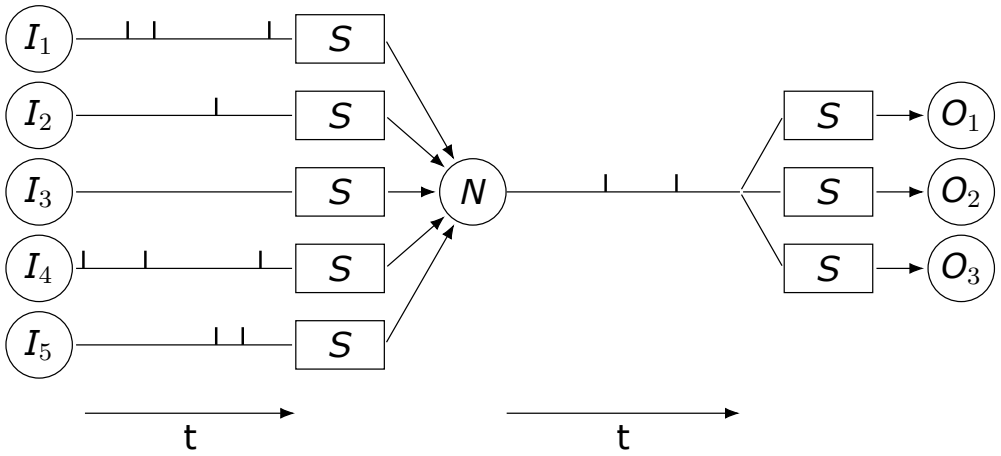
Simulateur N2S3 pour réseaux de neurones à spikes

Pierre Falez

Réseaux de neurones à impulsions



- Implémentions matérielles énergiquement efficaces (facteur x300000).
- Actuellement immatures pour résoudre des tâches complexes.
- Sujet interdisciplinaire (informatique, électronique, physique, biologie, neurosciences,...)

IRCICA

Axe ”traitements bioinspirés de l’information”

- Projet interdisciplinaire:
  - FOX: application vision
  - Émeraude: système embarqué
  - SPINE: électronique
- Besoin d’une plateforme de modélisation commune

Propriétés souhaitées

Flexible
Les SNNs sont encore immatures, et nécessitent beaucoup de prototypages.
Passage à l'échelle
La simulation à grande échelle est nécessaire pour les applications avancées.
Facile
L'outil doit pouvoir être utilisé par des électroniciens.
Compatible
Le simulateur doit pouvoir être utilisé sur la plupart des environnements.

Scala

+ Exécutable sur la JVM: Compatible.
+ Orienté objets: Facilite l'organisation du projet.
+ Fonctionnel: Code concis.
+ Permet de faire des langages dédiés (DSL): Facilite l'utilisation
– Peu optimisable (mais JIT disponible)
– Gourmand en mémoire (Garbage collector)

Parallélisation

SIMD

- Applique les mêmes instructions sur plusieurs données
- Requière une homogénéité (ex: un unqiue modèle de neurone)
- Met à jour tout les éléments à chaque fois
- Notamment optimisé sur GPU

