Topics:

- Wrap-up:
  – Few-Shot Learning
  – Architecture Search
  – Open directions in Deep Learning

CS 4803-DL / 7643-A
ZSOLT KIRA

- **Projects!**
  - Due May 3rd (May 5th with grace period)
  - Cannot extend due to grade deadlines!

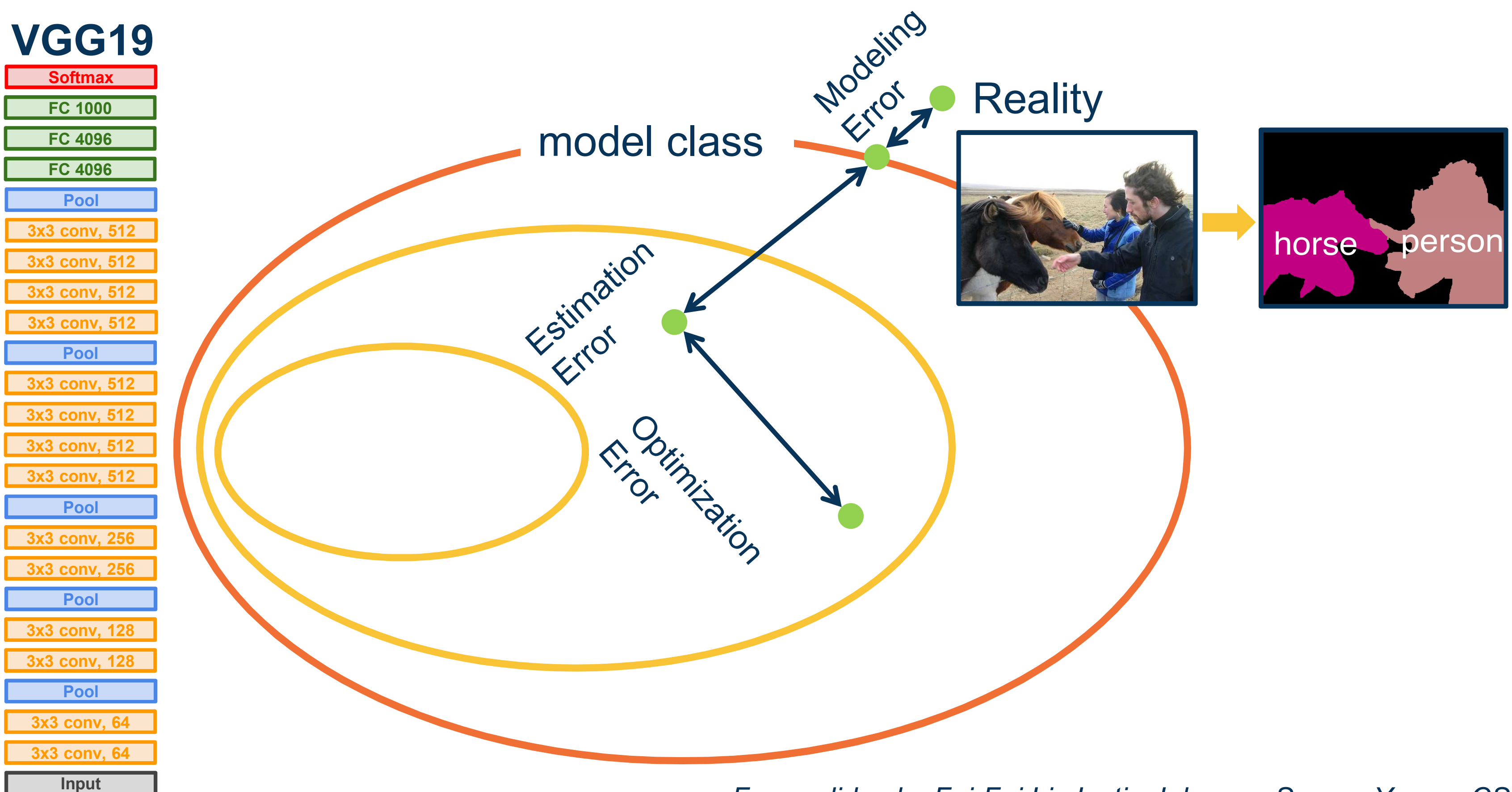
- CIOS
  - Please make sure to fill out! Let us know about things you liked and didn't like in comments so that we can keep or improve!

4803DL

7643A

Some existing works not covered…

- Current / Recent Past
  - AutoML
  - Meta-learning
  - 3D perception
  - Unsupervised, semi-supervised, domain adaptation, zero/one/few-shot learning
  - Memory
  - Visual question answering, embodied question answering
  - Adversarial Examples
  - Continual/lifelong learning without forgetting
  - World modeling, learning intuitive/physics models
  - Visual dialogue, agents, chatbots

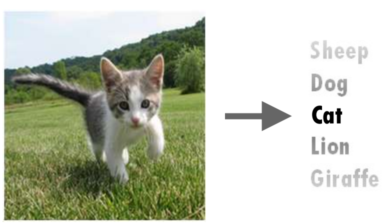


From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Generalization

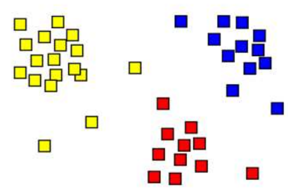
Supervised Learning

- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification



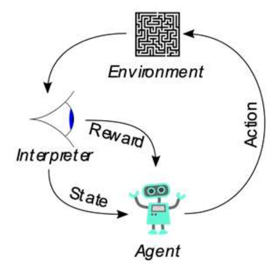
Unsupervised Learning

- Input: $\{X\}$
- Learning output: $P(x)$
- Example: Clustering, density estimation, etc.



