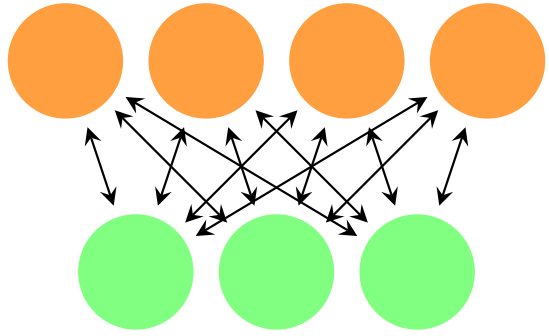
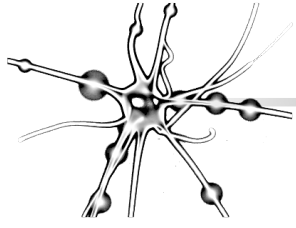
A REVIEW OF

REAL-TIME CLASSIFICATION AND SENSOR FUSION WITH A SPIKING DEEP BELIEF NETWORK

by P. O'CONNOR ET AL. (2013)



Pedro Mediano



Computational Neurodynamics Group

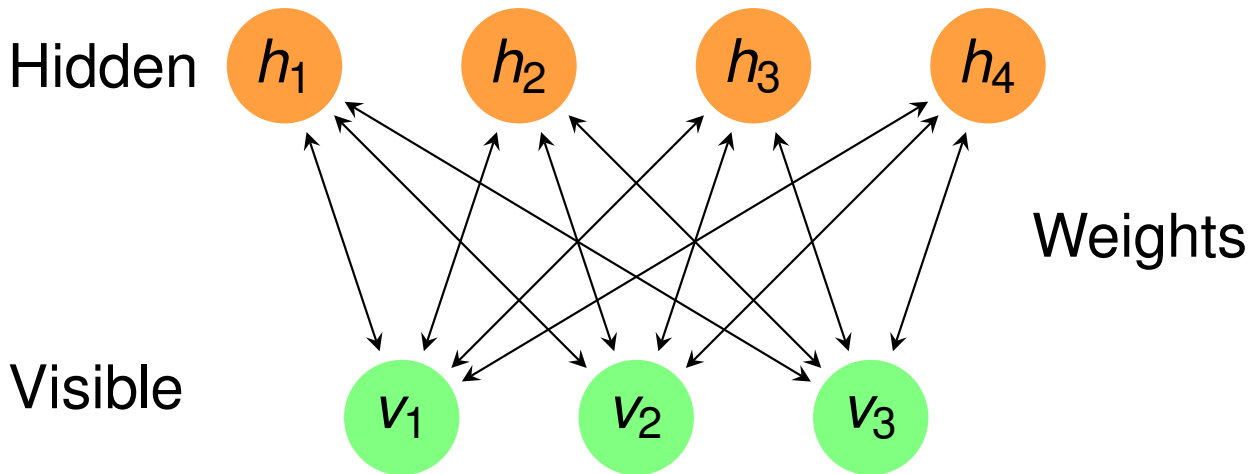
Department of Computing

pmediano@ic.ac.uk Huxley 444

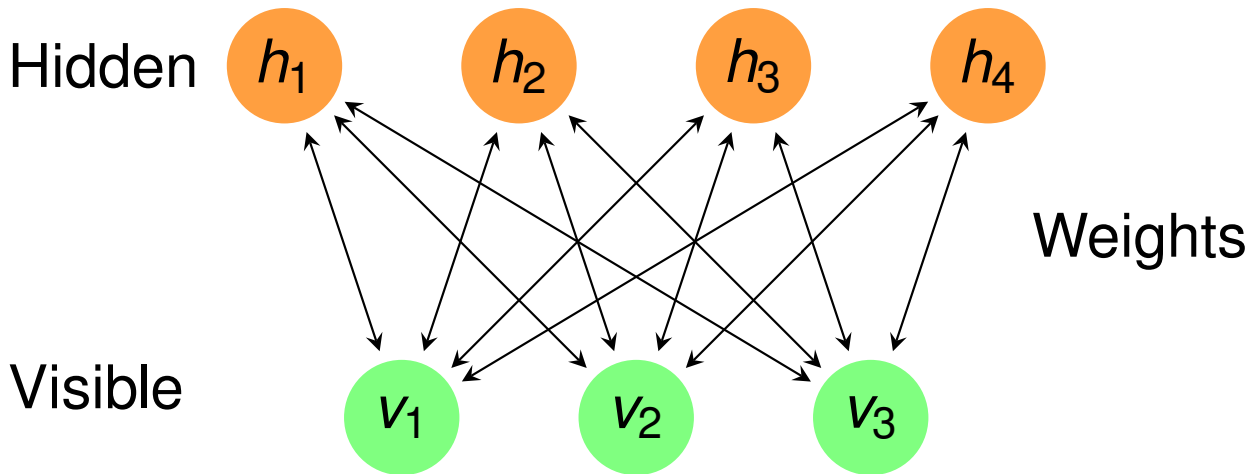
Table of contents

Background

DEEP BELIEF NETWORKS



DEEP BELIEF NETWORKS



- Each neuron is a binary variable.
- Neural activations are sigmoidal:

$$p(v_i = 1|\mathbf{h}, \theta) = \sigma\left(\sum_j w_{ij} h_j + b_i\right)$$

TRAINING A DBN

A DBN represents a probability distribution through an **energy function**,

$$p(\mathbf{v}, \mathbf{h}|\theta) \propto \exp\left(-E(\mathbf{v}, \mathbf{h}|\theta)\right)$$

$$E(\mathbf{v}, \mathbf{h}|\theta) = -\sum_{ij} w_{ij} v_i h_j - \sum_i b_i^{(v)} v_i - \sum_j b_j^{(h)} h_j$$

TRAINING A DBN

A DBN represents a probability distribution through an **energy function**,

$$p(\mathbf{v}, \mathbf{h}|\theta) \propto \exp\left(-E(\mathbf{v}, \mathbf{h}|\theta)\right)$$

$$E(\mathbf{v}, \mathbf{h}|\theta) = -\sum_{ij} w_{ij} v_i h_j - \sum_i b_i^{(v)} v_i - \sum_j b_j^{(h)} h_j$$

Maximize marginal $p(\mathbf{v}|\theta)$ with gradients

$$\Delta w_{ij} \propto \left\langle \frac{dE}{d\theta} \right\rangle_{data} - \left\langle \frac{dE}{d\theta} \right\rangle_{model}$$

TRAINING A DBN

