# Siamese Networks for Few-Shot Learning on Edge Embedded Devices

Lungu, Iulia Alexandra ; Aimar, Alessandro ; Hu, Yuhuang ; Delbruck, Tobi ; Liu, Shih-Chii

DOI: https://doi.org/10.1109/jetcas.2020.3033155

Accepted for publication in 2020 IEEE Journal on Emerging and
Selected Topics in Circuits and Systems

# Siamese Networks for Few-shot Learning on Edge Embedded Devices

Iulia-Alexandra Lungu, Alessandro Aimar, Yuhuang Hu, Tobi Delbruck, Shih-Chii Liu
*Institute of Neuroinformatics, University of Zürich and ETH Zürich*
Zürich, Switzerland
{iulialexandra, alessandro.aimar, yuhuang.hu, tobi, shih}@ini.uzh.ch

***Abstract*—Edge artificial intelligence hardware targets mainly inference networks that have been pretrained on massive datasets. The field of few-shot learning looks for methods that allow a network to produce high accuracy even when only a few samples of each class are available. Siamese networks can be used to tackle few-shot learning problems and are unique because they do not require retraining on the new samples of the new classes. Therefore they are suitable for edge hardware accelerators which often do not include on-chip training capabilities.**

**This work describes improvements to a baseline Siamese network and benchmarking of the improved network on edge platforms. The modifications to the baseline network included adding multi-resolution kernels, a hybrid training process as well a different embedding similarity computation method. This network shows an average accuracy improvement of up to 22% across 4 datasets in a 5-way, 1-shot classification task. Benchmarking results using three edge computing platforms (NVIDIA Jetson Nano, Coral Edge TPU and a custom convolutional neural network accelerator) show that a Siamese classifier can run on these devices at reasonable frame rates for real-time performance, between 3 frames per second (FPS) on Jetson Nano and 60 FPS on the Edge TPU. By increasing the weight sparsity during training, the inference time of a network with 25% weight sparsity increases by 10 FPS but with only 1% drop in accuracy.**

***Index Terms*—one-shot learning, few-shot learning, edge intelligence, deep networks, computer vision, Siamese networks, hardware accelerators, edge computing**

## I. Introduction

Edge artificial intelligence (AI) hardware accelerators usually target inference networks that have been trained offline on large servers [1]. Retraining these networks once they are deployed is both cost and energy prohibitive. When naively retrained on new data, a deep neural network (DNN) can learn new object classes but "forgets" most of the previous knowledge it had accumulated, unless previous data is fed to the network along with the new data [2].

Incorporating new classes without extensive amounts of samples and without requiring retraining on the entire original dataset is an active area of research in the field of few-shot learning. In this field, the number of stored samples in most algorithms is fewer than 20, usually either 1 or 5 [3]. This obviates the need to accumulate a large number of samples before training the system, which would be useful in edge deployment when data is scarce and the memory needed to store the samples is limited.

In recent years, multiple approaches to solving few-shot learning problems have been proposed. Transfer learning [4] trains a network in a fully-supervised manner and then fine-tunes the last layer on the novel classes or tasks. Data augmentation methods [5], [6] augment a small labelled dataset and turn the few-shot learning problem into a standard supervised learning problem. Meta-learning [7] allows for rapid fine-tuning of the model parameters by training a general meta-learner on a variety of tasks. The above-mentioned methods are not suitable for traditional edge implementations on inference accelerators because they require retraining whenever new classes of objects become available.

Siamese networks [8], [9] are another type of algorithm used for few-shot learning. Unlike standard classifiers, Siamese networks can classify samples from novel classes without requiring any retraining for the new classes. Because no retraining is needed, this architecture is suitable for edge devices which typically do not include on-chip learning. These networks are part of the metric learning approach to few-shot learning problems and are trained to recognize the similarity or dissimilarity between sample pairs. Training is first carried out offline on a vast number of sample pairs that belong either to the same class or to different classes. After the Siamese network is deployed, data to be classified are compared to labelled representative examples for each class, called *prototypes* in the rest of the paper. The incoming samples need not belong to classes seen during training. The winning class is the one corresponding to the highest similarity between the sample of interest and the saved prototype.

### A. Contributions

Although Siamese networks can classify novel objects without retraining, the classification accuracy of these networks on unknown classes is worse than that of incremental learning algorithms such as iCaRL [10]. In [11], we proposed several modifications to baseline Siamese networks to address the accuracy gap: different kernel size streams within one layer, concatenating the branch embeddings, adding several layers that combine the two embeddings before the similarity output, as well as a hybrid training process which makes use of both classification and similarity. By combining the four approaches, we showed state-of-the-art accuracy on one-

This work is partially funded by the Samsung Institute of Technology, the SNSF National Centre of Competence in Research (NCCR) Robotics, and the SNSF BRIDGE grant no. 40B2-0_181010.

shot classification with Siamese networks on four benchmark datasets, Omniglot [12], Tiny-Imagenet [13], CIFAR100 [14], and Roshambo [15].

This study provides the following contributions to the field of few-shot learning:

1) We benchmark Siamese networks on three different edge devices, specifically, the NVIDIA Jetson Nano [16], the Coral Edge TPU [17] as well as a custom hardware CNN accelerator to evaluate the real-time performance of a few-shot learning algorithm on edge platforms. We believe this is one of the first studies to show an edge hardware implementation of this algorithm.
2) We propose several modifications to the baseline Siamese networks: a hybrid training process which includes both classification and similarity-based losses, concatenation of branch embeddings, adding several layers that combine the two embeddings before the similarity output, as well as a branch architecture that contains multi-resolution convolutional layers. These modifications lead to an improvement in 5-class classification accuracy by up to 22.6%.
3) We show improved inference times from a baseline Siamese network by performing training with reduced bit precision and weight matrices with high sparsity levels.
4) We compare the inference times for a Siamese network against a one-branch classification network and argue why Siamese networks are a good approach to few-shot classification.

Section II discusses relevant work, Section III presents the proposed improvements, the datasets and the edge devices used in the study, while Section V describes the algorithmic and benchmarking results.

TABLE I
COMPARISON BETWEEN DIFFERENT STATE-OF-ART ALGORITHMS AND SIAMESE NETWORKS ON OMNIGLOT FOR 1-SHOT 5-WAY CLASSIFICATION

| | Accuracy | Retraining required? |
|---|---|---|
| **Matching networks [18]** | 98.1% | No |
| **Prototypical networks [19]** | 98.8% | No |
| **MAML (meta-learning) [20]** | 98.7% | Yes |
| **Siamese networks (ours)** | **99.0%** | No |

## II. RELEVANT WORK

Siamese networks (SNs) were introduced by [8] for solving a signature verification task. The idea of computing similarity between two samples using a deep artificial neural network (ANN) was then quickly adopted in applications such as face verification [21], few-shot classification [9], person re-identification [22], [23], image retrieval [24], [25], stereo matching [26] and object tracking [27].