Reinforcement Learning

- Evaluative feedback in the form of **reward**
- No supervision on the right action



Types of Machine Learning

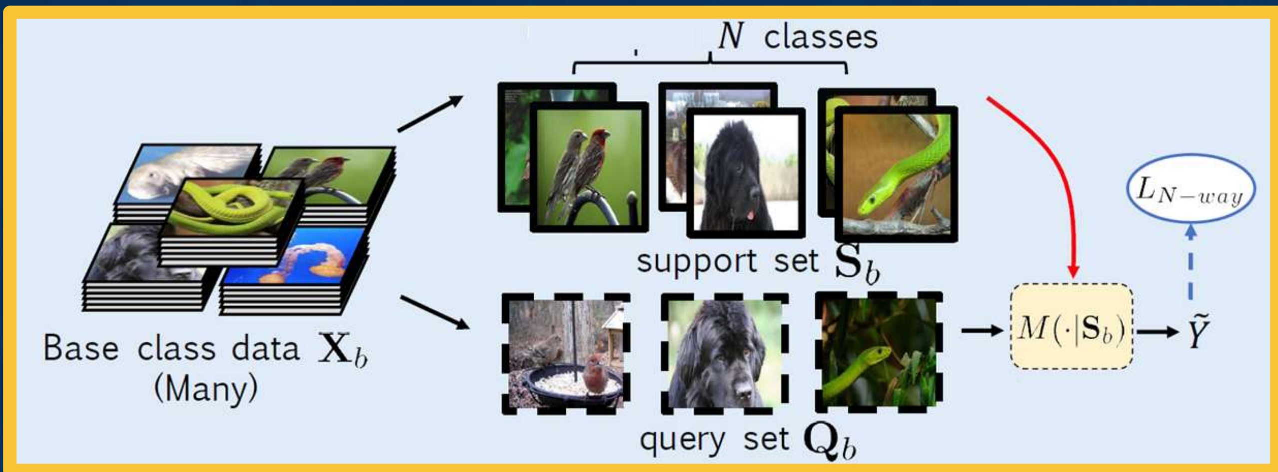
There is a large number of different low-labeled settings in DL research

Setting	Source	Target	Shift Type
Semi-supervised	Single labeled	Single unlabeled	None
Domain Adaptation	Single labeled	Single unlabeled	Non-semantic
Domain Generalization	Multiple labeled	Unknown	Non-semantic
Cross-Category Transfer	Single labeled	Single unlabeled	Semantic
Few-Shot Learning	Single labeled	Single few-labeled	Semantic
Un/Self-Supervised	Single unlabeled	Many labeled	Both/Task

Non-Semantic Shift

Semantic Shift

Few-Shot Learning



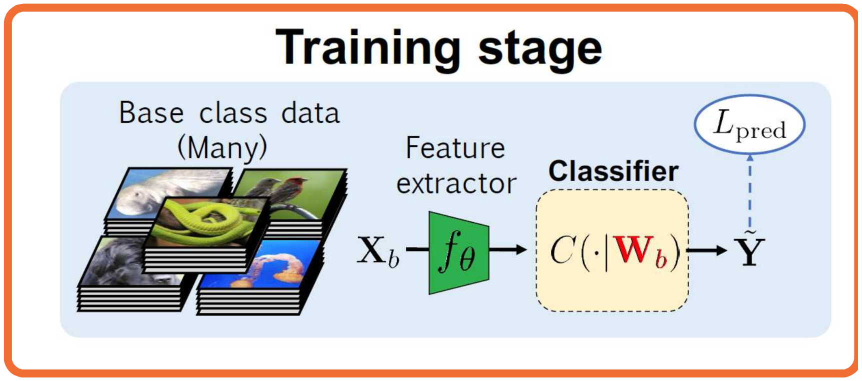
Lots of Labels (Base categories)

Very Few Labels (New categories)

Chen et al., A Closer Look at Few-Shot Learning

Few-Shot Learning

- Do what we always do: **Fine-tuning**
- Train classifier on base classes

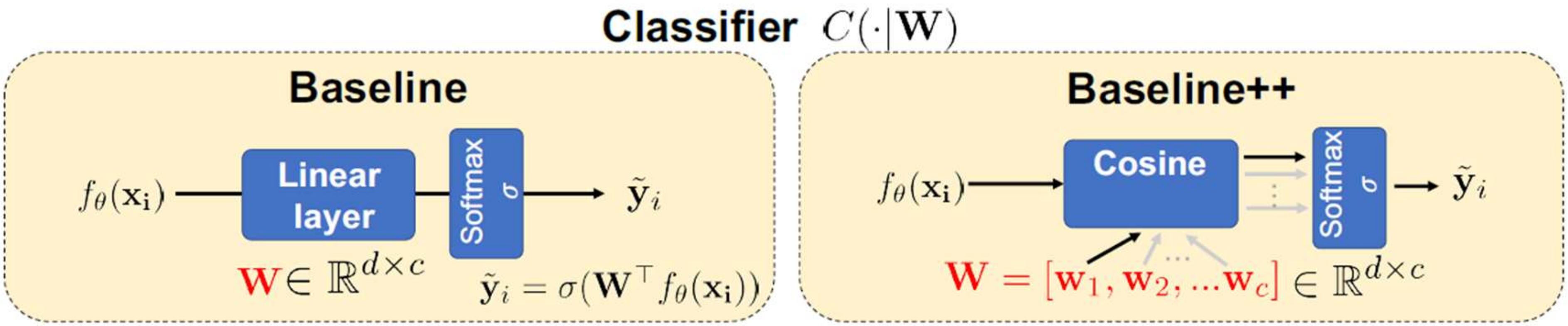


- Optionally freeze feature extractor
- Learn classifier weights for new classes using few amounts of labeled data (during "query" time)
  - Surprisingly effective compared to more sophisticated approaches (Chen et al., Dhillon et al., Tian et al.)

