Research Paper Reading

Backpropagation-Based Learning Techniques for Deep Spiking Neural Networks: A Survey

Ngo-Doanh NGUYEN
M5262108

2023-07-12

Overview

1. Key Contribution
2. Training Deep SNNs
3. Optimizing Deep SNNs
4. Impact of encoding, training, architecture on accuracy-latency trade-off
5. Conclusion

Overview

1. Key Contribution
2. Training Deep SNNs
3. Optimizing Deep SNNs
4. Impact of encoding, training, architecture on accuracy-latency trade-off
5. Conclusion

Key Contribution

- Survery on backpropagation-based learning method for SNN
  - List the trending methodology
  - List pros/cons, affection of each method

Overview

1. Key Contribution
2. Training Deep SNNs
3. Optimizing Deep SNNs
4. Impact of encoding, training, architecture on accuracy-latency trade-off
5. Conclusion

ANN vs SNN

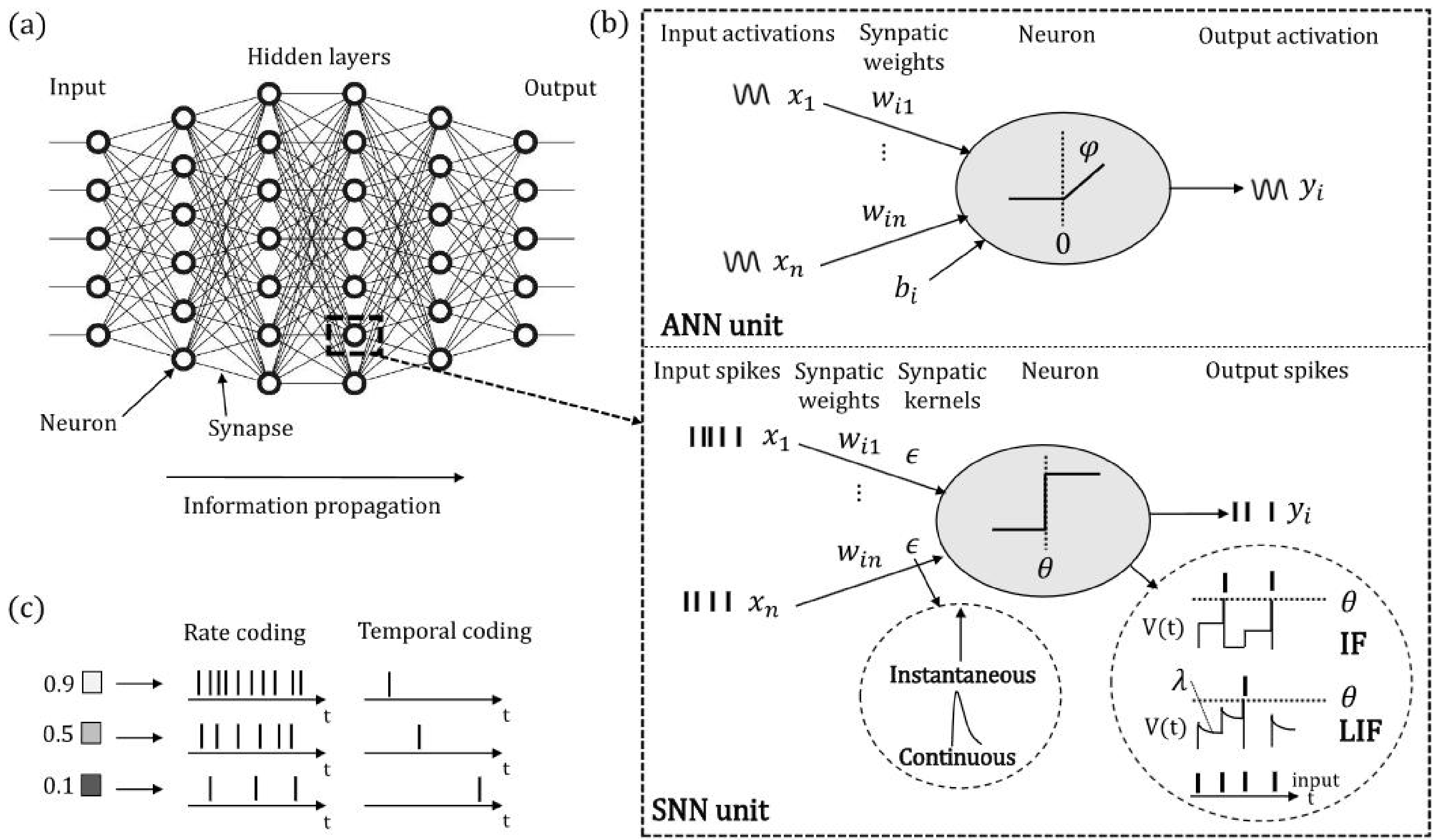


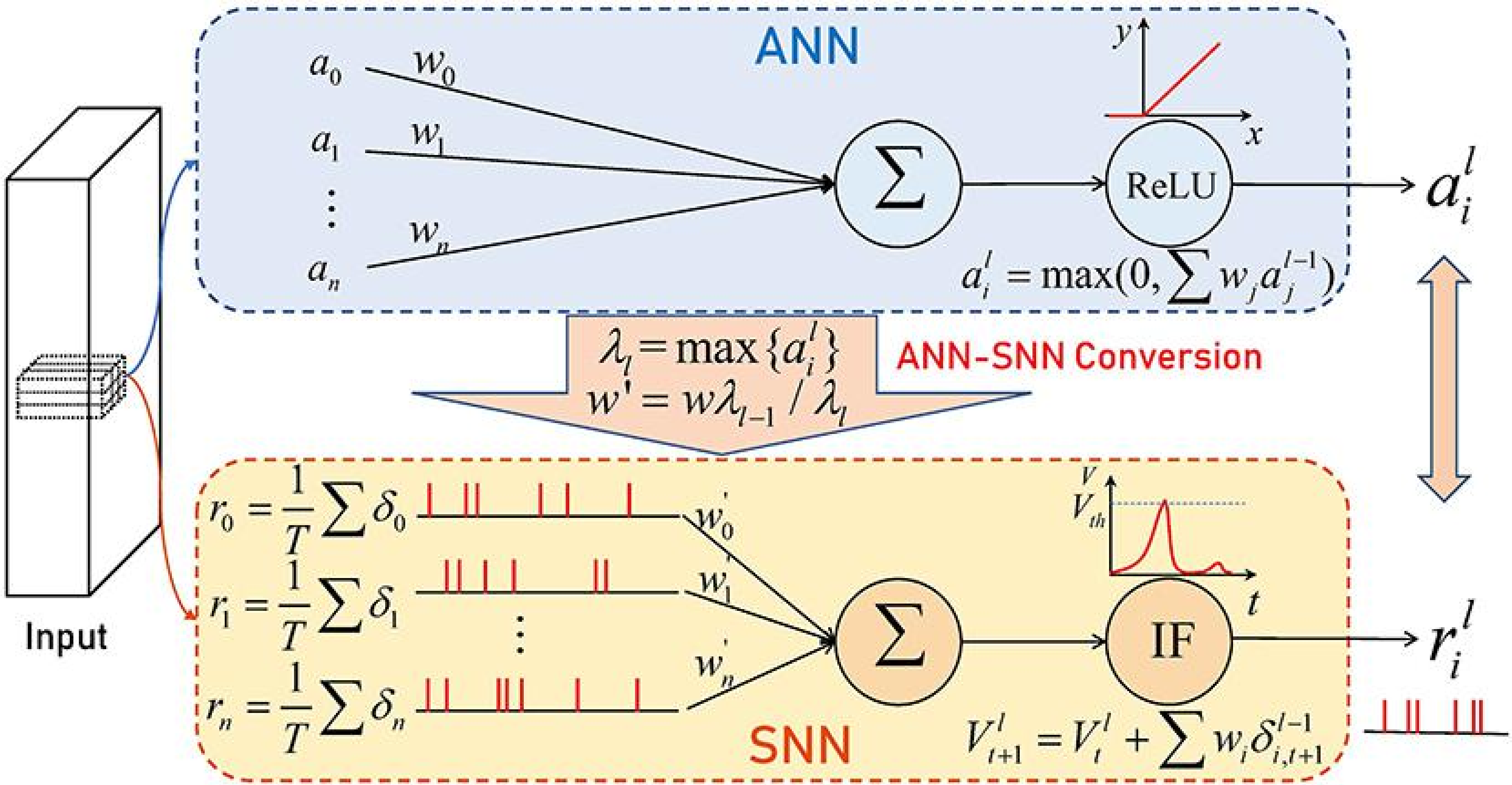
Fig. 1. (a) Feedforward fully connected neural network. (b) ANN and SNN neuron and synapse models. (c) Input encoding: example of pixel-to-spike conversion with a rate coding or temporal (latency) coding.

