MAX-POOLING OPERATIONS IN DEEP SPIKING NEURAL NETWORKS

Yuhuang Hu
Institute of Neuroinformatics,
University of Zürich and ETH Zürich,
yuhuang.hu@ini.uzh.ch

Supervisor:
Dr. Michael Pfeiffer

ABSTRACT

In converting Convolutional Neural Networks (ConvNets) to Spiking Neural Networks (SNNs), previous efforts avoid the max-pooling operation due to additional design and computation complexity. However, as the most default choices of implementing complex cells and sub-sampling method, max-pooling is an indispensable component in Deep SNNs. In this report, I propose three implementations of the max-pooling operation that result in a low performance loss during network conversion by addressing the activities of spike trains and timing. This work ships a missing building block in ANNs-SNNs conversion pipeline and hence allows a wider range of DNNs to be transformed into respective SNNs.

1 INTRODUCTION

As Deep Neural Networks (DNNs) revolutionized the field of Machine Learning and set groundbreaking results in various visual recognition challenges such as image and video classification (Krizhevsky et al., 2012; Karpathy et al., 2014), object tracking (Held et al., 2016; Zhu et al., 2016), semantic segmentation (Badrinarayanan et al., 2015; Shelhamer et al., 2016), visual Turing test (Vinyals et al., 2015; Johnson et al., 2016; Yao et al., 2015), etc, these data and energy hungry analog neural networks (ANNs) posed the need for low-cost specialized hardware acceleration for real-time intelligent applications. The nature of sparsely and event-driven firing mechanisms make Spiking Neural Networks (SNNs) a great candidate for mitigating the energy demands and expensive computation cost. Few past excellent researches have shown that it is possible to map ANNs to SNNs with nearly lossless performance (Perez-Carrasco et al., 2013; Cao et al., 2014; Diehl et al., 2015; Hunsberger & Eliasmith, 2015).

Former attempts of ANNs-SNNs conversion assume that the complex cell in Convolutional Neural Networks (ConvNets) is realized by average-pooling because of its simplicity (Cao et al., 2014; Diehl et al., 2015; Hunsberger & Eliasmith, 2015). However, max-pooling is largely adapted in most successful ConvNets since it gives extra advantage of translation invariance and is biologically inspired (Yu et al., 2002). As the operation functions differently in ANNs and SNNs, mapping max-pooling directly into SNNs without considering spiking activities causes a big performance drop. In this report,

- I propose three implementations of spiking max-pooling by taking advantage of spike trains and timing information. (Section 2)
- All three implementations are tested with various pooling settings (*e.g.* downscale size, pooling strides, number of pooling layers, etc) and result in a low performance drop in network conversion. (Section 3)
- The proposed implementation enabled a broader class of DNNs that can be mapped into SNNs paradigm. General discussion on experiments, the pooling method and possible future plans are in Section 4.

2 METHODS

2.1 MAX-POOLING IN CONVNETS

Max-Pooling is the most popular choice of implementing a complex cell in ConvNets. This non-linear sub-sampling greatly reduces the computation costs for subsequent layers and introduces a form of translation invariance (Goodfellow et al., 2016). Most state-of-the-art deep ConvNets employ max-pooling as the default choice (Krizhevsky et al., 2012; Karpathy et al., 2014; Simonyan & Zisserman, 2014; Szegedy et al., 2015).