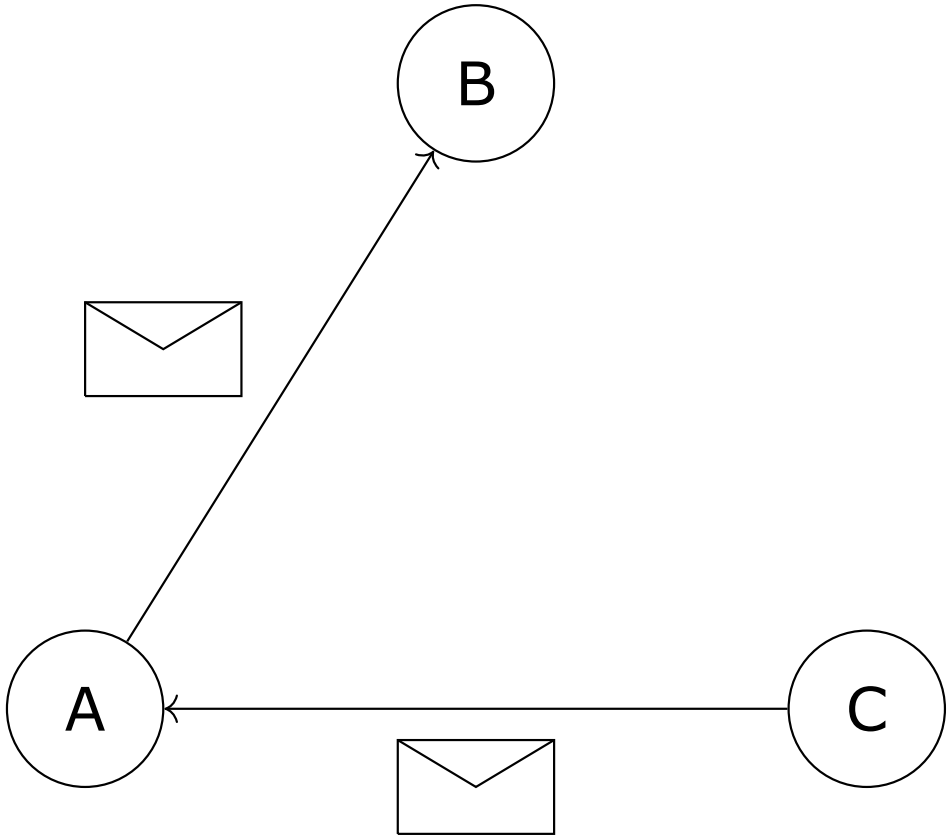
MIMD

- Instructions différentes en parallèle
- Permet une hétérogénéité (flexibilité)
- Mise à jour éparse
- CPU multi-coeurs/architectures hétérogènes
- **Requière une gestion avancée de la concurrence**