Chen et al., A Closer Look at Few-Shot Learning
Dhillon et al., A Baseline for Few-Shot Image Classification
Tian et al., Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need?

Finetuning Baseline

- We can use a **cosine (similarity-based)** classifier rather than fully connected linear layer



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Chen et al., A Closer Look at Few-Shot Learning
https://en.wikipedia.org/wiki/Cosine_similarity

Cosine Classifier

Cons of Normal Approach

- The training we do on the base classes **does not factor the task into account**
- No notion that we will be **performing a bunch of N-way tests**
- **Idea:** simulate what we will see during test time

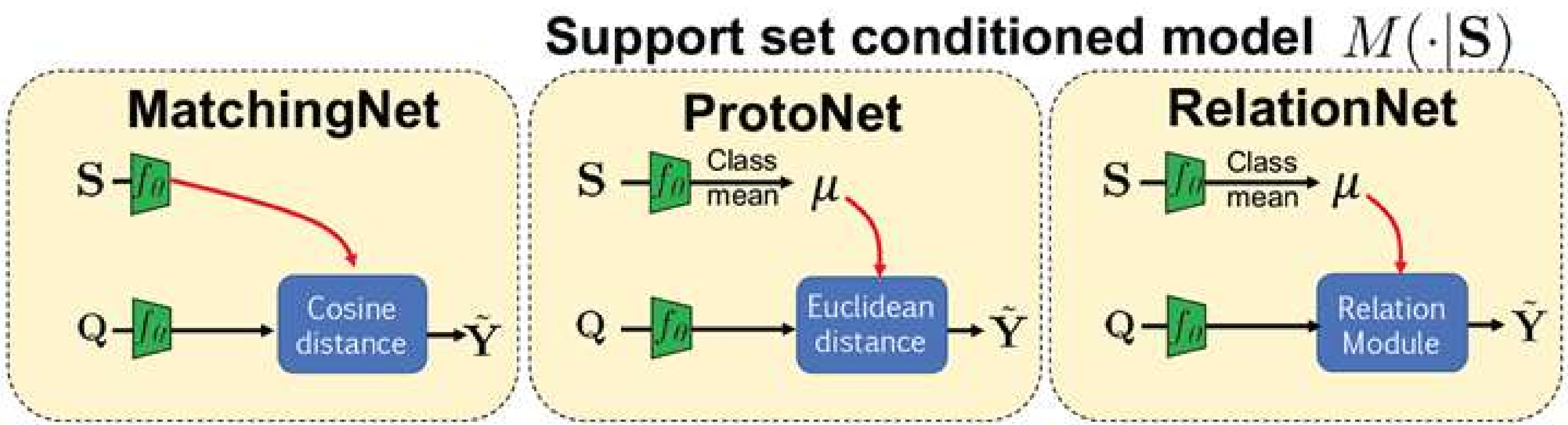
Meta-Training

Set up a set of **smaller tasks** during training which **simulates** what we will be doing during **testing: N-Way K-Shot Tasks**

- Can **optionally pre-train features** on held-out base classes

Testing stage is now the same, but with **new classes**

Learning a model conditioned on support set $M(\cdot|\mathbf{S})$



Chen et al., A Closer Look at Few-Shot Learning

Approaches using Meta-Training

How to parametrize learning algorithms?

Two approaches to defining a meta-learner:

- **Take inspiration from a known learning algorithm**
  - kNN/kernel machine: Matching networks (Vinyals et al. 2016)
  - Gaussian classifier: Prototypical Networks (Snell et al. 2017)
  - Gradient Descent: Meta-Learner LSTM (Ravi & Larochelle, 2017) , Model-Agnostic Meta-Learning MAML (Finn et al. 2017)
- **Derive it from a black box neural network**
  - MANN (Santoro et al. 2016)
  - SNAIL (Mishra et al. 2018)

Slide Credit: Hugo Larochelle

Learn gradient descent:

- Parameter initialization and update rules
- **Output:**
  - Parameter initialization
  - Meta-learner that decides *how* to update parameters

Learn just an initialization and use normal gradient descent (MAML)

- **Output:**
  - Just parameter initialization!
  - We are using SGD

Slide Credit: Hugo Larochelle

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Slide Credit: Hugo Larochelle

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

  - this is quite similar to LSTM cell state updates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Slide Credit: Hugo Larochelle

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

  - this is quite similar to LSTM cell state updates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

    - state c_t is model M's parameter space θ_t ← c_0 becomes a *learned initialization*

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Slide Credit: Hugo Larochelle

Meta-Learner LSTM

Georgia Tech

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

  - this is quite similar to LSTM cell state updates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