Siamese networks were first used in a few-shot classification task by [9]. Here, Siamese networks compute the similarity between the embeddings of image pairs to determine the class

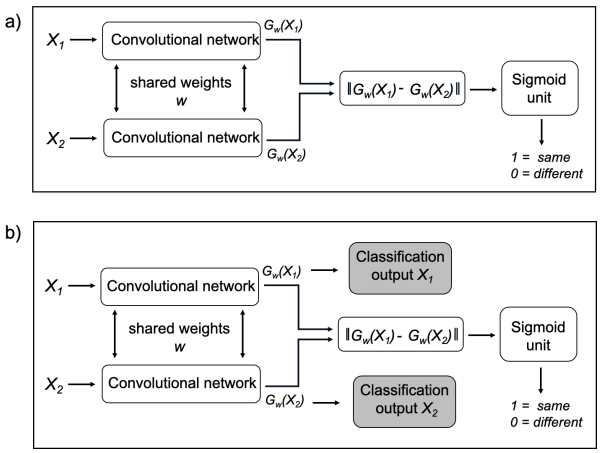



Fig. 1. General Siamese network architecture. Grey components are only used during training. (a) Networks are trained using only a similarity loss. (b) Both similarity and classification losses are employed at training.

identity of a query alphabet symbol. Other similarity-based few-shot learning methods include Matching Networks [28], Prototypical Networks [19], Relation Networks [29]. Another branch of few-shot learning aims to find good initializations of the employed models so that novel classes can be learned by adapting the existing model using only a limited number of samples [20], [30], [31]. The field of co-saliency detection resembles Siamese networks in their endeavour to uncover jointly informative areas in images. Some of the approaches suggested in [32], [33] could be applied to the field of few-shot learning to extend its ability to perform object identification without retraining.

The appeal of Siamese networks is that, unlike many of the aforementioned few-shot learning algorithms, Siamese networks are a simple, scalable algorithm that does not require re-training when new classes become available. As they do not require on-chip training resources, these networks should be easily adaptable to inference-focused edge hardware accelerators such as the Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Array (FPGA)-based accelerators [34]–[39]. Commercial edge computing platforms such as the NVIDIA Jetson series (*e.g.*, Nano, TX2) have an on-board, energy-efficient GPU to accelerate DNNs at runtime. Siamese networks are well-suited for re-identifying a sample that has been presented previously. Example tasks that are useful for edge applications include the re-identification of a tracked object across multiple camera views [23] or of a person's voice for speaker verification [40]. A recent study evaluated the inference and training time of ResNet-50 and MobileNet-v2 on a re-identification task using the NVIDIA Xavier which burns more power than the edge platforms used in this work [41].

## III. METHODS

The Siamese network first used for one-shot image recognition [9], shown in Fig. 1(a), consists of two branches that are two clones of the same Convolutional Neural Network (CNN).

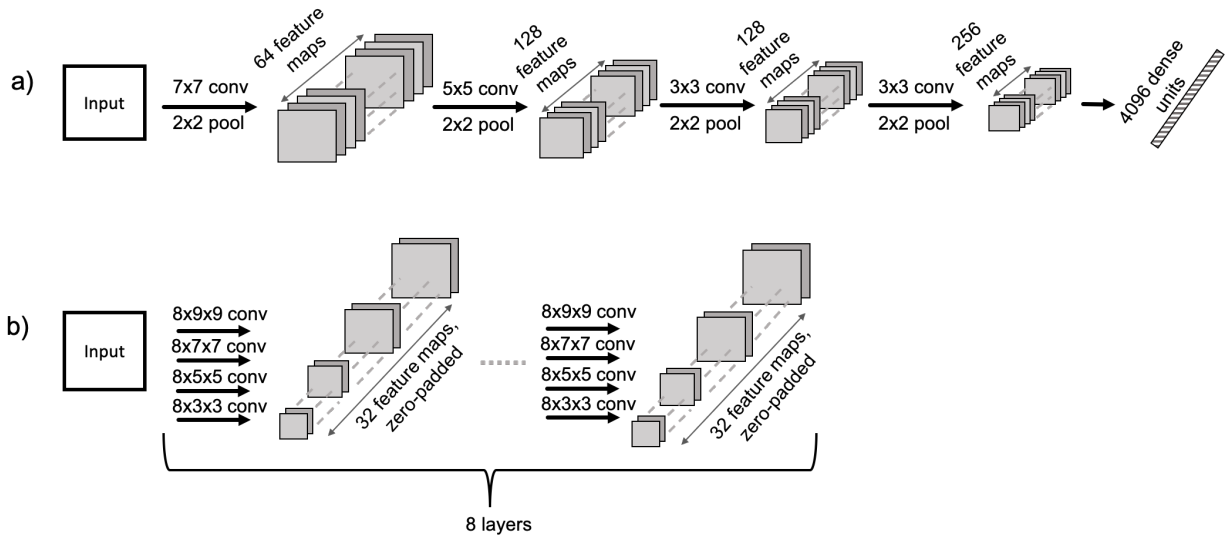



Fig. 2. Example of Siamese network branches used. (a) Branch similar to the one used in the original paper [9]. Used for networks *S-subtract*, *S-concat* and *S-concat-review* (b) Our own multi-resolution branch architecture, used in the *S-multires* network.