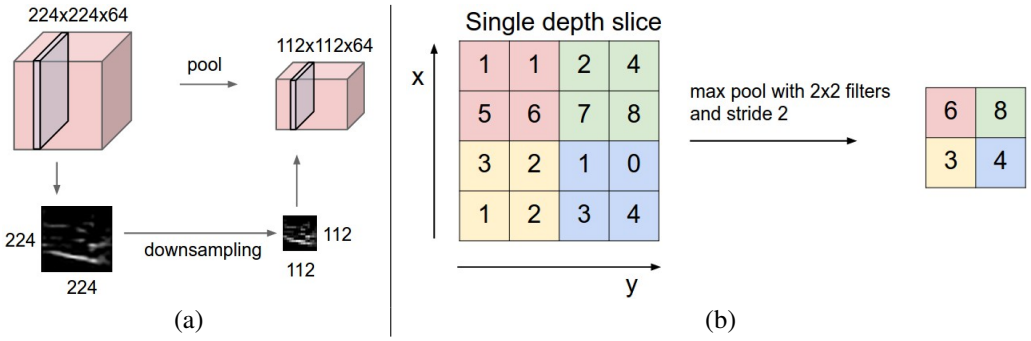

Figure 1: (a) A conceptual example of the MAXPOOLING function with the input of 64 224×224 feature maps, the pooled outcome are 64 112×112 feature maps. The MAXPOOLING function here has downscale factor $(2, 2)$ and strides $(2, 2)$; (b) An example of the MAXPOOLING function with downscale factor $(2, 2)$ and strides $(2, 2)$, the values in the pooled feature map are the maximum values in the respective regions. **Image Source:** *CS231n Convolutional Neural Networks for Visual Recognition* `http://cs231n.github.io/`.

To give a concrete description of max-pooling, let's define the MAXPOOLING function as a filter w where it has a downscale factor $d = (d_v, d_h)$, a stride $s = (s_v, s_h)$ and a zero-padding parameter $p = (p_v, p_h)$. Suppose there are N input feature maps $\mathcal{F} = \{f^{(i)}\}_{i=1}^{N}$ and $f^{(i)} \in \mathbb{R}^{n \times m}$. The output feature maps $\hat{\mathcal{F}} = \{\hat{f}^{(i)}\}_{i=1}^{N}$ can be computed by:

$$\hat{f}^{(i)} = \text{MAXPOOLING}(f^{(i)}, w) \tag{1}$$

The output feature map $\hat{f}^{(i)}$ is arranged in rectangular shape with the size $(\lfloor (n - d_v - 2p_v)/s_v + 1 \rfloor, \lfloor (m - d_h - 2p_h)/s_h + 1 \rfloor)$. Each entry $\hat{f}^{(i)}_{(x,y)}$ contains the maximum activation value from a subset of input feature map $f^{(i)}$'s entries:

$$\hat{f}^{(i)}_{(x,y)} = \max\{f^{(i)}_{(x',y')}; x' \in [xs_v, xs_x + d_v - 1], y' \in [ys_h, ys_h + d_h - 1]\} \tag{2}$$

where x (the row index) and y (the column index) start from 0. For example, in Fig. 1(b), it demonstrates a filter w with downscale factor $d = (2, 2)$, stride $s = (2, 2)$ and zero-padding $p = (0, 0)$, the input size is (4,4), and the output size is then (2,2).

2.2 MAX-POOLING IN SPIKING CONVNETS

Previous research strove to build spiking hierarchical model for object recognition by translating the HMAX model that is proposed in Riesenhuber & Poggio (1999). Zhao et al. (2014) presented the first conversion of the HMAX model where the architecture replaces Simple Cells (S layer) with the Leaky Integrate-and-Fire model and performs MAX-like neural competition by applying Time Domain Winner-Take-All in complex cells (C layer). Orchard et al. (2015) followed a similar approach where simple cells are characterized by a Integrated-and-Fire (IF) neuron model with linear decay and a refractory period and complex cells (C layer) apply a so-called "time to first spike" approach (which is as same as Zhao et al. (2014)'s idea).

In SNNs, the max-pooling operation connects the neuron that has the strongest response to the stimulus in the time domain. However as the dynamics of the neuron is changing over time with the

input spikes, the pooling operation should also adjust and rewire the neuron connections accordingly. In order to take advantage of the timing information of arrival spikes $\mathcal{F}(t) = \{f^{(i)}(t)\}_{i=1}^{N}$, we propose a pooling gate that is parameterized by a set of intermediate maps $\tilde{\mathcal{F}}(t) = \{\tilde{f}^{(i)}(t)\}_{i=1}^{N}$, which monitors the input neurons' activities. And the SPIKINGMAXPOOLING function can be formulated as Eqn. 3.

$$\hat{f}^{(i)}(t) = \text{SPIKINGMAXPOOLING}(f^{(i)}(t), \tilde{f}^{(i)}(t), w) \tag{3a}$$

$$x^*, y^* = \arg\max_{x',y'} \{\tilde{f}^{(i)}_{(x',y')}(t); x' \in [xs_v, xs_x + d_v - 1], y' \in [ys_h, ys_h + d_h - 1]\} \tag{3b}$$

$$\hat{f}^{(i)}_{(x,y)}(t) = f^{(i)}_{(x^*,y^*)}(t) \tag{3c}$$

Note that a new time variable t is introduced here for characterizing input and output spikes.

