Investigating Spiking Neural Networks for Energy-Efficient On-Board AI Applications. A Case Study in Land Cover and Land Use Classification

Andrzej S. Kucik, Gabriele Meoni*
European Space Agency
{andrzej.kucik, gabriele.meoni}@esa.int

Abstract

Spiking neural networks have been attracting the interest of researchers due to their potential energy efficiency. This feature makes them appealing for applications on board CubeSats or small Earth observation satellites, given their strict energy consumption requirements. However, the performance of spiking neural networks in terms of the accuracy on the space-scene classification datasets, and their benefits with respect to the energy efficiency still remain to be demonstrated. This work is a preliminary investigation on deploying spiking neural networks to land cover and land use classification problems. To train a spiking model, a VGG-16-based artificial neural network has been trained on EuroSAT RGB benchmark dataset. The parameters of this network were then used to initialise a spiking model, which was optimised by fine-tuning the connection weights and the synaptic filters parameters. By using the mean neuron activations, and the number of time-steps and their width as proxies for the energy consumption of the models, this study shows different trade-offs between accuracy, and energy efficiency, when comparing a spiking model to the deep learning approach. Moreover, some additional input data preprocessing strategies are investigated as a method of further enhancement of the energy benefits of the spiking models.

1. Introduction

Over the last years, researchers in the space community have been exploring the possibility of applying Artificial Intelligence (AI), and in particular deep Artificial Neural Networks (ANNs), on board the Earth Observation (EO) satellites [15, 20, 30, 12]. The applications of ANNs on-board include early-detection of natural disasters [15], filtering and discarding the data to mitigate downlink bandwidth requirements of small EO satellites, which has become stricter in the last years because of the growing resolution of space sensors [15, 41]. Nevertheless, one of the factors limiting the deployment of ANN models is the restricted energy-budget of *SmallSats*[7, 15]. This fact forces the research of complex design trade-offs, optimizing both the algorithms and hardware to perform the inference on the edge [20, 33, 37].

In this work, we propose to investigate Spiking Neural Networks (SNNs) [19, 40] for an on-board AI applications, which have been gaining the attention of researchers for their promising energy efficiency [36, 29]. This is due to the asynchronous nature of their neuromorphic computational paradigm. Indeed, according to a biologically-inspired model, each neuron of an SNN takes as the input the postsynaptic currents produced by the upstream neurons and combines them through connection weights. The resulting current is accumulated during different time-steps, altering the value of the membrane voltage. When the latter passes a fixed threshold, the membrane voltage is reset, and a spike is emitted. Such spike is processed by a post-synaptic filter, producing the output postsynaptic current. Since various neurons can emit a spike in different instants, the computation of an SNN model is normally sparse [29]. When inferred on neuromorphic hardware (for example, SpiNNaker [16, 17, 34], TrueNorth [39], or Loihi [10]) such sparsity is generally advantageous in terms of power consumption [22]. In addition, depending on the data format [6], and the information coding approach [36], and hardware implementation [6], SNNs may also provide benefits in terms of energy efficiency when compared to their artificial counterparts. In particular, whilst for the event-based input the energy benefits of spiking models are clear, the advantages of SNNs for static input images – such as those used for many EO applications – seem strictly dependent on the input data [6, 22]. Furthermore, even if the previous applications of SNNs to remote sensing exist [45, 5, 44], more research is needed to assess their practicality for space deployment.

In view of that, we propose an investigation of the relevancy of spiking models for on-board AI applications. In particular, we focused on land cover and land use classification

*Ordered alphabetically, equal contribution.

case study for the availability of benchmark datasets such as EuroSAT [23, 24] and benchmark solutions, which might foster future research.

In addition, the EO data about the land use is collected in order to better understand how human activity affects our planet and to help us solve the problems that said activity may cause, for example, the human-caused climate change. The reputation of the tremendous success of deep learning helps it propagate across different scientific domains and to be employed in a variety of problems. The EO research is not an exception here. However, this success story is not without shortcomings. The cost of growing deep neural networks [9, 21] is increased intensity of computation [2], and consequently – raise in power consumption and CO_2 emission [43]. This leads to a paradox, where to solve problems related to global warming, we use the tools which directly contribute to it – in a non-negligible way. Thus, an alternative approach to AI is desperately needed, and the SNNs, as well as the neuromorphic hardware designed for their implementation, may be just the toolbox that we need.

To this aim, we trained a VGG-16-based ANN model [42] on the EuroSAT RGB [23, 24] benchmark dataset. The resulting network parameters were used to initialise the weights of the spiking version of VGG-16 model, which was re-trained in the spike-domain using KerasSpiking framework [3].

To explore different trade-offs in terms of accuracy and energy efficiency, various levels of spiking regularisation was applied to the model, outlining many compromises that can be made in order to satisfy potential applications' constraints. The original ANN network and the various SNN models were compared in terms of accuracy and energy per inference, which can be roughly estimated on different hardware platforms through the approach described in Section 2.4.1. Moreover, to further investigate the dependence of energy efficiency on the input data, we proposed an image preprocessing approach relying on Prewitt filtering and input quantisation to foster data-sparsity and reduce the neuron activation rate.

Even if the strong assumptions made to estimate the energy per inferece make it impossible to claim the higher energy efficiency of SNNs with absolute certainty, the obtained results suggest that SNNs might have a potential for some AI applications, despite the reduction in accuracy compared to the ANN model.