    - state c_t is model M's parameter space θ_t ← c_0 becomes a *learned initialization*
    - state update $\tilde{c}_t$ is the negative gradient $-\nabla_{\theta_{t-1}} \mathcal{L}_t$

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Slide Credit: Hugo Larochelle

Meta-Learner LSTM

Georgia Tech

- Training a "**gradient descent procedure**" applied on some learner M
  - gradient descent starts from some initial parameters θ_0 and then performs the following updates:

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$$

  - this is quite similar to LSTM cell state updates:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

    - state c_t is model M's parameter space θ_t ← c_0 becomes a *learned initialization*
    - state update $\tilde{c}_t$ is the negative gradient $-\nabla_{\theta_{t-1}} \mathcal{L}_t$
    - f_t and i_t are LSTM gates: $i_t = \sigma\left(\mathbf{W}_I \cdot \left[\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}\right] + \mathbf{b}_I\right)$ ← *adaptive learning rate*

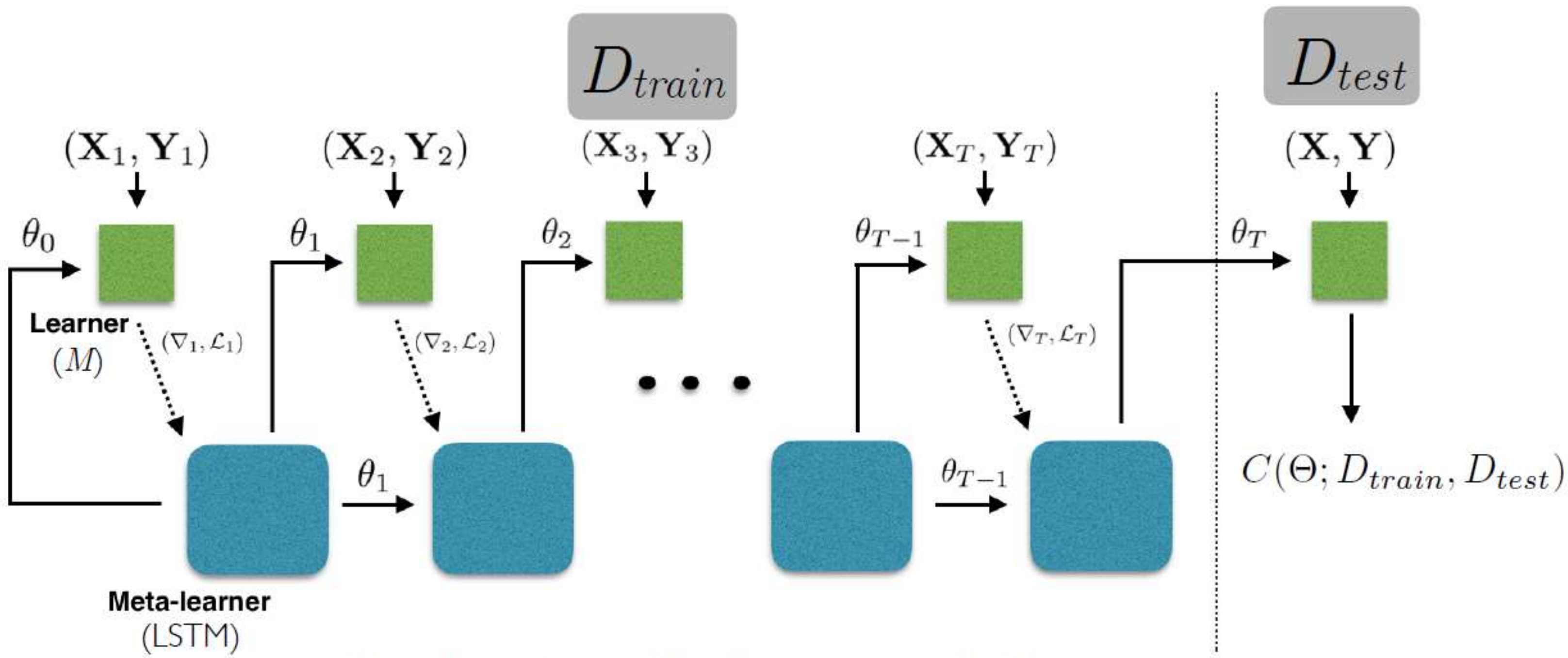
$f_t = \sigma\left(\mathbf{W}_F \cdot \left[\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}\right] + \mathbf{b}_F\right)$ ← *adaptive weight decay*

- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Slide Credit: Hugo Larochelle

Meta-Learner LSTM

- Training a "**gradient descent procedure**" applied on some learner M



- Optimization as a Model for Few-Shot Learning (2017)
  Sachin Ravi and Hugo Larochelle

Slide Credit: Hugo Larochelle

Meta-Learner LSTM

Georgia Tech

- Training a "**gradient descent procedure**" applied on some learner M
  - MAML proposes not to bother with training an LSTM for the gradient descent updates and constant step-size updates

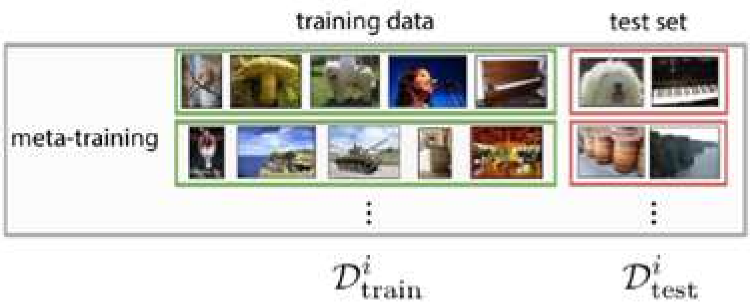
- Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks (2017)
  Chelsea Finn, Pieter Abbeel and Sergey Levine

