Materials Matter:
How biologically inspired alternatives to conventional neural networks improve meta-learning and continual learning

Jeff Clune
@jeffclune

Associate Professor, Computer Science
University of British Columbia

Research Team Leader
OpenAI
Evolution of Structural Organization

- Modularity
- Hierarchy
Modularity

- Localization of function in an encapsulated unit (Lipson 2007)
- Car (spark plug, muffler, wheel), bodies (organs), brains, software, etc.
- Enables increased complexity and adaptability
Modularity

• Rare in previous neuroevolution

• Suggests selection on performance alone does not produce modularity

Kashtan and Alon 2005
Evolutionary Origins of Modularity


Jeff Clune  Jean-Baptiste Mouret  Hod Lipson
Retina Problem

object on left side?

object on right side?

object on both sides? (L&R:)

L&R

L

R
Why does modularity evolve?

- Hypothesis from founding neuroscientist (Ramón y Cajal 1899)
  - Selection to minimize connection costs
Summary

- Performance Alone (PA)
- Performance & Connection Costs (P&CC)

Clune, Mouret, & Lipson. 2013. Proceedings of the Royal Society
• P&CC significantly more modular, higher-performing (p < 0.0001)
• Perfect decomposition in 56% of P&CC, never for PA (p < 0.0001)
• Significantly more evolvable (p < 0.0001)

Clune, Mouret, & Lipson. 2013. Proceedings of the Royal Society
Evolution of Structural Organization

- Modularity
- Hierarchy
Hierarchy

• recursive composition of lower-level units (Lipson 2007)
• important principle in brains
• also doesn’t occur in evolution by default
Hypothesis: Connection Costs also Cause Hierarchy

• Hierarchical networks are
  • sparse
  • composed of nested modules
Without a Connection Cost

Performance and Connection Cost (P&CC)

Performance Alone (PA)

% sub-problems solved

generations to adapt to a new environment

With a Connection Cost

Talk Overview

• Describe alternatives to conventional neural networks *loosely* inspired by biology
  • that can improve meta-learning, continual learning

• Deep dives
  • Differentiable Hebbian Plasticity
  • Differentiable Neuromodulated Hebbian Plasticity ("backpropamine")
  • ANML
Meta-Learning Algorithms

• Two major camps
  • Meta-learn good initial weights + SGD
    • e.g. MAML, Finn et al. 2017
  • Meta-learn RNN, which creates its own learning algorithm
    • Learning to Reinforcement Learn, Wang et al. 2016
    • RL², Duan et al. 2016
Meta-Learning Algorithms

• Two major camps
  • Meta-learn good initial weights + SGD
    • e.g. MAML, Finn et al. 2017
  • Meta-learn RNN, which creates its own learning algorithm
    • Learning to Reinforcement Learn, Wang et al. 2016
    • RL², Duan et al. 2016
• Outer loop: optimize RNN with parameters $\theta$ for “lifetime” performance
  • Inner loop: run $\theta$ (with reward as input)

• Et voila!
  • It learns an entire RL algorithm
  • Theoretically can learn any RL algorithm
LRL

(a) Labyrinth I-maze
(b) Illustrative Episode
(c) Performance


Learns to
- explore
- exploit
- all on its own!
LRL

Learns to
- build a model
- plan
- all on its own!
Rubik’s Cube

- Identifies properties of the world
  - friction, mass of cube, size of cube, etc.
- Exploits that information
Materials Matter

- Still have to decide the *materials* of the network
- RNNs forced to do all lifetime learning with *activations*
  - may be unstable
  - proposal: store information in *weights* too
Differentiable Hebbian Learning
Differentiable plasticity: training plastic neural networks with backpropagation
Miconi, Clune, Stanley. ICML. 2018

Thomas Miconi
Jeff Clune
Ken Stanley

UBER AI Labs
Differentiable Hebbian Learning

Differentiable plasticity: training plastic neural networks with backpropagation
Miconi, Clune, Stanley. ICML. 2018

