Generalizing from Few Examples with Meta-Learning

Hugo Larochelle
Work done at Twitter
Google Brain

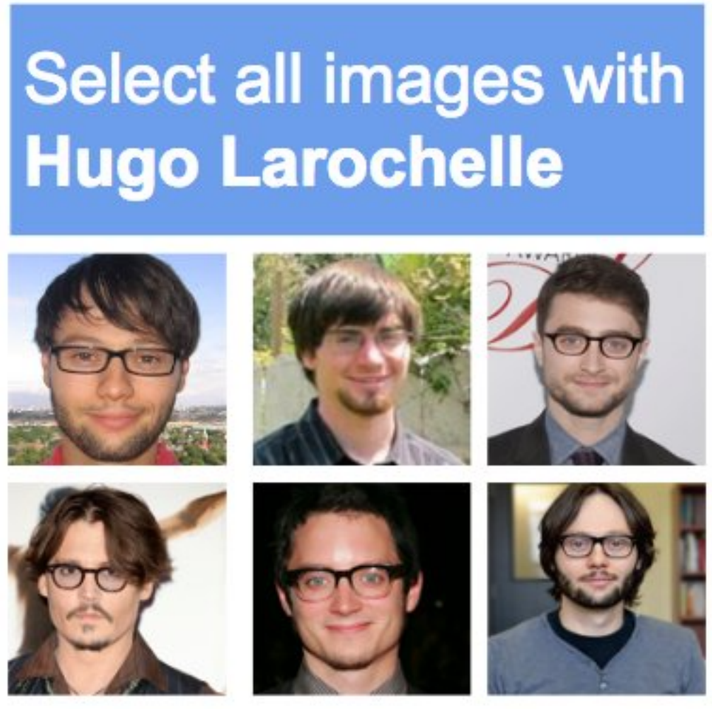
Joint work with Sachin Ravi

Optimization as a Model for Few-Shot Learning

Hugo Larochelle
Work done at Twitter
Google Brain

Joint work with Sachin Ravi

(challenge courtesy of Ian Goodfellow)

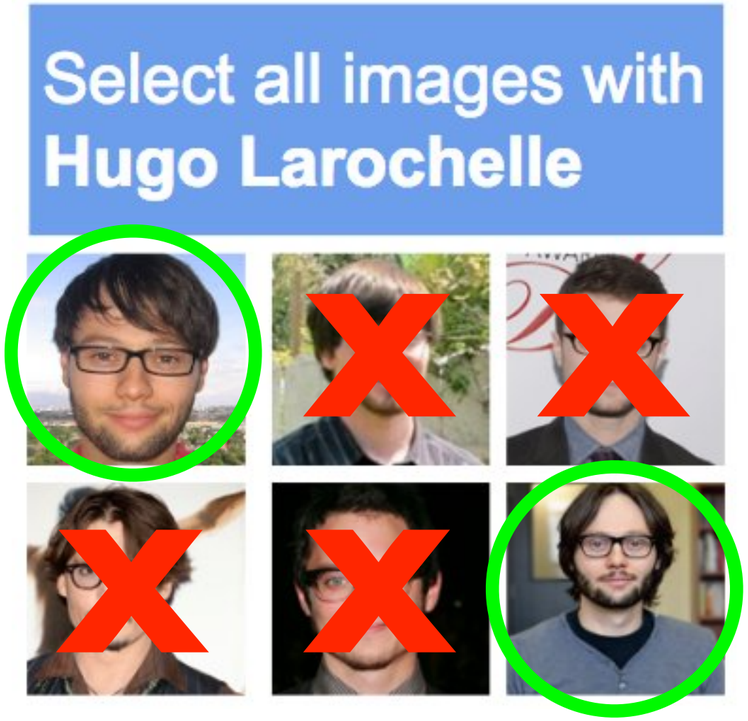


Optimization as a Model for Few-Shot Learning

Hugo Larochelle
Work done at Twitter
Google Brain

Joint work with Sachin Ravi

(challenge courtesy of Ian Goodfellow)




1
2
3
4
5
?
?
D_{train}
D_{test}

A RESEARCH AGENDA

- Deep learning successes have required a lot of labeled training data
  - collecting and labeling such data requires significant human labor
  - is that really how we'll solve AI ?