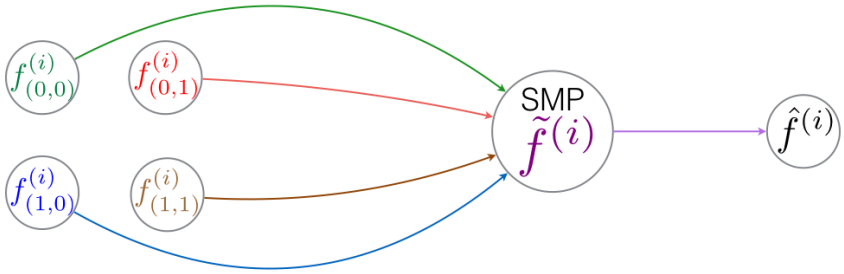

Figure 2: A conceptual example of SPIKINGMAXPOOLING function. The pooling gate (SMP) in the middle receives input from four input neurons at left (arranged in 2×2 grid), after evaluating $\tilde{f}^{(i)}$, it connects one of the input neurons to the output neuron at the right.

SPIKINGMAXPOOLING by online average spike rate (Avg. Max) By calculating the online average of input spike rate for each neuron, the pooling gate captures the strongest firing neuron over the to-date history (unlike moving average, which takes a limited time window). This method is different from the "time-to-first-spike" approach described in (Zhao et al., 2014; Orchard et al., 2015) as in the neuron that fires the maximum may not be the neuron that responds first.

$$\tilde{f}^{(i)}(t) = \tilde{f}^{(i)}(t) + \frac{f^{(i)}(t) - \tilde{f}^{(i)}(t)}{t} \tag{4}$$

SPIKINGMAXPOOLING by accumulated absolute spike rate (Fir. Max) In order to create a hybrid of the ideas of Avg. Max and "time-to-first-spike", one could calculate the absolute spike rate and accumulate it over time. This method slowly punishes the importance of later arriving spikes and still opens the chance that other neurons may replace the first-responding neuron when it has accumulated enough strength.

$$\tilde{f}^{(i)}(t) = \tilde{f}^{(i)}(t) + \frac{f^{(i)}(t)}{t} \tag{5}$$

This method is not a measure of spike rate instead of relative strength of spiking activities. Section 3 shows that this method represents the best performance in experiments.

SPIKINGMAXPOOLING by accumulated spike rate with exponential decay (Exp. Max) This method is to formulate a "time-to-first-spike" equivalent. The exponential decay 2^t ensures that only the first-responded neuron connects to the output neuron.

$$\tilde{f}^{(i)}(t) = \tilde{f}^{(i)}(t) + \frac{f^{(i)}(t)}{2^t} \tag{6}$$

Above mentioned methods can be extended flexibly by adding hyper-parameters to components in the right of the equations. The disadvantage of these methods is that the pooling gate has to monitor the activities for all input neurons, and this will raise the memory and computation cost. However this cost is necessary since unlike average-pooling, the neuron connections from input to output neurons are dynamical instead of static.

3 EXPERIMENTS

All experiments described in following two sections have the same parameter settings as in Table 1. The description and unit of each parameter are found at Appendix A. For completeness, the experiments are designed to test in different kinds of pooling configurations, *e.g.* downscale size, stride size, number of pooling layers, etc. Model specifications are listed in Appendix B and C.