• Can store info in weights (in addition to activations)
• Hebbian learning (trained via SGD)
Hebbian Learning

• neurons that fire together, wire together

\[ w_{ij}^{t+1} = w_{ij}^t + \eta x_i^t x_j^t \]

• many capabilities
  • unsupervised learning (e.g. PCA)
  • associative recall
  • …
Differentiable Hebbian Learning

Differentiable plasticity: training plastic neural networks with backpropagation
Miconi, Clune, Stanley. ICML. 2018

- Recurrent, Hebbian network
  - inner loop: network updates with no SGD
  - outer loop: differentiate through episode, update trainable parameters via SGD

\[
y_j = \tanh \left\{ \sum_{i \in \text{inputs}} \left( w_{i,j} + \alpha_{i,j} H_{i,j}(t) \right) y_i \right\}
\]

\[
H_{i,j}(t + 1) = \eta y_i y_j + (1 - \eta) H_{i,j}(t)
\]

Trainable parameters, optimized by SGD to maximize lifetime/episode reward

\[
H_{i,j} \quad \alpha_{i,j}
\]

Lifetime quantity (init=0)
Differentiable Hebbian Learning

Differentiable plasticity: training plastic neural networks with backpropagation
Miconi, Clune, Stanley. ICML. 2018

- Near then-SOTA on Omniglot

<table>
<thead>
<tr>
<th>Vinyals et al. (Matching Networks) (Vinyals et al., 2016)</th>
<th>Snell et al. (ProtoNets) (Snell et al., 2017)</th>
<th>Finn et al. (MAML) (Finn et al., 2017)</th>
<th>Mishra et al. (SNAIL) (Mishra et al., 2017)</th>
<th>DP (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.1%</td>
<td>97.4%</td>
<td>98.7% ± 0.4%</td>
<td>99.07% ± 0.16</td>
<td>98.5% ± 0.57</td>
</tr>
</tbody>
</table>

Table 1: Results for the 5-way, 1-shot omniglot tasks, including recent reported results and the new differentiable plasticity (DP) result (± indicates 95% CI). Note that these reports describe widely varying approaches and model sizes (see text).
Differentiable Hebbian Plasticity
Miconi, Clune, Stanley, ICML 2018

Diagram showing a neural network with input patterns and error over episodes.

- **Input Pattern 1**
  - Time: 0, 20000, 40000, 60000, 80000, 100000
  - Error: 0.25, 0.20, 0.15, 0.10, 0.05, 0.00
- **Input Pattern 2**
  - Time: 0, 20000, 40000, 60000, 80000, 100000
  - Error: 0.25, 0.20, 0.15, 0.10, 0.05, 0.00
- **Input Pattern 3**
  - Time: 0, 20000, 40000, 60000, 80000, 100000
  - Error: 0.25, 0.20, 0.15, 0.10, 0.05, 0.00
- **Test Pattern (partial 2)**
  - Time: 0, 20000, 40000, 60000, 80000, 100000
  - Error: 0.25, 0.20, 0.15, 0.10, 0.05, 0.00
- **Desired Output (full 2)**
  - Time: 0, 20000, 40000, 60000, 80000, 100000
  - Error: 0.25, 0.20, 0.15, 0.10, 0.05, 0.00

Graph showing error over episodes for different models:
- **LSTM (2050 neurons)**
- **Non-plastic RNN (2050 neurons)**
- **Plastic RNN (51 neurons)**
Differentiable Hebbian Learning

Differentiable plasticity: training plastic neural networks with backpropagation
Miconi, Clune, Stanley. ICML. 2018

• Image reconstruction: learn (memorize) an image, reconstruct it

• 2M+ parameters

LSTMs cannot solve this
Differentiable Hebbian Learning

Differentiable plasticity: training plastic neural networks with backpropagation
Miconi, Clune, Stanley. ICML. 2018