- Alternative solution : exploit other sources of data that are imperfect but plentiful
  - unlabeled data (unsupervised learning)
  - multimodal data (multimodal learning)
  - multidomain data (transfer learning, domain adaptation)

A RESEARCH AGENDA

- Deep learning successes have required a lot of labeled training data
  - collecting and labeling such data requires significant human labor
  - is that really how we'll solve AI ?

- Alternative solution : exploit other sources of data that are imperfect but plentiful
  - unlabeled data (unsupervised learning)
  - multimodal data (multimodal learning)
  - multidomain data (transfer learning, domain adaptation)

A RESEARCH AGENDA

- Let's attack directly the problem of **few-shot learning**
  - we want to design a learning algorithm A that outputs a good parameters $\boldsymbol{\theta}$ of a model M, when fed a small dataset $D_{train}=\{(\mathbf{X}_t, \mathbf{Y}_t)\}_{t=1}^{T}$

- Idea: let's *learn* that algorithm A, end-to-end
  - this is known as **meta-learning** or **learning to learn**

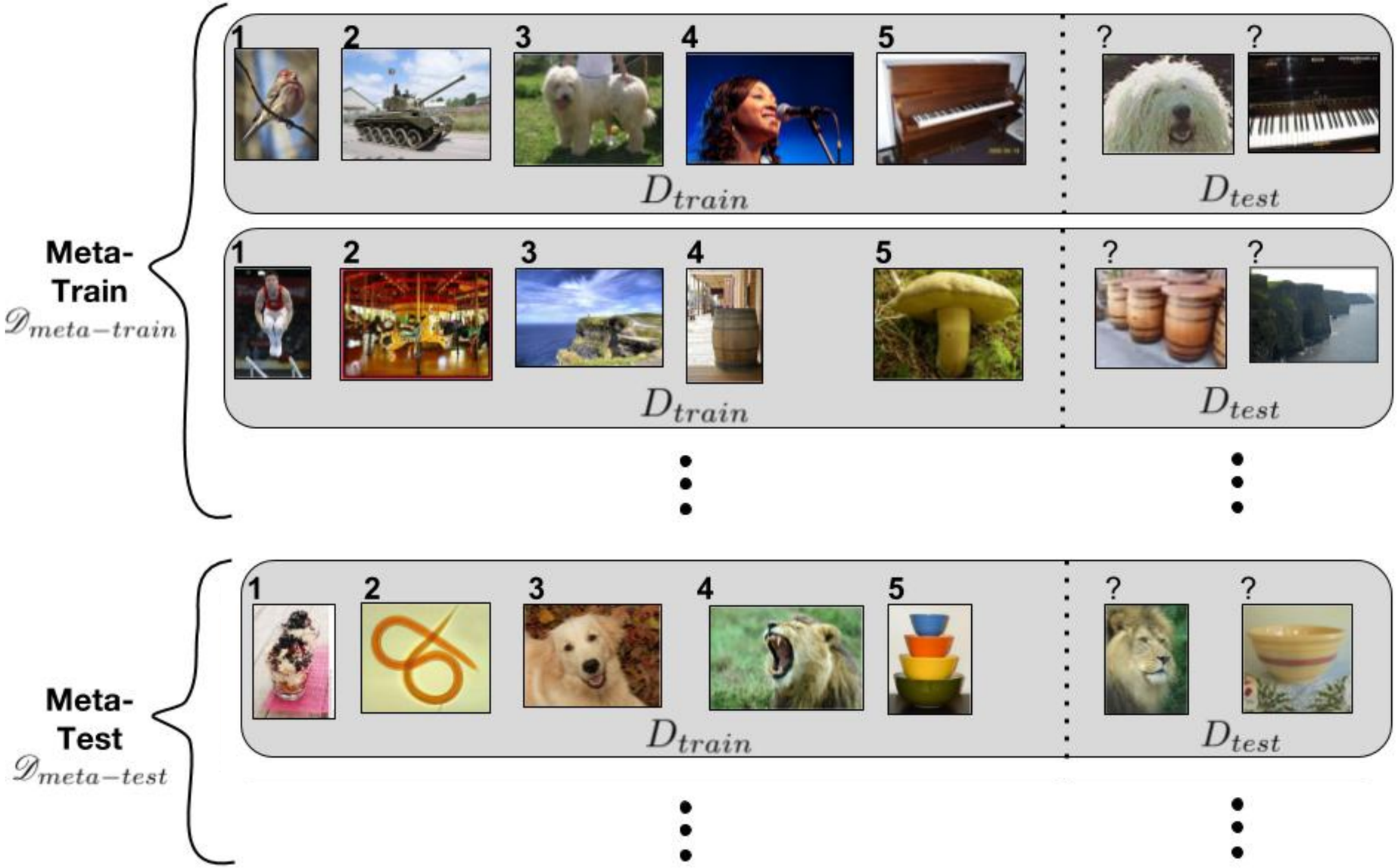
META-LEARNING

- Learning algorithm A
  - *input*: training set $D_{train}=\{(\mathbf{X}_t, \mathbf{Y}_t)\}$
  - *output*: parameters $\boldsymbol{\theta}$ model M (the **learner**)
  - *objective*: good performance on test set $D_{test}=(\mathbf{X}, \mathbf{Y})$

- Meta-learning algorithm
  - *input*: meta-training set $\mathscr{D}_{meta-train}=\{(D_{train}^{(n)}, D_{test}^{(n)})\}_{n=1}^{N}$
  - *output*: parameters $\boldsymbol{\Theta}$ algorithm A (the **meta-learner**)
  - *objective*: good performance on meta-test set $\mathscr{D}_{meta-test}=(D_{train}, D_{test})$

META-LEARNING

META-LEARNING



A META-LEARNING MODEL

- How to parametrize learning algorithms?
  - we take inspiration from the gradient descent algorithm:

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

  - we parametrize this update similarly to LSTM state updates:

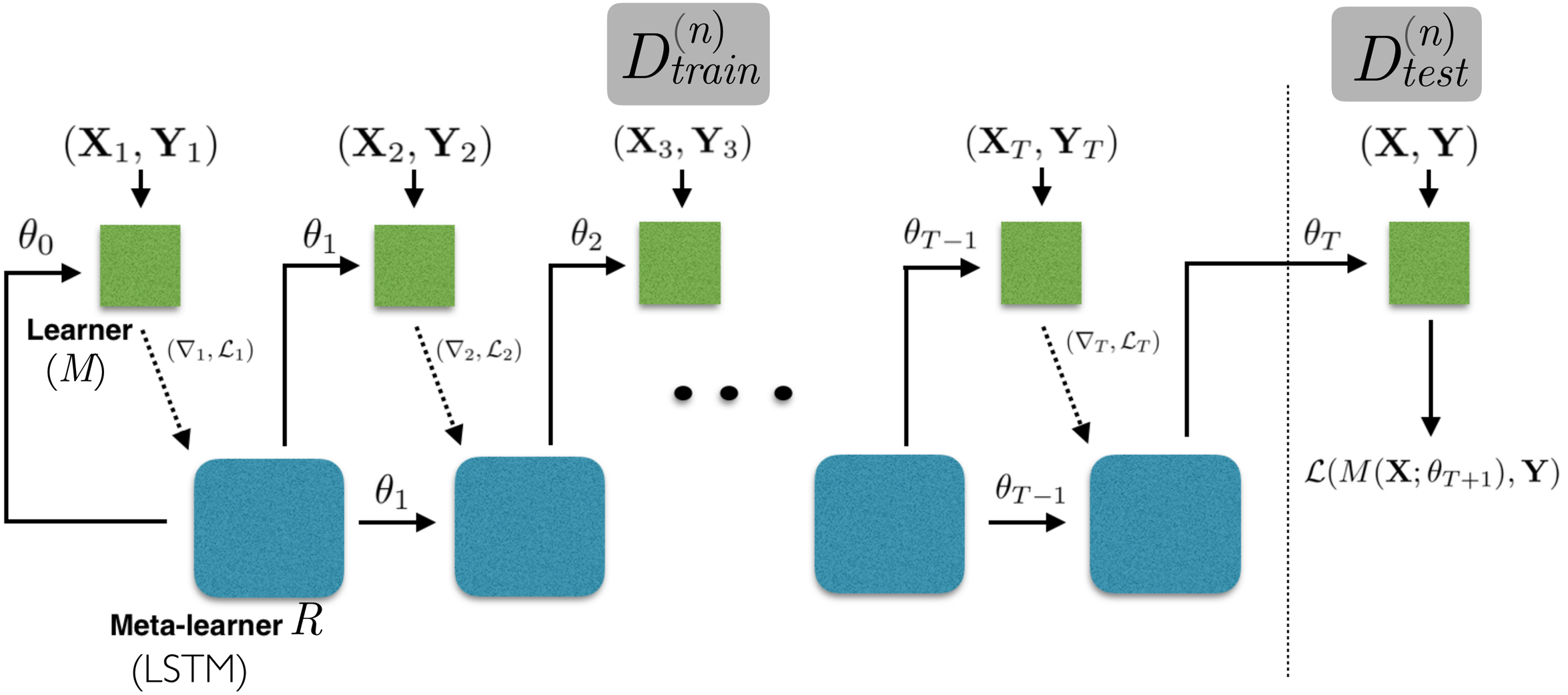
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

    - state c_t is model M's parameter space
    - state update $\tilde{c}_t$ is the negative gradient
    - f_t and i_t are LSTM gates:

$$i_t = \sigma\left(\mathbf{W}_I \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}\right] + \mathbf{b}_I\right)$$

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

META-LEARNING UPDATES



TO SUM UP

- We use our meta-learning LSTM to model parameter dynamics during training
  - LSTM parameters are shared across M's parameters (i.e. treated like a large minibatch)
  - learns c_0, which is like learning M's initialization
- It is trained to produce parameters that have low loss *on the corresponding test set*
  - possible thanks to backprop (though we don't ignore gradients through the inputs of the LSTM)
- Inputs to meta-learning LSTM are the loss, the parameter and its loss gradient
  - we use the preprocessing proposed by Andrychowicz et al. (2016)
- Model M uses batch normalization
  - we are careful to avoid "leakage" between meta-train / meta-validation / meta-test sets