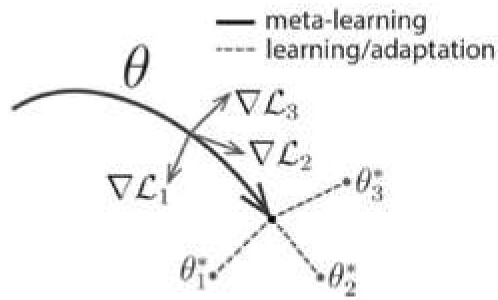
Slide Credit: Hugo Larochelle

Model-Agnostic Meta-Learning (MAML)

Georgia Tech

a general recipe:





$$\theta \leftarrow \theta - \beta \sum_i \nabla_\theta \underbrace{\mathcal{L}(\theta - \alpha\nabla_\theta\mathcal{L}(\theta, \mathcal{D}^i_{\text{train}}), \mathcal{D}^i_{\text{test}})}_{\text{“meta-loss” for task } i}$$

* in general, can take more than one gradient step here

** we often use 4 – 10 steps

Finn et al., “Model-Agnostic Meta-Learning”

Slide Credit: Hugo Larochelle

Model-Agnostic Meta-Learning (MAML)

Georgia Tech

supervised learning: $f(x) \rightarrow y$

supervised meta-learning: $f(\mathcal{D}_{\text{train}}, x) \rightarrow y$

model-agnostic meta-learning: $f_{\text{MAML}}(\mathcal{D}_{\text{train}}, x) \rightarrow y$

$$f_{\text{MAML}}(\mathcal{D}_{\text{train}}, x) = f_{\theta'}(x)$$

$$\theta' = \theta - \alpha \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \nabla_\theta \mathcal{L}(f_\theta(x), y)$$

Just another computation graph...

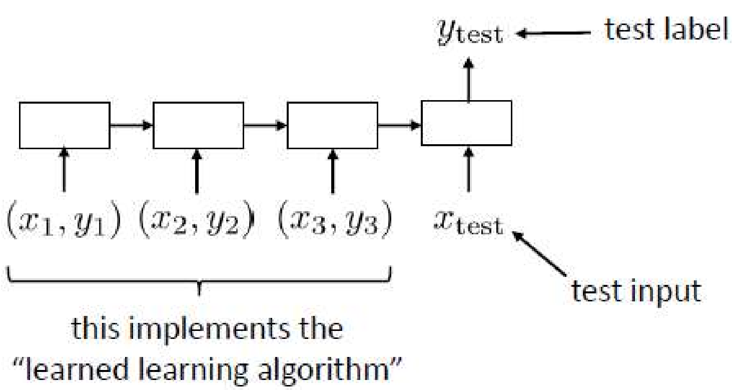
Can implement with any autodiff package (e.g., TensorFlow)

Slide Credit: Hugo Larochelle

Model-Agnostic Meta-Learning (MAML)

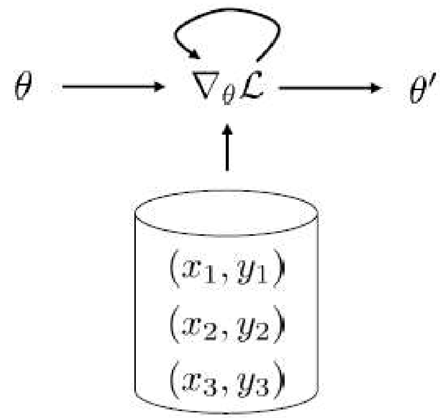
Georgia Tech

RNN-based meta-learning



- Does it converge?
  - Kind of?
- What does it converge to?
  - Who knows...
- What to do if it's not good enough?
  - Nothing...

