial threat model A, and under the threat model B with different parameters s and t of the filter.

7.1.3 SNN Robustness under Attack by Noise Filter-Aware Adversary

Finally, we test the resistance of the network also in model Ⓒ, which represents a situation in which the attacker is aware of the presence of the filter and sees it as an integral part of the network, i.e. as a preprocessing unit. Even with this assumption, the results are identical to those obtained for threat model Ⓑ, with a maximum accuracy of 91.67% when $s = 3, 4$ and $t = 10$ (see pointer ⑨ in Fig. 7.2). This testifies once again the effectiveness of our methodology in increasing the SNN safety and also demonstrates how the attack, despite being aware of the presence of the filter, cannot find a countermeasure to it.

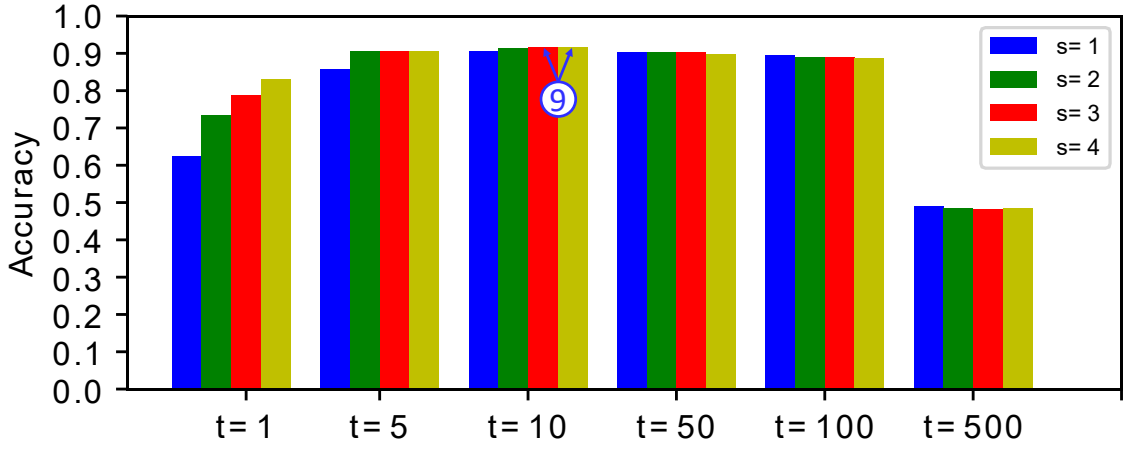

Figure 7.2: DVS128 Gesture: SNN robustness under adversarial threat model C.

7.2 NMNIST

7.2.1 SNN Robustness under Attack Without the Noise Filter

Threat model Ⓐ (see sec. 6.1), as before, is very effective. The impact on performance is even higher than in the previous case, with the accuracy that drops from 95% to only 4%. To achieve this result, 20 iterations are required, while with the DVS128 Gesture it is possible to obtain the maximum effectiveness already after the first execution.

In the case of NMNIST, after the first run of the algorithm, the SNN is still able to predict samples with an accuracy of 53%.

7.2.2 SNN Robustness under Attack by Noise Filter-Unaware Adversary

The trend shown by model Ⓑ is similar to that of the DVS128 Gesture. The filter turns out to be an effective defensive method against the attack, minimizing the drop in performance to a few percentage points, also for the NMNIST dataset. The results as s and t vary are reported in Fig. 7.3. We can easily notice how the accuracy with respect of model Ⓐ, is significantly higher peaking at 94% when $s = 3$ or $s = 4$ and $t = 2$. In any case, even varying the parameters over a wide range, the accuracy does not change much, always remaining around 90%, as happened with the DVS128 Gesture. The best results, as said before, were obtained with $t = 2$, while it is possible to notice a slight decrease in performance when $t = 1$ and when $t \geq 20$. In particular, when $t = 1$, the results are more dependant by the value of s, as already notice for the other dataset. Even if in this case the variation is less marked.

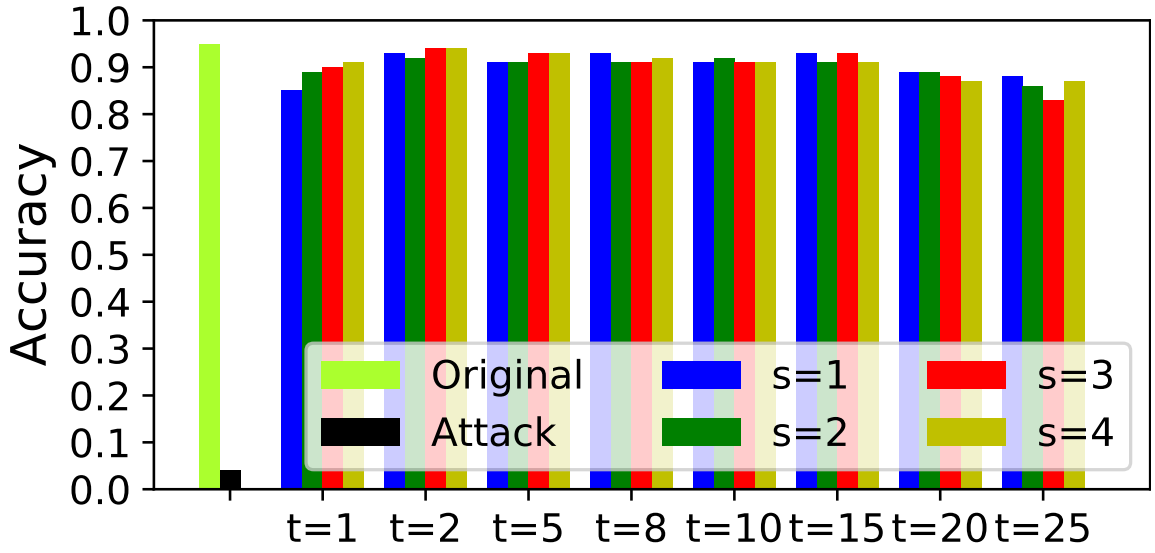

Figure 7.3: NMNIST: SNN robustness under the adversarial threat model A, and under the threat model B with different parameters s and t of the filter.

7.2.3 SNN Robustness under Attack by Noise Filter-Aware Adversary

The experiment has been repeated for threat model ©, inserting the filter as a preprocessing unit of the network, simulating an attacker aware of the filter's presence. As already appreciated in the previous case, the attack is not able to counter the presence of the filter and produce an effective perturbation. This confirms our hypothesis, that the Gradient Base attack is not able to overcome the filter, even when it is aware of its presence.

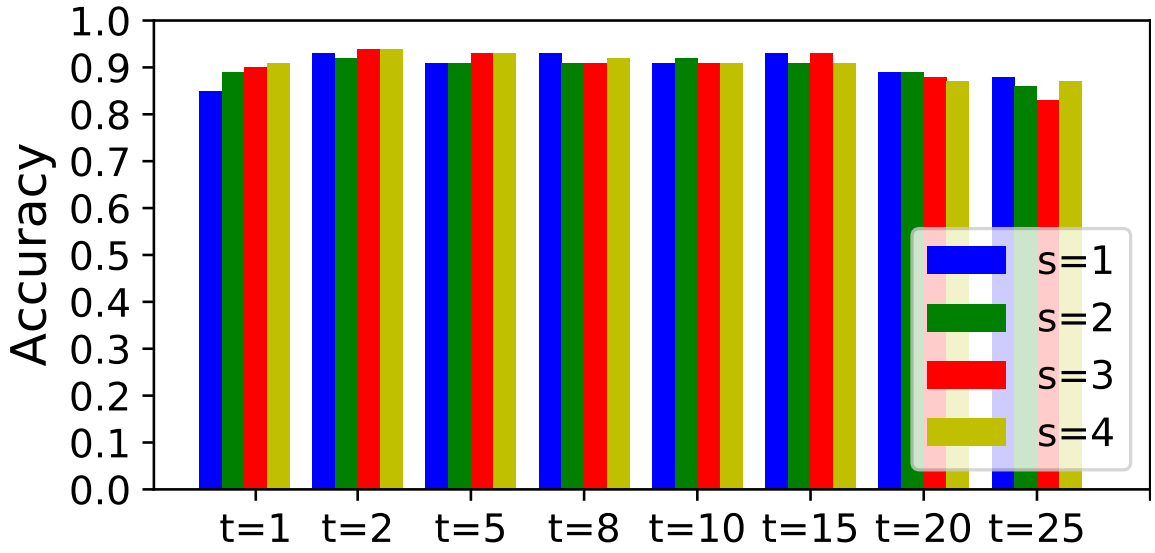

Figure 7.4: NMNIST: SNN robustness under adversarial threat model C.

7.2.4 Case Study: Output Probability Variation

In addiction to the above discussion, we also conduct a further investigation on a particular sample from *IBM DVS128 Gesture*, labeled *left hand wave*, that we chose as a case study. Fig. 7.5 reports the frames of events and the number of output spikes for each adversarial threat model presented in this work, as well as for the clean inputs and the filtered event series without attack. The number of output spikes is an important parameter, because it is connected to the output probability used by the attack, and also it is used by the SNN to make its prevision. The clean input, showed in Fig. 7.5-a,

is correctly classified by the SNN. As it easy to notice, *class 2* (i.e. *left hand wave*) produces the highest number of spikes, while the other classes have much lower values. Almost the same behaviour is noticeable in Fig. 7.5-b, where the sample has been filtered with a BAF with $s = 2$ and $t = 5$. Even if the frames are clearly different from before, since the majority of background noise was eliminated, the SNN behaves similarly to the previous case and guesses correctly. In this case, the prediction is even stronger, because the number of pulses for class 2 is slightly higher. Focusing on the events reported in Fig. 7.5-c, we can see that the difference between the clean input and the adversarial one is very difficult to spot, nevertheless if we analysed the number of spikes produced, we can notice that the attack caused a significant increase in the number of pulses for every class, which eventually led to a misclassification. In particular, *class 0* (i.e., *hand clap*) and *class 10* (i.e., *other gestures*), produce over 500 spikes and overcome even *class 2*, which was the right one. Note that, despite a great difference in the output probabilities, the modifications of the frames of events, compared to the clean event series, are barely noticeable (see Fig. 7.5-c). Finally, considering 7.5-d and 7.5-e , where we reported the adversarial model Ⓑ and Ⓒ, we can see that the SNN returns to correctly classify the input, thanks to the presence of the filter (again BA filter with $s = 2$ and $t = 5$). In this two cases the number of spikes is similar to the original one, with a high gap between the correct class and the other classes, showing the effectiveness of the filter in restoring the accuracy of the SNN.

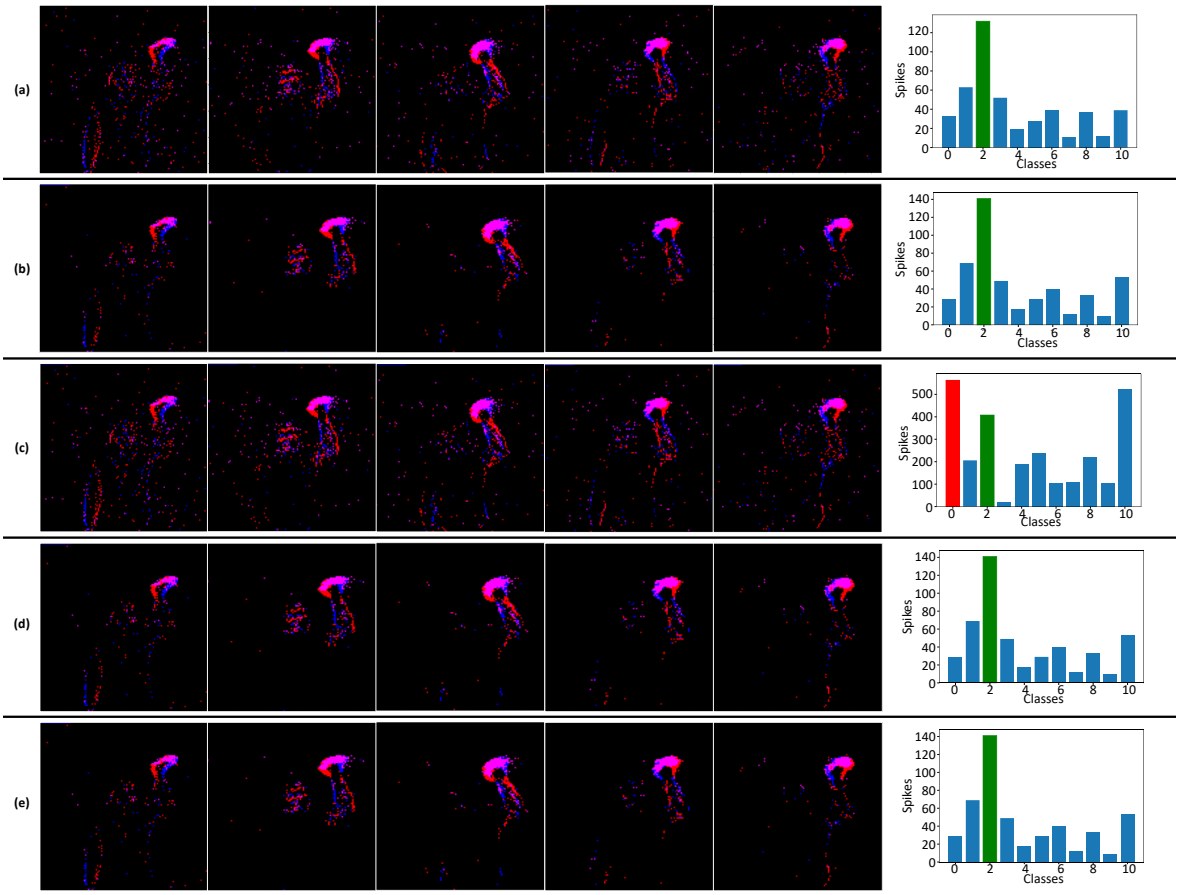

Figure 7.5: Detailed example of a sequence of event labeled as *left hand wave*. On the left, the frames of events are shown. The histograms on the right-most column report the number of spikes emitted by the neurons of the last layer, which correspond to the output classes. (a) Clean event series. (b) Event series filtered with $s = 2$ and $t = 5$. (c) Event series under the adversarial threat model A, unfiltered. (d) Event series under the adversarial threat model B, filtered with $s = 2$ and $t = 5$. (e) Event series under the adversarial threat model C, filtered with $s = 2$ and $t = 5$.

