```
import numpy

blist = [[0,0],[1,0],[2,0],[3,0],[80,0],[81,0],[82,0],[83,0],[160,0],[161,0],[162,0],[163,0],[240,0],[241,0],[242,0],
[243,0]]

acol = numpy.ndarray.tolist(numpy.add(blist,[320,20]))

arow = numpy.ndarray.tolist(numpy.add(blist,[4,1]))

#b.append(a)

for _ in range(19):

    for _ in range(29):

        blist.append(arow)

        arow = numpy.ndarray.tolist(numpy.add(arow,[4,1]))

    blist.append(acol)

    arow = acol

    arow = numpy.ndarray.tolist(numpy.add(arow,[4,1]))

    acol = numpy.ndarray.tolist(numpy.add(acol,[320,20]))


%%%%%%%%%%%

%New algorithm


x_preN_grid = 80

y_preN_grid = 80

NBpreN = x_preN_grid * y_preN_grid


%convergence factor could be 2^2,3^2,4^2,5^2,6^2

convergence_factor = 16

xfactor = sqrt(convergence_factor)

yfactor = sqrt(convergence_factor)



NBpostN = NBpreN/convergence_factor

preN = sqrt(NBpostN)
```

postN = sqrt()

———

Nov 1, 2019

A. A snapshot of pong environment (Gym-openAi) before preprocessing of image. B. Grayscale images are presented for time points t0, t1 and t4 together with a time-weighted maximum projection of grayscale images for 5 time points. This allows to capture temporal activity in the environment. C. The reduced map of input stimuli is presented here (1-1 mapping between image pixel and input stimulus), which converts individual pixels into firing rates. Baseline firing rate of stimulus is 5Hz which increases nonlinearly (using a sigmoid function) to 40Hz based on pixel intensity level. D. The input layer (R) which is driven by the poisson-processes based inputs. The dynamics of neurons in this layer extract spatial and temporal features of the environment. E. A simplified model of visual cortex (V1) receiving converging inputs from the input layer in a topographical manner. 16 neurons from input layer R map onto 1 neuron in the V1. The connections between R and V1 can adapt (using hebbian spike-time dependent plasticity rules) to allow encoding for object tracking in the environment. F. Raster plot showing the activity of neurons in the input layer R and visual cortex V1.

———

Oct 31, 2019

we have a prototype connection between video game environment and biologically realistic neuronal microcircuits. These models accept visual input with temporal history and adapt the connectivity patterns to encode the object tracking in the visual environment. This provides the first stage for learning appropriate behaviors in a dynamic environment.

———

Sept 12, 2019

model of thalamocortical circuitry with realistic dynamics
processing of time-varying visual inputs
model learning of goal-oriented behavior and decision making through biologically-realistic learning rules

Hypothesis:

our model will produce a neurobiologically-derived AI system that provides mechanistic insights into brain processes underlying decision making and behavior.

Aim1: develop biologically-principled model of a thamaocortical system including visual and motor networks in a closed-loop with simulated game environment.

Game —> Thalamus (LGN, TRN) <—**>** V1 <—**>** M1 —> Game

Reinforcement-learning-center (Global Reward/ Punisher Signaler<—>Error Evaluation) shape synaptic connectivity within V1, M1 and between V1-M1.

Part 1: V1 needs to decode/encode the visual representation of the environment (game).

How to implement part 1?

Option 1: Game —> Thalamus (LGN, TRN) <—> V1 (required in the project). Also need to understand the role of Thalamus other than relaying the information.
Option 2: Game —> V1 (Not accurate representation of biological network but here we assume that the visual inputs are only relayed to V1 through thalamus)

First step— Start with option 2 (below are the steps).

Visual stimuli (pixel intensities) from the game with activate a 2D array of time-varying poisson inputs, where the poisson firing will be modulated by pixel intensity.
These poisson inputs will then project to the modeled LGN topographically and then to **V1** and higher order visual areas**.**
RGB inputs —> grayscale inputs.
Pixel map of visual environment will be downsampled.
Unsupervised learning will occur in the visual areas while ignoring any initial random behaviors produced by the model. This will allow the model to learn the visual representation of objects occurring in the game environment. This type of learning is hypothesized to occur in the ventral visual stream, enabling invariant object representation- e.g. recognizing the same object in different positions, orientations, or scalings.
We will use spike-timing dependent plasticity with homeostatic synaptic scaling.