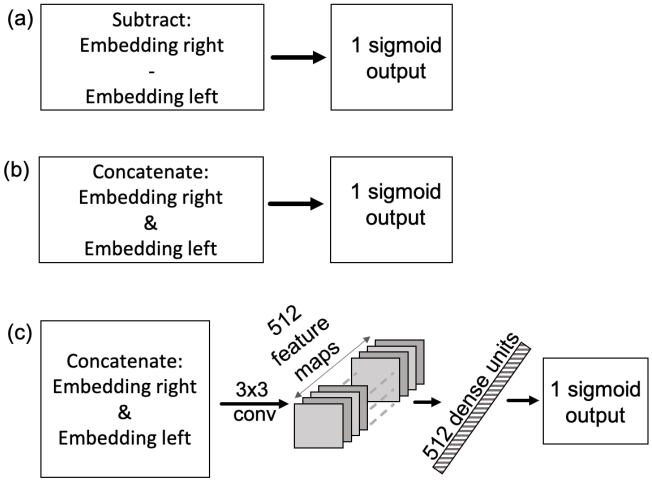



Fig. 3. Similarity modules (trunks) for different network architectures. (a) *S-subtract*: output embeddings of the two branches are subtracted; (b) *S-concat*: output embeddings of the two branches are concatenated; (c) *S-multires* and *S-concat-review*: concatenated embeddings, plus one convolutional and one dense layer before output

This means that the two branches receiving pairs of images as input could be seen as a single branch receiving two images sequentially. Weight sharing ensures that the order in which the images are fed to the network is not important. It also encourages similar images to be encoded in a similar way in feature space. The outputs of the two branches, $G_w(X_1)$ and $G_w(X_2)$, commonly referred to as embeddings, are then merged through a subtraction operation, followed by a sigmoid unit, which outputs the value 1 when the two input images belong to the same class, or 0 otherwise. The networks are trained by feeding them image pairs from as many classes as possible. It is important to have a balanced number of pairs of images from the same class and pairs of images from different classes, to not bias the network. Once deployed, Siamese networks can perform similarity analysis. However, to classify novel images, these images need to be compared to labelled prototypes. To start off, a single image can be labelled as belonging to class 0. To check if other query images belong to this class that we labelled 0, we compare them with the saved prototype. This is called 1-way classification, because we compare against a single labelled class. If we have saved prototypes belonging to *N* different classes, the *N*-way classification is carried out by computing the similarity for pairs constructed with the new query image and each of the *N* prototypes. The class decision then corresponds to the highest similarity score. If *K* labelled prototypes are stored for each class, *K-shot* classification is performed. Increasing the number of stored prototypes increases the classification accuracy.

In this study we chose to perform 5-way 1-shot classifications for ease of comparison with the existing literature.

### *A. Modifications to baseline*

The baseline architecture called *S-subtract* network in this work is similar to the Siamese network first used for one-shot image recognition [9], where the embeddings from the two branches are first subtracted and then passed through a sigmoid. The branch configuration of S-subtract is shown in Fig. 2(a), while the similarity module, also referred to as *trunk* in our paper, is described in Fig. 3(a). To improve the accuracy of the baseline Siamese network, we introduce the following modifications:

**Training on joint tasks**

While traditional CNNs used for classification (identification) integrate the information specific to each of the different classes in the dataset and deduce relationships between the

different classes, verification algorithms such as the Siamese networks only look at similarities between two images at a time. Both approaches have drawbacks: identification needs retraining for all classes to modify the mapping between all existing classes; verification algorithms are more generalizable but lack some of the discriminative power of identification models. By combining both tasks during training, [42], [43] showed that the performance across each of the tasks improves. We therefore introduced two classification layers, one for each input image, as shown in Fig. 1(b). Each layer receives as input the branch embeddings corresponding to one of the input images. During inference, the classification layer is discarded and only the similarity output is used to compute the accuracy.

**Concatenation of branch outputs**

By concatenating the embeddings of the two branches, we allowed the network to find the best means of comparing the two input images, instead of restricting it to a subtraction operation. This architecture, named *S-concat*, combines the Siamese branch in Fig. 2(a) and the similarity module in Fig. 3(b). S-concat is similar to S-subtract, the only difference being the use of concatenation instead of subtraction to combine the two embeddings.

**Convolutional similarity module**

In the *S-subtract* and *S-concat* architectures, the output of the two branches is passed directly to a dense layer. With this modification we explored how combining the two embeddings in a more complex way impacts accuracy. In *S-concat-review*, the two embeddings are concatenated and then passed through a convolutional layer with 512 $3 \times 3$ kernels followed by a dense layer and the sigmoid output. This architecture combines the Siamese branch in Fig. 2(a) and the similarity module in Fig. 3(c).

**Multiple resolution kernels**

Natural image datasets like CIFAR100 and Tiny-Imagenet contain objects of different resolutions. Smaller objects could be more easily captured by smaller kernels, while larger objects could benefit from larger receptive field sizes. We therefore tested an architecture, *S-multires*, that combines different sizes of kernels in the same layer, similar to the Inception architecture [44]. Each input image is convolved with kernels of size 9, 7, 5 and 3 at each layer. The branch has 8 multi-resolution layers. The embeddings are concatenated and passed through additional layers before reaching the sigmoid output, as explained for the *S-concat-review* architecture. Fig. 2(b) shows the branch, while the trunk can be seen in Fig. 3(c).

TABLE II
DETAILS OF THE DATASETS EMPLOYED

| | Omniglot | Tiny-Imagenet | CIFAR100 | Roshambo |
|---|---|---|---|---|
| **No. classes** | 1623 | 200 | 100 | 20 |
| **No. images / class** | 20 | 600 | 600 | 2000 |
| **Image size** | 105×105×1 | 64×64×3 | 32×32×3 | 64×64×1 |

### B. Datasets

We used four datasets to evaluate the architectures, summarised in Table II :

**Omniglot** [12] is a hand-written dataset containing 1623 characters from 50 different alphabets. There are only 20 samples per character. This was the first and is the most well-known dataset used as a few-shot learning benchmark. The image resolution is 105×105×1.

**Tiny-Imagenet** [13] is a dataset extracted from the ILSVRC competition Imagenet dataset [45]. Tiny-Imagenet has 200 classes, each with 500 training images, 50 validation images and 50 test images. The image resolution is 64×64×3.

**CIFAR100** [14] consists of 60k natural images of size 32×32×3, divided in 100 classes. There are 500 training images and 100 test images per class.

**Roshambo** [15] has 20 hand symbol classes recorded from an event-based DAVIS240C retina camera [46]. There are 1000 training images and 1000 test images per class. For compatibility with traditional convolutional neural networks used for vision tasks, we transform the event streams produced by the DAVIS240C into frames by accumulating events into 64×64 2D pixel histograms of a constant number of events, referred to as *constant-event* frames [47], [48].

All datasets were minimally preprocessed. Pixel values across all datasets were cast to a (0, 1) range for both training and testing. At training time, images were additionally augmented using random rotations, translations, crops, zoom and shear. For all experiments, training took place over 200 epochs using a batch size of 32 and an SGD optimizer with momentum. The starting learning rate was 0.001, geometrically decreasing at each epoch by a scaling factor of 0.99. The test accuracy results were obtained using the standard 5-way 1-shot comparison for evaluating Siamese networks [9]. For each dataset, 5 classes were set aside for testing, while the rest were used during training. In the case of the Omniglot, Tiny-Imagenet, CIFAR100 and Roshambo datasets, we have 1618, 195, 95 and 15 training classes respectively. To compute test accuracy, 1000 inference trials were conducted. For each trial, 5 pairings were created between the image to be classified and one prototype for each of the 5 classes to classify against. The pairings are random, so the prototypes differ at each trial.