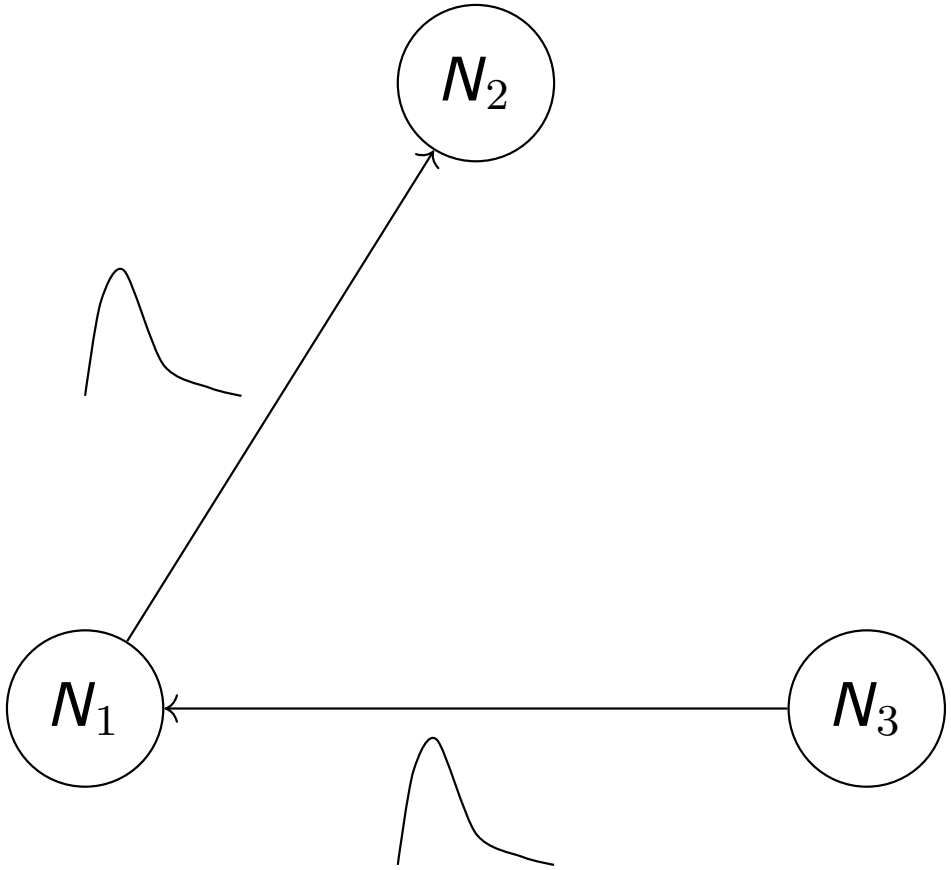
Programmation orientée acteurs

- Facilite la mise en place d’architectures MIMD.
- Chaque acteur a une mémoire indépendante.
- Les acteurs communiquent exclusivement par message.
- Retire la nécessitée de gérer la concurrence.

Programmation orientée acteurs



Programmation orientée acteurs



Similarités avec les SNNs!

Akka

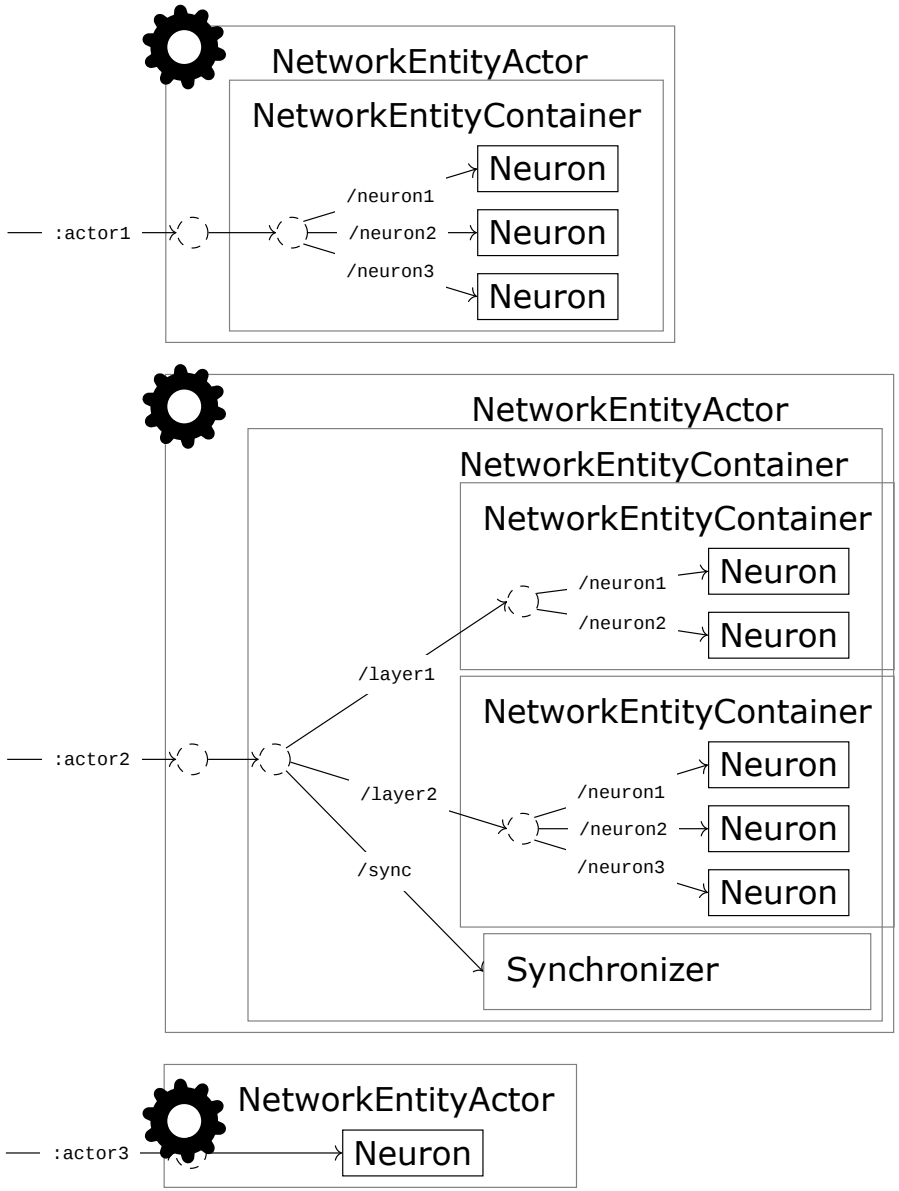
- Bibliothèque orientée acteurs
- Interface Scala
- Facilement distribuable (Couche d’abstraction)