Visual Input (preprocessed) —— modulates— —> Time varying Poison processes(retina) — — >LGN+TRN (plasticity??)— — plasticity— —> VC(plasticity)

How will we test, if V1(or visual cortex model) is identifying the objects correctly?

Sam proposed in the grant proposal——I need to understand this

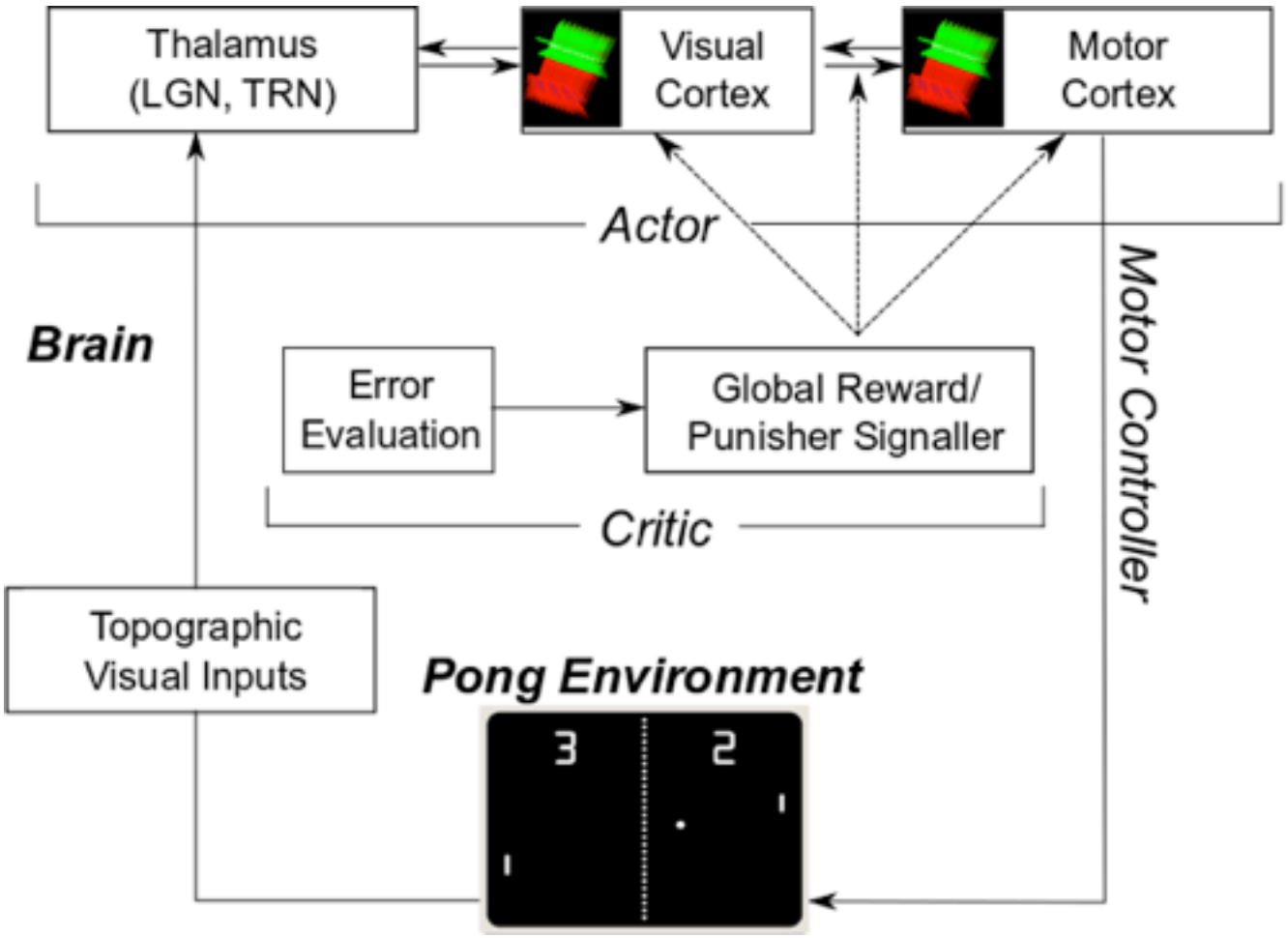
> **SA3.2 Test object recognition.** To test whether the model is successful in performing object recognition, which we hypothesized facilitates behavior, we will utilize pattern recognition algorithms on the higher order visual area spiking activity while applying geometric transformations to distinct game objects in the visual fields. These techniques were used in previous *in vivo* experiments[84]. This approach quantifies if there are distinct neuronal representations for distinct perceptual objects by providing supervised learning algorithms, such as support vector machines[38,85], with binned neuronal population spiking activity. If object recognition is successful, the spiking patterns for different objects are separable with the classification algorithms. In case classification algorithms are not able to distinguish the population spiking patterns, we can still measure specificity of neurons' responsiveness to stimulation with distinct perceptual objects.