MAML

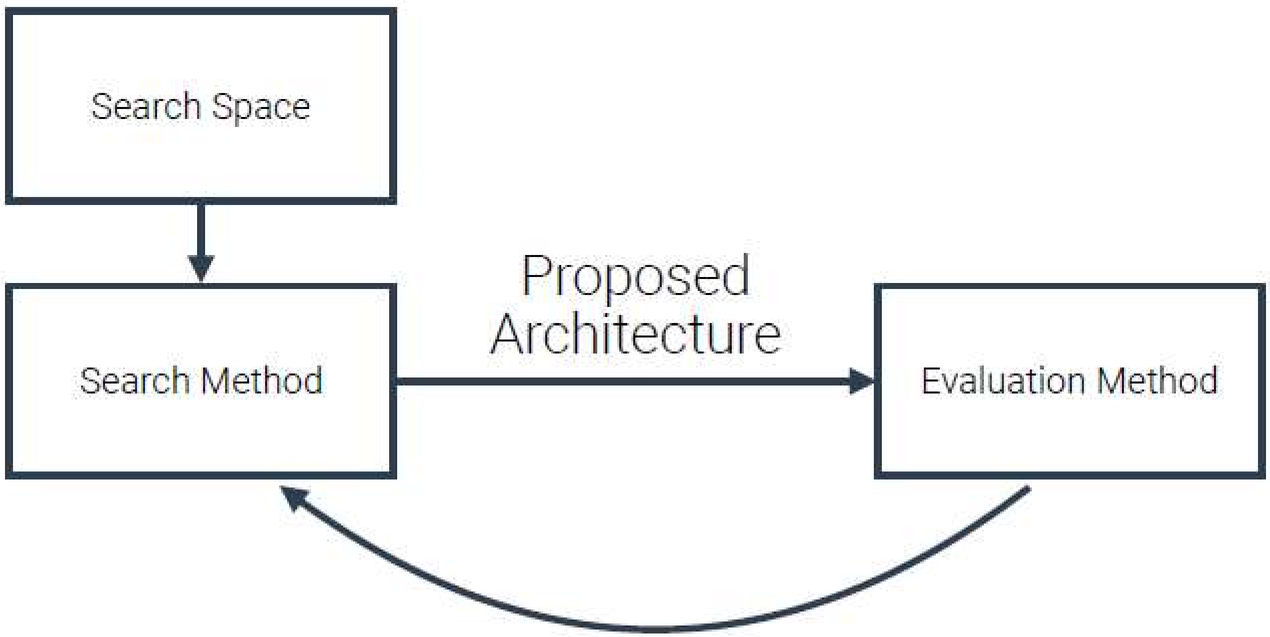


- Does it converge?
  - Yes (it's gradient descent...)
- What does it converge to?
  - A local optimum (it's gradient descent...)
- What to do if it's not good enough?
  - Keep taking gradient steps (it's gradient descent...)