N2S3

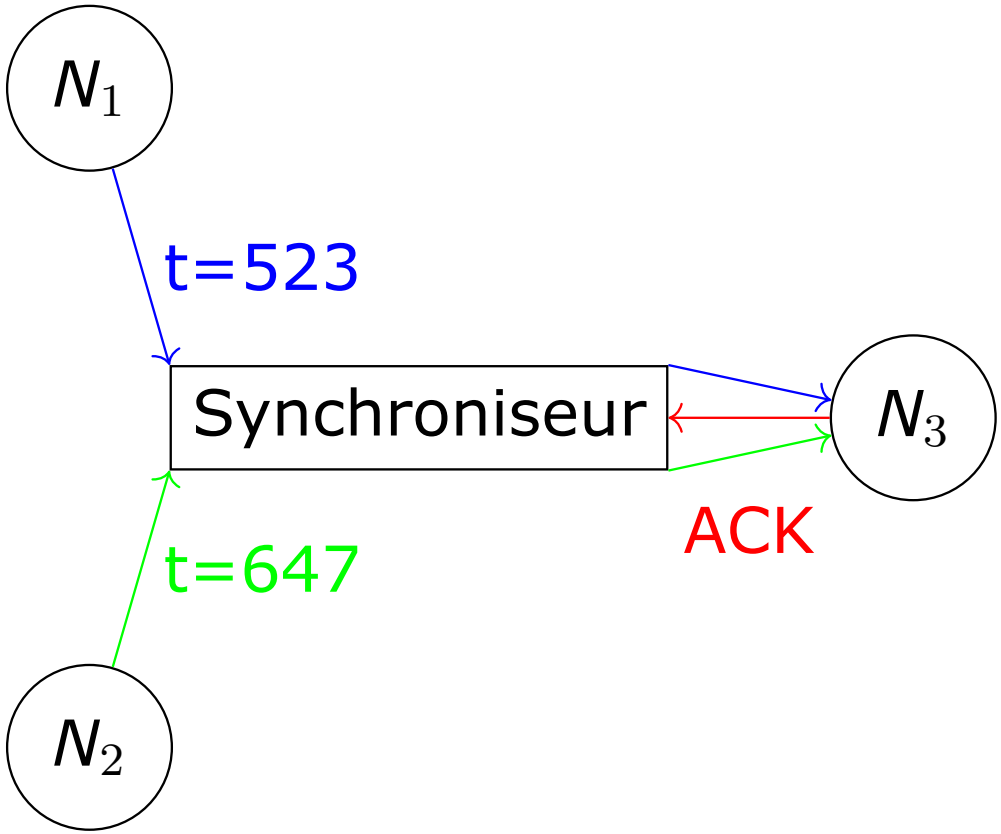
Neural **N**etwork **S**calable **S**piking **S**imulator

- Scala
- Akka
- Sbt (gestionnaire de projet)

N2S3


NetworkEntityActor
NetworkEntityContainer
Neuron
/neuron1
/neuron2
/neuron3
Neuron
Neuron
:actor1
NetworkEntityActor
NetworkEntityContainer
NetworkEntityContainer
/neuron1
/neuron2
Neuron
Neuron
/layer1
NetworkEntityContainer
:actor2
/layer2
/sync
Neuron
/neuron1
/neuron2
/neuron3
Neuron
Neuron
Synchronizer
NetworkEntityActor
:actor3
Neuron