## IV. EDGE EMBEDDED PLATFORMS

For the hardware study, we modified S-multires by removing batch normalization and the 9x9 kernels. These changes were necessary to optimally run our architecture on all the platforms tested. We benchmarked this architecture, *S-multires-edge*, on 3 different platforms targetting inference and edge applications:

*1) Carbon CNN accelerator:* Carbon is a customized version of our Nullhop CNN accelerator [39] with support for convolution layers (1D and 2D), pooling (max and average), ReLU and fully-connected layers. Carbon is optimized for operating with a batch size of 1. Because our accelerator does not support concatenation, this operation is executed by the system microcontroller rather than on the accelerator itself.

Similar to the Coral Edge TPU, Carbon requires a quantized network with fixed point 16-bit activations and 8-bit weights.

For the evaluation of the Siamese networks, we used an implementation that supports 16-bit activations and 8-bit weights, together with a full-precision 32-bit resolution for partial results accumulation. Biases are stored as 8-bit values. The accelerator has 128 multiply-accumulate units and an operating frequency of 500 MHz. Similar to Nullhop [39], this accelerator exploits sparsity of the feature maps by skipping over the multiplication operations for pixels with a zero activation. We extended the zero-skipping capabilities of the hardware with support for pruned neural networks.

**Performance** Because of the irregular and unpredictable sparsity pattern of the activations, it is not possible to precisely evaluate the peak performance of Carbon, which strongly depends on the specific input being processed. The minimum expected throughput is approximately 0.128 Tera Operations per sec (TOP/s) when processing networks with no sparsity. Assuming 75% activation sparsity and 50% weight sparsity, the throughput is expected to increase to up to 0.8 TOP/s according to the sparsity pattern. We experimentally verified that the upper throughput limit reaches 5.8 TOP/s when the sparsity values of both activations and weights are at 90% on a test layer with 256 $3\times3\times256$ kernels. The estimates of the computation speed are obtained from an RTL digital simulator (Cadence Incisive v18.09.005). We used a static power analysis technique [39] to obtain the power figures for both the core accelerator and DDR memory.

**Quantization and pruning** The version of S-multires-edge deployed on Carbon has been obtained using TensorFlow Keras v2.1.0 with in-training quantization. We exploited the built-in API for quantization and pruning [49], extending the basic functionality (designed for Coral Edge TPU) to support Carbon 16-bit activation/8-bit weight precision.

To increase the training speed, we implemented pruning with the same quantization API rather than using the dedicated one [50]. We applied an unstructured pruning based on weight absolute value [51]. For the training schedule, we started with 0% weight sparsity, increased it linearly every 100 batches until we reached a final sparsity after 20 epochs. After every 100 batches, we updated the pruning mask according to the magnitude of the weights. In the experiments reported here, we used 5 different sparsity values: 10%, 25%, 50%, 75% and 90%.

*2) NVIDIA Jetson Nano:* A 5 W device based on a 128-core NVIDIA Maxwell GPU, with claimed peak 472 GFLOPs throughput. It supports the CUDA programming interface and can compute all the operations/layers of a neural network with floating-point precision. The networks run on this device were implemented in Tensorflow 2.0 and were trained on a desktop NVIDIA GTX 1080 TI in floating-point 32 before being deployed on the Jetson Nano.

*3) Coral Edge TPU:* A single-board machine-learning inference accelerator that claims 4 TOP/s peak throughput at 2 W. However, our independent tests reported a real-world power consumption of more than 5 W. It only supports 8-bit integer quantized networks and a wide range of operations (detailed list available at [17]) including convolution, pooling, ReLU and the concatenation operation needed for running the S-multires-edge. The network deployed on this device was trained in floating-point 32 precision on a desktop NVIDIA GTX 1080 TI before being quantized to 8-bit precision post-training. The resulting quantized model was converted to an Edge TPU compatible version with the help of the Edge TPU compiler.

## V. Results

We present here results from the algorithmic experiments in Section V-A, the hardware benchmarking in Section V-B, considerations of weight sparsity in Section V-C, and comparisons between a classification network and the Siamese network in Section V-D.

### A. Algorithmic improvements

Table III shows the main network architectures and their accuracies for different datasets, reported as mean $\pm$ standard deviation over ten independent runs. Fig. 4 condenses the same information in easy to interpret barplots.

Across all datasets, adding a classification task during training increases accuracy by up to 17.7%. Adding a classification task has a higher impact on networks that do not incorporate the multi-resolution architecture. In particular, for the baseline architecture, *S-subtract*, we observe an average increase in accuracy of 7.7% across all datasets. In contrast, for *S-multires*, the average improvement is 1.8%. Multi-task learning and multi-resolution kernels have a similar effect on accuracy and we do not observe an additive effect of the two.

Using a concatenation for the embeddings instead of a subtraction seems to help especially in the case when a classification task is used during training. The accuracy increases in this case are 1%, 1.4%, 2.2% and 0% for Omniglot, Tiny-Imagenet, CIFAR100 and Roshambo respectively. Furthermore, when passing the embeddings through a convolutional and a dense layer (such as in *S-concat-review*), the accuracy increases further compared to the case where the two embeddings are only concatenated before being passed to the output (*S-concat*). The average accuracy increase across all datasets is 2.2%. Adding the multi-resolution kernels in the "with classification scenario" adds another 2.9 % to the average across-dataset accuracy. Overall, in this scenario, shown in Fig. 4(b), each modification to the baseline we implemented helped improve the accuracy.

For the case where no classification was used during training, shown in Fig. 4(a), the impact of the different proposed modifications is less evident. However, *S-multires* clearly improves accuracy across all datasets. Compared to *S-subtract*, *S-multires* adds 1.3%, 21.4%, 17.3% and 8.4%. As expected, adding multiple resolution kernels helps primarily natural image classification, where objects might be present at different resolutions throughout the same class, as is the case for Tiny-Imagenet and CIFAR100.