CHAPTER 8

Frame, Corner and Dash Attacks

In Chap. 6 and Chap. 7 we discussed the excellent results obtained by our methodology of defense against the Gradient Based Attack and the failure of the attack in recognizing the presence of the filter to adapt to it (threat model ©). We therefore decide to further investigate the robustness of SNN, developing new types of attack and defense methods, which are illustrated below.

8.1 Frame Attack

The first attack we develop is the Frame Attack, so called because it consists in adding a frame to all the samples. This attack is inspired by the one described in [34]. With the IBM DVS128 Gestures dataset, given the high number of pixels (128x128), the frame is not very visible and could escape a careless look, while the impact it has on the NMNIST is more evident since it is composed of 34x34 images.
In any case, as suggested in [34], this solution has the merit of not losing information on the content of the samples, as the subject is generally framed in the center of the image and the insertion of the frame deceives the neural networks but does not compromise the original image. This attack offers some clear advantages with respect to the GBA, first of all its simplicity: there is no need to calculate gradients, resulting in costly operations requiring a high use of resources; then its uniformity: its impact in terms of inserted events is the same for every sample, since the perturbation (i.e. the frame)

is the same; lastly its intrinsic resistance to the BA filter: since it does not insert isolated events, this type of attack should be resistant to the Background Activity filter. This fact is easily understood considering that space-time correlated pixels are inserted, which cannot be filtered by the BA filter. However, a possible throwback of the *Frame Attack* is the high overhead in terms of added events: this number can be easily computed considering that for every perimeter pixel, two events ('on' and 'off' polarity) are added for every frame. This means that for the DVS128 Gesture roughly 1.5 million events are added, while for the NMNIST only 95k. As we have been able to observe, the increase in the size of the samples has also led to an increase in the time required to process them, both by the filters and the network. The detailed algorithm is reported in Alg. 4:

Algorithm 4 : Frame Attack

Being D a neuromorphic Dataset
Being $d \subset D$ a $(C \times N \times N \times T)$ tensor, where C represents the channels, N represents the frame dimensions and T the sample duration
for d in D **do**
  for x in range(N) **do**
    for y in range(N) **do**
      if $(x == 0 \vee x == N-1 \vee y == 0 \vee y == N-1$ **then**
        $d[:, x, y, :] = 1$
      end if
    end for
  end for
end for

8.2 Corner Attack

To reduce the problem of excessive file size increase and also to make the attack less detectable to a human observer, we have developed a second version that we called *Corner Attack*, because it focuses on the corners of the affected images. The idea is to try to hit a corner, modifying only two pixels (two and not one to reduce the effectiveness of the BAF, creating a pair of space-time correlated neighbours). After the first iteration, all the samples that are already incorrectly interpreted by the network are eliminated and the algorithm passes to the second corner (no noise is added because the addition is made on the original images) and continues with the other angles. At each iteration, if the attack is successful on any sample, the algorithm gets faster and faster since it works on a smaller dataset. Once the 4 corners are completed, the number of modified pixels, y, increases and the procedure proceeds as before. If y increases until the row is complete and there are still correct samples, the algorithm changes row and starts again with $y = 2$. This attack turns out to be very effective and also quick. One possible flaw is that the lack of uniformity: in fact depending on how much a sample is resistant to the attack, the number of iterations can vary a lot, and with it the size of the perturbations. Even if the majority of the samples were successfully attacked after the first iteration (i.e, targeting only 2 pixels), some of them required more efforts and the associated perturbation is larger in size and therefore more recognizable. The algorithm describing the attack is reported in Alg. 5.

Algorithm 5 : Corner Attack

Being D a neuromorphic Dataset made of $(2\times N\times N\times T)$ tensors,where N represents the frame dimensions and T the sample duration
S is a list of the samples that compose D
$x = 0$
$y = 2$
$left = True$
while S is not empty **do**
  for s in S **do**
    for i in range(N) **do**
      for j in range(N) **do**
        if $i == x$ **or** ($left$ **and** $j < y$ **or** $\overline{left}$ **and** $j \geq N - y - 1$) **then**
          $s[:, i, j, :] = 1$
        end if
      end for
    end for
    The perturbed sample s is fed to the SNN, which produces a prevision P
    if P is incorrect **then**
      Remove s from S
    end if
  end for
  if $x == 0$ **then**
    $x = N - 1$
  else
    $left = left$ **xor** 1
    $x = 0$
    if $\overline{left}$ **then**
      $y = y + 1$
    end if
  end if
end while

8.3 Dash Attack

As suggested earlier, the problem with the Corner Attack is that not all samples are treated equally, but some may have a much higher noise level than others. To overcome this problem and to create an attack that was even more difficult to notice we have developed another version, *the Dash Attack*, derived from the previous one but with a substantial modification.

The *Dash Attack* methodology is designed taking inspiration from the *Corner Attack*. Indeed, the two algorithms are quite similar. The main difference is that in the *Dash Attack* only two pixels are targeted every time. The main structure of the algorithm is the same as for the *Corner Attack*, as the *Dash Attack* starts by targeting the top-left corner and by modifying the first two pixels. Moreover, the x, y coordinates are updated, in order for the attack to hit only two consecutive pixels (see lines 19-29 of Algorithm 6). Hence, this attack results to be very difficult to spot, and the introduced perturbations do not cause a large overhead of events on the samples. Moreover, all the samples under the *Dash Attack* are modified by the same amount of perturbations, even for the most resistant samples, for which multiple iterations are required to fool the network. The procedure is resumed in the following Alg. 6:

Algorithm 6 : Dash Attack

Being D a neuromorphic Dataset made of $(2\times N\times N\times T)$ tensors,where N represents the frame dimensions and T the sample duration
S is a list of the samples that compose D
$x = 0$
$y = 2$
$left = True$
while S is not empty **do**
  for s in S **do**
    for i in range(N) **do**
      for j in range(N) **do**
        if $i == x$ **or** ($left$ **and** $j < y$ **or** $\overline{left}$ **and** $j \geq N - y - 1$) **then**
          $s[:, i, j, :] = 1$
        end if
      end for
    end for
    The perturbed sample s is fed to the SNN, which produces a prevision P
    if P is incorrect **then**
      Remove s from S
    end if
  end for
  if $x == 0$ **then**
    $x = N - 1$
  else
    $left = left$ **xor** 1
    $x = 0$
    if $\overline{left}$ **then**
      $y = y + 1$
    end if
  end if
end while

CHAPTER 9

Frame, Corner and Dash Attacks : Results and Comments

In this chapter all the results obtained with the attack methodologies described in chap. 6 are reported and commented. As before, all the experiments are conducted on two different datasets: *IBM DVS128 Gesture* and *NMNIST*. The experimental setup coincides to the one already adopted in the case of the Gradient Base attack and presented in Chap. 3. In this case, however, we adopt also another filter: the Mask Filter, described in Sec. 4.1.2.

9.1 Frame Attack

The Frame Attack, described in Sec. 8.1, consists of the insertion of a frame into the samples. As already suggested, the visual impact of this attack may vary depending on the dimensions of the images. For example, this is much more evident in the case of the NMNIST than in the case of the IBMGesture. This is because the images are much more compact in the first case (34x34) than in the second (128x128) and therefore the percentage of pixels occupied by the frame is greater.

9.1.1 IBM DVS128Gesture

Two examples of the frame attack are reported in Fig. 9.1, as mentioned before the frame only occupies a small fraction of the image and it does not superimpose with the subject of the image. The resultant image is therefore perfectly recognisable by a human subject, but the network instead is unable to guess it correctly. After the attack the precision of the SNN drop from 92.04% to 9.84%, moreover most of the samples are interpreted as class *10* (*Other Gestures*) by the network. This value of accuracy is near 1/11, which correspond to the expected value of a random guesser.

Background Activity Filter

We try to filter the adversarial dataset with the BAF, varying both its parameters s and t. Unlike *the Gradient Based Attack* (see Chap.7), in this case the *BAF* is not able to restore the SNN's performances. The fact that we have targeted adjacent pixels seems to be sufficient to reduce the impact of the filter on the attack. The graph reported in fig. 9.2 shows the accuracy's values, that remain pretty low for every set of parameters. The accuracy peaks at 13.26% when $s = 3$ and $t = 10$, restoring less than 3.5%. In some cases, the filter has also a negative impact that leads to a further reduction of the accuracy (i.e. when $s = 3$ and $t = 1$).

Mask Filter

Given the inadequacy of the BAF in responding to an attack of this type, we decide to adopt another type of filter: the *Mask Filter*. The *Mask Filter* (see Sec. 4.0.2), is able to identify the pixels with an activity higher than the normal and to mask their output. As we can see in fig. 9.3 the filter works pretty well for a large interval of T values. Moreover, if we compare fig. 9.3 with fig. 4.3, we can notice how they are almost identical. The reason is easily explained: when a pixel is targeted by the *Frame Attack*, it produces large number of events, much higher than normal. If we add the fact that

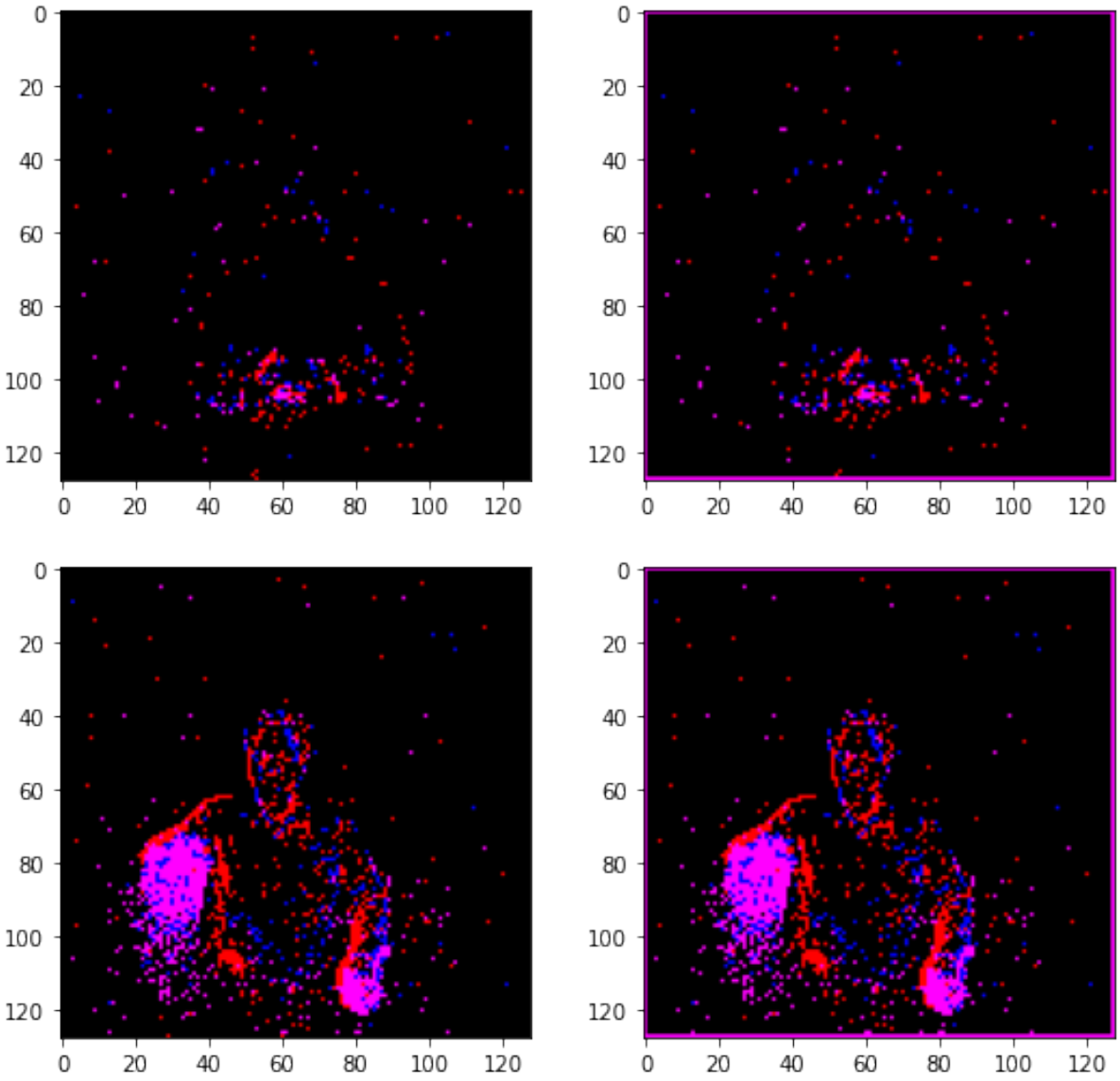

Figure 9.1: On the left: two images from IBM DVS128Gesture. On the right: the same images after being targeted by the Frame Attack. The frame is very thin and it may escape at first sight.