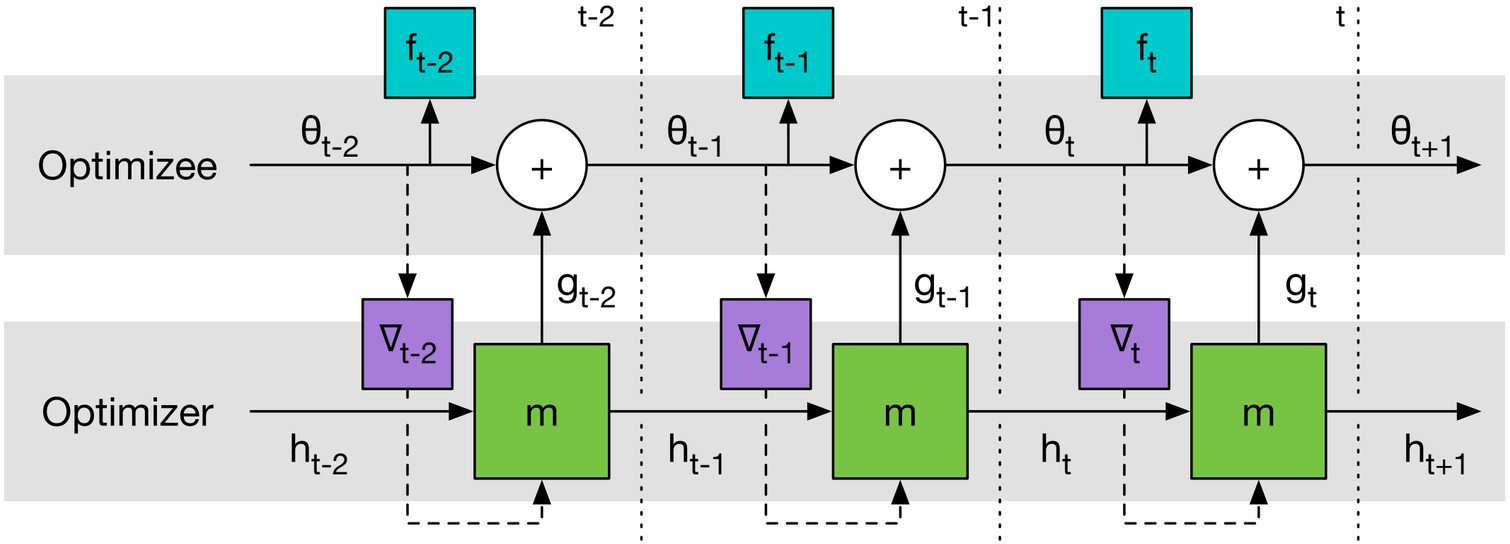
RELATED WORK: META-LEARNING

- Early work on learning an update rule
  - Learning a synaptic learning rule (1990)
    Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier
  - On the search for new learning rules for ANNs (1995)
    Samy Bengio, Yoshua Bengio, and Jocelyn Cloutier
- Early work on recurrent networks modifying their weights
  - Learning to control fast-weight memories: An alternative to dynamic recurrent networks (1992)