A DBN represents a probability distribution through an **energy function**,

$$p(\mathbf{v}, \mathbf{h}|\theta) \propto \exp\left(-E(\mathbf{v}, \mathbf{h}|\theta)\right)$$

$$E(\mathbf{v}, \mathbf{h}|\theta) = -\sum_{ij} w_{ij} v_i h_j - \sum_i b_i^{(v)} v_i - \sum_j b_j^{(h)} h_j$$

Maximize marginal $p(\mathbf{v}|\theta)$ with gradients

$$\Delta w_{ij} \propto \left\langle \frac{dE}{d\theta} \right\rangle_{data} - \left\langle \frac{dE}{d\theta} \right\rangle_{model} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

TRAINING A DBN

We compute gradients using **contrastive divergence**:

1. Clamp data to visible units.

(Hinton 2006)

TRAINING A DBN

We compute gradients using **contrastive divergence**:

1. Clamp data to visible units.
2. Sample hidden units.

(Hinton 2006)

TRAINING A DBN

We compute gradients using **contrastive divergence**:

1. Clamp data to visible units.
2. Sample hidden units.
3. Perform *k* visible-hidden sampling iterations.

(Hinton 2006)

TRAINING A DBN

We compute gradients using **contrastive divergence**:

1. Clamp data to visible units.
2. Sample hidden units.
3. Perform k visible-hidden sampling iterations.
4. Estimate weight gradient.

(Hinton 2006)

TRAINING A DBN

We compute gradients using **contrastive divergence**:

1. Clamp data to visible units.
2. Sample hidden units.
3. Perform *k* visible-hidden sampling iterations.
4. Estimate weight gradient.
5. Use your favourite gradient-based optimizer.