Slide Credit: Hugo Larochelle

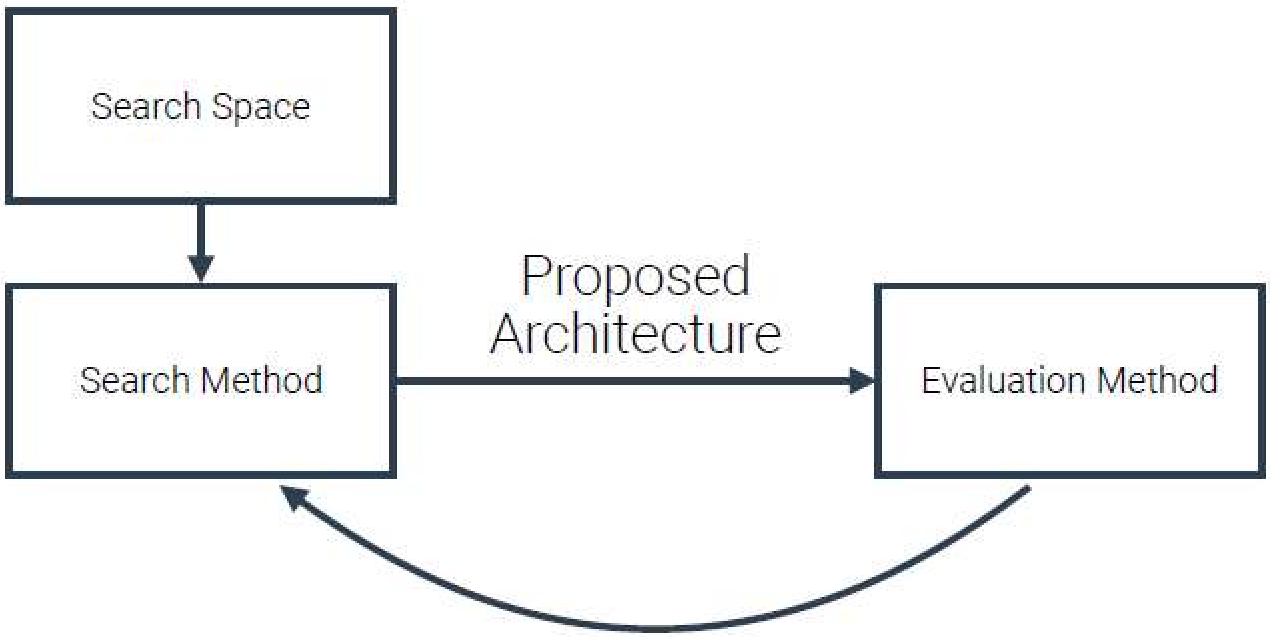
Comparison

Architecture Search



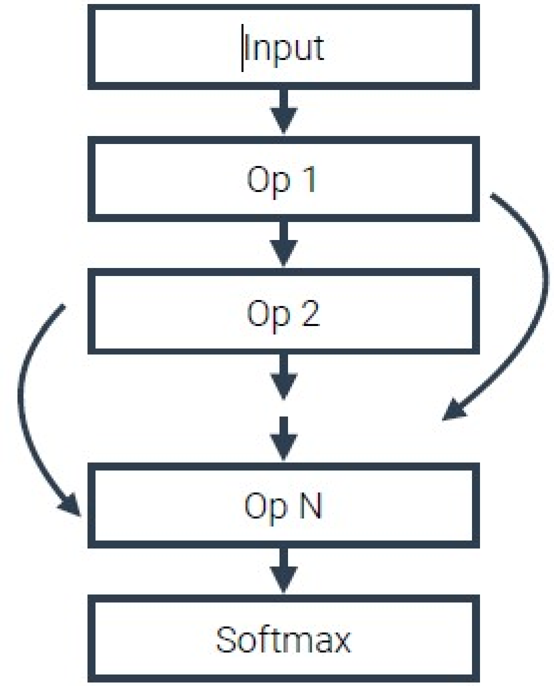
Slides by Erik Wijmans

Architecture Search

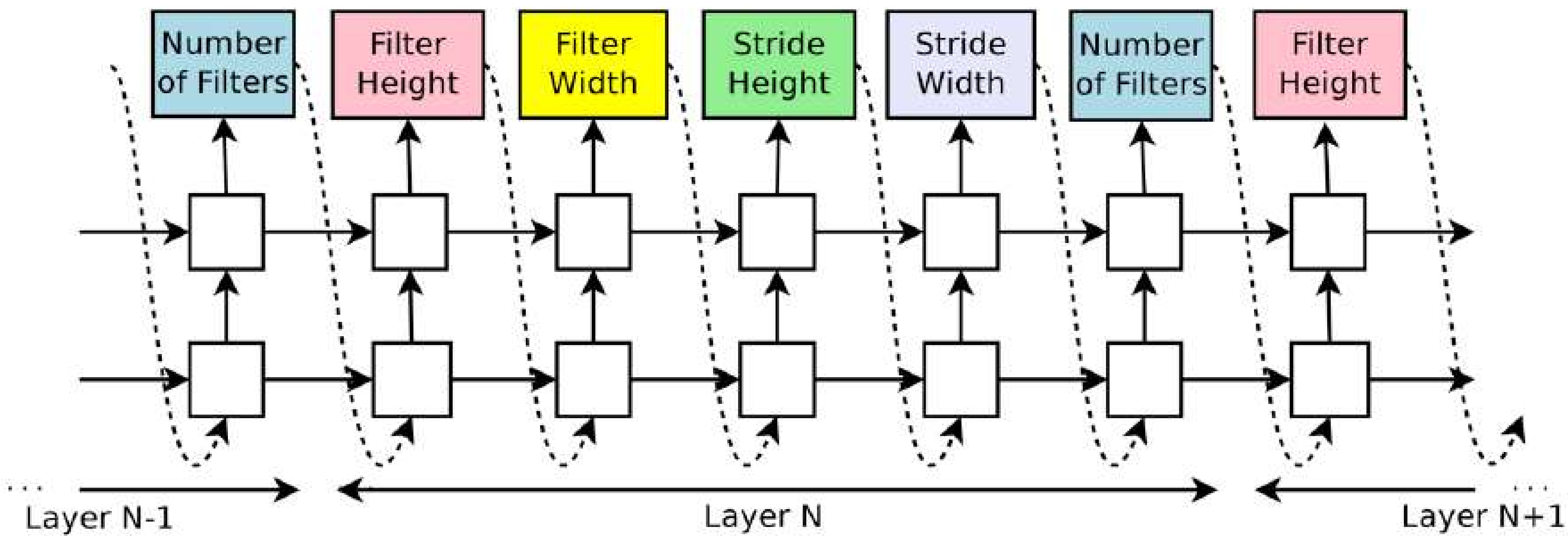


Slides by Erik Wijmans

- Motivated by the observation that a DNN architecture can be specified by a string of variable length (i.e. Breadth-first traversal of their DAG)
- Use reinforcement learning to train an RNN that builds the network



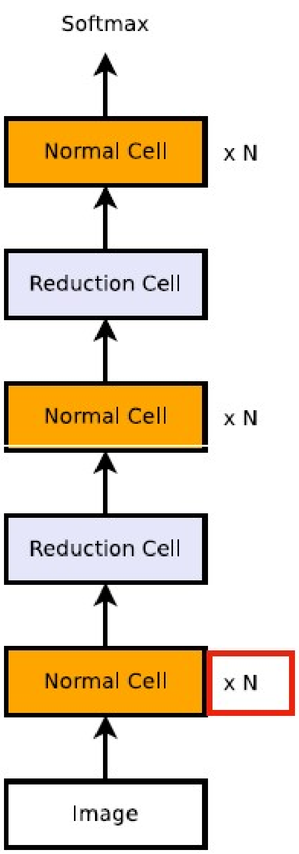
Slides by Erik Wijmans



Slides by Erik Wijmans

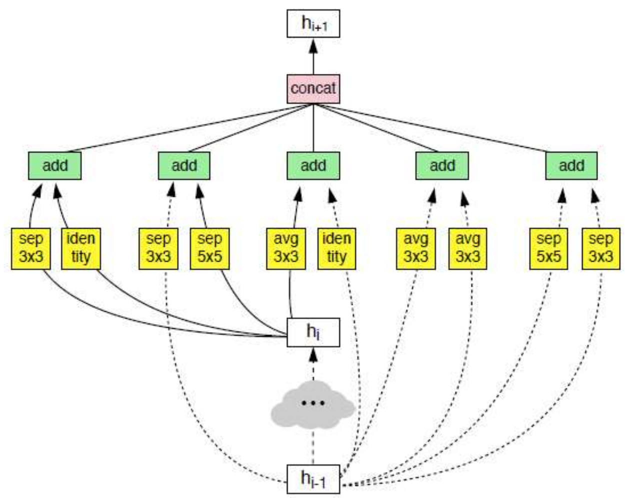
RNNs for Architecture Search

- This is a very general method
- The cost of that is compute: This used 800 GPUs (for an unspecified amount of time) and trained >12,000 candidate architectures
- Instead, limit the search space with “blocks”