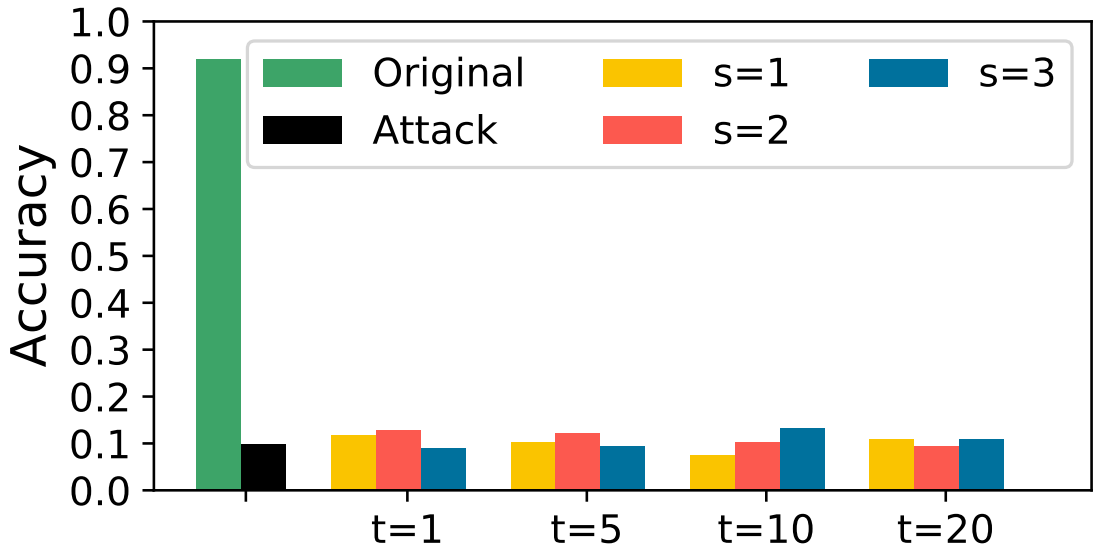

Figure 9.2: Frame Attack filtered with different BAFs: the filter is not able to counter the attack.

the attack hits both channels ("on" and "off"), this number doubles. For this reason the filter can easily spot the targeted pixels and it manages to filter the attack without compromising the original precision. Furthermore, considering that attack only affects the edge of the samples where subjects are generally not represented, after filtering no useful pixels are lost and the samples are restored to their original form. Any filter with $T \geq 75$ restores the accuracy to a value higher than 90%, this value diminishes as T decreases.

9.1.2 NMNIST

The frame in the case of the NMNIST dataset results much more evident, as it possible to notice from Fig. 9.4. In any case, also this time the main subject is not covered by the attack and could easily be identified by a human. On the contrary, the SNN interprets the majority of the samples as *class 0* (*Zero*). The accuracy drops from 95% to 8%.

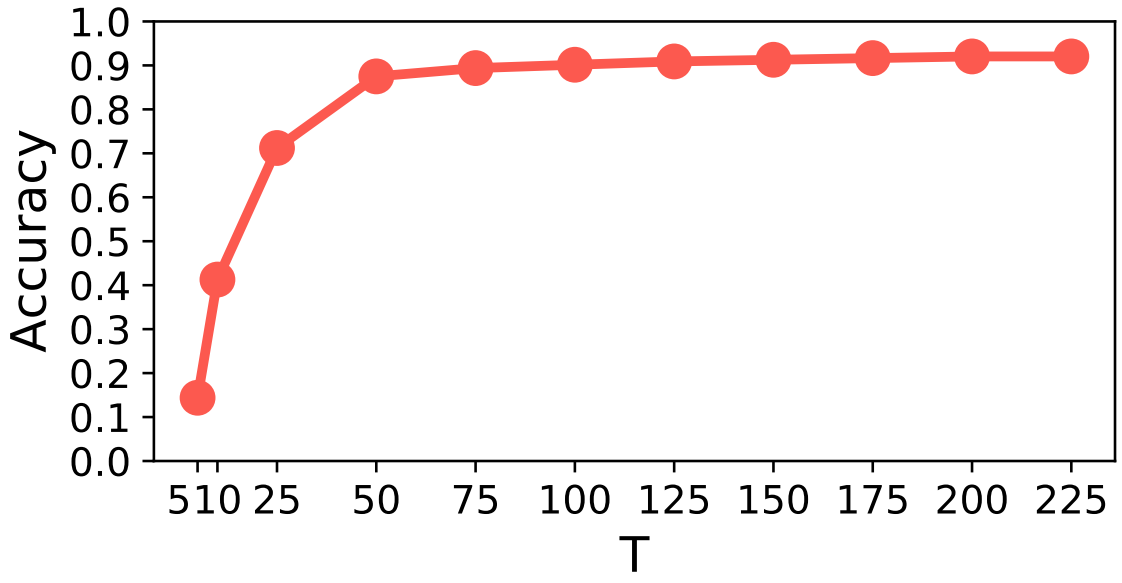

Figure 9.3: Frame Attack filtered with different Mask filters: the filter successfully restores the performances for a wide portion of T values.

Background Activity Filter

As happened in the case of the DVS128 Gesture, also with the NMNIST the BAF is not able to counter the Frame Attack. In this case the best results is obtained when $s = 3$ and $t = 1$, even if the filter manages only to recover one percentage point (from 8% to 9%). In all the other cases, the accuracy remain the same as after the attack, with the exception of $s = 1$, $t = 1$ and $s = 2$, $t = 2$ that even show a worsening compared to before (6% and 7% respectively). All results are reported in fig. 9.5.

Mask Filter

Also in this case the Mask Filter seems to be the ideal filter to deal with such an attack. When $t \geq 20$ the accuracy overcome 90%, and it even returns to the original value of 95% for $T > 25$, as we can see in fig. 9.6. Once again it is interesting to compare fig. 9.3 with fig. 4.4 that shows the analysis on the clean input. As happened before,

we can say that they are completely identical.

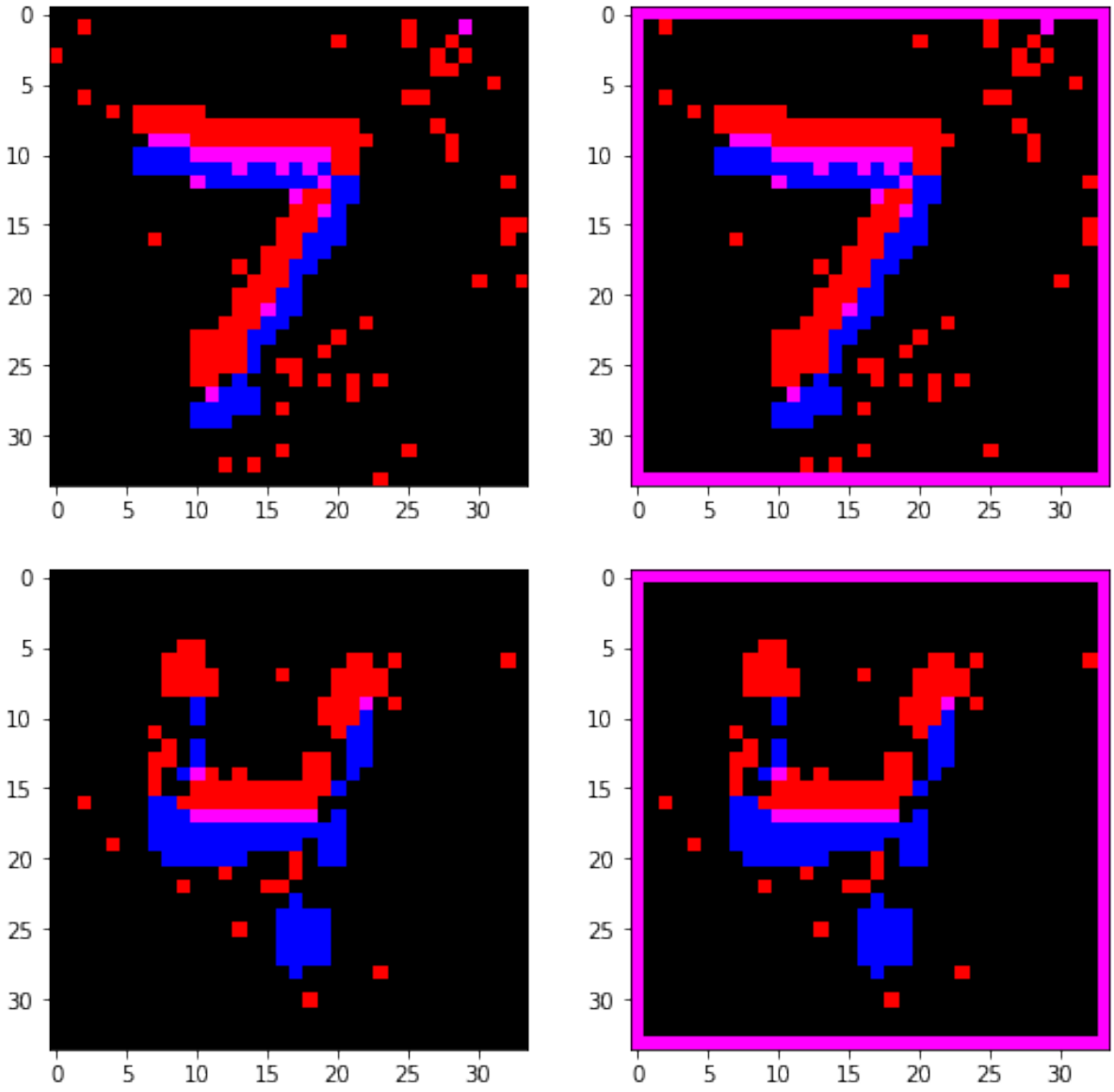

Figure 9.4: On the left: two images from NMNIST dataset. On the right: the same images after being targeted by the Frame Attack. The frame is quite evident, but it does not cover the subject.

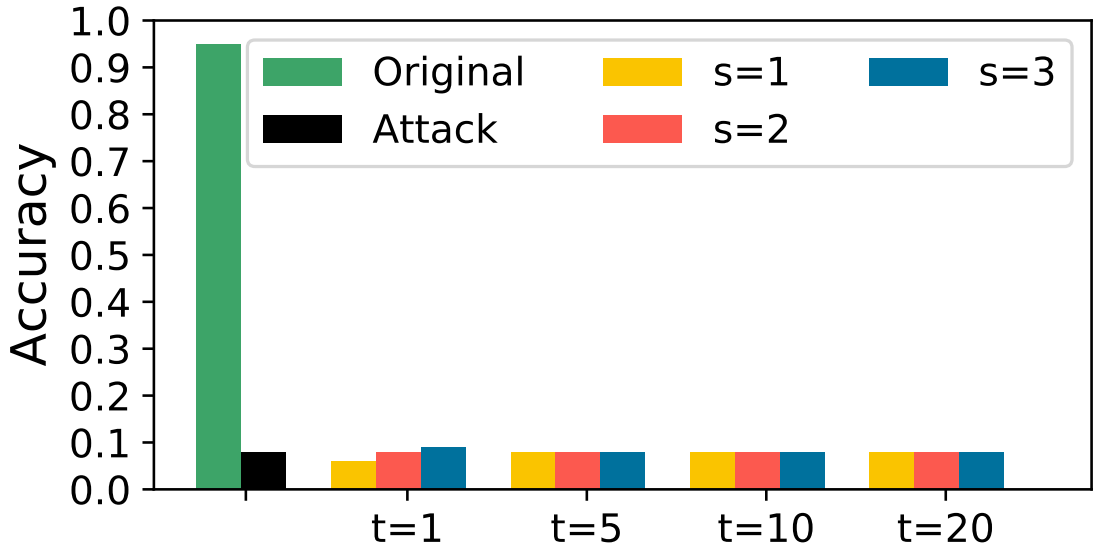

Figure 9.5: Frame Attack filtered with different BAFs: the filters are not able to counter the attack.

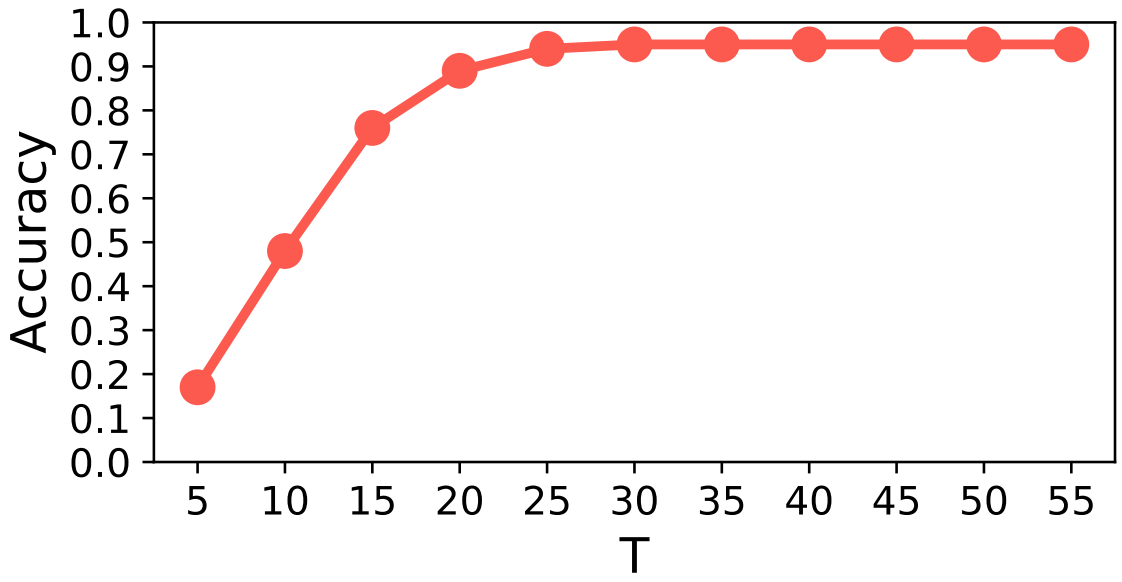

Figure 9.6: Frame Attack filtered with different Mask filters: the filter successfully restores the performances for a wide portion of T values.