The remainder of the paper is structured as follows: Section 2 describes the EuroSAT RGB dataset, the SNN and ANN models, the training framework and the approach used for the energy estimation. Section 3 provides the results in terms of accuracy and energy per inference and proposes a preprocessing pipeline based on Prewitt filtering and input quantisation that further improves our results. Section 4 outlines the main limitations of this work and proposes future improvements. In Section 5 we give our conclusions.

2. Methodology

2.1. The model

To demonstrate that an SNN deployed on a neuromorphic computing platform can achieve better inference energy efficiency than an ANN implemented on standard hardware, we initiated the experiments with one of the standard deep learning models – VGG-16 network [42], which we use as both: to set up the SNN topology and parameters, and to use it as a performance baseline (in terms of land cover and land use classification accuracy and single-image inference energy consumption). The VGG-16 network, albeit already dated and falling short in performance when compared to the contemporary state-of-the-art models, offers certain advantages in the SNN-ANN conversion procedure. First, it does not have batch normalization layers (invented after VGG architecture was published [27]) and skip connections, both of which are problematic to realise in the spiking paradigm; batch normalisation, because it involves global computation, and skip connections, because they require further synchronisation of spikes moving in parallel across a different number of layers. And secondly, VGG-16 is a relatively large model (over 14 million parameters), so it offers a reliable insight into the transferability of the presented methodology. That is, very often the ANN-SNN schemes are only implemented on elementary datasets, such as the classic MNIST (e.g. [28]), and executed with shallow networks, which does not guarantee the scalability to more complex datasets and model architectures.

In all our experiments we have used TensorFlow 2.4.1 [1] and KerasSpiking [8] software and two NVIDIA Quadro RTX 8000 Graphical Processing Units (GPUs). All our results are reproducible, with the code to run the experiments available at [31].

2.2. The dataset

To demonstrate the feasibility of employing spiking models in the context of EO, we selected the EuroSAT RGB land cover classification library [23, 24] as our task dataset. The original dataset consists of 27,000 64×64 Sentinel-2A satellite images covering 13 spectral bands and consisting of 10 classes, each represented by 2000-3000 images. Of those 13 bands 3 corresponding to the RGB colour channels are selected, so that there is compatibility with the original VGG-16 expected input channels.

We split the dataset into training, validation, and tests sets using 80%:10%:10% ratio. We normalise the images to the unit interval, and augment them by applying random zoom ($\pm 5\%$), crop, dihedral transformation, change of brightness ($\pm 20\%$), contrast (by a factor $c \in [.2, 1.8]$), hue ($\pm 10\%$), and saturation (by a factor $s \in [.9, 1.1]$), in order to avoid

over-fitting during the training.

2.3. Training procedure

2.3.1 ANN training

We use the VGG-16 pre-trained on ILSVRC-2012 (ImageNet) [13]. We remove the top three dense layers and replace them with global pooling and a single 10 output-units dense layer without bias. We swap the max pooling layers with average pooling layers because the former require lateral node connections in the spiking context, and there is no consensus on how they should be implemented in that setting.

The network is then trained for 1000 epochs, on batches of 2700 examples (8 training steps per epoch), using RMSProp optimizer [25] with 10^{-3} learning rate (reduced by a factor of 10^{-1} after 50 epochs of no improvement in the validation loss). We also use 10^{-3} kernel ℓ^2, and 10^{-4} bias ℓ^1 normalization for the convolutional layers. The final test set accuracy that we achieve is 95.07%. This is expectantly below the 98.57% reported in [24], because of the superior ResNet-50 architecture used therein. However, our aim was never to improve upon that result, but rather to show the performance transferability between ANN and SNN models in this context.

2.3.2 SNN conversion

After training the ANN model, we transform it into a spiking model in the following way. First, we remove the pooling layers from the model (apart from the final global pooling), by noting that a 3×3 stride-1 same-padding convolution followed by ReLU activation, and 2×2 stride-2 average pooling is equivalent to 4×4 stride-2 same-padding convolution followed by ReLU activation, with kernel parameters appropriately adjusted. This reduces the number of operations in the model and shortens the path that each spike has to traverse from the input to the output of the network. Next, we replace the ReLU functions with spiking activations corresponding to the one of an Integrate and Fire (IF) neuron, and a low-pass filter with a parameter τ. That is, the layer spiking rate (in Hz) is equal to the complementary ReLU activation value, and

$$y(t) := \left(1 - e^{-\Delta t/\tau}\right) x(t) + e^{-\Delta t/\tau} x(t-1),$$

where $x(t)$ is the output of the spiking activation at time t, $y(t)$ is the output of the low-pass postsynaptic filter, and Δt is the temporal resolution of the network (initially we fix it at $\Delta t = 1$ms). The conversion procedure is depicted in Figure 1.

If we also scale the firing rate of the neurons by a factor of 250 and let $\tau = 0.005$ we get that the spiking neural network achieves 94.59% classification accuracy after 200ms, without any further training. This is negligibly worse than the performance of the ANN. However, it comes at a non-trivial computational cost, and thus consumes a significant amount of energy during the inference. This is because, even though the higher firing rate of the neurons need not to increase the actual number of spikes (we can apply a linear scale to the input of all the neurons, and then divide their output by the same scale factor), the simulation length $T = 200$ steps requires a lot of accumulate operations, adding a hefty operational cost (see 2.4). Thus, our next step was to fine-tune the SNN in order to reduce the number of operations that it has to perform, without a serious drop in the classification accuracy performance.