(Hinton 2006)

SPIKING NEURONS

- They use Leaky Integrate-and-Fire (LIF) neurons,

$$\frac{dv}{dt} = \tau^{-1}(v_{rest} - v) + I$$

SPIKING NEURONS

- They use Leaky Integrate-and-Fire (LIF) neurons,

$$\frac{dv}{dt} = \tau^{-1}(v_{rest} - v) + I$$

- Differences wrt binary-sigmoid neurons:

SPIKING NEURONS

- They use Leaky Integrate-and-Fire (LIF) neurons,

$$\frac{dv}{dt} = \tau^{-1}(v_{rest} - v) + I$$

- Differences wrt binary-sigmoid neurons:
  - Real-valued,

SPIKING NEURONS

- They use Leaky Integrate-and-Fire (LIF) neurons,

$$\frac{dv}{dt} = \tau^{-1}(v_{rest} - v) + I$$

- Differences wrt binary-sigmoid neurons:
  - Real-valued,
  - Asynchronous (or *event-based*),

SPIKING NEURONS

- They use Leaky Integrate-and-Fire (LIF) neurons,

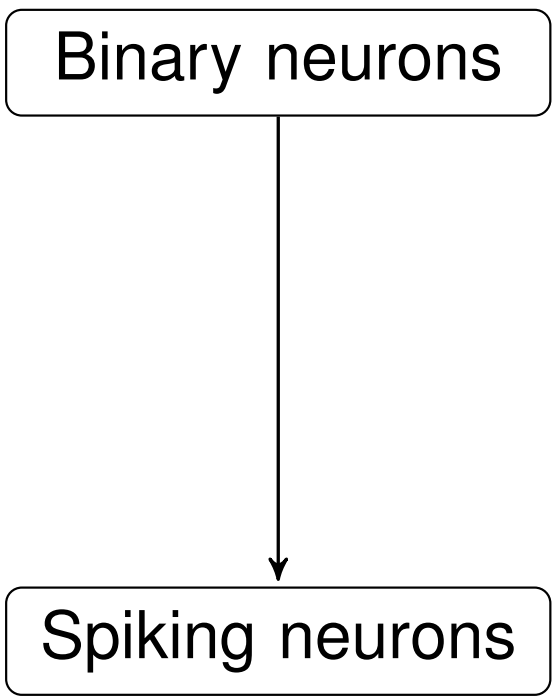
$$\frac{dv}{dt} = \tau^{-1}(v_{rest} - v) + I$$

- Differences wrt binary-sigmoid neurons:
  - Real-valued,
  - Asynchronous (or *event-based*),
  - Non-differentiable.

CONTRIBUTIONS

GOING SPIKY

OR HOW TO TRANSFORM YOUR DBN INTO A SNN



GOING SPIKY

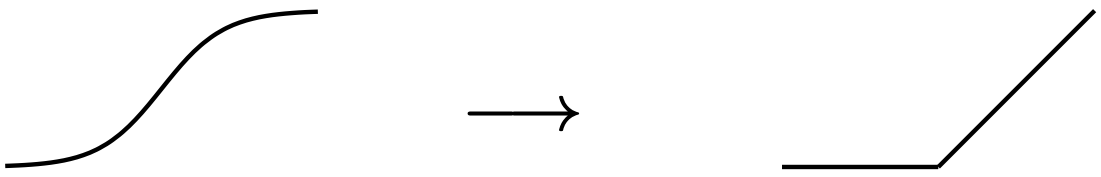
OR HOW TO TRANSFORM YOUR DBN INTO A SNN

Binary neurons

↓

NReLU

Change the activation function of your neurons.