Table 1: Experiment Parameter Settings

Cell Parameters							
v_thresh	1.0	v_reset	0.0	v_rest	0.0	e_rev_E	10.0
e_rev_I	-10.0	cm	0.09	tau_m	1000	tau_refrac	0.0
tau_syn_E	0.01	tau_syn_I	0.01	i_offset	0		
softmax_clockrate			300				
Simulation Parameters							
duration	150	dt	1.0	delay	1	input_rate	1000
timestep_fraction			0	diff_to_min_rate			0
diff_to_max_rate			0	online_normalization			false
percentile	99.9	reset	Reset by subtraction				

An experiment with above settings is tasked to compare the performance between a given ANN and the respective converted SNN. The performance measure in this report uses classification accuracy. The ANN is evaluated with the dataset in order to obtain the ANN classification accuracy (ANN Acc.). Then the ANN is fed into the SNN conversion pipeline. The first step of the conversion is model parameter normalization. The weights and biases of the ANN are rescaled according to 99.9 percentile of the network activation. Secondly, the neuron model that is used in the ANN is replaced with a spiking neuron model which mimics `ReLU` activation function. Additionally, for each max-pooling layer, a set of intermediate maps $\hat{\mathcal{F}}(t)$ is added for monitoring incoming spike trains. This conversion process is then completed and the converted model is also saved. The dataset is translated to spike trains for testing the converted SNN. Here the experiment sets the input rate as $1000\,Hz$ and the SNN has to classify a sample in $150\,ms$. Note that the proposed three max-pooling methods are tested independently. The SNN performance (SNN Acc.) for the max-pooling methods is documented finally.

3.1 MNIST

MNIST is a widely used visual recognition benchmark that consists of 70,000 handwritten digits in 10 classes (LeCun et al., 1998). Table. 2 summarizes the experiments with 8 different pre-trained MNIST models. The first column in Table. 2 has the IDs of the pre-trained models. The details of network configuration can be found in Appendix. B.

The input samples are converted to Poisson spike trains. The probability for a input neuron to fire is proportional to the analog value of the corresponding pixel, and limited by the parameter `input_rate`. The SPIKINGMAXPOOLING method Fir. Max delivers the best performance and stability across the majority of the experiments. In model `98.72.avg.max.hybrid`, the Fir. Max method even gives higher accuracy in SNN than ANN. The Exp. Max method gives the worst performance in general.

Table 2: Summary of MNIST Experiments.

Exp. Type	Nr. Param.	Nr. Layers	Pool Method	ANN Acc.	SNN Acc.	Acc. Diff.	Best
99.16	601K	5	Avg. Max	99.16%	99.16%	0.00%	⋆
			Fir. Max		99.15%	0.01%	
			Exp. Max		99.15%	0.01%	
99.17	601K	5	Avg. Max	99.17%	99.08%	0.09%	
			Fir. Max		99.12%	0.05%	⋆
			Exp. Max		99.13%	0.04%	
99.38.deeper. mnist	177K	8	Avg. Max	99.38%	99.36%	0.02%	
			Fir. Max		99.38%	0.00%	⋆
			Exp. Max		99.34%	0.04%	
99.44	1.1M	7	Avg. Max	99.44%	99.44%	0.00%	
			Fir. Max		99.44%	0.00%	
			Exp. Max		99.44%	0.00%	
98.97. larger.pool	27K	7	Avg. Max	98.97%	98.93%	0.04%	⋆
			Fir. Max		98.88%	0.09%	
			Exp. Max		97.87%	**1.10%**	
98.76.non. overlap.pool	474K	7	Avg. Max	98.76%	98.40%	0.36%	
			Fir. Max		98.68%	0.08%	⋆
			Exp. Max		98.33%	0.43%	
99.44. overlap.pool	3.1M	9	Avg. Max	99.44%	99.22%	0.22%	
			Fir. Max		99.30%	0.14%	⋆
			Exp. Max		98.21%	0.23%	
98.72.avg. max.hybrid	433K	10	Avg. Max	98.72%	98.64%	0.08%	
			Fir. Max		**98.75%**	**0.03%**	⋆
			Exp. Max		98.32%	0.40%	

Table 3: Summary of CIFAR10 Experiments.