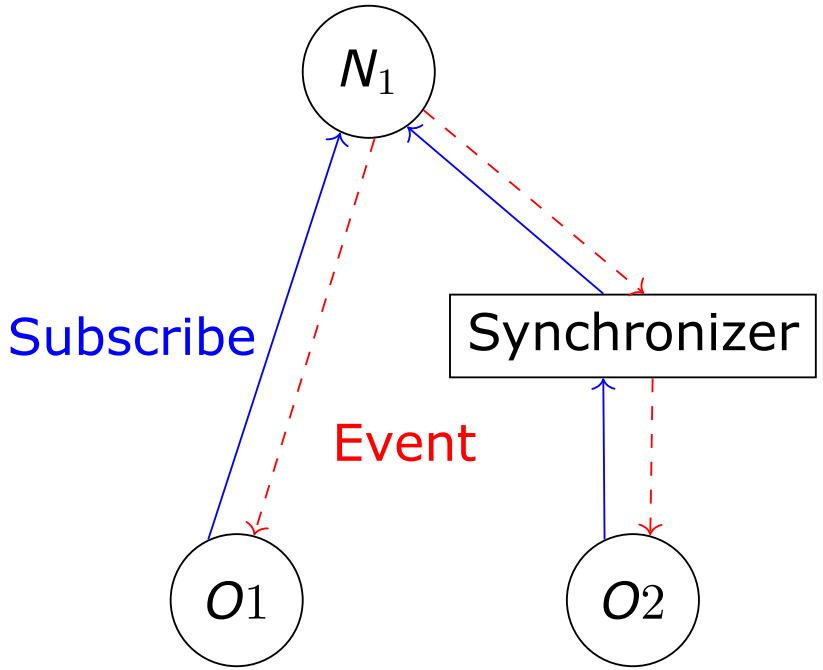
Utilisation d'un synchroniseur



N2S3

Système d'observateurs et d'évènements

- Récupération des propriétés des entités
- Évènements synchrones ou asynchrones
- Utilisé par les loggers, les visualisations et certains mécanismes utilisés par les SNNs.



N2S3

Simulateur orienté événements (Event-driven)

- Mise à jour éparse (seulement aux évènements)
- Grande précision dans la gestion du temps (pas de Δ_t)
- Forme analytique des modèles

Exemple avec le modèle LIF:

$$\tau_{\mathrm{m}} \frac{\partial V}{\partial t} = I(t) - V(t) \tag{1}$$

$$V_{t+\delta} = I_{\mathrm{s}} + V^{\frac{\delta}{\tau_{\mathrm{m}}}} \tag{2}$$

Entrée

Système de pipelines

- Générateurs/lecteurs
- Modificateurs (changement d’échelle)
- Injecteurs de bruits
- Convertisseurs (valeurs vers impulsions)

Exemples:

```
InputAER.Entry >> InputAER.Retina(128, 128) >> NoiseGenerator(0.8) >> N2S3Entry

InputMnist.Entry >> SampleToSpikeTrainConverter[Float, InputSample2D[Float]]
(150 MilliSecond, 350 MilliSecond) >> N2S3Entry
```

N2S3

Disponible de base dans le simulateur:

- **Entrées:** Generateurs (ex: Poisson), AER, Mnist, cifar.
- **Neurones:** IF, LIF, SRM, Izhichevich.
- **Synapses:** Statique, STDP biologique, STDP multiplicative.
- **Visualisations:** Évolution des potentiels, poids synaptiques.

Langage Dédié

```
implicit val network: N2S3SimulationDSL = N2S3SimulationDSL()
defaultQGBParameters

network hasInput InputMnist.Entry >>
    SampleToSpikeTrainConverter[Float, InputSample2D[Float]]
    (0, 23, 150 MilliSecond, 350 MilliSecond) >>
    N2S3Entry
network hasInputNeuronGroup "input"
network hasNeuronGroup "group_1" ofSize 30 ofModel LIF
"group_1" hasParameters (MembranePotentialThreshold -> 35.millivolts)

"group_1" connectsTo "group_1" using FullConnection withSynapse InhibitorySynapse
"input" connectsTo "group_1" using FullConnection withSynapse SimplifiedSTDP

network buildit

observeConnectionBetween("input", "group_1")

val dataFile = N2S3ResourceManager.getByName("mnist-train-images").getAbsolutePath
val labelFile =  N2S3ResourceManager.getByName("mnist-train-labels").getAbsolutePath


network trainOn MnistFileInputStream(dataFile, labelFile)

val dataTestFile = N2S3ResourceManager.getByName("mnist-test-images").getAbsolutePath
val labelTestFile =  N2S3ResourceManager.getByName("mnist-test-labels").getAbsolutePath


network testOn MnistFileInputStream( dataTestFile, labelTestFile)

network destroyit
```