9.2 Corner Attack

The Corner Attack, described in Sec. 8.2, targets the corners of the samples. Depending on the resistance to the attack, the number of iterations required to fool the SNN may vary. This fact will lead to a different visual impact on the sample: by increasing the number of iterations, more pixels are modified and the attack is more evident.

9.2.1 IBM DVS128Gesture

If we look at Fig. 9.7, we can see two samples from the DVS128Gesture dataset after they are hit by the Corner attack. As mentioned before, the impact of the attack is clearly different: in the first image the perturbation only affects two pixels and it is barely noticeable, in the second one instead the perturbation is wider and easier to detect, even if it remains moderate. Nevertheless the overall impact is lesser than in the case of the Frame Attack, and this influences also the size of the samples and the time required to processed them, which are reduced. The attack reaches a complete success in misleading the SNN: in fact the precision of the SNN drops from 92.04% to 0.0%, meaning that the SNN is unable to correctly recognize any image.

Background Activity Filter

In fig. 9.8 we can see the effect of the Background Activity filter on the Corner Attack. As we can see this type of filter is not suitable for countering such an attack, that targets groups of two or more adjacent pixels. The accuracy remains way below the original value, not exceeding 15.15%, value obtained when $s = 1$ and $t = 5$.

Mask Filter

The Mask Filter once again is able to restore the performances of the network. The explanation is similar to before: even if the Corner Attack targets fewer pixels, they show also in this case a high activity, making them an easy target for the filter. Also, as happened before, the perturbation is limited to the sides of the image, so that the informative pixels in the center are not affected. The accuracy graph is reported in fig. 9.9.

9.2.2 NMNIST

The Corner Attack is very efficient: in fact it is able to completely zero the network confidence in few iterations. The resulting images are than very difficult to distinguish from the original one. As we can see in fig. 9.10, the perturbation is concentrated in the corner (top left and bottom left respectively), and it is reduced in size. In particular, if we focus on the left image, we can see how the presence of natural noise makes it even more difficult to distinguish the presence of the attack.

Background Activity Filter

Looking at fig. 9.11, it is possible to immediately understand that the BAF is not the right filter to counter the Corner Attack. For all the set of s and t that we tried, the accuracy remained 0.0%, testifying once again the effectiveness of the attack.

Mask Filter

The graph reported in fig. 9.6 is once again coincident to the one obtained on the clean input, consequence of the weakness of this attack to the Mask Filter.

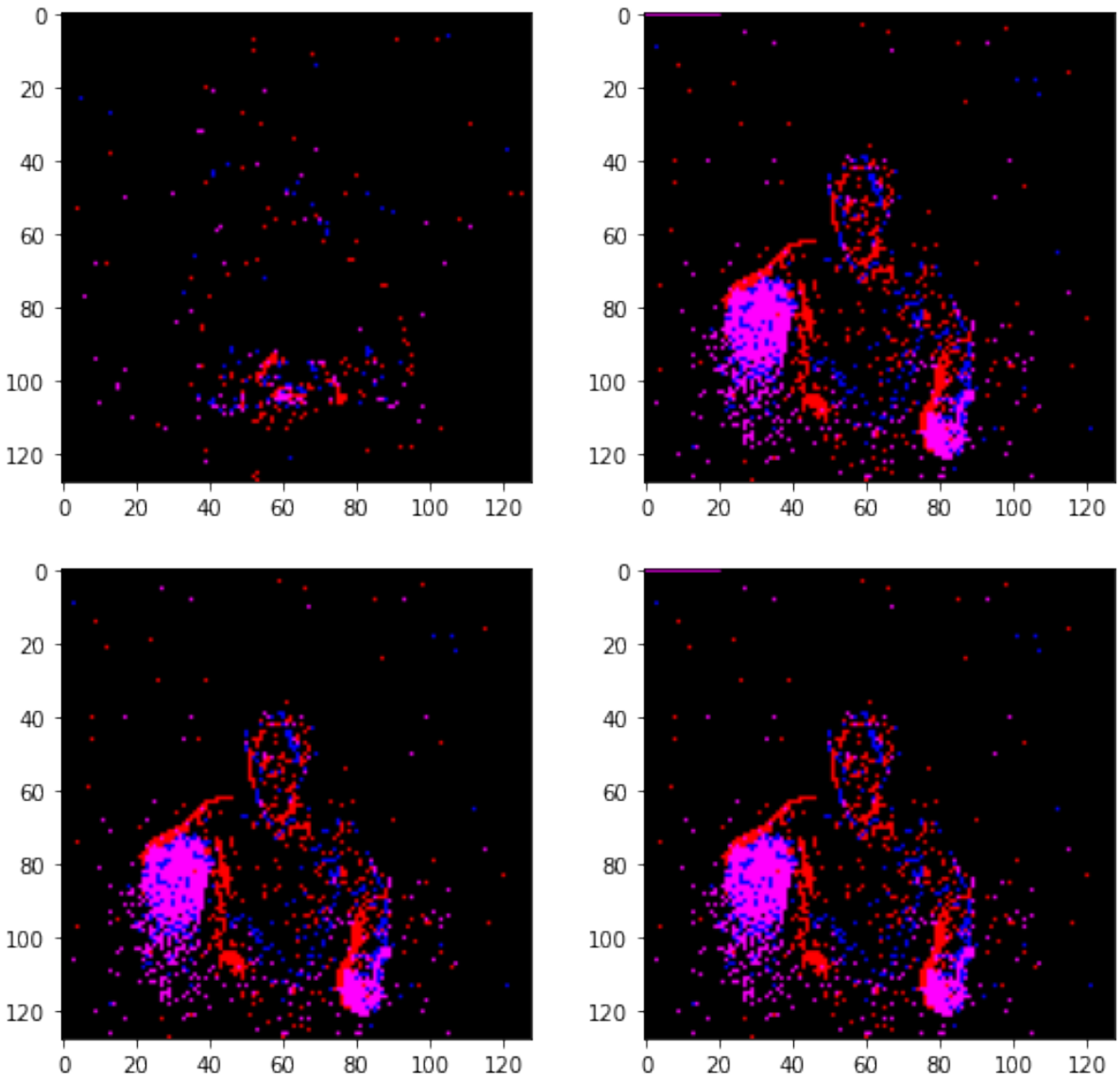

Figure 9.7: On the left: two images from IBM DVS128Gesture dataset. On the right: the same images after being targeted by the Corner Attack. The presence of the attack results more evident in the bottom image.

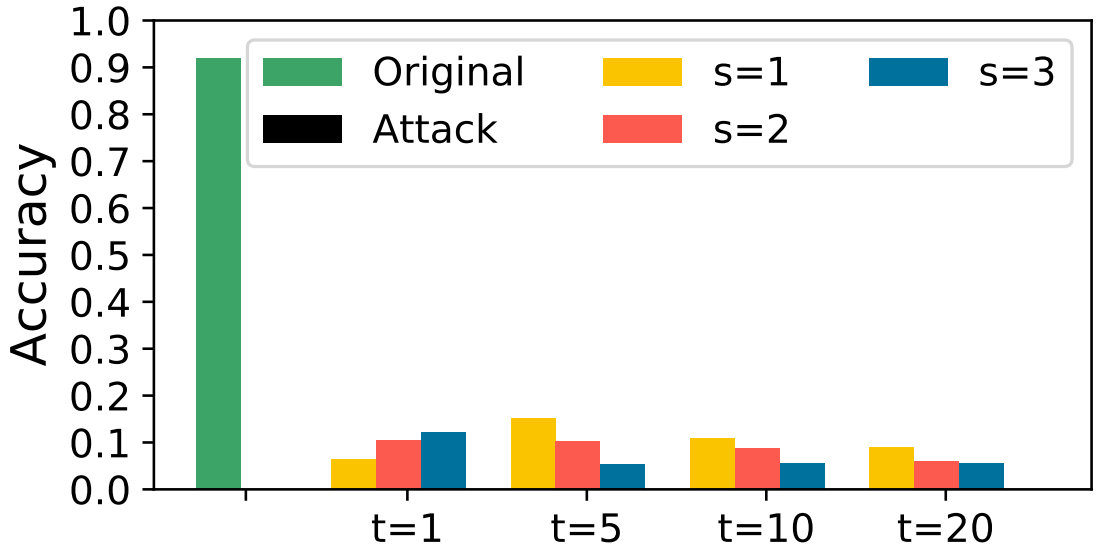

Figure 9.8: Corner Attack filtered with different BA filters: the filter is not able to counter the attack.

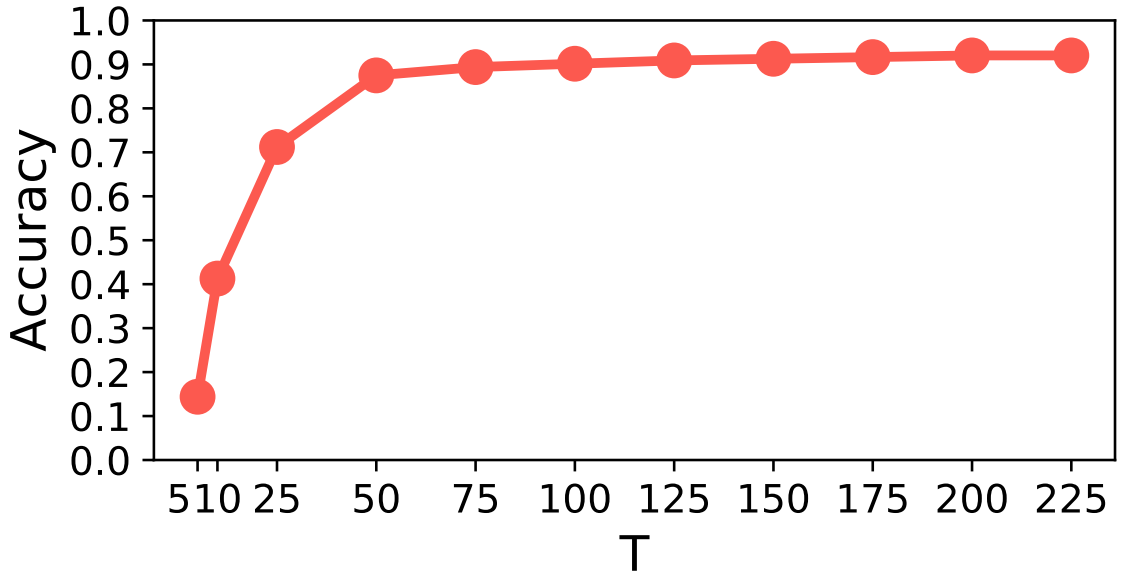

Figure 9.9: Frame Attack filtered with different Mask filters: the filter successfully restores the performances for a wide portion of T values.

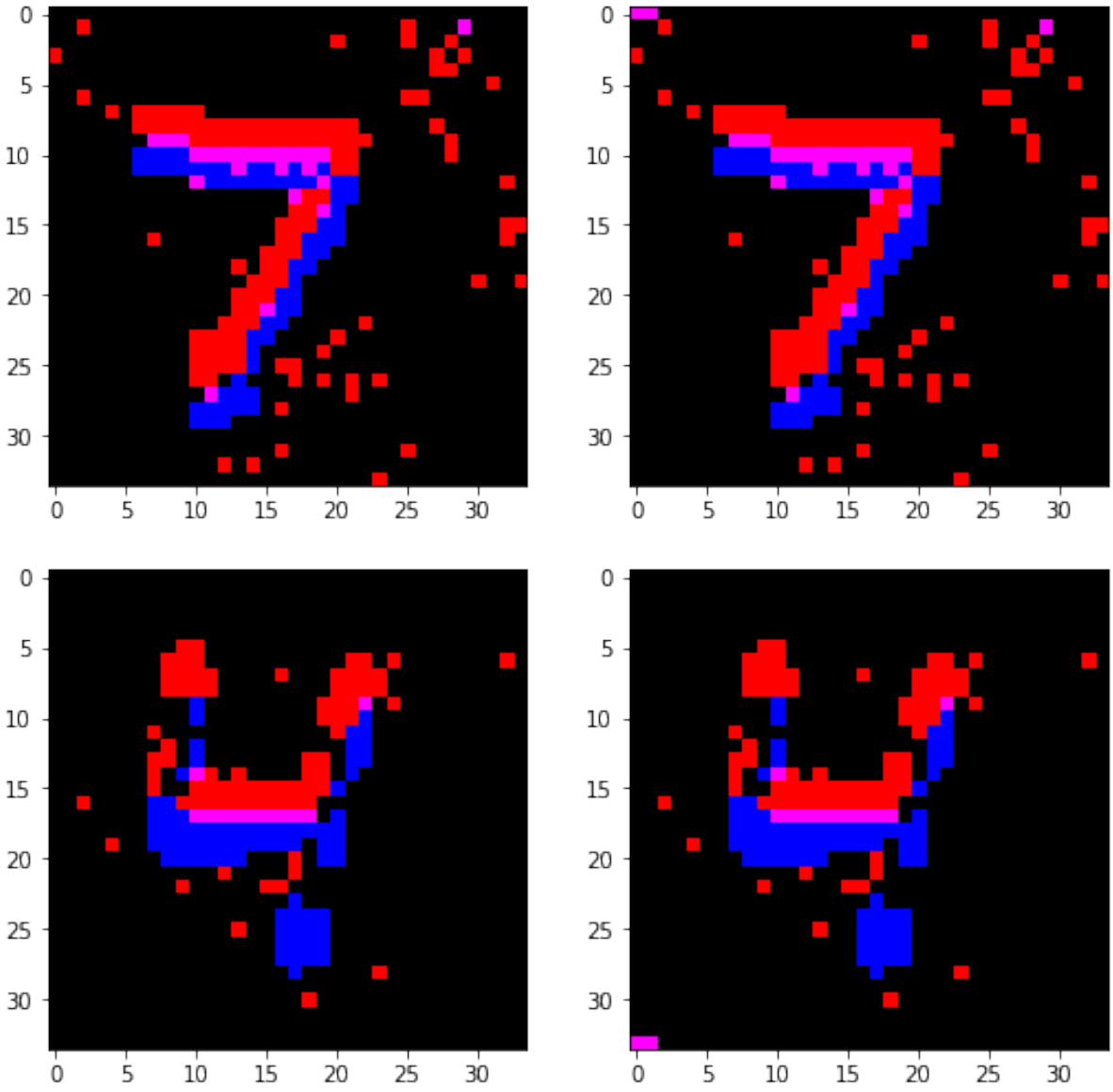

Figure 9.10: on the left: two images from NMNIST dataset. On the right: the same images after being targeted by the Corner Attack. The presence of noise makes the attack difficult to spot.