2.3.3 Integrate and Fire neuron model

As described in Section 2.3.2, the SNN conversion procedure converts the ReLU activation of the ANN network with the spiking activation implemented in KerasSpiking, and add postsynaptic low-pass filter modelling the dynamics of neural synapses. In this way, each block composed of a convolutional layer, spiking activation and low-pass filter is equivalent to a layer of IF neurons with postsynaptic filter. For the sake of simplicity, let us consider the case of fully-connected IF neurons. At time t a neuron N takes as input the postsynaptic current $J_M(t)$ from an upstream neuron M, as described in Equation 1 [28]:

$$J_M(t) = h_M(t) \circledast \sigma_M(t), \tag{1}$$

where $h_M(t)$ is the synaptic filter impulse response, $\circledast$ is convolution operation, $\sigma_M(t) = \sum_k \delta(t - t_{M,k})$ is the postsynaptic voltage spike train emitted by the neuron M, and $t_{M,k}$ is the the k^{th} firing instant of the neuron M. All the postynaptic currents J_M are linearly combined through the weights $w_{N,M}$, and a bias b_N is added to produce the total presynaptic current $J_{N-\text{in}}(t)$ as input to the neuron N. At time t, the $J_{N-\text{in}}(t)$ is added to the membrane voltage $V_N(t)$ (multiplied by a dimensional constant equal to 1 Ω) of the neuron N in order to update it [28], as shown in Equations 2:

$$\begin{cases} J_{N-\text{in}}(t) = \sum_M w_{N,M} \cdot J_M(t) + b_N \\ V_N(t) = V_N(t - \Delta t) + J_{N-\text{in}}(t), \end{cases} \tag{2}$$

where Δt is the time-step width. When $V_N(t)$ exceeds some threshold (usually set to 0 for IF neurons), $V_N(t)$ is reset and an output spike is generated by the neuron. In particular, at every time t, in a Δt interval, a neuron can produce more than one spike.

2.4. Spiking neural network objectives

2.4.1 Energy per inference estimation

To estimate energy per inference we adopted the approach exploited in KerasSpiking [8]. According to this method,

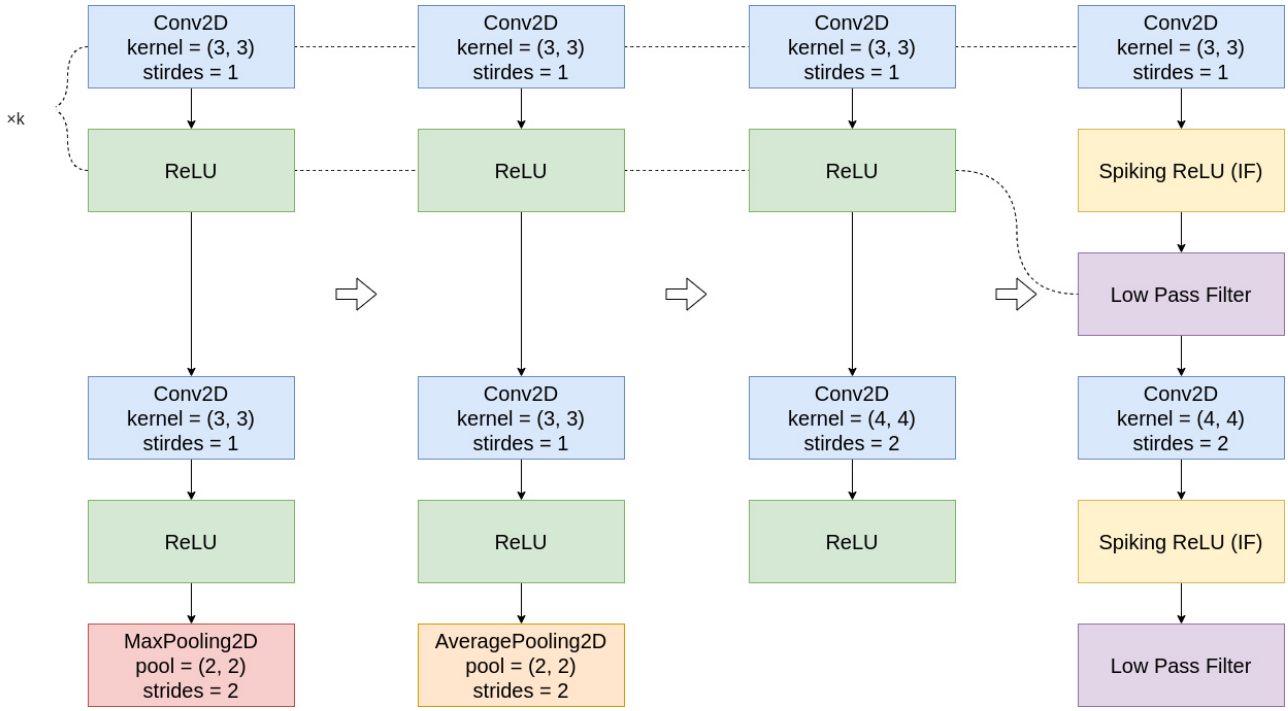

Figure 1. Flowchart showing the conversion of the VGG-16-based ANN convolutional block to an SNN convolutional block. From left to right. First, the max pooling layers in the original network are replaced with average pooling, and the network is trained. Then, the pooling layers are removed, and the preceding convolutional layer has its weights, kernel size, and strides altered accordingly, in order to preserve the expected output consistency; this version is used for estimating the energy consumption of the ANN model. Finally, the ReLU activations are replaced with spiking ReLU and a low-pass filter. The first convolution and activation in each block are repeated k times, where $k \in \{2, 3\}$. The network is concluded with global average pooling and a dense layer classifier with softmax activation.

the total energy per inference is given by two main contributions: the *energy per synaptic operations*: E_s and the *Energy per neuron update*: E_n.