Difference between ANN & SNN

Training Deep SNNs

- ANN-to-SNN conversion
  - Conversion with Rate Coding
  - Conversion with Temporal Coding
- Backpropagation-Based Learning Algorithms
  - Spatial Approaches
  - Spatiotemporal Approaches
  - Single-Spike Approaches

ANN-to-SNN conversion



*Y. Li et al. "BSNN: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons"

- **Activation transformation use rate-coding or temporal-coding**
- **Normalized weights**

ANN-to-SNN conversion with Rate Coding

- Problems:
  - The conversion process results in errors in some cases
    - the ANN activation is too high and cannot be accurately represented by the spike rate given a fixed simulation duration
- Methods:
  - (1) Weight Normalizaiton (rescaling weights in each layer)
  - (2) Balances threshold in each layer
  - (3) Uses ANN statistics to determine the normalization
  - (4) Soft reset (decrese a certain value not reset)

ANN-to-SNN conversion with Temporal Coding

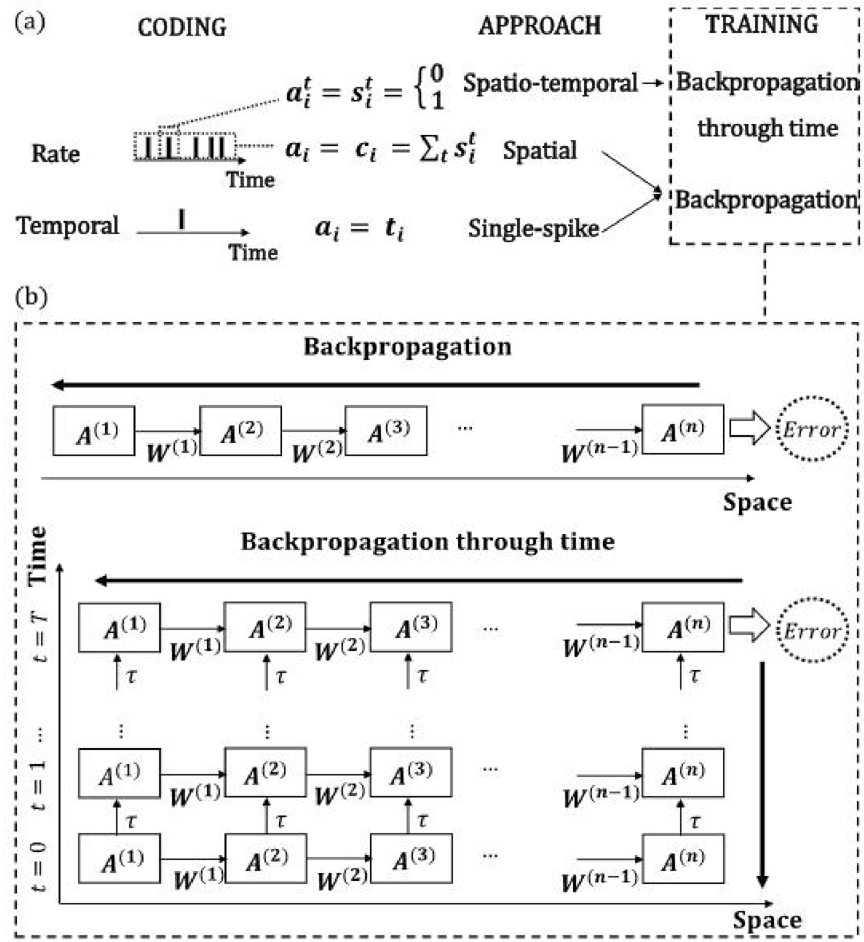
- Advantages:
  - Reduce the number of spikes compared to rate-coding
    - Reduce energy consumption
- Methods:
  - (1) Using equivalence for the ANN activations and the spike in SNN
    - Same accuracy with rate-coding
  - (2) Temoporal coding with two types of spike (positive and negative); (3) Threshold balancing; (4) Tranform to Log dimension
    - Increase the accuray with fewer timesteps
    - Increase the complexity of hardware
- Problems:
  - Less robust to noise in hardware implementation