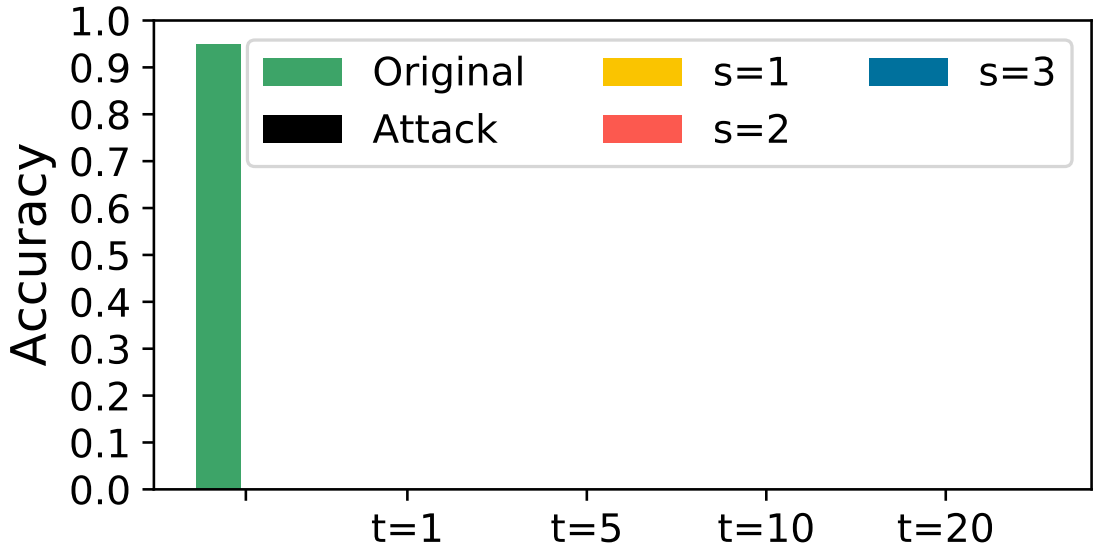

Figure 9.11: Corner Attack filtered with different BA filters: the filter does not lead to any improvement.

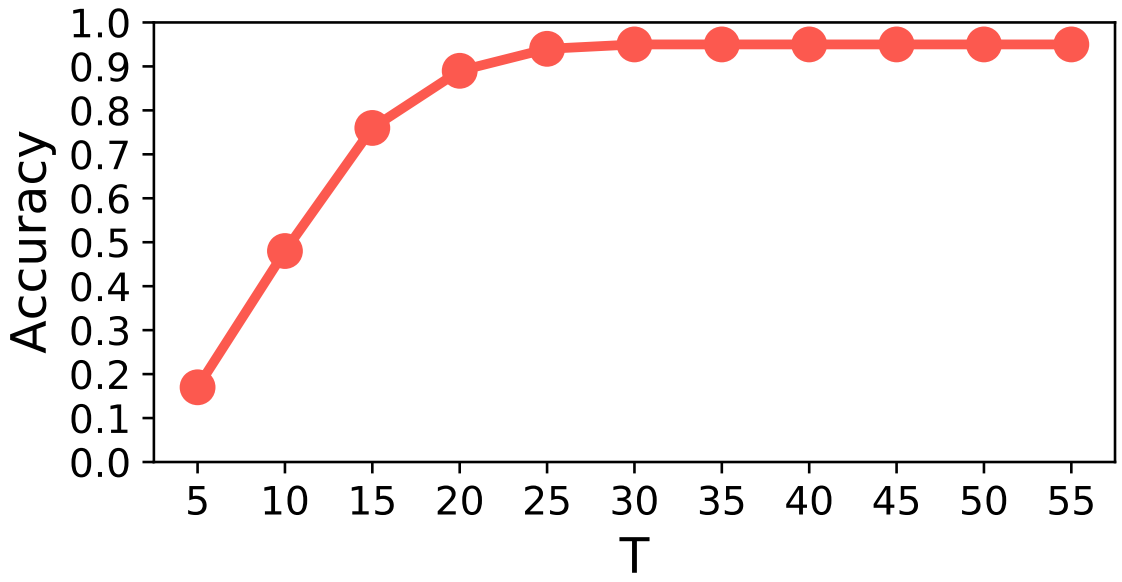

Figure 9.12: Corner Attack filtered with different Mask filters: the filter successfully restores the performances for a wide portion of T values.

9.3 Dash Attack

The Dash Attack, as mentioned in Sec. 8.3 derives from the Corner Attack. In the case of the NMNIST dataset, the difference among the two is not so marked, because the Corner Attack performed very well on every sample, reaching the complete corruption of the dataset after few iterations. On the contrary, when dealing with the DVS128 Gesture dataset, we noticed that some samples were more affected than others by the Corner Attack. This issue is solved by the Dash Attack, that affects all samples by the same amount.

9.3.1 IBM DVS128Gesture

After the attack the precision of the SNN drop from 92.04% to 0.00%. The adversarial attack generation time is slightly higher than in the case of the Corner Attack, but the final result is the same. Moreover, if we look at the two samples reported in Fig. 9.13, it can be seen that the perturbation introduced by the attack is so small that results invisible to the human eye. This demonstrates the power of this adversarial attack, which combines the effectiveness in deceiving the network, with the imperceptibility of the attack.

Background Activity Filter

If we compare the effect that the BA filter has on this attack, compared to the results obtained with the Frame and the Corner, we notice that they are slightly better. This is due to the fact that the introduced perturbation is limited to only two contiguous pixels and therefore the filter is able to work better than before. In any case, even if the recovery of performance is greater, this is still insufficient, reaching a maximum accuracy of 28.41% when $s = 1$ and $t = 10$. Such a value is too low, especially considering the use of SNN in safety-critical applications.

Mask Filter

As happened in the two cases mentioned above, the Mask Filter is able to completely cancel the effectiveness of the Dash Attack. By choosing a $T \geq 75$, the accuracy of the network is remains above 90%, reaching the original value of 92.04% with $T = 200$. The weakness of these attacks is the excessive increase in the activity of the pixels affected by the attack, for this we have developed another version that is aware of the presence of the filter and which aims to resist to it (see Chap. 10).

9.3.2 NMNIST

The behaviour with this Dataset is very similar to the one registered after the Corner Attack. A few iterations are enough to make the SNN completely unable to classify the samples correctly, in fact even in this case the accuracy drops to 0.0%. The attack is difficult to spot, confirming its excellent stealthiness.

Background Activity Filter

Compared to what happened with the other dataset, in this case we are faced with a completely different situation. The filter has no effect and does not bring any benefit in terms of improving the accuracy of the network. apart from the filter with $s = 3$ and $t = 20$ in fig. 9.17 that reaches 1%, all the other remain at 0.0%). As said before, the Dash Attack requires less iteration to converge when it is applied to the NMNIST than the DVS Gesture, therefore his behavior is very similar to the Corner Attack, which is why the results of the two attacks are almost coincident in this case.

Mask Filter

The filter behaves exactly as it does with the above discussed attacks. This was expected given the similarity with the *Corner Attack*. As before, every $t > 25$ is able to restore the original accuracy of the SNN.

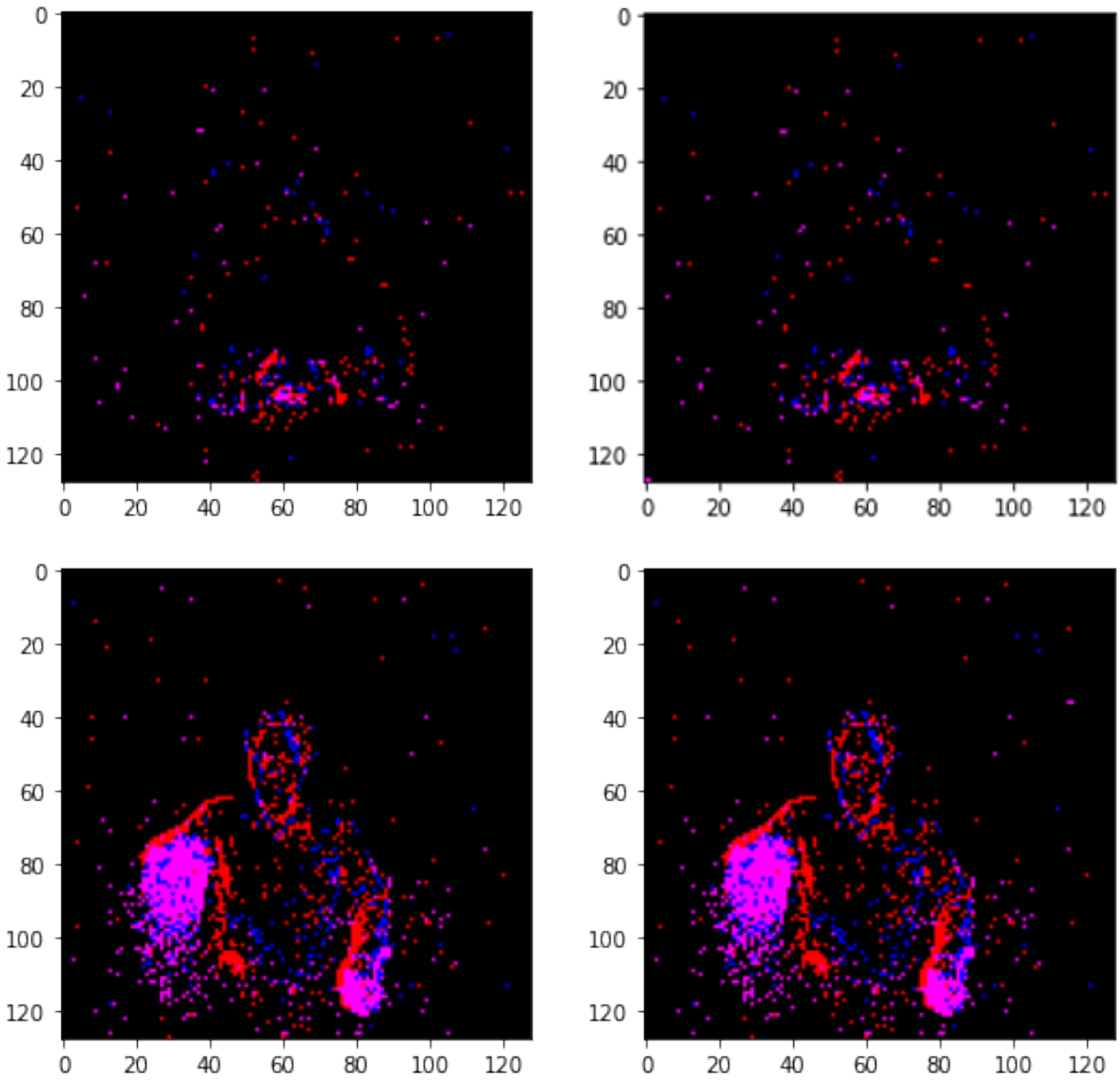

Figure 9.13: Two images from IBM DVS128Gesture targeted by the Dash Attack: the adversarial examples are undistinguishable from the originals.

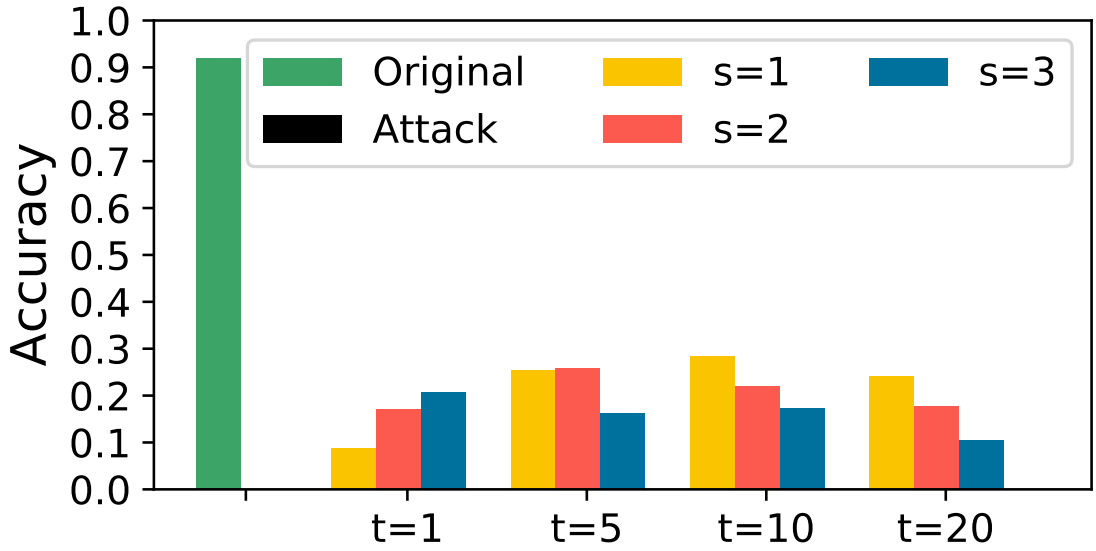

Figure 9.14: Dash Attack filtered with different BA filters: the filter achieves better results than before, but still not sufficient.