Specifically, for a spiking model E_s is the energy spent to update the input current of a neuron by weighting the post-synaptic currents from upstream neurons over the different time-steps [32, 14]. It is reasonable to assume that no energy is spent on a synapse update in a time-step if no current is provided as input. Because of that, for spiking models, E_s can be reduced by decreasing the average neuron firing-rates. On the contrary, for an artificial model, E_s is the energy required to linearly combine all the activations from upstream neurons through weight multiplication. But unlike for SNNs, ANN synaptic operations are only carried out once per each connection to perform the inference.

E_n is the energy required to update the neuron's activation. For an artificial neural layer, it is the energy required to calculate the activation on the input.

For a spiking neural layer, E_n is the energy required to update the neuron by performing the membrane potential updates and to generate the postsynaptic current. Thus, it is reasonable to assume that E_n also includes the energy spent on synaptic filtering.

E_n and E_s were measured by adding the contribution of all convolutional, and global average pooling layers. In particular, for each layer, we estimated

$$E_s = E_o \cdot S$$
$$E_n = E_u \cdot U,$$

where E_o and E_u are respectively the *energy per synaptic operation* and the *energy per neuron update* on the target hardware platform, whilst the S and U are the number of synaptic operations and neuron updates per inference.

The values of E_o and E_u provided in KerasSpiking, citing [11, 26, 10], for the hardware platforms considered here.

For Central Processing Units (CPUs) and GPUs, the energy required for a Multiply and ACcumulate (MAC) is used, assuming that a MAC operation is required for a synaptic operation and a neuron update. The values E_o and E_u for the above types of hardware are summarised in Table 1.

The number of synaptic operations $S(L)$ for a layer L was estimated according to Equation 3:

$$S(L) = \sum_{N} S_{\text{neuron}} \cdot f_{\text{in}} \cdot T \cdot \Delta t, \tag{3}$$

where S_{neuron} is the number of synaptic operations per neuron, matching the average number of connection per neuron,

Device	E_o [nJ]	E_u [nJ]
CPU [11] (Intel i7-4960X)	8.6	8.6
ARM [11] (Cortex-A)	0.9	0.9
GPU [11] (NVIDIA GTX Titan Black)	0.3	0.3
Spinnaker [26]	13.3	26
Spinnaker 2 [26]	0.45	2.19
Loihi [10]	0.02711	0.081

Table 1. E_o and E_u (in nanojoules) for the different hardware platforms.

f_{in} is the average spiking rate at the input layer of L, T is the number of timesteps, Δt is the time-step width, and the sum is taken over all the neurons N in L. When L is a convolutional layer, $S(L)$ depends on the position of the neuron in the layer. Indeed, if padding is used, neurons in the corner have a lower number of input connections than the internal ones. For the sake of simplicity, we assume that the number of input neurons is uniform across all the neurons N of L, leading to an overestimation of the energy required.

The number of neuron update per inference U for a layer was estimated as in Equation 4:

$$U = |N|T, \tag{4}$$

where $|N|$ is the cardinality of the set N (i. e. the number of neurons). For the artificial model, we take $T = 1$ and $f_{\text{in}} = 1/\Delta t$, making the energy independent of the values of T and Δt. For the spiking model, T and Δt are produced by the training procedure.

To estimate f_{in} of each layer, we performed the inference of the model on the 2700 test images, and we measured the average value of the layers' activations over the simulation time. The average activation value of the input layer is used for the first convolutional layer and the average low-pass filter output is used for the remaining layers and the global average pooling, which in turn is used to estimate the f_{in} of the final dense layer.

Note that our method relies on the assumption that synaptic filtering is performed as the last operation for a neuron. As a result, the output spikes of a neuron have different heights, forcing the synaptic operation to be a MAC operation, instead of only an accumulation. However, this also reduces the frequency of synaptic operations thanks to the low-pass filter.

Alternative implementations can exploit the synaptic filter as the first or the second operation performed by a neuron. Consequently, each neuron collects the spike current trains having unitary height as input from the upward neurons, which are then processed by the low-pass filter. This way, synaptic updates require only to accumulate spikes, but the frequency of synaptic operation is generally higher. Both implementations are possible to realise on hardware, with different advantages and drawbacks. In this work, we assumed that only the first approach is used for all the hardware platforms, even if they allow for the second approach to be implemented as well.

Finally, it is necessary to point out that the method used does not aim to provide the exact estimation of the energy per inference, nor can it. Indeed, it does not consider additional sources of power consumption such as the static power contribution, networks on chip congestion, accesses to off-chip memories, and other effects [8] that might significantly affect the estimation performed. Instead, it wants to be used as a tool to compare the different solutions depending on the values of T, Δt, and input preprocessing, indicating which setting may lead to a higher gain in energy efficiency when used in a real case scenario.