Pros & Cons of conversion

- Rate-coding:
  - Pros:
    - Simple, straight-forward to implement
    - Noise-resilience
  - Cons:
    - The conversion process results in errors in some cases
    - Large numer of spike; Hundreds to thousands of inference timesteps
- Temporal-coding:
  - Pros:
    - Reduce the timesteps compared rate-coding => reduce power consumption
  - Cons:
    - Less robust to noise in hardware implementation, lower accuracy compared to rate-coding
    - Hundreds to thousands of inference timesteps

Backpropagation-Based Learning

- The spatial approach
- The spatiotemporal approach
- The single-spike approach



Backpropagation-based learning methods

The spatial approach

- Approximating the SNN forward pass during the training
  - A lighter backpropagation, only in the spatial domain, as in ANN training
- The SNN is directly trained, but viewed as an ANN
- Problems:
  - surplus membrane potential of spiking neurons is not considered
  - spike discretization error can be reduced to zero
    - Affect training
  - not benefit from the spatiotemporal dynamics of SNNs

The spatiotemporal approach

- Propagate the gradient both in spatial and temporal dimensions using the BPTT (backpropagation through time)
- approximate the nondifferentiable spiking activity with a surrogate gradient
  - smoothing the spiking activity
  - degrades the accuracy
- The derivative of the error is decomposed into two factors
  - the interneuron dependencies
  - the intraneuron dependencies
- Problems:
  - Backpropagation when no spike

The single-spike approach

- Avoid the nondifferentiability problem from spatiotemporal approach (at least 1 fire per neuron)
  - Directly differentiating the spike times (temporal coding)
    - Not use BPTT, direct backpropagation on spike times
    - Cannot use BP in non-fire neuron => Use weight regularization => force fire spike at least once
    - Negative spike times to encourage neurons to fire => analyzie input-output relationship
  - Use IF neurons with linear synapses
    - Reduce the exploding gradient and dead neurons compared to alpha neurons
    - Derivation of the linear synapse =/= 0 and no leak model
      => More likely to fire
    - Approximate with instantaneous synapse => Gradient not exact
- Problems:
  - Only backpropagation on presynaptic neuron that had spikes
    - Fewer spike => Slow backpropagation => Slow convergence in loss minimization

Pros & Cons of Backpropagation-based

- Pros:
  - Better accuracy & latency with spatial and single-spike approaches
  - Spatiotemporal approaches are compatible with dynamic input data
  - BPTT reduce number of timestep => reduce latency & energy
- Cons:
  - Cost of BPTT is huge
    - Require storing all actiavtions & computing gradients all timesteps
  - Use approximate derivative for spiking activities
    - Not accurate => accumulate through layers => low accuracy
  - Vanishing & exploding gradients

Overview

1. Key Contribution
2. Training Deep SNNs
3. Optimizing Deep SNNs
4. Impact of encoding, training, architecture on accuracy-latency trade-off
5. Conclusion

Optimizing Deep SNNs

- Improvement for backpropagration approaches
  - Improve accuracy, latency, spike sparsity
- Use similar techniques for backpropagation in ANNs
  - Transferring ANN techniques (Batch Norm., Drop Out, Regularization, ...)
  - Improve encoding & decoding
  - Use wider network architecture
  - Hybridization in training (mix ANN & SNN)
  - Tuning parameters specificially to SNNs

ANN techniques for SNN

- Regularization: (Decrease overfitting in training => improve the generalization of model)
  - Neuron Normalization Method
    - using auxiliary neruons at each convolutional layer
    - weight summation of spike count (from pre-neuron to post-neuron) => balance input current => increase accuracy
    - Require additional multiplications => increase complexity
- Dropout: (regularization - avoid training focus on one neuron)
  - Use in BPTT - keep random subset dropped in each timestep
    - Better convergence
- Batch Normalization: (reduce the training timesteps, increase scalability)
  - 9x efficiency compared non-BN model
  - Threshold-dependent spatiotemporal BN => Norm. the variance of the inputs => high accuracy, fewer timesteps
- Optimize network architectures
  - ResNet, VGG => increase accuracy

Improving encoding & decoding

- Encoding:
  - Poisson spike generation
    - Famous in spatial & spatiotemporal approaches
  - Discrete cosine transformation (make use of temporal dimension)
    - Decomposing the input into a basis of spectral component
    - Fine grain spike vectors => increase accuracy, reduce latency
    - But 2 matrix multiplications => increase complexity
  - Add encoding layer (real-value not binary) - hybrid layers
    - As first layer
    - 1 spike layer, 1 encoding layer