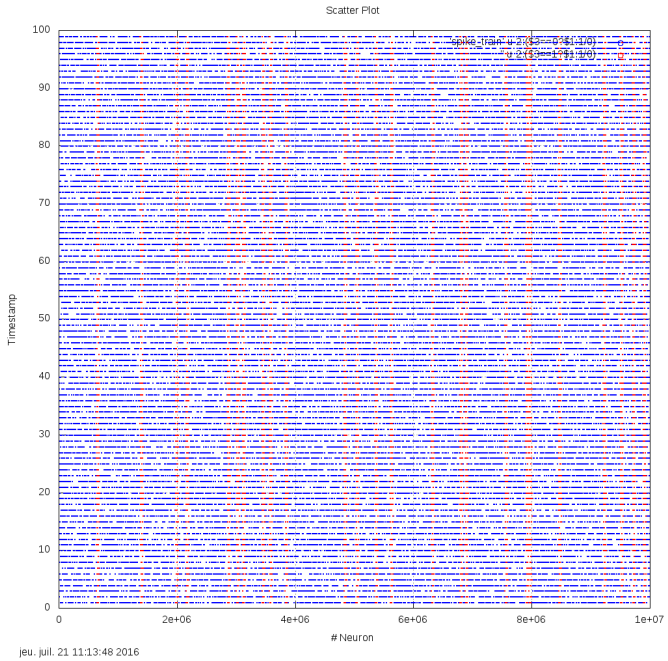
Performance

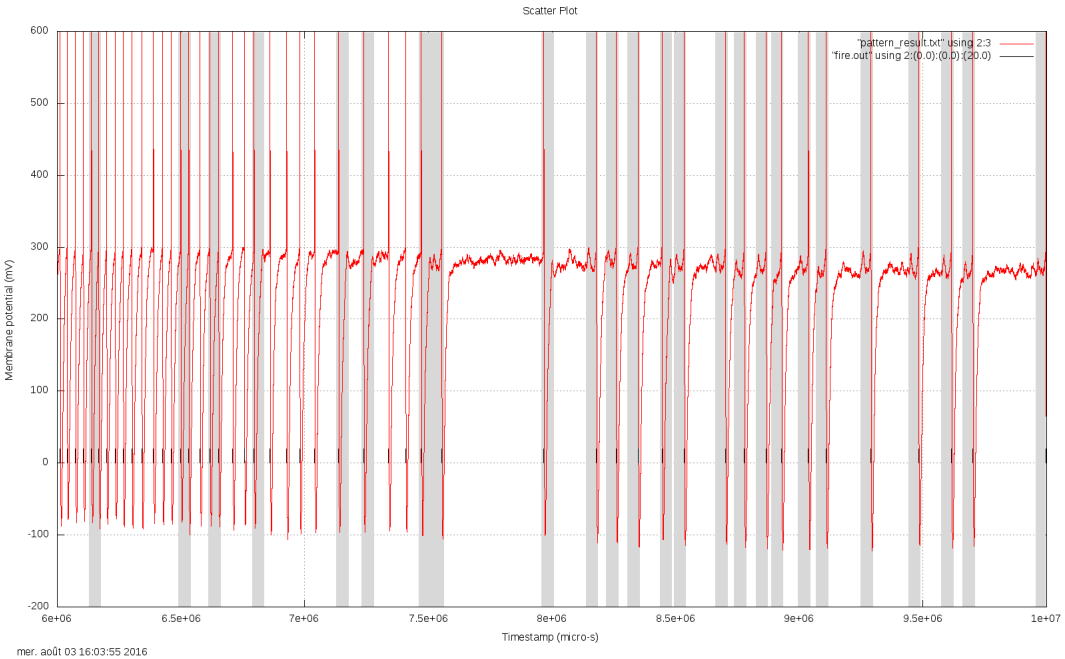
Pour PyNEST et Brian, $\Delta_t = 1ms$

Experiment	Measure	PyNEST	Brian	N2S3
MNIST, 100N	CPU time	15:05:16	9:39:15	3:42:03
	Memory	85 MB	2822 MB	1331 MB