2.4.2 Spiking-aware training

Bearing these assumptions in mind, we have a new training objective for the network: energy efficiency. We aim to reduce the spiking rate of the neurons and the temporal resolution of the simulation while keeping the total simulation length below some threshold. We initialise the network as outlined in 2.3.2, setting $T = 1$ steps, $\tau = 0.1$, and $\Delta t = 1$s, where the latter two are trainable parameters, and each τ is independent for each network node (i.e. during the training they will start to diverge from 0.1 across the layer). Note that setting the simulation length and resolution to one second does not mean that the entire inference will only last a single second; rather it is the simulation time per layer, meaning that each layer will be simulated subsequently for 1s, and hence the entire inference takes 18s (16s for the convolutional layers, and 1s for global pooling and dense layers each). The network is then trained for 512 epochs, on batches of 1024 examples, using RMSProp optimizer with 3^{-5} learning rate (reduced by a factor of 10^{-1} after 128 epochs of no improvement in the validation loss). We iterate the process, each time doubling the simulation length T, and halving the number of training epochs, batch size, the base learning rate (i.e. the value of the learning rate in the previous iteration before any reduction on the validation loss plateau), and the learning rate decay patience. Our target simulation length is $T = 32$ steps, and upon reaching it, we stop the training. To increase the sparsity of the spiking events, we also encourage the neurons to spike with a frequency between 10-20 Hz by applying ℓ^2 activity regularization to the spiking activation layer. The regularization penalty is chosen from $\{2 \cdot 10^{-9}, 10^{-9}, 5 \cdot 10^{-10}\}$.

Figure 2. Representatives of each of the land-cover classed of the EuroSAT dataset, in RGB (top row), and after applying the (normalised) Prewitt filter (bottom) row.

3. Results

We summarise the results in Figure 3 and Table 2. We see that the drop in the classification accuracy is almost 10%, which not insignificant. However, the estimated gain in energy efficiency as compared to a GPU is almost sixteen-fold, which is very substantial.

Furthermore, the gain in energy efficiency is not only due to the hardware properties but also because of the algorithmic advancement. To illustrate it, we included a comparison to a hypothetical implementation of the ANN on the Loihi chip, assuming that an ANN might be inferred by using the same E_o and E_u for spiking models. And even in this case, the SNN consumes $1.43\times$ less energy.

3.1. Input filtering

Brain-inspired computing is not limited to neuromorphic hardware and spiking neural network. It also encompasses neuromorphic sensors, such as event-based cameras, also known as dynamic vision sensors (DVS), or silicon retinas [18]. They are characterised by their asynchrony, low-energy consumption, high dynamic range, latency, and sparsity. The latter is of particular interest to us. The DVS only record (as a sequence of events) those pixels, where a change of light intensity (above some predefined threshold) occurred. This is normally caused by the motion (real or apparent) of the observed objects against a background of a different shade. That is, the information is registered primarily on the edges.

In land-cover machine learning problems, based on Earth observation data, the images are often composed of regions of identical visual patterns, meaning that the boundaries of these uniform patches contain the most pertinent information. Furthermore, the feasibility of deploying a DVS in the high-radiation hostile conditions of a space flight was recently demonstrated in [38].

Now, to the best of the our knowledge, there exists no event-based dataset of land-cover images, neither collected by on-board sensors nor synthetic, so we cannot verify how well a classification model would perform in this framework. Nevertheless, we can simulate the information load, if not qualitatively, then at least quantitatively in terms of the number of pixels that are being used as the input for the classification pipeline. To that end, we preprocess the EuroSAT RGB dataset by applying the (normalised) Prewitt filter to all the images, discarding all the resulting values which are below the 0.0078 threshold. That is, for all input images $\mathbf{X}$, we get the preprocessed image $\mathbf{X}'$ by the Equation (5):

$$\mathbf{X}' := \max\left(c\sqrt{(\mathbf{G}\circledast\mathbf{X})^2+(\mathbf{G}^\top\circledast\mathbf{X})^2},\, 0.0078\cdot\mathbb{1}\right), \tag{5}$$

where

$$\mathbf{G} := \begin{pmatrix}1 & 1 & 1\end{pmatrix}^\top \begin{pmatrix}1 & 0 & -1\end{pmatrix},$$

$\circledast$ denotes 2-dimensional convolution operation, c is the normalising constant, $\mathbb{1}$ is a tensor of ones with the same shape as $\mathbf{X}$, and the maximum is taken entrywise. This effectively leaves us with the edges of regions of similar colour and brightness, similar to what an on-board DVS camera could record, due to its ego-motion.

As shown in Figure 2, We feed these modified images to the VGG-16-based ANN, and conduct the training in the exact same manner as described in 2.3.1. The network scores 90.19% of accuracy performance, which is understandably lower than in case of the RGB images, because we have degraded the network input in some sense. We then proceed to train this network in the spiking context, as described in 2.3.1. The final top accuracy that we get is 87.89%, which is is higher than in the case of SNN without Prewitt filtering of the input. This might be a stochastic coincidence, but it is also possible that removing obsolete information from the input acts like denoising, allowing the meaningful spikes to propagate more easily. We can also gain more than twice the energy efficiency, when compare to the SNN with unfiltered input. The results are summarised in Table 2 and Figure 3.