TABLE III
TEST ACCURACY ON 5 NOVEL CLASSES FOR DIFFERENT NETWORK ARCHITECTURES AND DIFFERENT DATASETS.

| | Omniglot | Tiny-Imagenet | CIFAR100 | Roshambo |
|---|---|---|---|---|
| **Training without classification** | | | | |
| S-subtract | 97.2±2.4% | 33.0±2.9% | 35.4±4.6% | 50.5±7.1% |
| S-concat | 97.2±1.9% | 35.8±4.8% | 35.0±4.3% | 41.1±6.7% |
| S-concat-review | 95.4±4.5% | 33.1±5.6% | 35.4±5.8% | 50.8±6.1% |
| S-multires | 98.5±1.8% | 54.4±4.1% | 52.7±8.0% | 58.9±8.1% |
| **Training including classification** | | | | |
| S-subtract | 94.8±5.0% | 50.7±6.6% | 49.7±6.8% | 51.8±6.6% |
| S-concat | 95.8±4.8% | 52.1±6.5% | 51.9±7.2% | 51.8±4.9% |
| S-concat-review | 97.8±3.7% | 54.4±7.0% | 52.5±7.4% | 55.6±5.4% |
| S-multires | 99.1±1.3% | 55.6±4.5% | 54.4±6.8% | 62.6±7.0% |

Across all conditions, the accuracy improvement on Omniglot is not as dramatic due to a saturation effect. The accuracy achieved with S-subtract was already above 90%.

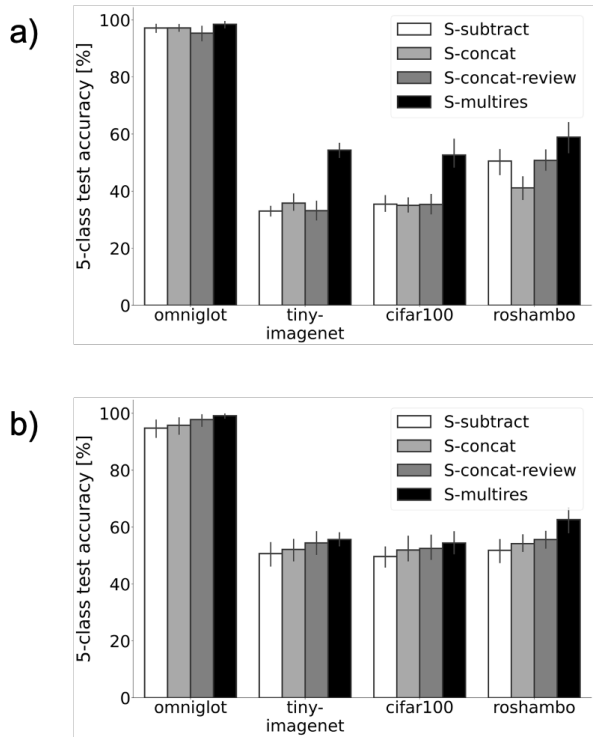



Fig. 4. Test accuracy on 5 novel classes for different network architectures and different datasets. (a) No additional classification task was used during training. (b) An additional classification task was used during training

The different levels of accuracy improvement across all datasets are partially due to the fact that two of the datasets consist of natural images and that different numbers of classes are available for training in each dataset. Siamese networks generalize better when the dataset has more classes. We confirmed this trend on Omniglot, which consists of 1618 classes for training. Fig. 5 shows that even on this simple dataset, we need at least 250 training classes to reach the maximum accuracy. For Roshambo, only 15 distinct classes were used during training, while for Tiny-Imagenet we had 195 classes.

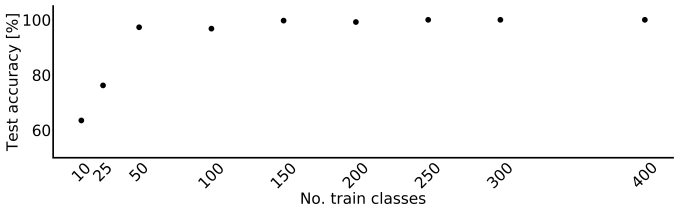



Fig. 5. 5-way 1-shot classification accuracy on Omniglot for an increasing number of training classes.

### B. Hardware performance metrics

We compare the performance of the S-multires-edge network on 4 different devices which use different bit precision values for activations and weights. The desktop GPU (1080 Ti) and the Jetson Nano support floating-point bit precision but the other 2 platforms support lower bit precision for weights and activations for reduced energy consumption. Table IV shows the performance metrics of running this network including bit precision, inference time, and test accuracy on 5 novel classes. The evaluation was done on the Tiny-Imagenet dataset. The S-multires-edge network for this dataset has 4.75 million parameters, about 25.8% lower than the 5.5 million parameters of the S-multires network. Its accuracy of 49.6% for 32-bit precision is about 5.33% lower than the S-multires but the smaller network can be benchmarked on all 4 devices. S-multires-edge achieves 49.3% when quantized to 8-bit weight precision and 16-bit activation precision. When quantized to 8-bit for weights and activations, the accuracy stands at 49.8%.

TABLE IV
PERFORMANCE OF S-MULTIRES-EDGE NETWORK ON DIFFERENT DEVICES.

| | GTX 1080 Ti | Jetson Nano | Edge TPU | Carbon |
|---|---|---|---|---|
| **Weight precision** | 32-bit | 32-bit | 8-bit | 8-bit |
| **Activation precision** | 32-bit | 32-bit | 8-bit | 16-bit |
| **Accuracy** | 49.6% | 49.6% | 49.8% | 49.3% |
| **Inference time [ms]*** | 15.8±0.6 | 329.4±40.0 | 19.8±0.4 | 35.2±0.1** |
| **Frames per second [FPS]** | 63.3 | 3.0 | 50.6 | 28.4 |
| **Average throughput [GOP/s]** | 577 | 28 | 462 | 259 |
| **Peak utilization** | 5.1% | 5.9% | 11.5% | 4.5% |

* Numbers represent the mean and standard deviation over 1000 5-way classifications. All measurements are performed on the Tiny-Imagenet dataset.

** Due to the time length of Carbon simulations, we limited the trials to 100 frames.

TABLE V
TEST RESULTS ON 5 NOVEL CLASSES FOR DIFFERENT WEIGHT SPARSITY LEVELS OF S-MULTIRES-EDGE ON CARBON ON THE TINY-IMAGENET DATASET.

| **Target weight sparsity** | 0% | 10% | 25% | 50% | 75% | 90% |
|---|---|---|---|---|---|---|
| **Accuracy** | $49.3\% \pm 8.8\%$ | $47.5\% \pm 7.5\%$ | $48.0\% \pm 9.7\%$ | $44.4\% \pm 7.7\%$ | $40.4\% \pm 6.1\%$ | $36.0\% \pm 5.5\%$ |
| **Inference time [ms]** | $35.2 \pm 0.1$ | $33.9 \pm 0.2$ | $26.6 \pm 0.2$ | $19.8 \pm 0.1$ | $13.6 \pm 0.2$ | $10.9 \pm 0.1$ |
| **Average throughput [GOP/s]** | 259 | 266 | 339 | 461 | 661 | 837 |

Both accuracy values are within the ±4.52% accuracy standard deviation for S-multires and the Tiny-Imagenet dataset (Table III), confirming that it is possible to quantize these models without sacrificing accuracy.