Exp. Type	Nr. Param.	Nr. Layers	Pool Method	ANN Acc.	SNN Acc.	Acc. Diff.	Best
74.16	1.4M	7	Avg. Max	74.16%	72.25%	1.91%	
			Fir. Max		73.79%	0.37%	⋆
			Exp. Max		73.26%	0.90%	
78.08	624K	8	Avg. Max	78.08%	77.99%	0.09%	
			Fir. Max		**78.21%**	0.13%	⋆
			Exp. Max		78.07%	0.01%	
82.65	1.3M	8	Avg. Max	82.65%	**82.73%**	0.08%	⋆
			Fir. Max		82.51%	0.14%	
			Exp. Max		81.75%	0.90%	
81.11	1.3M	8	Avg. Max	81.11%	81.01%	0.10%	
			Fir. Max		81.11%	0.00%	⋆
			Exp. Max		80.75%	0.36%	
70.96. larger.pool	81K	6	Avg. Max	70.96%	66.31%	4.65%	
			Fir. Max		70.47%	0.49%	⋆
			Exp. Max		61.97%	8.99%	
73.28.non. overlap.pool	163K	6	Avg. Max	73.28%	73.02%	0.26%	
			Fir. Max		**73.87%**	**0.50%**	⋆
			Exp. Max		**72.75%**	0.47%	
69.40. overlap.pool	3.8M	9	Avg. Max	69.40%	61.60%	7.80%	⋆
			Fir. Max		61.57%	7.83%	
			Exp. Max		61.39%	**8.01%**	
76.70.avg. max.hybrid	100K	9	Avg. Max	76.70%	75.50%	1.20%	
			Fir. Max		76.15%	0.55%	⋆
			Exp. Max		74.06%	2.64%	
73.11. deeper.max	100K	10	Avg. Max	73.11%	71.00%	2.11%	
			Fir. Max		72.03%	1.08%	⋆
			Exp. Max		68.65%	4.46%	

3.2 CIFAR-10

CIFAR-10 is a collection of 60,000 tiny images (available in 32×32) that contains 10 classes of objects (Krizhevsky, 2009). The benchmarks is one of the most popular image recognition datasets. We evaluated 9 pre-trained models for testing SPIKINGMAXPOOLING function (model details in Appendix. C). The results are presented in Table. 3.

Unlike MNIST dataset, Poisson input is switched off during the experiments. All other configurations remain same. Similar conclusions can be drawn from the results: Fir. Max performs the best and Exp. Max offers the worst.

4 DISCUSSION

In this report, I proposed three methods of implementing the max-pooling operation in Deep Spiking Neural Networks. The methods are tested and proven to be successful in capturing correct network dynamics in converting from ANNs to SNNs. This enabled a broader class of ANNs that can be translated into SNNs, such as object detection, semantic segmentation, and tracking. These task-specific networks can potentially be converted given the availability of spiking max-pooling operations.

Experiments show that the Fir. Max method has the best performance and stability. By addressing both early arrival spikes and later arrival spikes, it shows that considering only first-responded neuron may not enough, in fact, the theoretical equivalent method Exp. Max reports the worst performance among three proposed implementations.

The pooling gate only monitors the spike activities from incoming layer, therefore it is mostly influenced by its updating mechanisms instead of other parameters in the network. As pointed out in section 2, the realization of such pooling-gate introduces a set of parameters as large as number of input neurons. However this memory and computation cost is necessary and can be alleviated by computing them in parallel. Future plans include testing larger image recognition networks and other task-specific deep ANNs, integrating more not-yet-available ANNs features to SNNs (*e.g.* weighted pooling methods, up-sampling operations, recurrent cells such as LSTM, GRU, etc.) and loading such converted SNNs to specialized hardware acceleration platform.

A DESCRIPTION OF EXPERIMENT PARAMETERS

There are 12 cell parameters. All experiments share the same settings.

Table 4: Description of cell parameters.

Cell Parameter	Description
v_thresh	Threshold in mV defining the voltage at which a spike is fired.
v_reset	Reset potential in mV of the neurons after spiking.
v_rest	Resting membrane potential in mV.
e_rev_E	Reversal potential for excitatory input in mV.
e_rev_I	Reversal potential for inhibitory input in mV.
i_offset	Offset current in nA.
cm	Membrane capacitance in nF.
tau_m	Membrane time constant in milliseconds.
tau_refrac	Duration of refractory period in milliseconds of the neurons after spiking.
tau_syn_E	Decay time of the excitatory synaptic conductance in milliseconds.
tau_syn_I	Decay time of the inhibitory synaptic conductance in milliseconds.
softmax_clockrate	the firing rate in Hz of an external Poisson clock. Note that this rate is $\leq 1000 \times 1/dt\ Hz$.