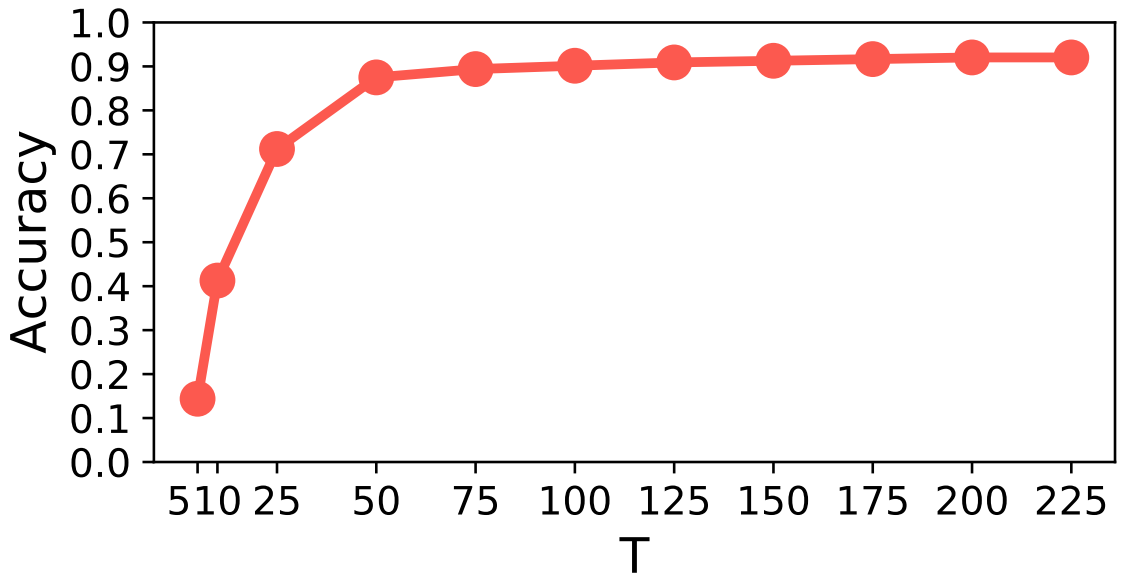

Figure 9.15: Frame Attack filtered with different Mask filters: the filter successfully restores the performances for a wide portion of T values.

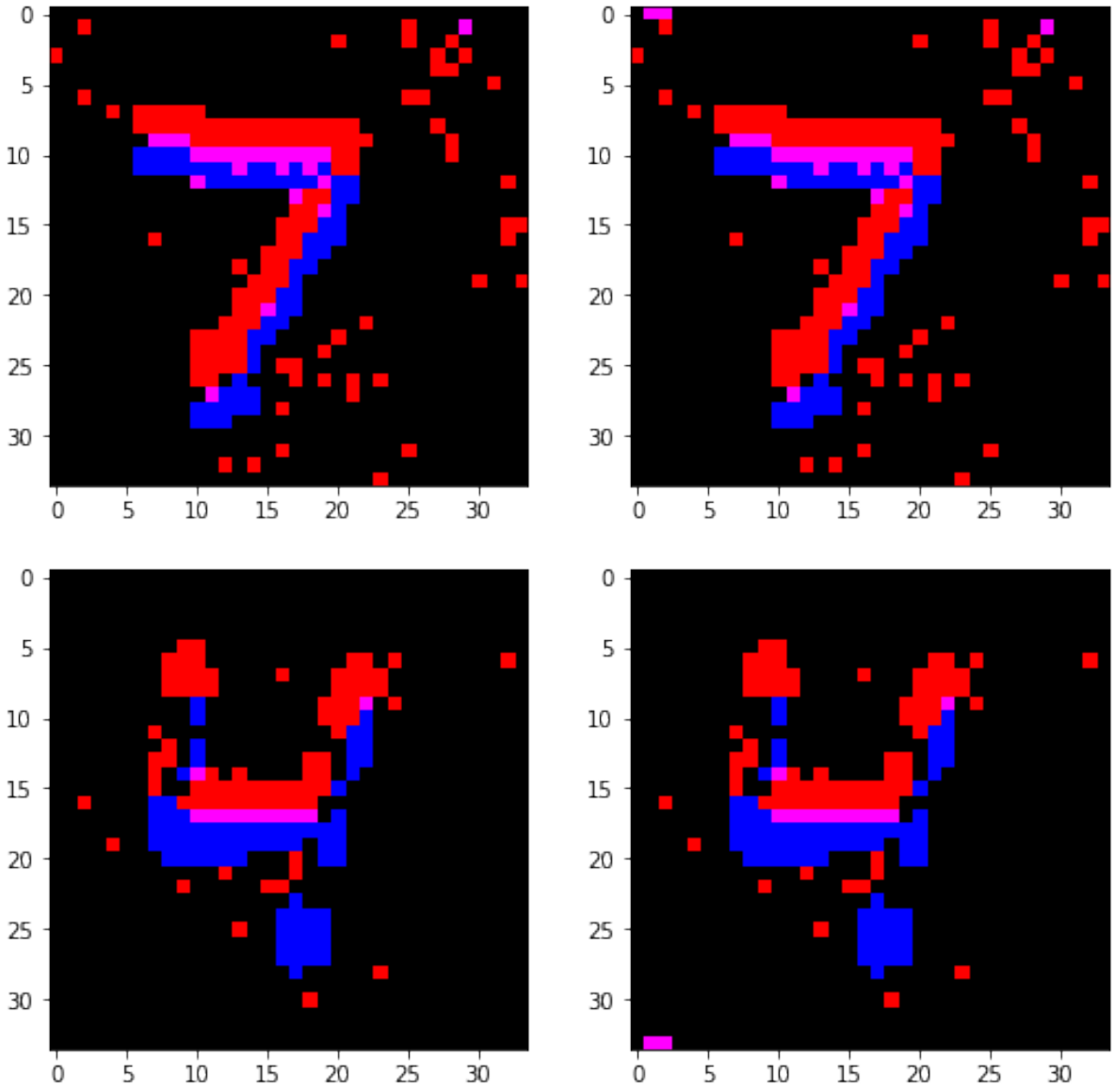

Figure 9.16: On the left: two images from NMNIST dataset. On the right: the same images targeted by the Dash Attack. The attack is difficult to distinguish because it blends in with the background noise.

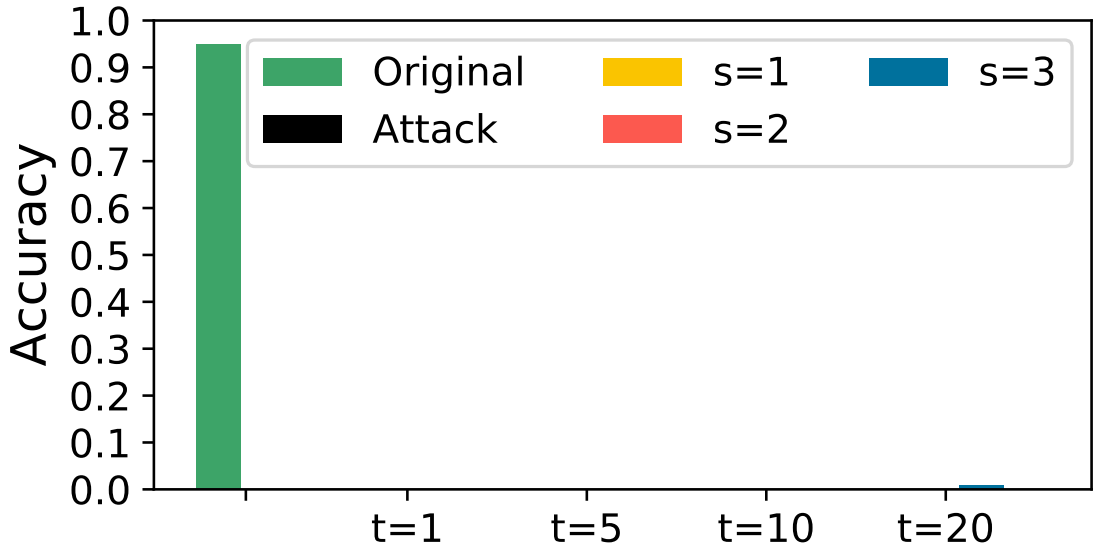

Figure 9.17: Dash Attack filtered with different BA filters: the filter is not able to counter the attack.

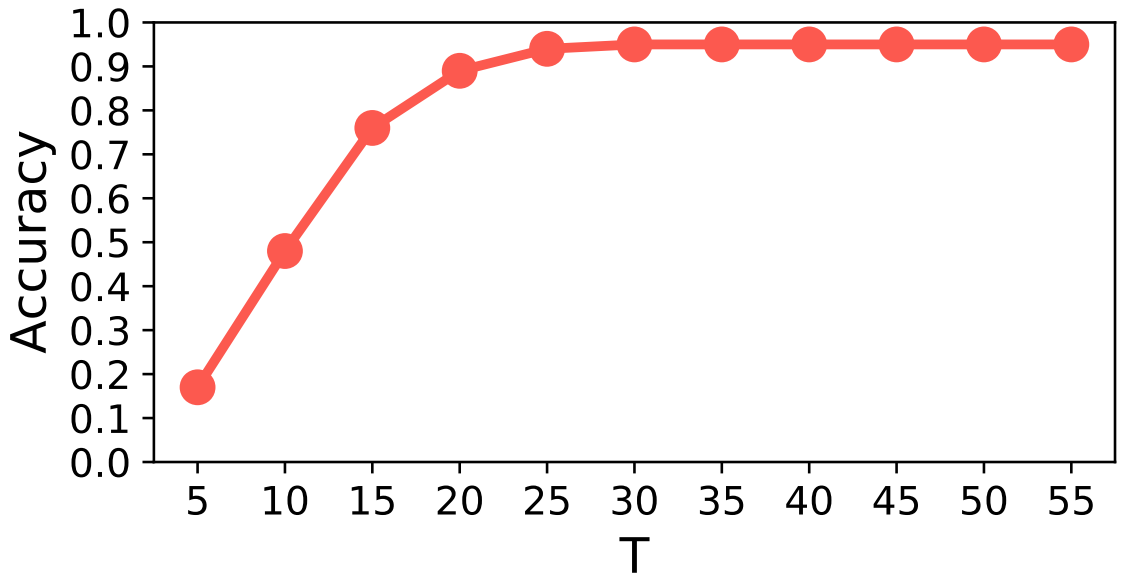

Figure 9.18: Dash Attack filtered with different Mask filters: the filter successfully restores the performances for a wide portion of T values.

CHAPTER 10

MF Aware Dash Attack

The attacks introduced in Chaps. 8 and 9, had the advantage of overcoming the Background Activity filter, starting from the awareness of not being able to attack scattered pixels but having to focus on neighbouring pixels in space (and time). However, it is true that the effectiveness of the attacks in case a Mask Filter is inserted is totally nullified. As already mentioned, the problem of the previous attacks was to indiscriminately attack all the frames of the samples, creating pixels whose activity was significantly higher than normal, and therefore easily identifiable by the filter, which was able to eliminate them without problems. If we add the fact that they focus on the boundaries of the image, thus they do not tend to cover the main subject, that is usually well-centred, we understand how the filter is able to completely restore the original accuracy of the SNN. To overcome this problem we took a feature of the first attack that had not been applied to the second round: *Sparsity*. The idea is to hit only a few frames in order to reduce the number of additional events and consequently the activity of the individual pixels, this should reduce the effectiveness of the Mask Filter, while maintaining the resistance to BAF, by keeping to attack contiguous pixels.

10.1 Implementation

The last attack we present therefore aims to resist the Mask Filter, for which it was called MF Aware Dash Attack. The attack structure, visible in the algorithm below (see Alg. 7), is similar to the Dash Attack's. In this case though there is an additional parameter: th , which represents the maximum threshold of frames that can be modified by the attack for the same pixel. This parameter is related to the T parameter of the Mask Filter, which imposes a limit on the maximum activity allowed to each pixel. Recalling briefly the filter mechanism (see Alg. 2), any pixel that produces a number of events greater than T is masked and its output does not reach the network. By passing this parameter to the attack, we provide the necessary information to evade the mask filter, at least if it has a $T \geq th$, otherwise, as we will see in the results section, the filter continues to work.

The operation of the attack is as follows: always starting from the corner, 2 adjacent pixels are perturbed, for th frame, then the attack selects the next two, and so on. The effect is that of a dash moving along a straight line, the smaller th the faster the dash appears to move. Furthermore, unlike before, only one channel is affected, leading to a further halving of the number of events produced per pixel. As before, all samples are initially affected by the same perturbation and are progressively removed from the list if the attack is successful. At each iteration the attack is repeated on the remaining samples, updating the starting position of the attack. As was the case for the Dash Attack, therefore, only two pixels per frame are affected, but in this case the pixels change over time, instead of always remaining the same.