    => reduce timesteps, increase complexity
- Decoding:
  - Apply loss function, high-precision activations on output neurons
    - Increase accuracy
    - Increase parameters, more neurons => increase complexity

Wide Network Architecture

- Increase number of neurons per layer
  - Improve accuracy, low latency
  - ResNet, VGG
  - Increase in width more benefit in depth
    - Width - Increase quality for backpropagation
    - Depth - Difficult to backpropagation

Training Hybridization

- ANN-SNN Network Hybridization
  - Improve encoding accuracy
  - Good benefit from small dataset (CIFAR-10)
  - Low benefit from big dataset (ImageNet)
- Tandem Learning
  - Training process is reduced
  - Temporal information cannot used
- Conversion and direct training Hybridization
  - Good accuracy
  - High number of inference timestep

Leveraging the specificity of SNN

- Neuron's leak and threshold
  - Control input-output => good for backpropagation
  - Reduce timesteps
- Synapse Dynamics
  - Add filter to increase the accuracy but increase complexity
- Surrogate gradient
  - Approximate the drivative of spike
    - Training is fast, but not good in big scale

Results (1)

COMPARISON OF BACKPROPAGATION-BASED ALGORITHMS ON STATIC VISION DATASETS

Learning strategy	Paper	Coding	Neurons + synapses	Architecture	Regularization method	Additional training strategy	Timesteps	Acc. (%)
CIFAR-10								
Spatial	[45]	rate	IF + instantaneous	VGG-8	/	/	/	89.99
Spatio-temporal	[51]	rate	LIF + instantaneous	VGG-8	neuron normalization, dropout	encoding layer, voting layer	12	90.53
	[53]	rate	LIF + instantaneous	ResNet-11	dropout	/	100	90.95
	[54]	rate	LIF + exponential	VGG-8	/	encoding layer	5	91.41
	[58]	rate	IF + instantaneous	ResNet-11	batch normalization, dropout	surrogate gradient tuning	20	90.20
	[56]	rate	LIF + instantaneous	VGG-8	batch normalization, dropout	encoding layer, voting layer	8	93.50
	[67]	rate	LIF + instantaneous	VGG-9	batch normalization	/	25	90.50
	[57]	rate	LIF + instantaneous	ResNet-19	batch normalization	encoding layer, voting layer	6	93.16
Single-spike	[64]	time	IF + exponential	VGG-16	weight regularization	/	/	92.68
CIFAR-100								
Spatio-temporal	[58]	rate	IF + instantaneous	ResNet-50	batch normalization, dropout	surrogate gradient tuning	40	58.5
	[67]	rate	LIF + instantaneous	VGG-11	batch normalization	/	30	65.8
ImageNet								
Spatio-temporal	[57]	rate	LIF + instantaneous	ResNet-34	batch normalization	encoding layer, voting layer	6	67.05
	[68]	rate	LIF + instantaneous	ResNet-152	batch normalization	encoding layer	4	69.26
Single-spike	[64]	time	IF + exponential	GoogLeNet	weight regularization	/	/	68.8

Increase accuracy, reduce latency with optimization techniques

Results (2)

COMPARISON OF BACKPROPAGATION-BASED ALGORITHMS ON NEUROMORPHIC VISION DATASETS

Learning strategy	Paper	Coding	Neurons + synapses	Architecture	Regularization method	Additional training strategy	Timesteps	Acc. (%)
CIFAR-10-DVS								
Spatio-temporal	[51]	rate	LIF + instantaneous	VGG-5	neuron normalization, dropout	encoding layer, voting layer	20	60.5
	[56]	rate	LIF + instantaneous	VGG-6	batch normalization, dropout	encoding layer, voting layer	20	74.8
	[67]	rate	LIF + instantaneous	VGG-7	batch normalization	/	20	63.2
	[57]	rate	LIF + instantaneous	ResNet-19	batch normalization	encoding layer, voting layer	10	67.8
DVSGesture								
Spatio-temporal	[55]	rate	LIF + continuous	5-layer CNN	/	synapse kernel optimization	/	96.09
	[56]	rate	LIF + instantaneous	VGG-7	batch normalization, dropout	encoding layer, voting layer	20	97.57
	[57]	rate	LIF + instantaneous	ResNet-17	batch normalization	encoding layer, voting layer	40	96.87