Maze Navigation

Episode 0
Episode 500,000

Learned to Explore & Exploit (Better)
Differentiable Neuromodulated Plasticity

“Backpropamine”: Miconi, Rawal, Clune, Stanley, ICLR, 2018

- Hebbian learning is local (hard optimization problem)
- Better: turn learning on in some weights only in certain contexts
  - e.g. if I am playing chess AND I just won, THEN:
    - increase learning in only chess playing parts of the brain
Differentiable Neuromodulated Plasticity

“Backpropamine”: Miconi, Rawal, Clune, Stanley, 2018

**Hebbian Learning**

\[
x_j(t) = \sigma\left\{ \sum_{i \in \text{inputs to } j} (w_{i,j} + \alpha_{i,j}\text{Hebb}_{i,j}(t))x_i(t-1) \right\}
\]

**Neuromodulated Hebbian Learning**

\[
\text{Hebb}_{i,j}(t + 1) = \text{Clip}(\text{Hebb}_{i,j}(t) + \eta x_i(t-1)x_j(t))
\]

**Eligibility Trace Version**

\[
\text{Hebb}_{i,j}(t + 1) = \text{Clip}(\text{Hebb}_{i,j}(t) + M(t)x_i(t-1)x_j(t))
\]

\[
E_{i,j}(t + 1) = (1 - \eta)E_{i,j}(t) + \eta x_i(t-1)x_j(t).
\]
Differentiable Neuromodulated Plasticity

“Backpropamine”: Miconi, Rawal, Clune, Stanley, 2018

Simple Task
network says if one of the symbols just shown is the secret symbol

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline LSTM (Zaremba et al., 2014)</td>
<td>104.26 ± 0.22</td>
</tr>
<tr>
<td>LSTM with Differential Plasticity</td>
<td>103.80 ± 0.25</td>
</tr>
<tr>
<td>LSTM with Simple Neuromodulation</td>
<td>102.65 ± 0.30</td>
</tr>
<tr>
<td>LSTM with Retroactive Neuromodulation</td>
<td>102.48 ± 0.28</td>
</tr>
</tbody>
</table>

Word prediction, Penn-Tree Bank
Learning to Continually Learn

Shawn Beaulieu  Lapo Frati  Joel Lehman  Thomas Miconi  Ken Stanley

Jeff Clune*  Nick Cheney*

*co-senior authors

ECAI 2020

The University of Vermont  DARPA  UBER AI Labs
Catastrophic Forgetting

• Achilles Heel of machine learning
• In sequential learning
  • Learn task A, then learn task B
  • ML overwrites A when learning B
    • forgets catastrophically
  • Animals, including humans
    • pick up where we left off
    • forget gradually
• Must solve catastrophic forgetting to continually learn
Many Proposed Solutions: All Manual

• Rehearsal techniques
• Pseudo-patterns
• Activation sharpening
• Sparse representations
• Progressive networks
• Elastic weight consolidation
• PathNet
• Intelligent synapses

• Experience replay
• Generative replay
• Progress & Compress
• etc.
Many Proposed Solutions: All Manual

- Rehearsal techniques
- Pseudo-patterns
- Activation sharpening
- **Sparse representations**
- Progressive networks
- Elastic weight consolidation
- PathNet
- Intelligent synapses

- Experience replay
- Generative replay
- Progress & Compress
- etc.
Frequent Manual Path Philosophy

- Optimize for one thing and hope for in other
  - e.g. optimize for sparse representations, hope for decreased catastrophic forgetting
Meta-Learning Philosophy

• Don’t optimize for one thing and hope for another
• Optimize for what you want
Hypothesis

- There’s a good chance humans are not smart enough to manually build systems that continually learn well.
Proposal: Use \textit{meta-learning} to learn to continually learn

- Optimize for \textit{what}
  - Learn a \textit{sequence} of tasks
  - Be good on \textit{all of them} at the end
Meta-Learning Algorithms

• Two major camps
  • Meta-learn good initial weights + SGD
    • e.g. MAML, Finn et al. 2017
  • Meta-learn RNN, which creates its own learning algorithm
    • Learning to Reinforcement Learn, Wang et al. 2016
    • RL², Duan et al. 2016
    • OpenAI et al. 2019, Rubik’s Cube
Meta-Learning Algorithms