Models of homeostatic synaptic scaling

1. **Rowan, Neymotin, Lytton (2014). Electrostimulation to reduce synaptic scaling driven progression of Alzheimer's disease. Front. Comput. Neuroscience.**
2. **Angulo, Orman, Neymotin, Liu….(2017) Tau and amyloid-related pathologies in the entorhinal cortex have different effects in the hippocampal circuit. Neurobiol. Dis.**
3. **Synaptic scaling balances learning in a spiking model of neocortex. in Adaptive and Natural Computing Algorithms. (2013)**

Models of V1

1. **Eguchi, Neymotin, Stringer (2014). Color opponent receptive fields self-organize in a biophysical model of visual cortex via spike-timing dependent plasticity. Front. Neural Circuits.**
2. **Eguchi, Neymotin, Horwitz, Albright (2016). Representation of color.**



Sept 9, 2019

python
import gym

```
env = gym.make("Pong-v0")
env.reset()
for _ in range(1000):
env.render()
action = random.randint(3,4)
observation, reward, done, info = env.step(action)
#env.step(env.action_space.sample())
env.close()
```

ACTION— there are 6 actions in this game 0-5.

each action is repeatedly performed for 2,3 or 4 frames. This causes different displacements of the racket.

```
env.action_space
output = Discrete(6)
```

env.action_space.sample() —>random sample out of possible actions in action space… so it chooses 1 of 6 discrete values

observation, reward, done, info = env.step(action) —> performs the action

```
env1.unwrapped.get_action_meanings()
['NOOP', 'FIRE', 'RIGHT', 'LEFT', 'RIGHTFIRE', 'LEFTFIRE']
```

OBSERVATION

```
env1.observation_space
Box(210, 160, 3)
```

SHOW THE OBSERVATION

SHOW THE OBSERVATION

if a3[0] is observation

a3[0][209][159][2] gives the last element of the array.

a3[0] gives a 3 dimensional RGB pixel map of a screen as given by env.observation_space Box(210, 160, 3)

from matplotlib import pyplot as plt

>>> plt.imshow(observation)

>>> observation.shape

(210, 160, 3)

for gray scale image:

gray = numpy.zeros(shape=(210,160))

>>> for i in range(210):

... for j in range(160):

... gray[i][j]= 0.2989*observation[i][j][0] + 0.5870*observation[i][j][1] + 0.1140*observation[i][j][2]

plt.imshow(gray, cmap=plt.get_cmap('gray'), vmin=0, vmax=255)

<matplotlib.image.AxesImage object at 0x11f284b38>

>>> plt.show()

for down sampling

```
dc=a[:,range(0,a.shape[1],2)]
drdc=dc[range(0,a.shape[0],2),:]
```

>>> gray.shape

(210, 160)

>>> gray_ds = gray[:,range(0,gray.shape[1],2)]

>>> gray_ds.shape

(210, 80)

>>> gray_ds = gray_ds[range(0,gray_ds.shape[0],2),:]

>>> gray_ds.shape

(105, 80)

gray_ds = gray(range(0,gray.shape[0],2),range(0,gray.shape[1],2))

——

i guess in this game, we can only play defense.

to score a point/reward, the agent must be able to

1. predict the trajectory of the ball
2. based on the prediction, choose the action and adjust the speed of the action such that it hits the ball. (thats defense)
3. predict the action + speed of the computer or the other player
4. based on the prediction in 3, deflect the ball so that the other player/computer can't pick the ball. (thats attack)