Increase accuracy, reduce latency with optimization techniques

Overview

1. Key Contribution
2. Training Deep SNNs
3. Optimizing Deep SNNs
4. Impact of encoding, training, architecture on accuracy-latency trade-off
5. Conclusion

Impact (1) - CIFAR 10

Paper	Architecture	Encoding layer	Training	Timesteps	Acc. (%)
CIFAR-10					
[38]	ResNet-20	×	conversion (rate)	2048	91.36
[38]	ResNet-20	×	conversion (rate)	256	89.37
[39]	ResNet-20	×	conversion (time)	2048	91.42
[39]	ResNet-20	×	conversion (time)	256	90.10
[40]	ResNet-20	✓	conversion (rate)	16	91.62
[66]	ResNet-20 (L)	×	conversion (rate)	250	91.12
[66]	ResNet-20 (L)	×	conversion (rate) + backpropagation	250	92.22
[70]	ResNet-20 (L)	✓	conversion (rate) + backpropagation	5	90.29
[70]	ResNet-20 (L)	✓	conversion (rate) + backpropagation (+ leak & threshold tuning)	5	91.78
[57]	ResNet-19 (L)	✓	backpropagation	6	93.16
[53]	ResNet-11 (L)	×	backpropagation	100	90.95
[58]	ResNet-11 (L)	×	backpropagation (+ batch normalization + surrogate gradient tuning)	20	90.20
[72]	ResNet-20	/	ANN (+ batch normalization)	/	91.25
[70]	ResNet-20 (L)	/	ANN	/	92.79

Number of parameters of the architectures (estimated according to the details given in the associated papers): ResNet-20: 0.27M. ResNet-20 (L): 11M. ResNet-19 (L): 13M. ResNet-11 (L): 18M. ResNet-34: 21M. ResNet-34 (M): 22M. ResNet-34 (L): 85M. VGG-16: 138M.

Impact (2) - ImageNet

ImageNet					
[38]	ResNet-34	×	conversion (rate)	4096	69.89
[38]	ResNet-34	×	conversion (rate)	256	≈20
[39]	ResNet-34	×	conversion (time)	4096	69.93
[39]	ResNet-34	×	conversion (time)	256	55.65
[40]	ResNet-34	✓	conversion (rate)	64	72.35
[66]	ResNet-34 (M)	×	conversion (rate)	250	56.87
[66]	ResNet-34 (M)	×	conversion (rate) + backpropagation	250	61.48
[57]	ResNet-34	✓	backpropagation (+ batch normalization)	6	63.72
[57]	ResNet-34 (L)	✓	backpropagation (+ batch normalization)	6	67.05
[68]	ResNet-34	✓	backpropagation (+ batch normalization)	4	67.04
[73]	ResNet-34	/	ANN (+ batch normalization)	/	73.31
[38]	VGG-16	×	conversion (rate)	4096	73.09
[38]	VGG-16	×	conversion (rate)	256	48.32
[39]	VGG-16	×	conversion (time)	2560	73.46
[39]	VGG-16	×	conversion (time)	256	69.71
[40]	VGG-16	✓	conversion (rate)	64	72.85
[66]	VGG-16	×	conversion (rate)	250	62.73
[66]	VGG-16	×	conversion (rate) + backpropagation	250	65.19
[70]	VGG-16	✓	conversion (rate) + backpropagation	5	64.32
[70]	VGG-16	✓	conversion (rate) + backpropagation (+ leak & threshold tuning)	5	69.00
[73]	VGG-16	/	ANN (+ batch normalization)	/	73.36

Number of parameters of the architectures (estimated according to the details given in the associated papers): ResNet-20: 0.27M. ResNet-20 (L): 11M. ResNet-19 (L): 13M. ResNet-11 (L): 18M. ResNet-34: 21M. ResNet-34 (M): 22M. ResNet-34 (L): 85M. VGG-16: 138M.

Overview

1. Key Contribution
2. Training Deep SNNs
3. Optimizing Deep SNNs
4. Impact of encoding, training, architecture on accuracy-latency trade-off
5. Conclusion

Conclusion

- Based the expectation to choose the suitable techniques
  - High Accuracy, low latency
  - Low complexity, low-power, low area cost

Thank you

for your attention.