

Neural Network Scalable Spiking Simulator

N2S3 [nesi]

Q. Bailleul

✉ quentin.bailleul@etudiant.univ-lille1.fr

W. Gouzer

✉ william.gouzer@etudiant.univ-lille1.fr

Lille, le 19 juin 2014



1 Objectifs

2 Simulateur N2S3

3 Futur du projet

Pourquoi un simulateur ?

- Expérimenter des architectures
- Tester différents codages d'information, procédures d'apprentissage
- Utiliser différents type de composants (e.g. memristors)

Pourquoi créer « Yet Another Simulator » ?

- Performances (Brian)
- Extensibilité
- Open source (Xnet)

Propriétés désirées du simulateur

- Comparable avec les simulateurs existants (ex : Brian, Xnet)
- Scalable
- Générique

Flexibilité et extensibilité des modèles :

- Définir / Redéfinir un comportement
- Ajouter des fonctionnalités
- Mixer les modèles
 - Composants
 - Temps
 - Monitoring



Neural Network Scalable Spiking Simulator

Choix techniques :

- Simulation événementielle
 - Modèles analytiques

Choix techniques :

- Simulation événementielle
 - Modèles analytiques
- Langage Scala
 - Modulaire
 - Extensible
 - Scalable

Choix techniques :

- Simulation événementielle
 - Modèles analytiques
- Langage Scala
 - Modulaire
 - Extensible
 - Scalable
- Acteurs Akka
 - Gestion de la concurrence
 - Système distribué

Définition d'un neurone

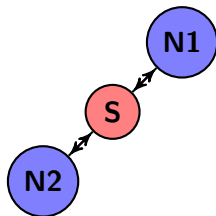
```
abstract class Neuron(  
  layer : Int,  
  var inputs : Seq[ActorRef] = Seq(),  
  var outputs : Seq[ActorRef] = Seq()  
) extends Actor { ... }
```

Définition d'un synapse

```
abstract class Synapse(  
  pre : ActorRef,  
  post : ActorRef  
) extends Actor { ... }
```

Définition d'un Spike

```
trait Spike
```



Définition d'un neurone

```
abstract class Neuron(  
  layer : Int,  
  var inputs : Seq[ActorRef] = Seq(),  
  var outputs : Seq[ActorRef] = Seq()  
) extends Actor { ... }
```

Définition d'un Spike

```
trait Spike
```

Création du réseau

- création des neurones
- création des synapses
- liaison neurones/synapses

Ajouter un nouveau modèle

- Etendre la classe Network et Model
- Créer le modèle de neurone et de synapse
- Ajouter des Spikes (impulsions)

Ajouter un nouveau modèle

- Etendre la classe Network et Model
- Créer le modèle de neurone et de synapse
- Ajouter des Spikes (impulsions)

Main.scala

```
val nbNeuronPerLayer = Seq(5, 2, 3, 1)
```

```
val threshold = 8
```

```
val net = new NetworkDefault(nbNeuronPerLayer, threshold)
```

```
// val net = new NetworkPji(nbNeuronPerLayer, threshold)
```

```
// val net = new Demo1(nbNeuronPerLayer, threshold)
```

```
net.initiateNetwork()
```

```
net.firingSpikes()
```

Projets à court terme :

- mise en place d'un modèle de temps
- automatisation de la création d'un réseau (DSL)
- entrées / sorties
 - création de fichier de sortie
 - parsing du fichier d'entrée de type AER

Projets à long terme :

- instauration d'une architecture distribuée
- rendre l'interface plus conviviale
- ajout de graphiques de monitoring

Synthèse

- Fonctionne actuellement :
 - Le cœur du simulateur (structure, acteurs)
 - Le système de modèles
 - Des simulations simples (sans gestion du temps)



Synthèse

- Fonctionne actuellement :
 - Le cœur du simulateur (structure, acteurs)
 - Le système de modèles
 - Des simulations simples (sans gestion du temps)
- En cours de développement :
 - Modèle de temps
 - Monitoring des acteurs



Synthèse

- Fonctionne actuellement :
 - Le cœur du simulateur (structure, acteurs)
 - Le système de modèles
 - Des simulations simples (sans gestion du temps)
- En cours de développement :
 - Modèle de temps
 - Monitoring des acteurs
- En projet futur :
 - Automatisation : DSL, lecture de fichiers .AER
 - Exploiter les données de sortie (graphique etc)