——

good resources related to pong, DL and RL

https://blog.floydhub.com/spinning-up-with-deep-reinforcement-learning/

——

NOTE: the methods are for env Pong-v0
Try understanding the module env : help(env)
some useful functions— may be other functions are important but i don't see how

1. reset() — resets the state of the environment and returns an initial observation.
2. step(action) — accepts an action and returns a tuple (observation: a3[0], reward: a3[1], done: a3[2], info: a3[3])
3. close() — override close in your subclass to perform any necessary cleanup.
4. compute_reward(self, achieved_goal,desired_goal,info) ——— don't understand this
5. render(mode='human' or mode='rgb_array')

——

pong
brick out

read pixel map from the game ——> to extract the state space.
read the score from the game—> to extract the reward/punishment to be used with the reinforcement learning paradigm.

RL motor cortex model

It will be good to have DL models with:

It will be good to have DL models with:
1. intracellular molecular dynamics and calcium signaling which enable complex learning and adaptation at multiple time-scales
2. the dendritic tree of pyramidal neurons which allows integration, recognition and storage of complex spatiotemporal inputs through modification of arrays of synaptic weights.
3. 6 layered cortical structure
4. dynamically generated neural network oscillations which are detected in EEG and are hypothesized to entrain to and predict complex spatiotemporal inputs.
5. feed forward and recurrent connectivity patterns
6. nonlinear excitability properties.

———

August 30, 2019

Goal:

I want to develop a biological neural network model that can learn and computer play a game.

I ask myself this question: How do we learn how to play a computer game and later use that experience to play games.

- for video game playing, it is common to use a stack of convolutional layers followed by recurrent layers and fully connected feed-forward layer. Why all these layers together?
- Supervised(mapping between data and labels): model is asked to make a decision for which the correct answer is known. Difference between the provided answer and correct answer is used as a loss or penalty to update the model. The goal is to achieve a model that can generalize beyond the training data and performs well with unseen data. More data ——> better performance.
- Unsupervised: Instead of learning a mapping between data and its labels, the objective of unsupervised learning is to discover patterns in the data. Mostly used for clustering or pattern recognition. For games with spares rewards (such as Montezuma's Revenge), such methods can be used. A prominent unsupervised learning technique in deep-learning is **autoencoder: 2 parts - an encoder that maps the input x to a low-dimensional hidden vector h and a decoder that attempts to reconstruct x from h.**
- Reinforcement learning: an agent learns a behavior by interacting with an environment that provides a rewarding signal back to the agent.
  - a key challenge in applying RL to games with sparse rewards is to determine how to assign credit to the many previous actions when a reward signal is obtained
  - Markov Decision Process: the agent can build a probability tree of future states and their rewards. The probability tree can then be used to calculate the utility of the current state

————————————————

August 29, 2019

so far what i understand is that gym ai game environments require actions and based on those actions, the environment is updated and therefore the state variables in the game.

- Identify the output of the Motor cortex model.
  - sensory (S) and motor (M) populations with each having:
    - 192 excitatory cells (ES and EM).
    - 44 fast spiking inhibitory cells (IS and IM).
    - 20 low threshold inhibitory cells (ILS and ILM).

    - Recurrent connectivity:

- Recurrent connectivity:
  - ES->ES; EM->EM (within each class)
  - IS->IS; IM->IM (within each class)
  - ILS->ILS; ILM->ILM (within each class)
  - ES->IS->ES; EM->IM->EM; ES->ILS->ES; EM->ILM->EM (between the excitatory and inhibitory neurons of each population)
  - **IS->ILS->IS; IM->ILM->IM (i could not comprehend that from text in METHODS)**
  -
  - ES->EM->ES (between the two main excitatory cell classes)
- Identify the input of the Motor cortex model.
  - Inputs from the virtual arm to the neural network was provided bu the proprioceptive (P) population, which consisted of 192 NetStims (NEURON spike generators) and encoded muscle length.
  - Muscle Length (or change in it) drives P.
  - P is divided in 4 subpopulations each responsible for representing mean length of one of four muscle groups: P1 for shoulder extensor; P2 for shoulder flexors; P3 for elbow extensors; P4 for elbow flexors.
  - Units employed population coding to represent the muscle lengths, such that within each subpopulation, individual units only fired to a small range of lengths.???
- How is the model learning (or update of synaptic parameters)?
  - By employing reinforcement learning - reward to punishment depending on whether the arm is getting closer or farther from the target - to modify the synaptic weights via STDP, the network can learn to drive the arm to a target.
    - how is target defined?
- Limitation: Currently, the network can only be trained to reach one target at a time. However, it can be extended to learning multiple targets by adding a population to encode target selection as shown by Dura-Bernal et al. 2015 and Spuler et al. 2015