Algorithm 7 : MF-Aware Dash Attack

Being D a neuromorphic Dataset made of $(2 \times N \times N \times T)$ tensors,where N represents the frame dimensions and T the sample duration
S is a list of the samples that compose D
$x = 0$, $y_0 = 2$, $left = True$
while S is not empty **do**
  for s in S **do**
    $Th = Th_0$, $y = y_0$
    for t in T **do**
      for i in range(N) **do**
        for j in range(N) **do**
          if $i == x$ **and** $t < Th$ **and** ($left$ **and** ($j == y$ **or** $j == y - 1$) **or** $\overline{left}$ **and** ($j == N - y$ **or** $j == N - y + 1$)) **then**
            $s[0, i, j, t] = 1$
          end if
        end for
      end for
      if $t == Th$ **then**
        $Th = Th + Th_0$, $y = y + 2$
      end if
    end for
    The perturbed sample s is fed to the SNN, which produces a prevision P
    if P is incorrect **then**
      Remove s from S
    end if
  end for
  if $x == 0$ **then**
    $x = N - 1$
  else
    $left = left$ **xor** 1 , $x = 0$
    if $\overline{left}$ **then**
      $y_0 = y_0 + 1$
    end if
  end if
end while

10.2 Results

10.2.1 IBM DVS128Gesture

Looking at Fig. 10.1, we can understand how difficult is to spot this attack, the two images in fact seems totally equivalent to the original ones. The perturbation is very small, but nonetheless it is able to sensibly decrease the network's confidence.
From our experiments we have been able to observe how the parameter th influences the effectiveness of the attack and the convergence time. In fact, as th decreases, the attack becomes less effective, leaving a higher number of correct samples, a number which however is limited to a few units. The accuracy drops to 1.89% when $th = 150$, a number which doubles itself to 3.78% when $th = 100$ and again to 7.95% when $th = 50$. These values are by no means sufficient to guarantee the level of performance required to the SNN and therefore present a not negligible safety hazard.

BAF

As we can easily see from the results obtained on the DVS Gesture dataset, shown in Fig. 10.2, the BAF gets quite good results compared to before, especially when $t = 1$ and $s = 3$. In this case, in fact, we obtain the maximum recovery for all three attacks conducted with $th = 50,\ 100,\ 150$, obtaining respectively $59.09\,\%,\ 54.92\,\%,\ 56.06\,\%$. The filter with $t = 1$ and $S = 2$ also works fairly well, restoring the SNN's accuracy above 40%. In the other cases, however, the results are not satisfactory, because in any case the accuracy does not exceed 32%, which is too low a value. Anyhow, the nature of this attack, which is more widespread than the previous ones, as it affects different pixels, seems to make it more vulnerable to the BAF, which identifies events that do not have a strong space-time correlation as noise.

MF

This attack was mainly developed to resist the Mask Filter, so it is reasonable to expect a different behaviour than in previous cases. If we look at the figure, we immediately notice how the attacks that have $th \geq t$ do not receive any benefit from the presence of *MF*, which on the contrary remains effective when $th < t$. When $t \geq th$, in fact the SNN accuracy peaks at 23.5% in the case of the attack with $th = 50$, while it does not overcome 15% when $th = 100,\ 150$. As mentioned, the effectiveness of the attack also depends on the t parameter, and decreases as t decreases. However, it is equally true that an attack with a lower t has a greater chance of not being opposed, being able to resist a greater number of filters, even those with a low th. When deciding which t to use, it should also be considered that filters with th below a certain threshold have a greater impact on the network, causing a decrease in performance (see Fig. 4.3), it is therefore unlikely that these filters are chosen as pre-processing units.

10.2.2 NMINST

Also in the case of the NMNIST dataset, the attack remains stealthy. As before, the dash moves along a line, with a speed that depends on th. Unlike what happened with the DVS Gesture dataset, this time the attack has a complete success, whatever the th. In fact, all the three attack considered, with $th = 20,\ 15,\ 10,$ lead to a final accuracy of 0%. As before though, the smaller th the longer the attack takes to converge.

BAF

Similar to what happened above, the filter gets the best results when $t = 1$. As we can see from the graph, in fact in this case the precision of the network is markedly

superior to the others. Unlike before, however, with this dataset the best result does not exceed 30 %, a number that remains too low to be considered acceptable. If then, we consider the filters with $t > 1$, the precision remains even lower, with a maximum of 10 % which makes the SNN completely useless.

The *BAF* therefore, which had proved a viable solution in mitigating the MF Dash aware attack with the DVS Gesture, fails to re-establish an acceptable level of performance with the NMNIST dataset.

MF

The behaviour of the filter is completely similar to that of the DVS Gesture. As you can see in the graph (Fig. 10.6) all filters with $T \geq th$ are not able to counter the attack and in this case the precision of the network does not exceed 2%. As before, if $t < th$ its behavior recalls that already shown by the above discussed attacks. However, the best *MF-Aware Dash Attack*, the one with $th = 10$, is able to lower the accuracy to only 22%, with a reduction of 73% with respect to the original one.

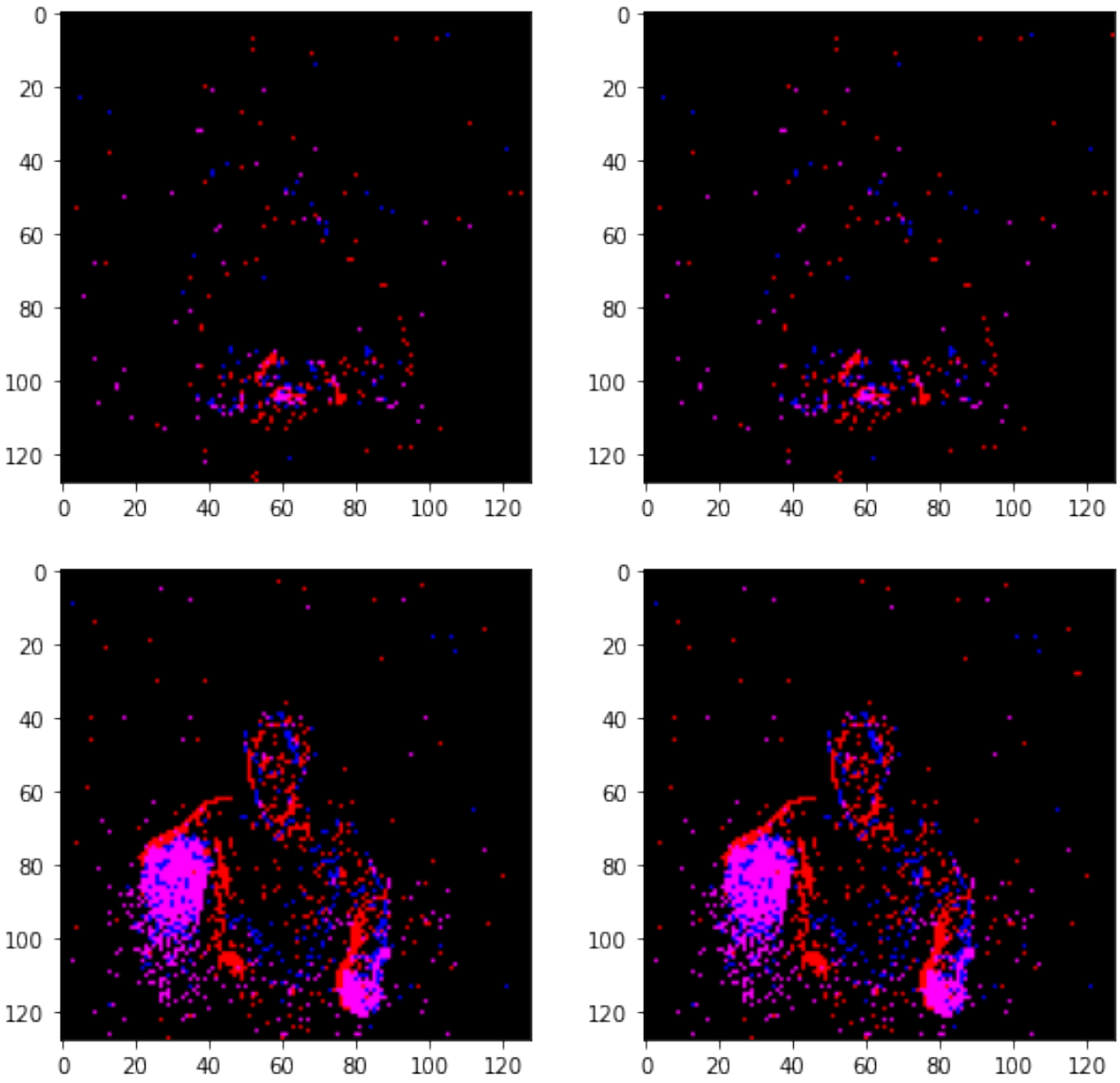

Figure 10.1: On the left: two samples from IBM DVS128Gesture dataset. On the right: the same samples after being attacked by the MF Aware Dash Attack, th=150.

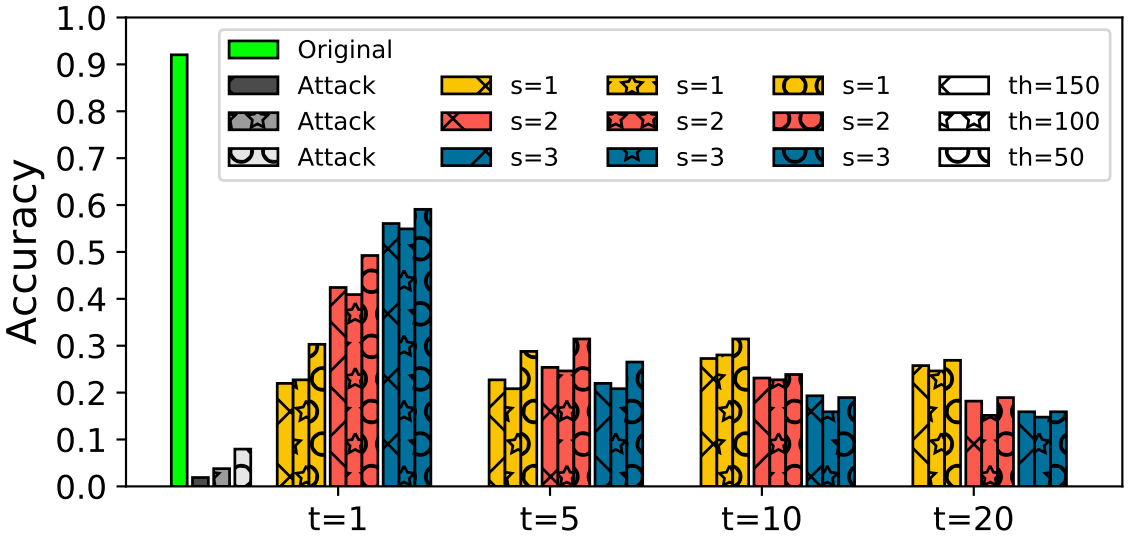

Figure 10.2: Various BAFs applied after 3 different MF Aware Attacks: results may be acceptable in same cases, but the lack of performances remains large.

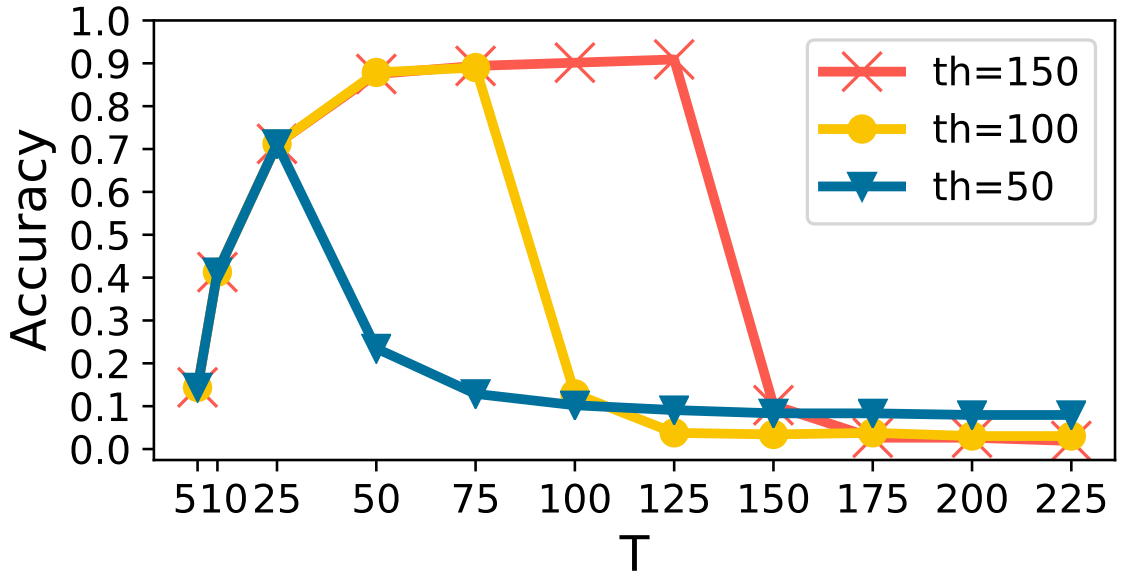

Figure 10.3: Various Mask Filters applied after 3 different MF Aware Attacks: it is clear how the filter ceases to work when $T \geq th$.