AER, 60N	CPU time		10:03:40	3:34:41
	Memory		914 MB	1448 MB

Exemples

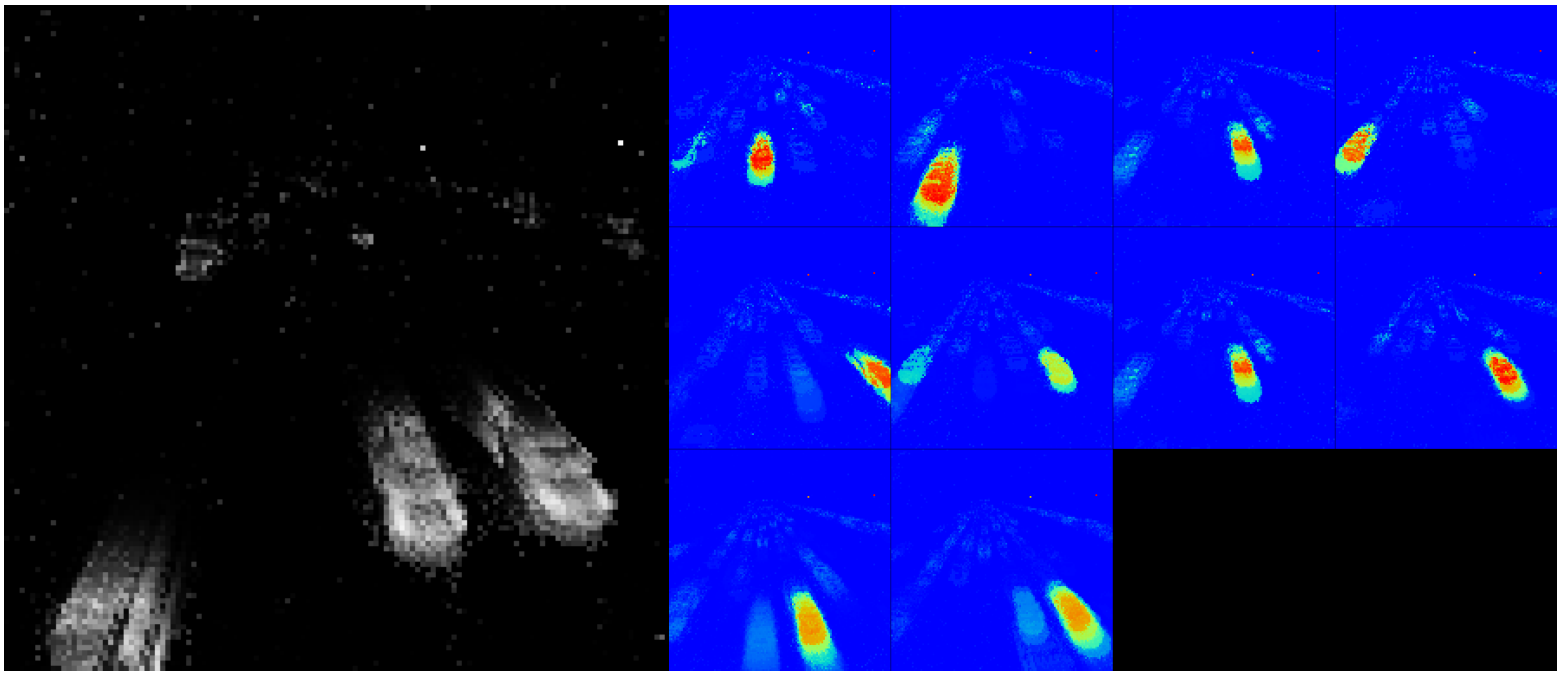
Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains, T. Masquelier, R. Guyonneau, and S. J. Thorpe