Spiking neurons

(Nair and Hinton 2010)

GOING SPIKY

OR HOW TO TRANSFORM YOUR DBN INTO A SNN

Binary neurons

↓

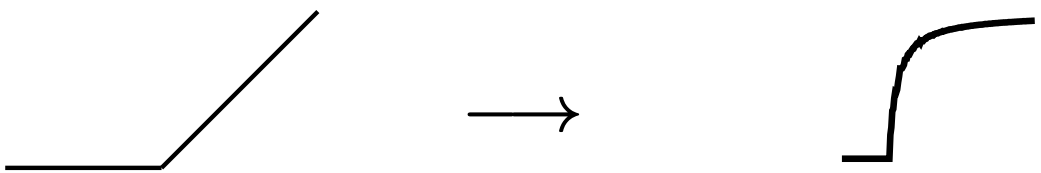
NReLU

↓

Siegert neurons

Spiking neurons

Change again to a more realistic activation function.

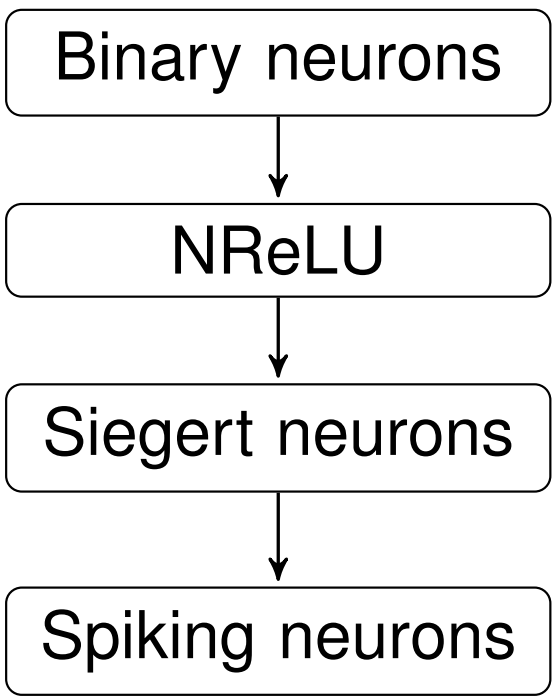


Note that we can still use CD.

(Siegert 1951)

Going spiky

Or how to transform your DBN into a SNN



Siegert neurons map directly to LIF.

Note that Siegerts are **long-time** approximation of a **single** LIF neuron.

FULL ALGORITHM

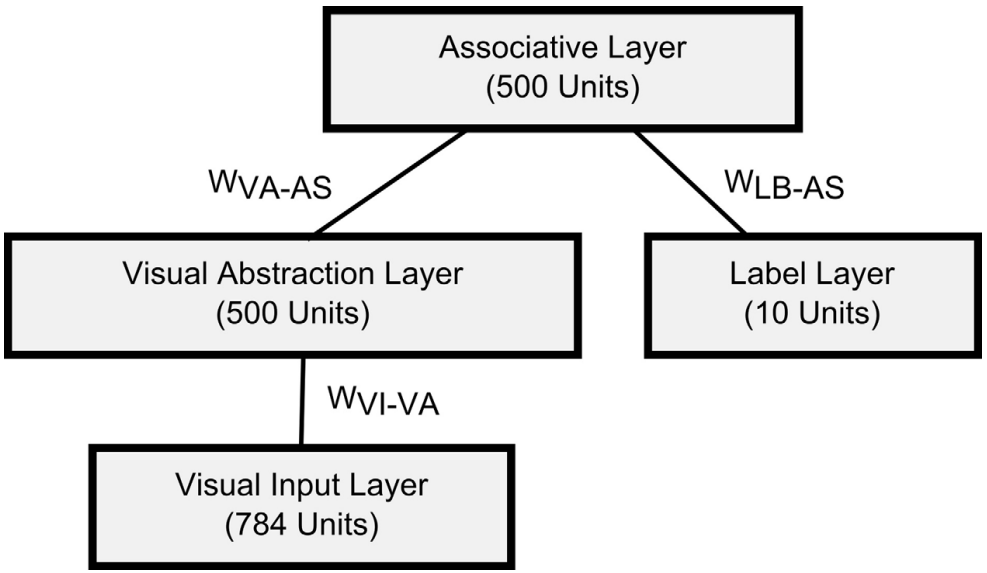
1. Set up Siegert DBN.
2. Train it with CD.
3. Replace neurons with LIF.
4. Replace pixels with Poisson sources.