  - Spüler,M.,Nagel,S.,andRosenstiel,W.(2015).“Aspikingneuronalmodel learningamotorcontroltaskbyreinforcementlearningandstructural synaptic plasticity,”in *NeuralNetworks(IJCNN),2015InternationalJointConferenceon* (Killarney:IEEE),1–8.doi:10.1109/IJCNN.2015.7280521

  - Dura-Bernal,S.,Kerr,C.,Neymotin,S.,Suter,B.,Shepherd,G.,Francis,J.,etal. (2015a).“Large-scalem1microcircuitmodelwithplasticinputconnections frombiologicalpmdneuronsusedforprostheticarmcontrol,”in *24thAnnual ComputationalNeuroscienceMeeting(CNS15),BMCNeuroscience* (Prague).

——

August 23, 2019

Things to do today: August 21,2019

```
mpiexec -n 4 nrniv -python -mpi tut1.py ——> This doesn't work

instead Robert suggested using

mpirun -n 4 python tut1.py
```

- Install NetPyNe

- Install NetPyNE
  - 1: go to http://www.netpyne.org
  - 2: while installing netpyne using pip, I encountered the following problem: “Installing collected packages: kiwisolver, six, cycler, subprocess32, backports.functools-lru-cache, python-dateutil, matplotlib, future, matplotlib-scalebar, netpyne

    Found existing installation: six 1.4.1

    ERROR: Cannot uninstall 'six'. It is a distutils installed project and thus we cannot accurately determine which files belong to it which would lead to only a partial uninstall.”
    - To deal with this problem I used the following command: “sudo pip install netpyne --ignore-installed six"
- start playing with the model “Cortical spiking model driving a virtual arm”
  - Starting models: 36,38,43,49–53,63
  - Neymotin et al. 2017: Optimizing computer models of corticospinal neurons to replicate in vitro dynamics. J Neurophys.
    - Mice P23-P32, T = 34C, **Corticospinal neurons** (**thick tufted pyramidal neurons in motocortex upper layer 5B** that project caudally via the medullary pyramids to brain centers or directly yo the spinal cord- Sutter et al 2013; these cell are the **main output of the motor cortex**) show strong sag with hyperpolarization, lack of adaptation, and a nearly linear FI curve.
    - Other excitatory cell population of L5, the **corticostriatal cells in lower layer 5A (any model available? what role?)**, which also differ anatomically by having smaller apical tufts, are involved in various loops via other telencephalon structures: thalamus, striatum, cerebellum and contralateral cortex.
    - Read the article Anderson_Sheets_Kiritani_Shepherd_NatNeuroscience2010 to understand corticospinal and corticostrial pathways within the mouse motor cortex (**L2/3 -> L5**).
      - Of many (?) classes projecting various targets, two projection classes corticospinal and corticostriatal neurons are centrally involved in motor control.
      - In mouse, L5B is relatively thick layer and consists of multiple sub-layers.
      - **corticospinal neurons (CSp) were distributed from upper to lower layer 5B.**
      - **inputs from L2/3 neurons were strongest for upper L5B CSp neurons but fell steeply as a function of increasing soma position towards lower L5B.**
      - **In contrast, relatively weak intralaminar perisomatic inputs from L5B were a consistent feature.**
      - **corticostriatal neurons (IT type or non-PT type) were distributed from upper 5A to lower 5B**.
      - **these type of corticostriatal neurons received strong L2/3 input in lower 5A and very weak input from L2/3 in upper 5A and 5B. Weak intralaminar inputs from L5 were present throughout the layer**.
  - Neymotin et al. 2016: Multitarget multiscala simulation for pharmacological treatment of dystonia in motor cortex. Front Pharmacol.