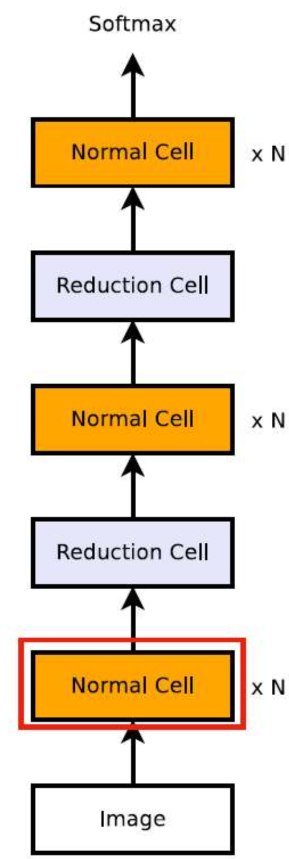


Slides by Erik Wijmans

Evolving Modular Blocks

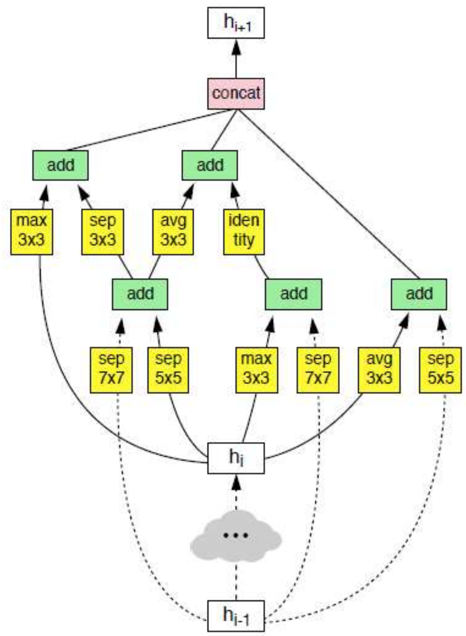


Normal Cell

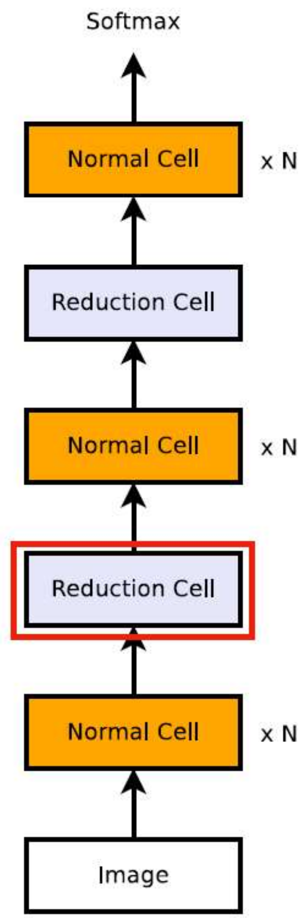


Slides by Erik Wijmans

Evolving Modular Blocks



Reduction Cell

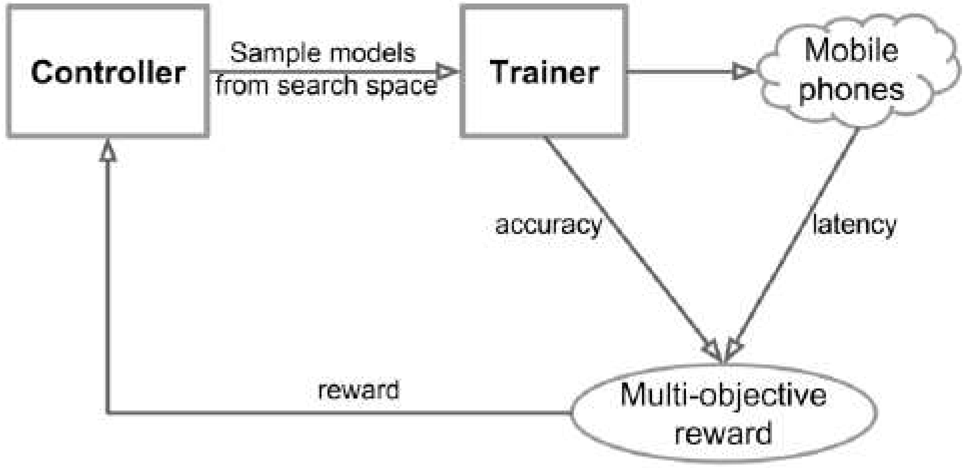


Slides by Erik Wijmans

Evolving Modular Blocks

Georgia Tech

- One benefit of search via RL is that validation performance need not be the *only* metric



Slides by Erik Wijmans

Wrap-up

Some current/upcoming topics

- More recent
  - Transformers for vision, audio, etc.
  - Fixing reinforcement learning
    - First you have to admit you have a problem
  - Simulation frameworks, joint perception, planning, and action
    - Navigation, mapping
  - Uncertainty quantification, robustness
  - Deep Learning and logic!
  - Just scaling everything up and watch the magic!
    - Especially multi-modal, multi-task problems

Things to Watch out For

- Research is cyclical
  - SVMs, boosting, probabilistic graphical models & Bayes Nets, Structural Learning, Sparse Coding, Deep Learning
  - Deep learning is unique in its depth and breadth, but...
  - Deep learning may be improved, reinvented, combined, overtaken

- Learn fundamentals for techniques across the field:
  - Know the span of ML techniques and choose the ones that fit your problem!
  - **Be responsible** in 1) how you use it, 2) promises you make and how you convey it

- Try to understand landscape of the field
  - Look out for what is coming up next, not where we are

- Have fun!