• Two major camps
  • Meta-learn good initial weights + SGD
    • e.g. MAML, Finn et al. 2017
  • Meta-learn RNN, which creates its own learning algorithm
    • Learning to Reinforcement Learn, Wang et al. 2016
    • RL², Duan et al. 2016
    • OpenAI et al. 2019, Rubik’s Cube
\[ \theta + \text{SGD (e.g. MAML)} \]

\begin{align*}
\theta_1 & \\
\nabla_{\theta_1}(L_{\text{meta}}(\theta_1^n)) & \\
\theta_2 & \\
& \cdots \\
\theta_m & \\
\end{align*}

inner-loop learning

\[
\begin{pmatrix}
\theta_1^1 & \theta_1^2 & \cdots & \theta_1^n \\
\theta_2^1 & \theta_2^2 & \cdots & \theta_2^n \\
& & \cdots \\
& & & \\
& & & \\
& & & \\
\end{pmatrix}
\]

\[ L_{\text{meta}}(\theta_1^n) \]

meta-training (outer-loop learning)
"meta-training"

\[
\begin{align*}
\theta_1 & \xrightarrow{} \begin{bmatrix} \theta_1^1 & \theta_1^2 & \cdots & \theta_1^n \end{bmatrix} \\
\nabla_{\theta_1}(\mathcal{L}_{\text{meta}}(\theta_1^n)) & \xrightarrow{} \mathcal{L}_{\text{meta}}(\theta_1^n) \\
\theta_2 & \xrightarrow{} \begin{bmatrix} \theta_2^1 & \theta_2^2 & \cdots & \theta_2^n \end{bmatrix} \\
\vdots & \hspace{1em} & \vdots & \hspace{1em} & \vdots \\
\theta_m & \xrightarrow{} \begin{bmatrix} \theta_m^1 & \theta_m^2 & \cdots & \theta_m^n \end{bmatrix} \\
\nabla_{\theta_m}(\mathcal{L}_{\text{meta}}(\theta_m^n)) & \xrightarrow{} \mathcal{L}_{\text{meta}}(\theta_m^n)
\end{align*}
\]
“meta-training”

\[ \theta_1 \rightarrow \theta_1^1 \theta_1^2 \cdots \theta_1^n \rightarrow L_{meta}(\theta_1^n) \]

\[ \theta_2 \rightarrow \theta_2^1 \theta_2^2 \cdots \theta_2^n \rightarrow L_{meta}(\theta_2^n) \]

inner-loop learning

\[ \nabla_{\theta_1}(L_{meta}(\theta_1^n)) \]

meta-training (outer-loop learning)
inner-loop learning

$\theta_1 \rightarrow \begin{bmatrix} \theta_1^1 & \theta_1^2 & \cdots & \theta_1^n \end{bmatrix} \rightarrow \mathcal{L}_{\text{meta}}(\theta_1^n)$

inner-loop learning

$\theta_2 \rightarrow \begin{bmatrix} \theta_2^1 & \theta_2^2 & \cdots & \theta_2^n \end{bmatrix} \rightarrow \mathcal{L}_{\text{meta}}(\theta_2^n)$

... \\

$\theta_m \rightarrow \begin{bmatrix} \theta_m^1 & \theta_m^2 & \cdots & \theta_m^n \end{bmatrix} \rightarrow \mathcal{L}_{\text{meta}}(\theta_m^n)$
meta-learning for continual, multi-task learning

\[
\theta_1 \quad \nabla_{\theta_1} (\mathcal{L}_{\text{meta}}(\theta_1^n))
\]

\[
\theta_2 \quad \ldots
\]

\[
\theta_m
\]

inner-loop learning

Task 1
\[\theta_1^1 \ldots \theta_1^k\]
\[k \text{ updates}\]

Task 2
\[\theta_2^1 \ldots \theta_2^k\]
\[k \text{ updates}\]