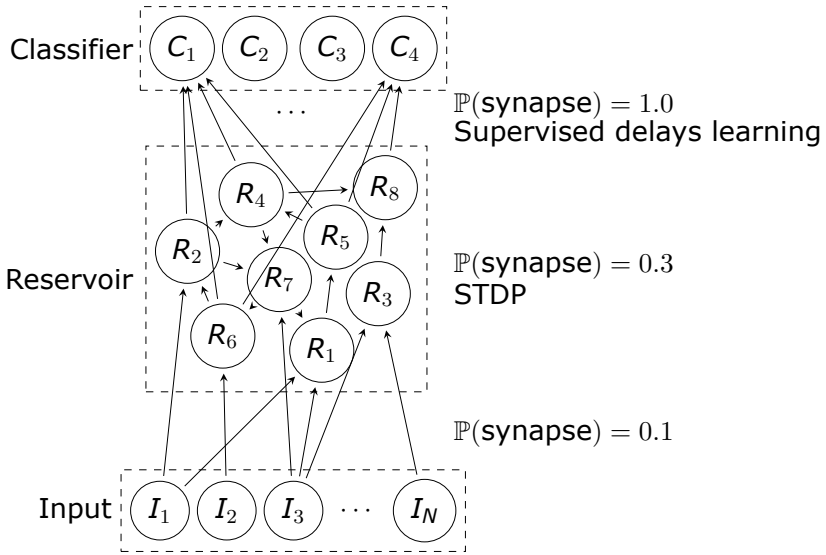
Exemples

Unsupervised features extraction from asynchronous silicon retina through spike-timing-dependent plasticity, O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat

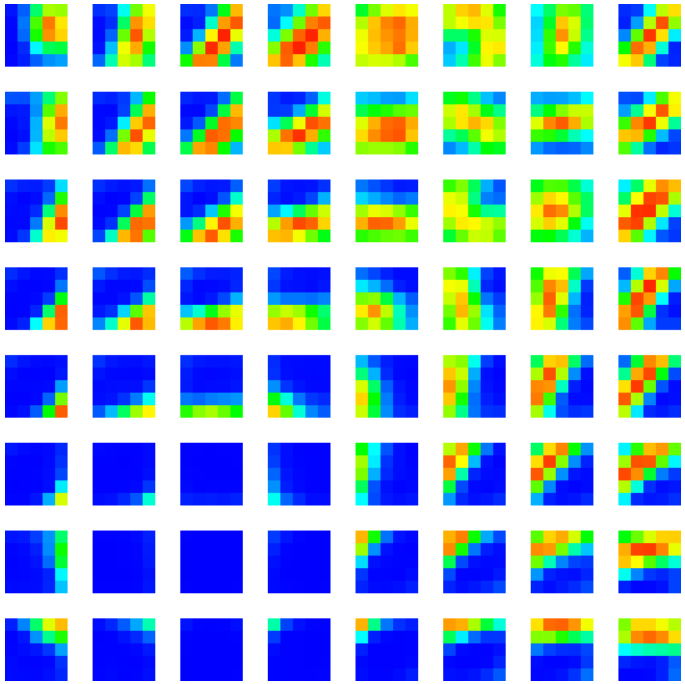


Exemples

Reservoir Computing



Cartes auto-adaptatives



Perspectives

- Améliorer la distribution (désynchronisation)
- Faciliter le déploiement (Docker)
- Continuer le DSL
- Étendre la documentation
- Augmenter le niveau de détails matériels

Vers un outil d'aide à la conception matérielle (exportation de HDL)

N2S3

Neural **N**etwork **S**calable **S**piking **S**imulator

Open-source (CeCILL-B)

`https://sourcesup.renater.fr/wiki/n2s3/start`