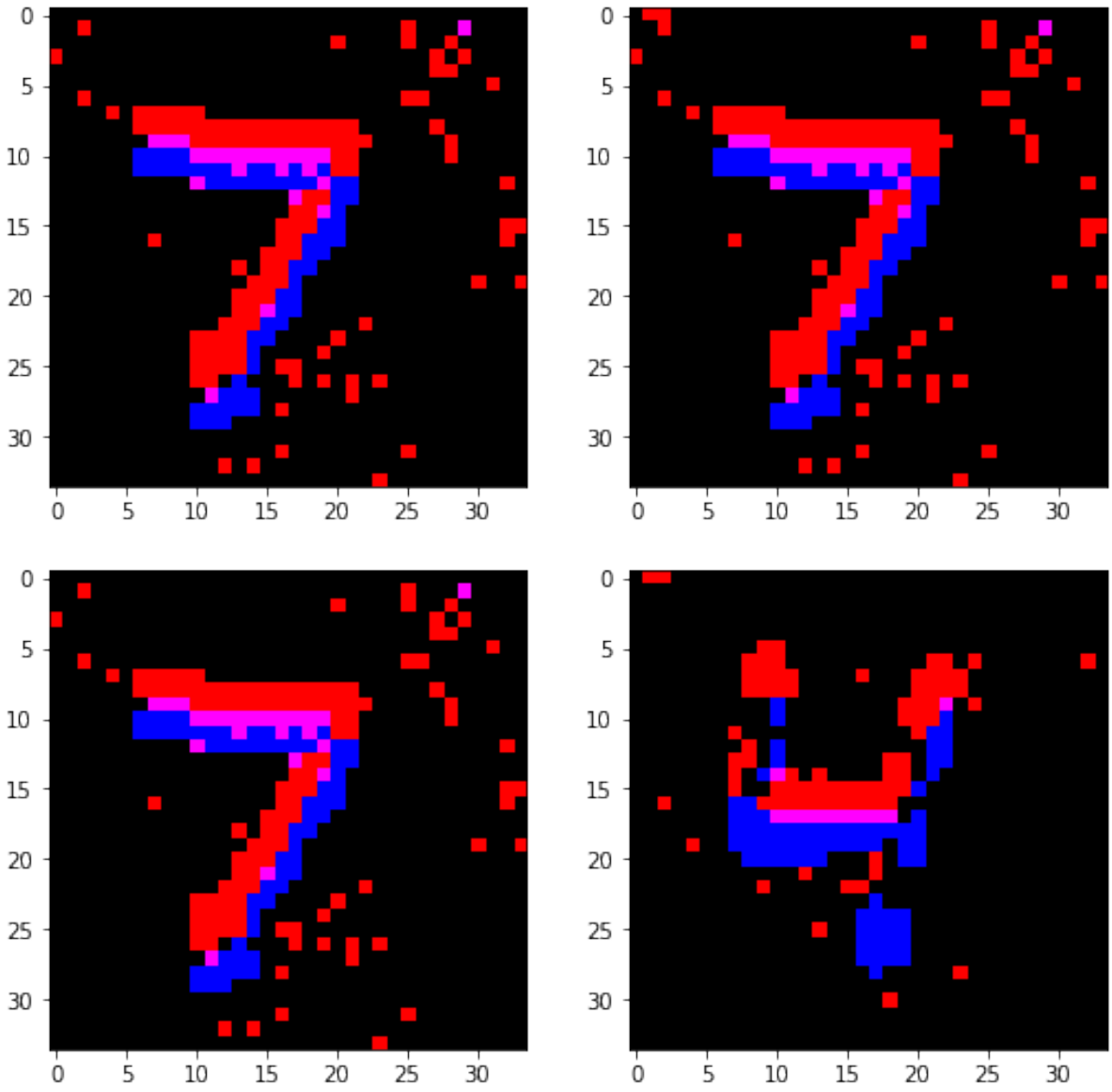

Figure 10.4: On the left: two samples from NMNIST dataset. On the right: the same samples after being attacked by MF Aware Dash Attack, th=20.

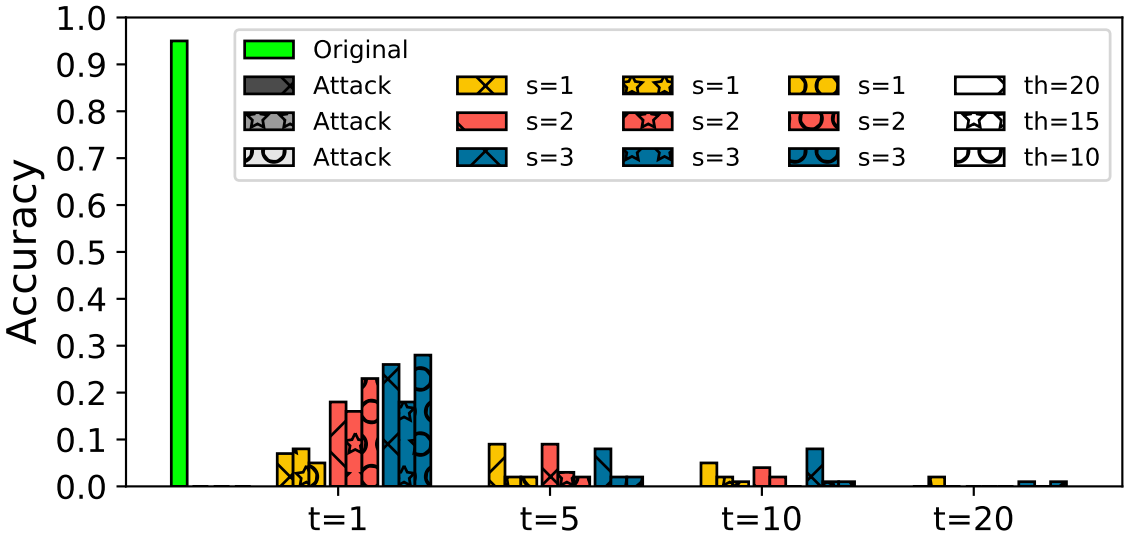

Figure 10.5: Various BAFs applied after 3 different MF Aware Attacks: results may be acceptable in same cases, but the lack of performances remains large.

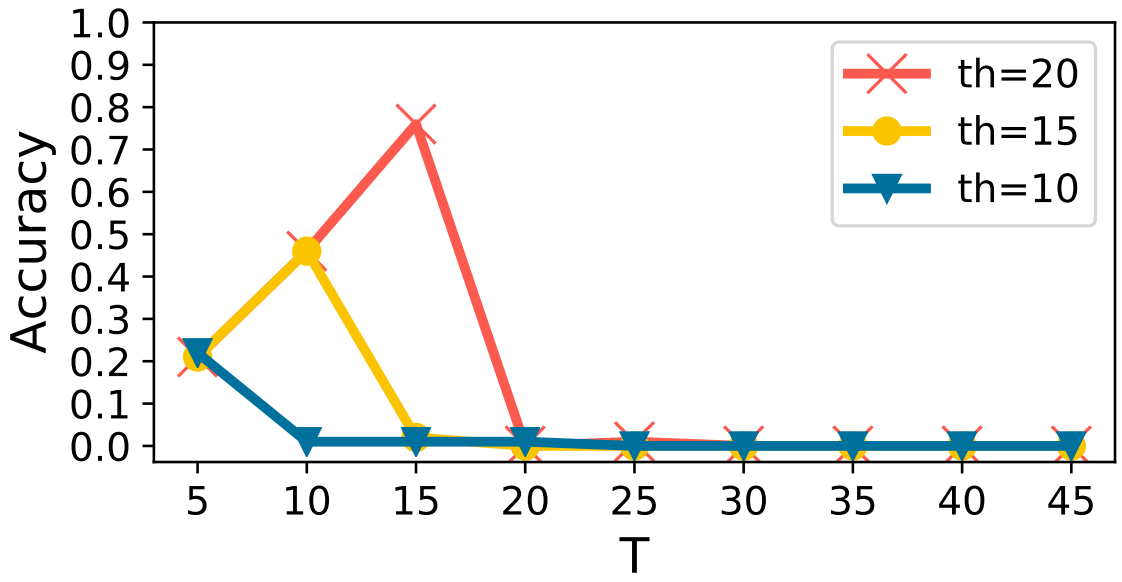

Figure 10.6: Various Mask Filters applied after 3 different MF Aware Attacks: it is clear how the filter ceases to work when $T \geq th$.

CHAPTER 11

Conclusions

Spiking Neural Networks (SNNs) represent a valid alternative for a low powered and energy efficient learning system, mostly when they are implemented on neuromorphic chips and coupled with event based cameras, such as Dynamic Vision Sensors (DVS). However, as we have demonstrated in this work, they suffer from the same security threats as other non-spiking Deep Neural Networks, in particular we have shown how they are vulnerable to adversarial attacks. In fact, all the attacks that we have developed have been able to significantly reduce and in some cases completely reset the SNNs' performances. The same networks that were originally able to recognize samples with an accuracy greater than 90% have therefore become completely useless after our attacks. The defense methodology we proposed, based on two types of noise filters for DVS cameras, however, proved to be effective in countering the aforementioned attacks. In fact, depending on the attack, the two types of filter proposed have proved to be more or less effective, but have always been able to bring the network back to adequate performance levels. As demonstrated by our experiments, in 4 out of 5 cases (Gradient Based, Frame, Corner and Dash Attacks), after the insertion of the filters the SNNs recover up to more than 90% of accuracy. The most resistant attack proved to be the MF Aware Dash Attack. This result is not surprising, considering that it was designed with the purpose of resisting against our filters. This attack represent a critical security threat for event-based SNNs, as it reduces the accuracy by more than 20% in the case of the DVS128 Gesture dataset and by more than 65% for the

NMNIST dataset, even in the presence of filters. This shows that, although effective, our defense system has limitations, and that before using SNNs in safety-critical applications, it would be advisable to investigate stronger defenses for further increasing their robustness.

Bibliography

[1] S. Zhou *et al.*, "Deep scnn-based real-time object detection for self-driving vehicles using lidar temporal data," *IEEE Access*, 2020.

[2] C. D. V. Gonzalez, J. H. S. Azuela, J. Antelis, and L. E. Falcón, "Spiking neural networks applied to the classification of motor tasks in eeg signals," *Neural networks*, 2020.

[3] G. Tang and K. P. Michmizos, "Gridbot: An autonomous robot controlled by a spiking neural network mimicking the brain's navigational system," *ArXiv*, vol. abs/1807.02155, 2018.

[4] F. Ponulak and A. Kasiński, "Introduction to spiking neural networks: Information processing, learning and applications," *Acta neurobiologiae experimentalis*, 2011.

[5] M. Capra *et al.*, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access*, 2020.

[6] L. Deng *et al.*, "Rethinking the performance comparison between snns and anns," *Neural networks*, 2020.

[7] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, 2014.

[8] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, 2018.

[9] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120db 30mw asynchronous vision sensor that responds to relative intensity change," in *ISSCC*, 2006.

[10] R. Paz, F. Gomez-Rodriguez, M. A. Rodriguez, A. Linares-Barranco, G. Jimenez, and A. Civit, "Test infrastructure for address-event-representation communications," in *Computational Intelligence and Bioinspired Systems* (J. Cabestany, A. Prieto, and F. Sandoval, eds.), (Berlin, Heidelberg), pp. 518–526, Springer Berlin Heidelberg, 2005.

[11] A. Bagheri, O. Simeone, and B. Rajendran, "Adversarial training for probabilistic spiking neural networks," in *SPAWC*, 2018.

[12] A. Marchisio *et al.*, "Is spiking secure? a comparative study on the security vulnerabilities of spiking and deep neural networks," in *IJCNN*, 2020.

[13] S. Sharmin *et al.*, "A comprehensive analysis on adversarial robustness of spiking neural networks," in *IJCNN*, 2019.

[14] L. Liang *et al.*, "Exploring adversarial attack in spiking neural networks with spike-compatible gradient," *ArXiv*, vol. abs/2001.01587, 2020.

[15] A. Madry *et al.*, "Towards deep learning models resistant to adversarial attacks," in *ICLR*, 2018.

[16] H. Zhang *et al.*, "Theoretically principled trade-off between robustness and accuracy," in *ICML*, 2019.

[17] F. Khalid *et al.*, "Fademl: Understanding the impact of pre-processing noise filtering on adversarial machine learning," in *DATE*, 2019.

[18] A. Linares-Barranco *et al.*, "Low latency event-based filtering and feature extraction for dynamic vision sensors in real-time fpga applications," *IEEE Access*, 2019.

[19] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[20] R. Brette, "Philosophy of the spike: Rate-based vs. spike-based theories of the brain," *Frontiers in Systems Neuroscience*, vol. 9, p. 151, 2015.

[21] D. Beeman, *Hodgkin-Huxley Model*, pp. 1–13. New York, NY: Springer New York, 2013.

[22] Z. WANG, L. GUO, and M. ADJOUADI, "A generalized leaky integrate-and-fire neuron model with fast implementation method," *International Journal of Neural Systems*, vol. 24, no. 05, p. 1440004, 2014. PMID: 24875788.

[23] R. El-Allami, A. Marchisio, M. Shafique, and I. Alouani, "Securing deep spiking neural networks against adversarial attacks through inherent structural parameters," 2020.

[24] S. B. Shrestha and G. Orchard, "SLAYER: spike layer error reassignment in time," *CoRR*, vol. abs/1810.08646, 2018.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.

[26] S. Sharmin, P. Panda, S. S. Sarwar, C. Lee, W. Ponghiran, and K. Roy, "A comprehensive analysis on adversarial robustness of spiking neural networks," 2019.

[27] L. Liang, X. Hu, L. Deng, Y. Wu, G. Li, Y. Ding, P. Li, and Y. Xie, "Exploring adversarial attack in spiking neural networks with spike-compatible gradient," 2020.

[28] V. Venceslai *et al.*, "Neuroattack: Undermining spiking neural networks security through externally triggered bit-flips," in *IJCNN*, 2020.

[29] X. Wei, J. Zhu, and H. Su, "Sparse adversarial perturbations for videos," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 03 2018.

[30] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120db 15us latency asynchronous temporal contrast vision sensor," vol. 43, pp. 566–576, 01 2007.

[31] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," in *CVPR*, 2017.

[32] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop on Autodiff*, 2017.

[33] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, p. 437, 2015.

[34] K. Zolna, M. Zajac, N. Rostamzadeh, and P. O. Pinheiro, "Adversarial framing for image and video classification," 2019.