Task t
\[\theta_t^1 \ldots \theta_t^k\]
\[k \text{ updates}\]

\[
\mathcal{L}_{\text{meta}}(\theta_1^n)
\]

all t tasks

\[
\mathcal{L}_{\text{meta}}(\theta_2^n)
\]

all t tasks
Online-aware Meta-Learning (OML)

Javed & White, NeurIPS, 2019

• validates the vision of meta-learning solutions to continual learning
• we were
  • inspired by it
  • compare to it
meta-learn then freeze representation, SGD for PLN
• Performs well
  • After sequentially training on 150 classes of Omniglot
    • 97% on meta-test training set (near-perfect memorization)
    • ~63% on meta-test test set (worse at generalizing, but still impressive)

• Learns a sparse representation
• Gets a lot right
• But is still ultimately subject to SGD
  • which was not optimized for continual learning
  • has to find a representation that avoids CF when SGD is applied
Can we do better?

- We propose: allowing control over SGD via neuromodulation
Traditional Neuromodulation

- NM neurons change learning rates in other neurons
- Enables data-dependent, thus task-specific, learning

Soltoggio et al. (2008)
Neuromodulation Solves CF on Simple Networks & Problems

Scaling Traditional Neuromodulation

- Struggled to scale it up
- Insight (Shawn Beaulieu)
  - maybe it is because the forward pass is not affected
  - thus forward-pass interference still exists

Soltoggio et al. (2008)
Activation-Based Neuromodulation

- Neuromodulation that
  - directly modulates activations: 
    - selective activation
  - indirectly modulates learning: 
    - selective plasticity

Soltoggio et al.
A Neuromodulated Meta-Learning algorithm (ANML)
A Neuromodulated Meta-Learning algorithm (ANML)

Neuromodulatory network ($\theta^{NM}$)

Prediction network ($\theta^p$)

element-wise multiplication

Class = 3
Normal Deep Learning

Inference everywhere
Learning everywhere
Enables:

- Selective activation (inference)
- Selective plasticity (backprop)
Domain

- Omniglot, following OML
- each character type is a class/task
Ideally, differentiate through 600 tasks

\[ \theta_1 \]

\[ \nabla_{\theta_1}(\mathcal{L}_{\text{meta}}(\theta^n_1)) \]

\[ \theta_2 \]

\[ \vdots \]

\[ \theta_m \]

inner-loop learning

Task 1

\[ \theta_1^1 \ldots \theta_1^k \]

k updates

Task 2

\[ \theta_1^1 \ldots \theta_1^k \]

k updates

\[ \vdots \]

Task t

\[ \theta_1^1 \ldots \theta_1^k \]

k updates

\[ \mathcal{L}_{\text{meta}}(\theta^n_1) \]

all t tasks

\[ \mathcal{L}_{\text{meta}}(\theta^n_2) \]

all t tasks
Approximation: train on task t+1
validate on t+1 & some previous tasks

inner-loop learning

Task 1
\( \theta_1^1 \ldots \theta_1^k \)
\( k \) updates

Task 2
\( \theta_1^1 \ldots \theta_1^k \)
\( k \) updates

Task t
\( \theta_1^1 \ldots \theta_1^k \)
\( k \) updates

\( \mathcal{L}_{meta}(\theta_1^n) \)
all t tasks

inner-loop learning

Task 1
\( \theta_1^1 \ldots \theta_1^k \)
\( k \) updates

Task 2
\( \theta_1^1 \ldots \theta_1^k \)
\( k \) updates

Task t
\( \theta_1^1 \ldots \theta_1^k \)
\( k \) updates

\( \mathcal{L}_{meta}(\theta_2^n) \)
all t tasks

\( \theta_1 \)

\( \nabla_{\theta_1}(\mathcal{L}_{meta}(\theta_1^n)) \)

\( \theta_2 \)

\( \ldots \)

\( \theta_m \)

meta-learning (outer-loop learning)
Learn sequentially on one class in the inner-loop

- \( \theta_0 \)
- Inner-loop train step 0
- \( \theta_1 \)
- Inner-loop train step 1
- \( \theta_2 \)
- Inner-loop train step 2
- \( \vdots \)
- Inner-loop train step 19
- \( \theta_{20} \)
Backpropagate through the SGD steps

Inner-loop train step 0
θ₀

Inner-loop train step 1
θ₁

Inner-loop train step 2
θ₂

...  