S-multires-edge performs 1.5 Giga Operations (GOp) for a pair of images. The deep-learning-dedicated hardware enabled both the Edge TPU (19.75 ms or 50.6 FPS) and Carbon (35.21 ms or 28.4 FPS) to achieve an inference time comparable to a much more powerful and power hungry NVIDIA GTX 1080 TI workstation GPU (15.80 ms or 63.3 FPS). By contrast, the Jetson Nano is not suitable for real-time inference at 3 FPS.

### C. Effect of weight sparsity

Training a network for weight sparsity is useful because the required memory footprint of the network will be lower. In this subsection, we report on results for the inference times on Carbon which supports zero-skipping, and the accuracy of S-multires-edge trained for 5 sparsity levels as described in the previous section. The other edge devices benchmarked here do not support zero-skipping. Table V shows that increasing the weight sparsity in S-multires-edge leads to a significant improvement of the inference time, with a variation in accuracy of about 1% for a target sparsity of 25% and a drop of 4.9% for 50% sparsity. However, higher levels of sparsity strongly impact the classification capabilities of the network: while still being better than the 20% chance level, in the extreme case of 90% target weight sparsity, the accuracy drops to 36.60%. Better training methods for weight sparsity will help to improve the accuracy [51]. The reported sparsity refers to the target provided to the algorithm at the beginning of the training. The effective value can differ from it, especially for quantized networks. In our tests, the effective sparsity is constantly 5% to 15% higher than the target, e.g. our target sparsity of 25% results in our tests in an effective sparsity of about 40%.

### D. One-branch vs Siamese classification

Unlike Siamese networks, traditional one-branch classification models that do not compute similarity cannot classify new objects without retraining. This results in high latencies whenever new objects must be learned. In this section we compared the inference time of the S-multires architecture to a similar one-branch classification model taking into account the time it takes to do the retraining for the one-branch classifier. We designed a standard classification network using one branch of the S-multires network along with the trunk. As there is only one input to the network, the concatenation layer was removed. The number of output units is increased from 1 to 5 and the sigmoid activation is replaced by a softmax. This architecture can be seen in Fig. 6. Except for the output layer, this architecture has the same number of parameters as the S-multires Siamese network.

On Tiny-Imagenet, the one-branch model takes 5.04 ms to classify an image, while the Siamese network requires 15.8 ms for the same task. However, in the case of Siamese networks, parts of the network graph can be pre-computed. Because Siamese networks perform comparisons with immutable class

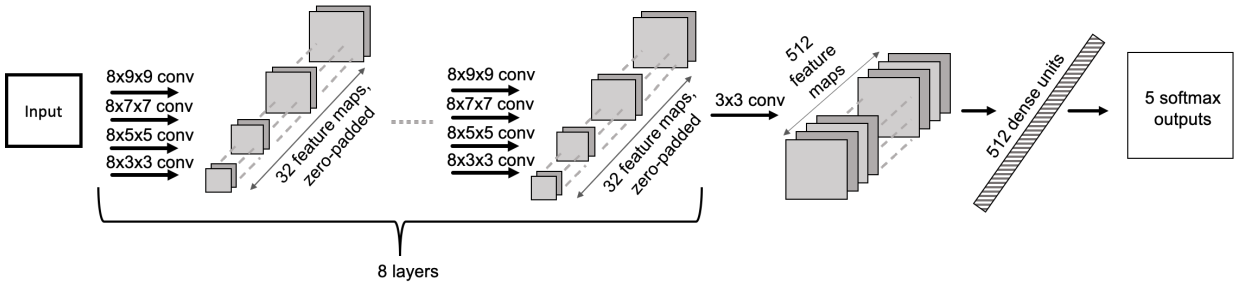



Fig. 6. One-branch conventional classifier, based on the S-multires architecture

TABLE VI
ONE-BRANCH VERSUS SIAMESE CLASSIFIER INFERENCE TIMES.

| | Omniglot | Tiny-Imagenet | CIFAR100 | Roshambo |
|---|---|---|---|---|
| **Input size** | 105×105×1 | 64×64×3 | 32×32×3 | 64×64×1 |
| **One-branch network inference time** | 6.0±0.3 ms | 5.0±0.4 ms | 4.3±0.7 ms | 5.0±0.3 ms |
| **Siamese (S-multires) inference time** | 25.3±1.4 ms | 15.8±0.6 ms | 7.8±0.9 ms | 16.3±1.9 ms |

Inference time in ms for a normal one-branch classification CNN (1 branch, 5 outputs) *vs.* Siamese network (2 branches, 1 output, 5-way classification).

All measurements performed on an NVIDIA GTX 1080Ti GPU using the S-multires architecture

prototypes, the branch embeddings for each of the prototypes can be stored instead of computed at each run. This means that whenever we have an image from a new class that the network has not seen before, all we have to compute is the embedding for a prototype representing that class and the embedding for the query image. In addition, we need to perform the comparisons between the test image and all the stored prototypes. In total, we need to run the branch twice and the comparisons as many times as there are classes to compare against. For example, when we want to classify a 5th novel class, the inference time would be $2 \times 2.54$ ms (the time it takes to run a branch) + $5 \times 1.18$ ms (one comparison for each of the 5 classes). The total latency would be 11 ms. Although this inference time for a 5-class classification with a Siamese network is two times longer than that of the one-branch classifier, Siamese networks have the advantage that they do not need retraining. To retrain on the 5 novel classes would require 60 s for each epoch (with about 1000 samples per novel class), or around 1.5 hours for a full training on a desktop GPU, which is roughly 491.000 times more expensive in terms of time than running a Siamese network.

## VI. DISCUSSION AND CONCLUSION

### A. Siamese Network Model

In an attempt to reduce the accuracy gap for novel classes between one-shot and incremental learning algorithms, in [11] we proposed novel architectural modifications to the original Siamese networks and adopted a hybrid training mechanism where classification and similarity tasks are combined at training time, but not at inference time. In this paper we describe an additional exploration that combines the original branch architecture seen on Fig. 2(a) with the convolutional similarity module seen on Fig. 3(c). We achieve overall improvements of 1.9%, 22.6%, 19.0% and 12.1% respectively compared to the baseline *S-subtract* Siamese network when tested on Omniglot, Tiny-Imagenet, Cifar100 and Roshambo. Our multi-resolution Siamese architecture contributes the most to the increase in accuracy, by almost 10%.