There are 11 simulation parameters. The parameter maxpool_type is specified by particular experiment settings. The remaining 10 parameters are preset and used in all experiments.

Table 5: Description of simulation parameters.

Sim. Param.	Description
duration	Runtime of simulation of one input in milliseconds.
dt	Time resolution of spikes in milliseconds.
delay	Delay in milliseconds. Must be equal to or greater than the resolution.
input_rate	Poisson spike rate in Hz for a fully on pixel of the input image. Note that the input_rate$\leq 1000 \times 1/dt\ Hz$.
timestep_fraction	If set to 10 (default), the parameter modification mechanism described in online_normalization will be performed at every 10th timestep.
diff_to_min_rate	When The firing rates of a layer are below this value, the weights will NOT be modified in the feedback mechanism described in online_normalization.
diff_to_max_rate	If the highest firing rate of neurons in a layer drops below the maximum firing rate by more than diff_to_max_rate, we set the threshold of the layer to its highest rate. Set the parameter in Hz.
online_normalization	Normalization eliminates saturation but introduces under-sampling. If they drop below the maximum firing rate by more than diff_to_max_rate, we set the threshold of the layer to its highest rate.
percentile	Use the activation value in the specified percentile for normalization. Set to 50 for the median, 100 for the max.
reset	Choose the reset mechanism to apply after spike. **Reset to zero:** After spike, the membrane potential is set to the resting potential. **Reset by subtraction:** After spike, the membrane potential is reduced by a value equal to the threshold.
maxpool_type	Implementation variants of spiking MAXPOOLING layers, based on avg_max: running average of firing rate; fir_max: accumulated absolute firing rate; exp_max: exponentially accumulated absolute firing rate.

B SUMMARY OF MNIST PRETRAINED MODELS

There are total 8 pretrained MNIST models for testing MAXPOOLING operations: `99.16` (in Table 6), `99.17` (in Table 7), `99.38.deeper.mnist` (in Table 8), `99.44` (in Table 9), `98.97.larger.pool` (in Table 10), `98.76.non.overlap.pool` (in Table 11), `99.44.overlap.pool` (in Table 12), `98.72.avg.max.hybrid` (in Table 13).

Table 6: MNIST model `99.16`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	128			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 7: MNIST model `99.17`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	128			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 8: MNIST model `99.38.deeper.mnist`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	500			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 9: MNIST model `99.44`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$20 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$50 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	800			`ReLU`
Dropout	drop rate: 0.50			
Dense	500			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 10: MNIST model `98.97.larger.pool`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	3×3	3×3	`valid`	
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	4×4	4×4	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	500			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 11: MNIST model `98.76.non.overlap.pool`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	3×3	`valid`	
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	3×3	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	800			`ReLU`
Dropout	drop rate: 0.50			
Dense	500			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 12: MNIST model `99.44.overlap.pool`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$20 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	1×1	`valid`	
Convolution	$20 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	1×1	`valid`	
Convolution	$20 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	1×1	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	800			`ReLU`
Dropout	drop rate: 0.50			
Dense	500			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 13: MNIST model `98.72.avg.max.hybrid`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Average Pooling	2×2	2×2	`valid`	
Convolution	$20 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	800			`ReLU`
Dropout	drop rate: 0.50			
Dense	500			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

C SUMMARY OF CIFAR-10 PRETRAINED MODELS

There are total 9 pretrained CIFAR-10 models for testing MAXPOOLING operations: `74.16` (in Table 14), `78.08` (in Table 15), `82.65` (in Table 16), `81.11` (in Table 17), `70.96.larger.pool` (in Table 18), `73.28.non.overlap.pool` (in Table 19), `69.40.overlap.pool` (in Table 20), `76.70.avg.max.hybrid` (in Table 21), `73.11.deeper.max` (in Table 22).