    Jürgen Schmidhuber
  - A neural network that embeds its own meta-levels (1993)
    Jürgen Schmidhuber

[see related work section of Learning to learn by gradient descent by gradient descent (2016)]

RELATED WORK: META-LEARNING

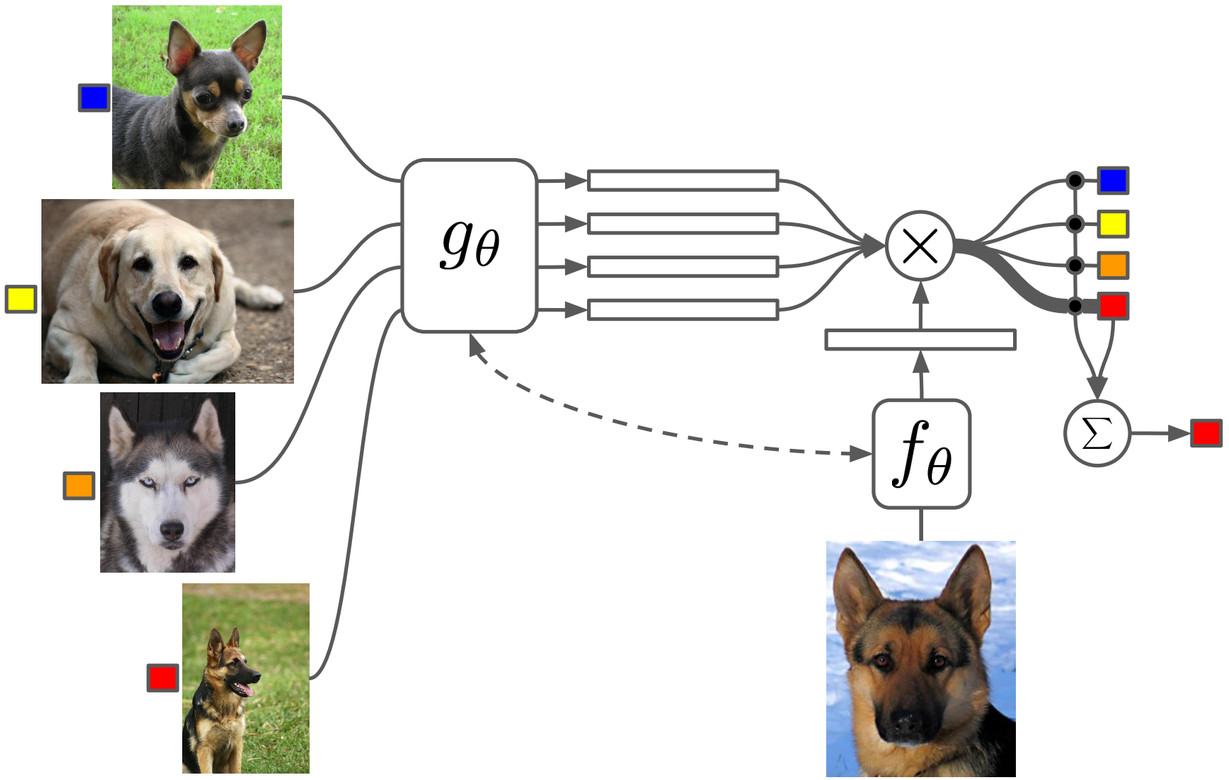
- Training a recurrent neural network to optimize
  - outputs update, so can decide to do something else than gradient descent