The state-of-art accuracy on Omniglot is 99.5% and is achieved with meta-learning [52]. Although our best result on Omniglot is slightly below state-of-art, we achieve this result using a much simpler and more flexible algorithm. Similarly, state-of-the-art results have been achieved with meta-learning [53] on different subsets of the ImageNet and CIFAR100 benchmarks. Because of the different dataset subset selections, our results are not directly comparable with these works.

Our accuracy results are lower than what some incremental learning methods achieve [10], [54], but could be improved by performing a K-shot instead of the 1-shot classification we are currently testing. However, this increase in accuracy will come at the cost of higher memory requirements and additional computation.

Our algorithm improves upon the inference times of the incremental learning algorithms mentioned above while maintaining a much higher than chance accuracy. It also has the advantage of not using backpropagation. For example, in [55]

we showed how iCaRL [10] can be successfully used to learn event-based hand symbols incrementally. However, the algorithm required two minutes of training time for every two novel classes, even with a powerful NVIDIA GTX 1080 Ti GPU. The Siamese networks used in this study can classify novel samples from the same event-based Roshambo dataset in only 15 ms on an edge device.

### *B. Hardware evaluation*

To evaluate if our proposed architectural model modifications are compatible with modern edge neural network accelerators, we tested the improved Siamese network on three different platforms (Coral Edge TPU, NVIDIA Jetson Nano and our custom Carbon accelerator), using a workstation GPU (NVIDIA GTX 1080 TI) as reference.

Quantization is a widely adopted technique to reduce the computational load of neural networks. We show how it can be safely used for Siamese networks, achieving a small accuracy reduction in the case of Carbon quantization (-0.4%) and a slight increase (+0.2%) for the Edge TPU (Table IV). Moreover, the networks that we run on the Edge TPU were quantized post-training. This was required due to an incompatibility between the version of Tensorflow we used and in-training quantization process needed for the Edge TPU. In-training quantization is known to be superior to post-training quantization [56], [57]. Therefore, the accuracy of the network on the quantized Edge TPU platform might be even higher.

Weight pruning is another popular technique to accelerate neural network inference by increasing sparsity. To assess its validity in the context of Siamese networks, we trained our S-multires-edge model with multiple levels of weight sparsity and tested it on our Carbon platform. Our results (Table V) show how pruning can be used to obtain performance that is on par with that of a workstation: running a 5-way classification with an effective sparsity of 85.98% takes 13.60 ms whereas the same classification for an effective sparsity of 94.67% takes 10.88 ms. While quantizing the network reduced latency without a significant accuracy drop, weight pruning impacted the accuracy severely.

An important question we aimed to answer with this work was how efficiently edge devices can handle complex network topologies such as the proposed Siamese network architecture. In Table IV, we report the peak utilization, which is the ratio between the maximum throughput of each platform and the effective measured performance. Overall, all the tested platforms underperform, ranging from 4.47% resource utilization (Carbon) to 11.54% (Edge TPU). All the three edge platforms are designed to exploit the high level of parallelism and regularity of CNNs. The complexity of the S-multires-edge model with its multiple kernel resolutions, few kernels per layer and many concatenation operations seems to undermine the ability of the edge platforms to make full use of the available resources. In the last decade, convolutional layers have been the research focus for CNNs accelerators. As the compute time of convolutions decreases, the relative cost of other operations such as concatenation increases, opening new possible research threads on hardware. Similarly, the software handling the mapping of the models to the hardware have a clear margin of improvement and can be the focus of future major work.

Because Siamese Networks do not require retraining unlike other incremental learning techniques in few-shot learning, they are ideal for edge platforms which do not include on-chip training resources. The benchmarking results show that we can run our Siamese classifier at 60 FPS on the Edge TPU, making it suitable for real-time applications. Further algorithmic research will be carried out to determine alternate ways for training Siamese networks so that their classification accuracies during inference approach the state-of-art results from pure classification networks.

## REFERENCES


[1] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[2] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, G. H. Bower, Ed. Academic Press, 1989, vol. 24, pp. 109–165.

[3] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," *ICLR*, no. 2018, pp. 1–16, 2019. [Online]. Available: http://arxiv.org/abs/1904.04232

[4] E. Bart and S. Ullman, "Cross-generalization: learning novel classes from a single example by feature replacement," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 672–679 vol. 1.

[5] Y. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7278–7286.

[6] A. Antoniou, A. Storkey, and H. Edwards, "Augmenting image classifiers using data augmentation generative adversarial networks." in *International Conference on Artificial Neural Networks*, pp. 594-603. Springer, Cham, 2018.

[7] J. Vanschoren, "Meta-learning." in *Automated Machine Learning*, pp. 35-61. Springer, Cham, 2019.

[8] J. Bromley, J. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *Int. J. Patt. Recognit. Artif. Intell.*, 1993.

[9] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *Proceedings of the 32nd International Conference on Machine Learning, Lille, France*, vol. 2, 2015.

[10] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[11] I. A. Lungu, Y. Hu, and S. Liu, "Multi-resolution siamese networks for one-shot learning," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 183–187.

[12] B. M. Lake, R. Salakhutdinov, and J. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, pp. 1332 – 1338, 2015.

[13] "Tiny ImageNet Visual Recognition Challenge." [Online]. Available: https://tiny-imagenet.herokuapp.com/

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2012.

[15] I. A. Lungu, S.-C. Liu, and T. Delbruck, "Fast event-driven incremental learning of hand symbols," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2019, pp. 25–28.

[16] NVIDIA, "Jetson nano." [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[17] Coral, "Tensorflow models on the edge tpu." [Online]. Available: https://coral.ai/docs/edgetpu/models-intro/

[18] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3630–3638.

[19] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4077–4087.

[20] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, pp. 1126–1135.

[21] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 539–546 vol. 1.

[22] R. R. Varior, M. Haloi, and G. Wang, "Gated siamese convolutional neural network architecture for human re-identification," in *European Conference on Computer Vision*. Springer, 2016, pp. 791–808.

[23] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang, "Person re-identification with deep similarity-guided graph neural network," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 508–526.