FULL ALGORITHM

1. Set up Siegert DBN.
2. Train it with CD.
3. Replace neurons with LIF.
4. Replace pixels with Poisson sources.

- Note:
  They actually use a small modification called Persistent CD, plus fast weights, momentum and sparsification.

NETWORK ARCHITECTURE



- 2k neurons smaller than first MNIST DBN.
- 5k neurons smaller than MNIST state-of-the-art.

(Hinton 2006, Ciresan 2011)

RESULTS

MNIST, of course.

Network type	**Accuracy**
Binary-sigmoid	97.48
Siegert	95.20
LIF	94.09

Experimental results in video (using event camera):

`https://sites.google.com/site/thebrainbells/`

SENSOR FUSION

- Play a sound for each number.

SENSOR FUSION

- Play a sound for each number.
- Get data with spiking cochlea.

SENSOR FUSION

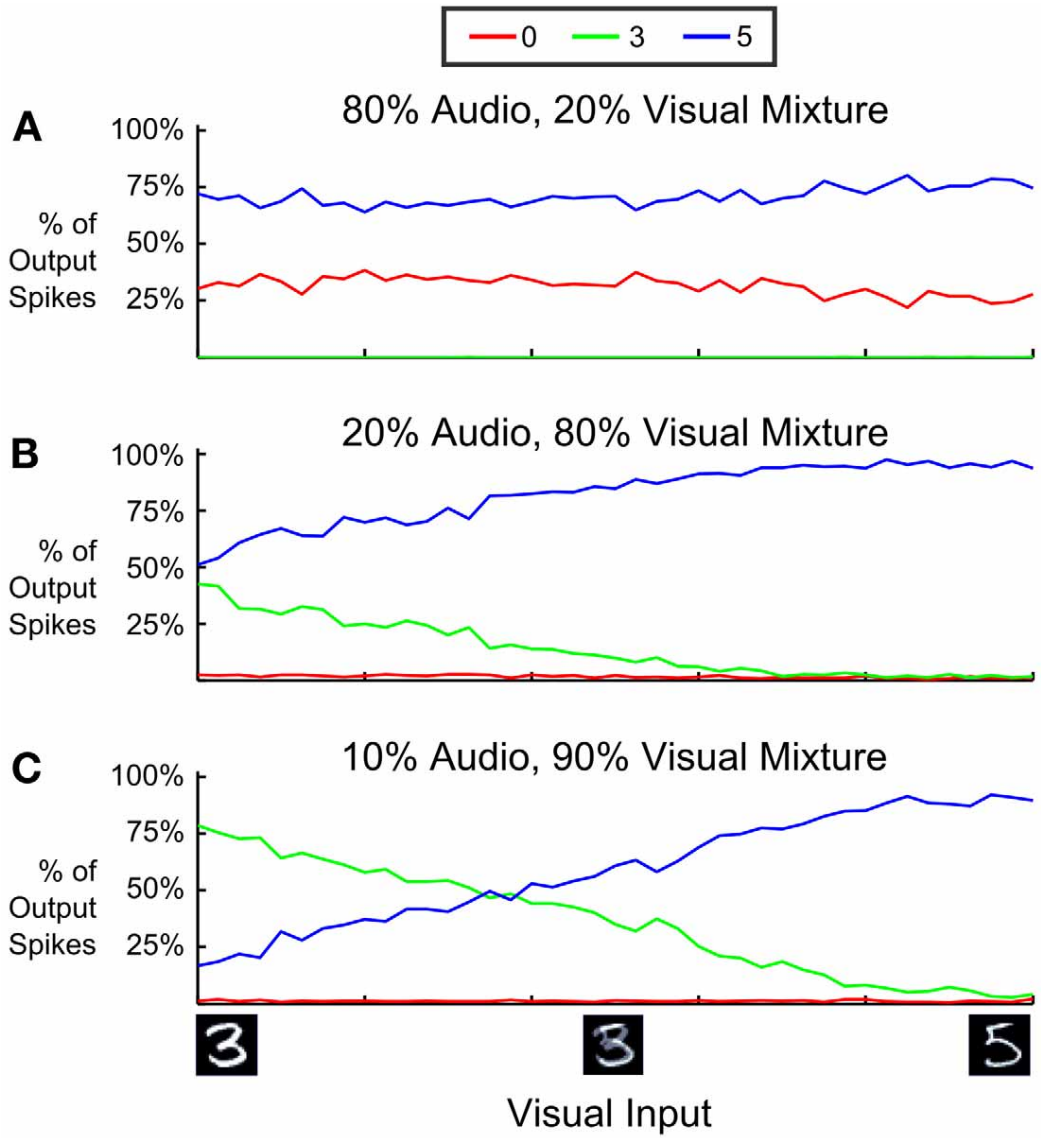
- Play a sound for each number.
- Get data with spiking cochlea.
- Add an auditory layer to DBN.