Model	Acc. (%)	T	Δt	Energy	Hardware platform					
					SpiNNaker	SpiNNaker 2	Loihi	CPU	ARM	GPU
ANN	95.07	1	–	E_s	*3.09195*	*0.10461*	*0.00630*	1.99931	0.20923	0.06974
				E_n	*0.01901*	*0.00160*	*0.00006*	0.00629	0.00066	0.00022
(+ Prewitt)	(90.19)			Total	*3.11096*	*0.10622*	*0.00636*	2.00559	0.20989	0.06996
SNN	**85.11**	4	0.0381	E_s	2.06081	0.06973	0.00420			
				E_n	0.07604	0.00640	0.00024			
				Total	2.13685	0.07613	0.00444			
SNN	84.11	**2**	**0.0626**	E_s	2.12170	0.07179	0.00432			
				E_n	0.03802	0.00320	0.00012			
				Total	2.15972	0.07499	0.00444			
SNN	83.74	2	0.0663	E_s	**1.56687**	**0.05301**	**0.00319**			
				E_n	**0.03802**	**0.00320**	**0.00012**			
				Total	**1.60489**	**0.05622**	**0.00331**			
SNN + Prewitt	**87.89**	4	0.0403	E_s	2.21900	0.07508	0.00452			
				E_n	0.07604	0.00640	0.00024			
				Total	2.29504	0.08148	0.00476			
SNN + Prewitt	85.07	**1**	**0.0813**	E_s	**0.97873**	**0.03312**	**0.00199**			
				E_n	**0.01901**	**0.00160**	**0.00006**			
				Total	**0.99774**	**0.03472**	**0.00205**			

Table 2. The best models' accuracy and energy consumption (in joules) per inference on selected hardware platforms. Values in *italics* are hypothetical referenced values. Values in **bold** are the optimal experimental values for the SNN and SNN with Prewitt filter applied to the input respectively. That is, the highest accuracy, the lowest simulation time ($T \cdot \Delta t$), and the lowest energy consumption per inference. All the SNN models have the same architecture, and the difference in the performance levels results from different parameters' initial values.

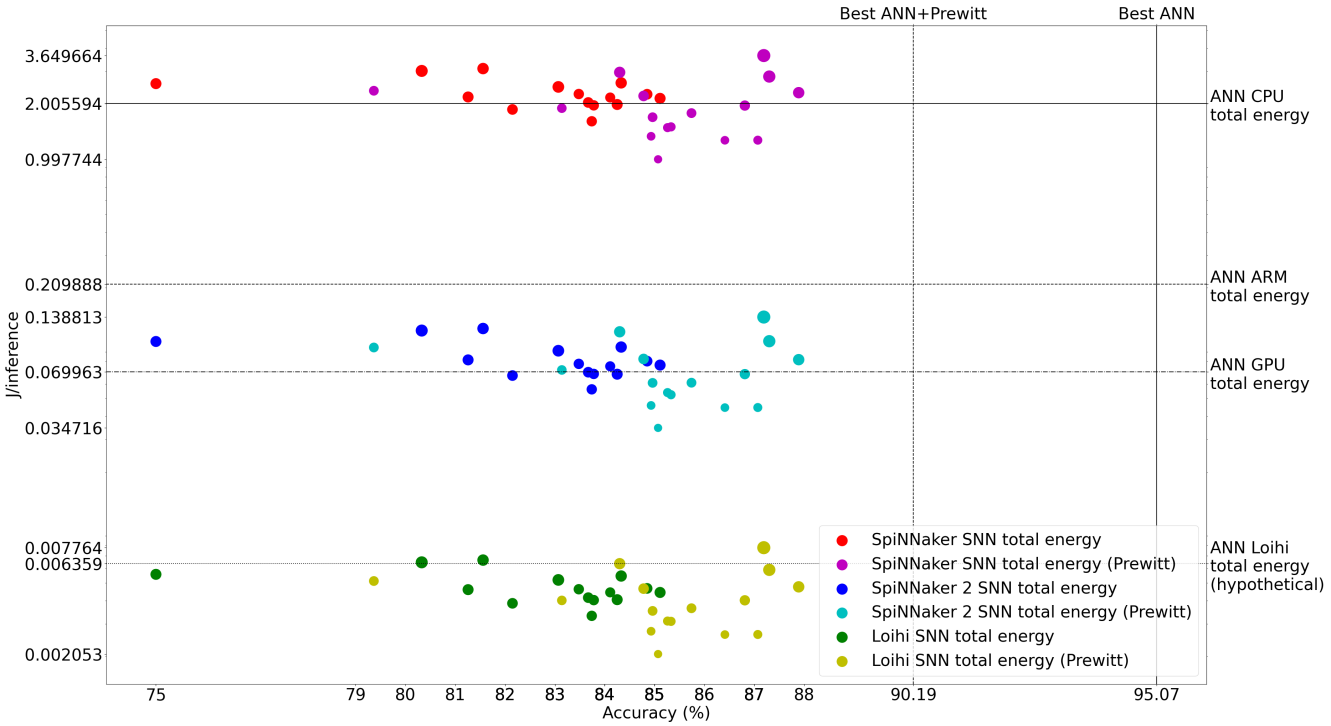

Figure 3. Distribution of trained model's accuracy against the energy consumption (in joules) per inference (log scale). Best to be viewed in colour. The size of the dots corresponds to the duration of the simulation ($T \cdot \Delta t$), which tends to stay close (standard deviation $\sigma = 0.0308$) to the mean value $\mu = 0.1369$s. Note that the implementation of the SNN on the Loihi chip remains more energy efficient than an implementation of the ANN on a hypothetical chip with the same MAC energy consumption as Loihi.