Inner-loop train step 19
θ₁₉

Inner-loop train step 20
θ₂₀

Chars just seen

Others from meta-train training set

meta-loss on both
Backpropagate through the SGD steps

\[ \theta_0 \leftarrow \theta_0 - \eta \frac{\partial \mathcal{L}}{\partial \theta_0} \]
\[ \theta_1 \leftarrow \theta_1 - \eta \frac{\partial \mathcal{L}}{\partial \theta_1} \]
\[ \theta_2 \leftarrow \theta_2 - \eta \frac{\partial \mathcal{L}}{\partial \theta_2} \]
\[ \vdots \]
\[ \theta_{19} \leftarrow \theta_{19} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{19}} \]
\[ \theta_{20} \leftarrow \theta_{20} - \eta \frac{\partial \mathcal{L}}{\partial \theta_{20}} \]

Inner-loop train step 0
Inner-loop train step 1
Inner-loop train step 2
\ldots
Inner-loop train step 19

Chars just seen
Others from meta-train training set

meta-loss on both

\[ \mathbb{u}, \mathbb{v}, \mathbb{w}, \mathbb{y}, \ldots, \mathbb{u}, \mathbb{v}, \mathbb{y}, \ldots, \mathbb{v}, \mathbb{n} \]
META-TESTING
Meta-test-training and meta-test-testing

meta-testing post-training on a held-out set of 200 classes

Meta-test training

train on first class (15 instances)
Meta-test-training and meta-test-testing

Meta-test-training post-training on a held-out set of 200 classes

Meta-test-training
train on first class (15 instances)

Meta-test-testing
test on remaining 5 instances of first class

Meta-test test accuracy
Meta-test-training and meta-test-testing

meta-testing post-training on a held-out set of 200 classes

class

Meta-test training

sequentially train on 2 classes
(15 instances each)

Meta-test testing

test on remaining 5 instances
of 2 classes

Meta-test test accuracy
Meta-test-training and meta-test-testing

meta-testing post-training on a held-out set of 600 classes

Meta-test training sequentially
train on 600 classes
(15 instances each)

Meta-test testing test on remaining
5 instances
of all 600 classes
Reminder: Continual Learning is Hard

• Normal Deep Learning
  • IID sampling (no catastrophic forgetting)
  • Multiple passes through data

• Sequential Learning
  • Catastrophic Forgetting
  • One pass through data
Results

![Graph showing accuracy over number of classes seen for different treatments: Scratch, Pretrained, OML, OML-OLFT, ANML. Each treatment shows a decrease in accuracy as more classes are seen, with Scratch dropping sharply at the beginning and then flattening out, Pretrained starting high and dropping steadily, OML and OML-OLFT showing gradual decreases, and ANML maintaining a high accuracy throughout.]

sequential learning, one epoch
vs. IID Oracles, Relative Performance Drop

• Oracles eliminate CF
• Oracle - Sequential
  • isolates performance drop due to CF
vs. IID Oracles, Relative Performance Drop

<table>
<thead>
<tr>
<th>Performance Drop</th>
<th>Scratch</th>
<th>Pretrain &amp; Transfer</th>
<th>OML</th>
<th>ANML</th>
</tr>
</thead>
<tbody>
<tr>
<td>99%</td>
<td>67%</td>
<td>47%</td>
<td>8%</td>
<td></td>
</tr>
</tbody>
</table>

after one pass through 600 classes

Suggests ANML has mostly solved CF is in this problem
Learned Sparsity

Both OML and ANML: No dead neurons!
vs. ~14% with sparsity auxiliary loss
(Javed & White 2019)
Update