SENSOR FUSION

- Play a sound for each number.
- Get data with spiking cochlea.
- Add an auditory layer to DBN.
- Allows study of ambiguous stimuli.

SENSOR FUSION

- Play a sound for each number.
- Get data with spiking cochlea.
- Add an auditory layer to DBN.
- Allows study of ambiguous stimuli.



Discussion

GOOD STUFF

GOOD STUFF

- It works with event camera despite being trained on Poisson sources.

GOOD STUFF

- It works with event camera despite being trained on Poisson sources.
- Even through Siegert is a long-time approximation, classification takes 5.8 ms.

Good stuff

- It works with event camera despite being trained on Poisson sources.
- Even through Siegert is a long-time approximation, classification takes 5.8 ms.
- Effective proof of concept – SNNs *can* work.

GOOD STUFF

- It works with event camera despite being trained on Poisson sources.
- Even through Siegert is a long-time approximation, classification takes 5.8 ms.
- Effective proof of concept – SNNs *can* work.
- Opens door to more accesible hardware implementations.

GOOD STUFF

- It works with event camera despite being trained on Poisson sources.
- Even through Siegert is a long-time approximation, classification takes 5.8 ms.
- Effective proof of concept – SNNs *can* work.
- Opens door to more accesible hardware implementations.
- Insights in neuroscience?

BAD STUFF

BAD STUFF

- Not very well written.

BAD STUFF

- Not very well written.
  - Many buzzwords.

BAD STUFF

- Not very well written.
  - Many buzzwords.
  - Misplaced citations.

BAD STUFF

- Not very well written.
  - Many buzzwords.
  - Misplaced citations.
- Spiky hacks (e.g. spike normalization).

BAD STUFF

- Not very well written.
  - Many buzzwords.
  - Misplaced citations.
- Spiky hacks (e.g. spike normalization).
- Misleading information (NReLU, sensor fusion, STD).

BAD STUFF

- Not very well written.
  - ◇ Many buzzwords.
  - ◇ Misplaced citations.
- Spiky hacks (e.g. spike normalization).
- Misleading information (NReLU, sensor fusion, STD).
- They claim only 1% accuracy loss.
  $\rightarrow$ It's actually closer to 3.5%.

Follow-up work

FOLLOW-UP WORK

Since 2013, the authors have:

FOLLOW-UP WORK

Since 2013, the authors have:

- ✓ Applied same approach to more DL models.

FOLLOW-UP WORK

Since 2013, the authors have:

- ✓ Applied same approach to more DL models.
- ✓ Designed new tricks for better accuracy.

Follow-up work

Since 2013, the authors have:

- ✓ Applied same approach to more DL models.
- ✓ Designed new tricks for better accuracy.
- ✓ Implemented in specialized hardware (SpiNNaker).

Follow-up work

Since 2013, the authors have:

- ✓ Applied same approach to more DL models.
- ✓ Designed new tricks for better accuracy.
- ✓ Implemented in specialized hardware (SpiNNaker).
- ✗ Succeeded at learning with spiking neurons.

FOLLOW-UP WORK

- 33 citations since 2013.
- First example in Wikipedia “Spiking neural network”.

Follow-up work

- 33 citations since 2013.
- First example in Wikipedia “Spiking neural network”.

If interested, check out new paper on CNNs:

Diehl et al. *Fast-Classifying, High-Accuracy Spiking Deep Networks Through Weight and Threshold Balancing*. IJCNN 2015 (to appear).