4. Discussion

This work offers a preliminary investigation of the energy efficiency of SNNs models for static images scene classification problems. As stated in Section 2.4.1, the methodology used for energy estimation relies on strong assumptions on the implementation of neuron models and neglects some other energy contribution factors that might significantly affect the estimated results presented in Section 3. However, it is not uncommon in the literature to use the number of synaptic operations and floating-point operations as proxies of the energy consumption for spiking [32, 14] and artificial models' [4] operations. Therefore, despite the impreciseness of the values of the energy per inference, the proposed method can be considered as a valid theoretical solution to benchmark ANNs and SNNs for land cover and land use scene classifications. In that respect, results provided in Section 2.4.1 should encourage future research on the use of SNNs for image classification, given the promising energy efficiency demonstrated by these algorithms in our benchmark. Furthermore, the study also suggests the values of T and Δt giving a clue about the best trade-offs between accuracy and energy efficiency.

Consequently, we intend to continue this work in the future by deploying to the models on neuromorphic hardware, enabling us to perform more accurate measurements, which might be used to validate the results conjectured in this paper, and, shall the result of this investigation be positive, consider the deployment of an SNN on an on-board neuromorphic platform. In addition, we also plan to extend the benchmark to modern hardware accelerators for edge inference of ANN models, such as Myriad 2 [35], which was already exploited to accelerate ANN models on board EO satellites [20].

Another limitation of the presented work relates to the use of the VGG-16 model, whose performance lags behind the state-of-the-art-models [24]. This is due to the requirement to avoid skip connections, max pooling, alternative activation functions for which the corresponding spiking implementations are less straightforward. None the less, future works will hopefully also extend its scope to more sophisticated networks, which may further reduce the accuracy and energy-efficiency trade-offs.

Finally, another contribution of the work is the exploration of potential benefits of the proposed preprocessing strategy, i.e. quantisation and Prewitt filtering. Such preprocessing enables both an increase of the accuracy and the energy efficiency, by ensuring a high sparsity of the input. These potential advantages might be due to the typical structure of the data of scene classification. Indeed, the presence of rivers, fields, and other vegetation in the image leads to mostly uniformly colored areas, whose information is condensed in the edge by the preprocessing pipeline. In particular, it shall be noted that the energy spent to implement the preprocessing is not included in the estimated energy consumption per inference. This is due to the fact that in the real-world implementations additional convolutional preprocessing operations might be performed on the input data, in which Prewitt filtering might be included, thus not increasing the overall energy consumption. Also, in the future the static data may be replaced by dynamic senors' output [18], resulting in the information workload similar to gradient preprocessed input, but with no need to perform any additional filtering operations. Otherwise, the additional energy budget for Prewitt filtering must be added to the estimates shown in Table 2.

5. Conclusions

In this paper, we presented a theoretical investigation of the potential energy consumption benefits of the SNNs for on-board AI applications, focusing on the scene classification case study. Accordingly, different ANN/SNN models were trained and compared in terms of energy efficiency and accuracy trade-offs on the EuroSAT RGB dataset. Furthermore, a preprocessing pipeline including Prewitt filtering and input quantisation is proposed, leading to an improvement of said energy accuracy concessions.

Despite the inaccuracy of the suggested energy estimation method, the preliminary results show that the use of SNNs might ensure potential benefits in terms of energy efficiency for this particular application with limited loss of accuracy, which should encourage future research involving the deployment of these models on neuromorphic hardware.

6. Acknowledgements

The authors of this paper would like to thank the Advanced Concepts Team and Φ-Lab divisions of the European Space Agency for their continued support of this research. We would also like to express our gratitude to Lenovo Italy for providing us with the GPU hardware infrastructure (ThinkStation P920) necessary to carry out the experiments.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang

Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Dario Amodei, Danny Hernandez, Girish Sastry, Jack Clark, Greg Brockman, and Ilya Sutskeve. AI and compute, May 16 2018.

[3] Applied Brain Research. KerasSpiking. https://www.nengo.ai/keras-spiking/project.html- Last access on 20/04/2021.

[4] Gionata Benelli, Gabriele Meoni, and Luca Fanucci. A low power keyword spotting algorithm for memory constrained embedded systems. In *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 267–272. IEEE, 2018.

[5] Pritam Bose, Nikola K Kasabov, Lorenzo Bruzzone, and Reggio N Hartono. Spiking neural networks for crop yield estimation based on spatiotemporal analysis of image time series. *IEEE Transactions on Geoscience and Remote Sensing*, 54(11):6563–6573, 2016.

[6] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2):1–35, 2019.

[7] Jasper Bouwmeester and J Guo. Survey of worldwide pico-and nanosatellite missions, distributions and subsystem technology. *Acta Astronautica*, 67(7-8):854–862, 2010.

[8] Applied Brain Research. KerasSpiking - Estimating model energy. https://www.nengo.ai/keras-spiking/examples/model-energy.html - Last access on 20/04/2021.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[10] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[11] B. Degnan, B. Marr, and J. Hasler. Assessing trends in performance per watt for signal processing applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(1):58–66, 2016.

[12] Bradley Denby and Brandon Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 939–954, 2020.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Charlotte Frenkel, Jean-Didier Legat, and David Bol. Morphic: A 65-nm 738k-synapse/mm2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning. *IEEE transactions on biomedical circuits and systems*, 13(5):999–1010, 2019.