• Sara Pelivani et al. at UCL / Evolution.ai found results are ~just as good without the NM network (she will share more soon)

• We had controlled num params, so made red smaller
  • Turns out being smaller is the key driver of improved performance

• We are still investigating
  • why smaller models do better
  • where neuromodulation helps
    • e.g. for domain transfer: https://arxiv.org/abs/2108.12056
ANML Conclusions

- OML/ANML can learn 600 sequential tasks, and still perform pretty well on all on average
- Learns to produce sparse representations
  - and likely many other things to solve CF
- Future work:
  - more and harder domains
  - other flavors of meta-learning (e.g. RNNs)
Artificial General Intelligence (AGI)
or Human-level AI, if you prefer

• Long way to go

• How will we get there?
Manual Path to AI

- Dominant paradigm in ML
- Phase 1: Identify key building blocks
Key Building Blocks?

- convolution
- attention mechanisms
- spatial transformers
- batch/layer norm
- a learned loss (e.g. evolved policy gradients)
- hierarchical RL, options
- structural organization (regularity, modularity, hierarchy)
- intrinsic motivation (many different flavors)
- auxiliary tasks (predictions, autoencoding, predicting rewards, etc.)
- good initializations (Xavier, MAML, etc.)
- catastrophic forgetting solutions
- universal value functions
- hindsight experience replay
- LSTM cell machinery variants
- complex optimizers (Adam, RMSprop, etc.)

- Dyna
- variance reduction techniques
- activation functions
- good hyperparameters
- capsules
- gradient-friendly architectures (skip connections, highway networks)
- value functions, state-value functions, advantage functions
- recurrence (where?)
- multi-modal fusion
- trust regions
- Bayesian methods
- Active learning
- Probabilistic models
- Distance metrics (latent codes)
- etc.

how many more? hundreds? thousands? can we find them all?
Manual Path to AI

- Dominant paradigm in ML
- Phase 1: Identify key building blocks

- Phase 2: Combine building blocks into complex thinking machine
  - Herculean task
    - complex, non-linear interactions
    - debugging, optimizing would be a nightmare
    - massive team required (e.g. CERN, Apollo)
Clear Machine Learning Trend:
Hand-designed pipelines are ultimately outperformed by learned solutions

- Features
- Architectures
- Hyperparameters & data augmentation
- RL algorithms

suggests alternate path
AI-Generating Algorithms
Clune 2019

- Learn as much as possible
- Bootstrap from simple to AGI
- Expensive outer loop
  - produces a sample-efficient, intelligent agent
- Existence proof
  - Earth
AI-Generating Algorithms
Clune 2019

Three Pillars

1. Meta-learn architectures
2. Meta-learn learning algorithms
3. Generate effective learning environments

Handcrafting each is slow, limited by our intelligence/time
Better to learn them. Let ML+compute do the heavy lifting
AI-Generating Algorithms
Clune 2019

Three Pillars

1. Meta-learn architectures
2. Meta-learn learning algorithms
3. Generate effective learning environments
Three Pillars

1. **Meta-learn architectures**
   - Evolved NAS Real et al. 2017
   - Generative Teaching Networks Such et al. ICML 2020.
   - Synthetic Petri Dish. Rawal et al. 2020
AI-Generating Algorithms
Clune 2019

Three Pillars

1. Meta-learn architectures
2. Meta-learn learning algorithms
3. Generate effective learning environments

CORL Keynote, see jeffclune.com/videos.html
Overall Conclusions

• Described
  • Differentiable Hebbian plasticity
  • Differentiable neuromodulated Hebbian Plasticity
  • ANML: Learning to continually learn via neuromodulation
  • AI-Generating Algorithms

• In all, **materials matter**
  • Hebbian plasticity vs. normal RNN
  • Neuromodulation

• What other materials should we be building with?
• Might we be able to search for them?
Thanks!

Main collaborators

• Thomas Miconi
• Shawn Beaulieu
• Ken Stanley

• Nick Cheney
• Joel Lehman
• Lapo Frati

Join us at U. British Columbia!