[24] U. Chaudhuri, B. Banerjee, and A. Bhattacharya, "Siamese graph convolutional network for content based remote sensing image retrieval," *Computer Vision and Image Understanding*, vol. 184, pp. 22–30, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314219300578

[25] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *International Journal of Computer Vision*, vol. 124, no. 2, pp. 237–254, 2017. [Online]. Available: http://www.xrce.xerox.

[26] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, 2016, pp. 5695–5703. [Online]. Available: http://www.cs.toronto.edu/

[27] R. P. Pflugfelder, "Siamese learning visual tracking: A survey," *CoRR*, vol. abs/1707.00569, 2017.

[28] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching Networks for One Shot Learning," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3630–3638.

[29] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[30] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *CoRR*, vol. abs/1803.02999, 2018.

[31] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BJgklhAcK7

[32] C. Wang, S. Dong, X. Zhao, G. Papanastasiou, H. Zhang, and G. Yang, "SaliencyGAN: Deep learning semisupervised salient object detection in the Fog of IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2667–2676, 2020.

[33] D. Zhang, D. Meng, and J. Han, "Co-saliency detection via a self-paced multiple-instance learning framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 5, pp. 865–878, 2017.

[34] L. Cavigelli and L. Benini, "Origami: A 803-GOp/s/w convolutional network accelerator," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 11, pp. 2461–2475, Nov 2017.

[35] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. S. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," *CoRR*, vol. abs/1708.04485, 2017.

[36] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-X: An accelerator for sparse neural networks," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2016, pp. 1–12.

[37] V. Gokhale, A. Zaidy, A. X. M. Chang, and E. Culurciello, "Snowflake: An efficient hardware accelerator for convolutional neural networks," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[38] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, "BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28nm CMOS," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*, April 2018, pp. 1–4.

[39] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu, and T. Delbruck, "NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2018.

[40] E. Ceolini, J. Anumula, S. Braun, and S.-C. Liu, "Event-driven pipeline for low-latency low-compute keyword spotting and speaker verification system," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 7953–7957.

[41] M. Baharani, S. Mohan, and H. Tabkhi, "Real-time person re-identification at the edge: A mixed precision approach," in *International Conference on Image Analysis and Recognition*. Springer, 2019, pp. 27–39.

[42] Z. Zheng, L. Zheng, and Y. Yang, "A discriminatively learned cnn embedding for person reidentification," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 1, pp. 13:1–13:20, 2017.

[43] X. Liu, Y. Zhou, J. Zhao, R. Yao, B. Liu, and Y. Zheng, "Siamese convolutional neural networks for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 8, pp. 1200–1204, Aug 2019.

[44] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[46] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240$\times$180 130dB 3$\mu$s latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 2333–2341, 2014.

[47] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr, and T. Delbruck, "Steering a predator robot using a mixed frame/event-driven convolutional neural network," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, June 2016, pp. 1–8.

[48] I. Lungu, F. Corradi, and T. Delbruck, "Live demonstration: Convolutional neural network driven by Dynamic Vision Sensor playing RoShamBo," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–1.

[49] Tensorflow, "Quantization aware training comprehensive guide." [Online]. Available: https://www.tensorflow.org/model_optimization/guide/quantization/training_comprehensive_guide

[50] ——, "Pruning comprehensive guide." [Online]. Available: https://www.tensorflow.org/model_optimization/guide/pruning/comprehensive_guide

[51] D. W. Blalock, J. J. G. Ortiz, J. Frankle, and J. V. Guttag, "What is the state of neural network pruning?" in *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*, I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, Eds. mlsys.org, 2020.

[52] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layer-wise metric and subspace," in *ICML*, 2018.

[53] L. Song, J. Liu, and Y. Qin, "Fast and generalized adaptation for few-shot learning," *ArXiv*, vol. abs/1911.10807, 2019.

[54] F. M. Castro, M. J. Marin-Jimenez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[55] I. A. Lungu, S. Liu, and T. Delbruck, "Fast event-driven incremental learning of hand symbols," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2019, pp. 25–28.

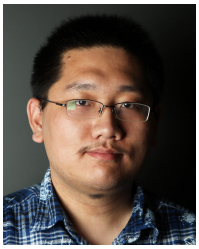
[56] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "WRPN: Wide reduced-precision networks," in *International Conference on Learning Representations*, 2018.

[57] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S.-C. Liu, "Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms," *Frontiers in Neuroscience*, vol. 9, p. 222, 2015.


## AUTHOR INFORMATION

**Iulia-Alexandra Lungu** received her BSc in Bioinformatics from the University Claude Bernard Lyon and her MSc in Computational Neuroscience from the Technical University of Berlin. She is now pursuing her PhD degree at the Institute of Neuroinformatics, where she works with talented hardware designers to develop efficient and powerful algorithms and hardware solutions for AI. Her main interests revolve around incremental and few-shot learning.

**Alessandro Aimar** received his B.Sc. degree in Physical Engineering from Politecnico di Torino (Italy) and his M.Sc. degree in Nanotechnologies from a joint program of Politecnico di Torino, INP Grenoble (France) and EPFL (Switzerland). After working as engineer at Imagination Technologies (UK) he joined the Institute of Neuroinformatics for his PhD. In 2019 he founded Synthara AG, a startup focused on deep learning and neuromorphic hardware.



**Yuhuang Hu** received his BSc in Computer Science from the University of Malaya in 2015 and Joint MSc in Neural Systems and Computation from the University of Zürich and ETH Zürich in 2017. He is currently a PhD student at the Institute of Neuroinformatics, University of Zürich and ETH Zürich. His main research interests include deep neural networks, self-supervised learning, event-based processing, and computer vision.

**Tobi Delbruck** ( M'99–SM'06–F'13) received the B.Sc. degree in physics from University of California in 1986 and PhD degree from California Institute of Technology in 1993. Currently, he is a Professor of Physics and Electrical Engineering with the Institute of Neuroinformatics, University of Zurich and ETH Zurich, where he has been since 1998. His interests include dynamic vision sensor silicon retina event cameras, data-driven deep neural network hardware accelerators, robotics, and dynamical systems control applications of these technologies.

**Shih-Chii Liu** (M'02–SM'07) received her PhD degree in Computation and Neural Systems from the California Institute of Technology, Pasadena, in 1997. She worked at various companies, including Gould American Microsystems, LSI Logic, and Rockwell International Research Labs. Currently, she is a professor with the Institute of Neuroinformatics at the University of Zurich, Switzerland. Her interests include low-power neuromorphic event-driven sensor design; bio-inspired and deep learning algorithms and hardware for energy-efficient, real-time, adaptive intelligent systems.