Table 14: CIFAR-10 model `74.16`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$20 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$50 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	800			`ReLU`
Dropout	drop rate: 0.50			
Dense	500			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 15: CIFAR-10 model `78.08`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.40			
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.40			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 16: CIFAR-10 model `82.65`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Convolution	$64 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$64 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 17: CIFAR-10 model `81.11`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Convolution	$64 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$64 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 18: CIFAR-10 model `70.96.larger.pool`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	3×3	3×3	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	4×4	4×4	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 19: CIFAR-10 model `73.28.non.overlap.pool`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	3×3	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	3×3	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 20: CIFAR-10 model `69.40.overlap.pool`

Layer Type	**Description**			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	1×1	`valid`	
Convolution	$32 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	1×1	`valid`	
Convolution	$32 \times 5 \times 5$	1×1	`valid`	`ReLU`
Max Pooling	2×2	1×1	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Dropout	drop rate: 0.25			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 21: CIFAR-10 model `76.70.avg.max.hybrid`

Layer Type	Description			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Average Pooling	2×2	2×2	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`valid`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

Table 22: CIFAR-10 model `73.11.deeper.max`

Layer Type	Description			
	FS/PS/OS	**Strides**	**Border Mode**	**Activation**
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Convolution	$32 \times 3 \times 3$	1×1	`same`	`ReLU`
Max Pooling	2×2	2×2	`valid`	
Dropout	drop rate: 0.25			
Flatten				
Dense	512			`ReLU`
Dropout	drop rate: 0.50			
Dense	10			`Softmax`

REFERENCES

Badrinarayanan, Vijay, Kendall, Alex, and Cipolla, Roberto. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

Cao, Yongqiang, Chen, Yang, and Khosla, Deepak. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2014. ISSN 1573-1405.

Diehl, Peter U., Neil, Daniel, Binas, Jonathan, Cook, Matthew, Liu, Shih-Chii, and Pfeiffer, Michael. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pp. 1–8, 2015.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. URL `http://www.deeplearningbook.org`. Book in preparation for MIT Press.

Held, David, Thrun, Sebastian, and Savarese, Silvio. Learning to track at 100 FPS with deep regression networks. *CoRR*, abs/1604.01802, 2016.

Hunsberger, Eric and Eliasmith, Chris. Spiking deep networks with LIF neurons. *CoRR*, abs/1510.08829, 2015.

Johnson, Justin, Karpathy, Andrej, and Fei-Fei, Li. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.

Karpathy, Andrej, Toderici, George, Shetty, Sanketh, Leung, Thomas, Sukthankar, Rahul, and Fei-Fei, Li. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. Technical report, 2009.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Orchard, G., Meyer, C., Etienne-Cummings, R., Posch, C., Thakor, N., and Benosman, R. Hfirst: A temporal approach to object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2028–2040, Oct 2015. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2392947.

Perez-Carrasco, J. A., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., and Linares-Barranco, B. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2706–2719, Nov 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.71.

Riesenhuber, Maximilian and Poggio, Tomaso. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

Shelhamer, Evan, Long, Jonathan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1605.06211, 2016.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL `http://arxiv.org/abs/1409.4842`.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 3156–3164, 2015.

Yao, L., Torabi, A., Cho, K., Ballas, N., Pal, C., Larochelle, H., and Courville, A. Describing videos by exploiting temporal structure. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 4507–4515, Dec 2015. doi: 10.1109/ICCV.2015.512.

Yu, Angela J., Giese, Martin A., and Poggio, Tomaso A. Biophysiologically plausible implementations of the maximum operation. *Neural Comput.*, 14(12):2857–2881, December 2002. ISSN 0899-7667. doi: 10.1162/089976602760805313. URL http://dx.doi.org/10.1162/089976602760805313.

Zhao, B., Chen, S., and Tang, H. Bio-inspired categorization using event-driven feature extraction and spike-based learning. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 3845–3852, July 2014. doi: 10.1109/IJCNN.2014.6889541.

Zhu, Gao, Porikli, Fatih, and Li, Hongdong. Robust visual tracking with deep convolutional neural network based object proposals on pets. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.