[15] Gianluca Furano, Gabriele Meoni, Aubrey Dunne, David Moloney, Veronique Ferlet-Cavrois, Antonis Tavoularis, Jonathan Byrne, Léonie Buckley, Mihalis Psarakis, Kay-Obbe Voss, et al. Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. *IEEE Aerospace and Electronic Systems Magazine*, 35(12):44–56, 2020.

[16] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[17] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown. Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467, 2013.

[18] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[19] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International Journal of Neural Systems*, 19(4):295–308, 2009.

[20] Gianluca Giuffrida, Lorenzo Diana, Francesco de Gioia, Gionata Benelli, Gabriele Meoni, Massimiliano

Donati, and Luca Fanucci. Cloudscout: A deep neural network for on-board cloud detection on hyperspectral images. *Remote Sensing*, 12(14):2205, 2020.

[21] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, and Piotr Bojanowski. Self-supervised pretraining of visual features in the wild, 2021.

[22] Bing Han, Aayush Ankit, Abhronil Sengupta, and Kaushik Roy. Cross-layer design exploration for energy-quality tradeoffs in spiking and non-spiking deep artificial neural networks. *IEEE Transactions on Multi-Scale Computing Systems*, 4(4):613–623, 2017.

[23] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018.

[24] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[25] George Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6e —RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[26] S. Höppner, B. Vogginger, Y. Yan, A. Dixius, S. Scholze, J. Partzsch, F. Neumärker, S. Hartmann, S. Schiefer, G. Ellguth, L. Cederstroem, L. A. Plana, J. Garside, S. Furber, and C. Mayr. Dynamic power management for neuromorphic many-core systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(8):2973–2986, 2019.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[28] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[29] Saeed Reza Kheradpisheh and Timothée Masquelier. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(06):2050027, 2020.

[30] Vivek Kothari, Edgar Liberis, and Nicholas D Lane. The final frontier: Deep learning in space. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, pages 45–49, 2020.

[31] Andrzej S. Kucik and Gabriele Meoni. SNN for Space. https://github.com/AndrzejKucik/SNN4Space, 2021.

[32] Erwann Martin, Maxence Ernoult, Jérémie Laydevant, Shuai Li, Damien Querlioz, Teodora Petrisor, and Julie Grollier. Eqspike: spike-driven equilibrium propagation for neuromorphic implementations. *iScience*, page 102222, 2021.

[33] Abhas Maskey and Mengu Cho. Cubesatnet: Ultralight convolutional neural network designed for on-orbit binary image classification on a 1u cubesat. *Engineering Applications of Artificial Intelligence*, 96:103952, 2020.

[34] Christian Mayr, Sebastian Hoeppner, and Steve Furber. Spinnaker 2: A 10 million core processor system for brain simulation and machine learning, 2019.

[35] David Moloney, Brendan Barry, Richard Richmond, Fergal Connor, Cormac Brick, and David Donohoe. Myriad 2: Eye of the computational vision storm. In *2014 IEEE Hot Chips 26 Symposium (HCS)*, pages 1–18. IEEE, 2014.

[36] Priyadarshini Panda, Sai Aparna Aketi, and Kaushik Roy. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Frontiers in Neuroscience*, 14, 2020.

[37] Philippe Reiter, Philipp Karagiannakis, Murray Ireland, Steve Greenland, and Louise Crockett. FPGA acceleration of a quantized neural network for remote-sensed cloud detection. In *7th International Workshop on On-Board Payload Data Compression*, 2020.

[38] Seth Roffe, Himanshu Akolkar, Alan D. George, Bernabé Linares-barranco, and Ryad Benosman. Neutron-induced, single-event effects on neuromorphic event-based vision sensor: A first step towards space applications, 2021.

[39] J. Sawada, F. Akopyan, A. S. Cassidy, B. Taba, M. V. Debole, P. Datta, R. Alvarez-Icaza, A. Amir, J. V. Arthur, A. Andreopoulos, R. Appuswamy, H. Baier, D. Barch, D. J. Berg, C. Di Nolfo, S. K. Esser, M. Flickner, T. A. Horvath, B. L. Jackson, J. Kusnitz, S. Lekuch, M. Mastro, T. Melano, P. A. Merolla, S. E. Millman, T. K. Nayak, N. Pass, H. E. Penner, W. P. Risk, K. Schleupen, B. Shaw, H. Wu, B. Giera, A. T. Moody, N. Mundhenk, B. C. Van Essen, E. X. Wang, D. P. Widemann, Q. Wu, W. E. Murphy, J. K. Infantolino, J. A. Ross, D. R. Shires, M. M. Vindiola, R.

Namburu, and D. S. Modha. Truenorth ecosystem for brain-inspired computing: Scalable systems, software, and applications. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 130–141, 2016.

[40] Stefan Schliebs and Nikola Kasabov. Evolving spiking neural network – a survey. *Evolving Systems*, 4(2):87–98, 2013.

[41] Daniel Selva and David Krejci. A survey and assessment of the capabilities of cubesats for earth observation. *Acta Astronautica*, 74:50–68, 2012.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[43] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[44] Shilpa Suresh, Devikalyan Das, and Shyam Lal. A framework for quality enhancement of multispectral remote sensing images. In *2017 Ninth International Conference on Advanced Computing (ICoAC)*, pages 9–14. IEEE, 2017.

[45] Xiaoli Tao and Howard E Michel. Novel artificial neural networks for remote-sensing data classification. In *Optics and Photonics in Global Homeland Security*, volume 5781, pages 127–138. International Society for Optics and Photonics, 2005.