    - **https://senselab.med.yale.edu/ModelDB/ShowModel. cshtml?model=189154**

    - multi scale model of primary motor cortex (molecular to cellular to network)
    - 1715 model-neurons with ion channels and intracellular molecular dynamics.
    - connectivity based on electrophysiology.
    - single compartment inhibitory neurons and 5 compartment pyramidal neurons.
    - connectivity matrix based on Weiler et al. 2008 (Top-down laminar organization of the excitatory network in motor cortex. Nat Neuroscience)
    - brute-force
  - Neymotin et al. 2011: Emergence of physiological oscillation frequencies in a computer model of neocortex.Front Comput Neurosci
  - Neymotin et al. 2016: Calcium regulation of HCN channels supports persistent activity in a multi scale model of neocortex. Neuroscience.
  - Eguchi et al. 2014: Color opponent receptive fields self-organize in a biophysical model of **visual**

- Eguchi et al. 2014: Color-opponent receptive fields self-organize in a biophysical model of **visual cortex** via **spike-timing dependent plasticity**. Front Neural Circuits.
- Chadderdon et al. 2014: **Motor cortex** microcircuit simulation based on brain activity mapping. Neural Computation.
- Vierling-Claassen et al. 2010: Computational modeling of distinct neocortical oscillations driven by cell-type selective ontogenetic drive: separable resonant circuits controlled by low-threshold spiking and fast-spiking interneurons. Front. Human Neuroscience.
- Sherman et al. 2016. Neural mechanisms of transient neocortical beta rhythms: Converging evidence from humans, computational modeling, monkeys and mice. PNAS.
- Neymotin et al. 2011: Synaptic information transfer in computer models of neocortical columns. J Comput. Neurosci.

- Other models:

———————————————————————

August 19, 2019

visual cortex models:

184182: Pvalb-IRES-Cre neuron from layer 5 of the mouse primary visual cortex—from Allen institute.

184327: model of a Nr5a1-Cre neuron from layer 2/3 of the mouse primary visual cortex - pyramidal intratelencephalic GLU cell;

184161: Rorb-IRES2-Cre-D VISp layer 2/3 - Neocortex V1 L2/6 pyramidal intratelencephalic GLU cell

184231: same as 184161 but probably a different cell

247848: Arkhipov A, Gouwens NW, Billeh YN, Gratiy S, Iyer R, Wei Z, Xu Z, Abbasi-Asl R, Berg J, Buice M, Cain N, da Costa N, de Vries S, Denman D, Durand S, Feng D, Jarsky T, Lecoq J, Lee B, Li L, Mihalas S, Ocker GK, Olsen SR, Reid RC, Soler-Llavina G, Sorensen SA, Wang Q, Waters J, Scanziani M, Koch C (2018) Visual physiology of the layer 4 cortical circuit in silico. PLoS Comput Biol14:e1006535[PubMed]

———————————————————

August 19, 2019

Dura-Bernal et al. 2016 Frontiers in Neuroscience paper:

- spiking network model of sensorimotor cortex was trained to drive a realistic virtual musculoskeletal arm to reach a target.
- Lesion was simulated by either silencing the neurons or removing synaptic connections.
- The remaining cells (?? how are remaining cells defined) were then systematically probed with a set of single and multiple-cell stimulations(?? what kinds of stimulation), and results were used to build an inverse model of the neural system (?? what kind of model).
- The inverse model was constructed using a kernel adaptive filtering method, and was used to predict the neural stimulation pattern required to recover the pre-lesion neural activity.
- Applying the derived neurostimulation to the lesioned network improved the reaching behavior performance.

Comprehensive model description:

1. Neymotin et al 2013 - Reinforcement learning of two-joint virtual arm reaching in a computer model of sensorimotor cortex. Neural Computation.

model of sensorimotor cortex. Neural Computation.
2. Dura-Bernal et al. 2015b - Cortical spiking network interfaced with virtual musculoskeletal arm and robotic arm. Front. Neurorobot.

Other paper

deep convolutional network— which uses hierarchical layers of tiled convolutional filters to mimic the effects of receptive fields.

xxxxxxxxxxxxxxxx———————————