- Learning to learn by gradient descent by gradient descent (2016)
  Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas
- Learning to learn using gradient descent (2001)
  Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell

RELATED WORK: FEW-SHOT LEARNING

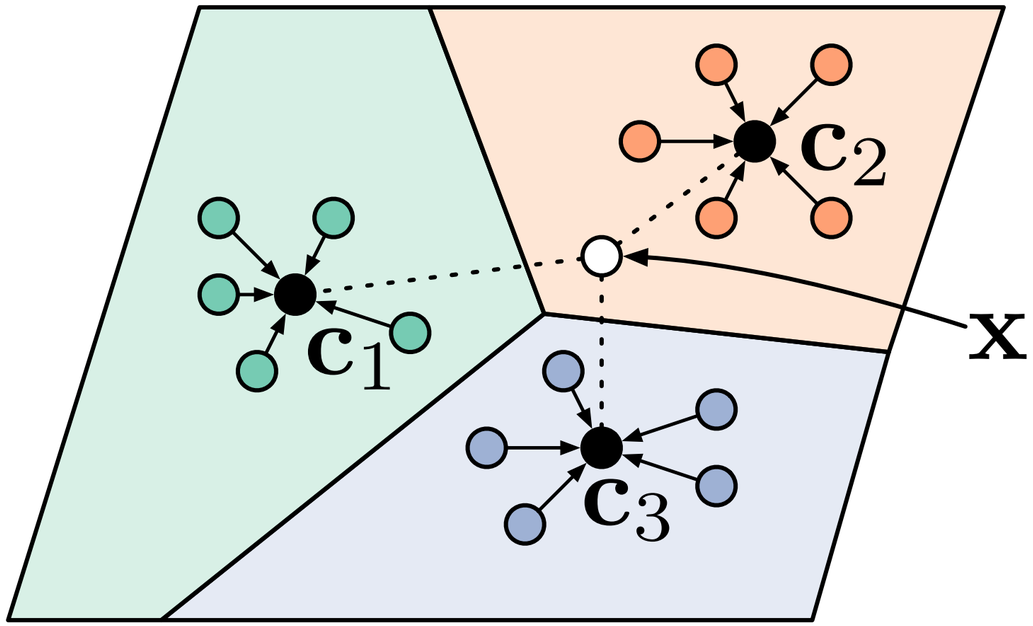
- Training a "**pattern matcher**" to optimize each episode's test set performance
  - no notion of learning an update rule



- Matching networks for one shot learning (2016)
  Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra

RELATED WORK: FEW-SHOT LEARNING

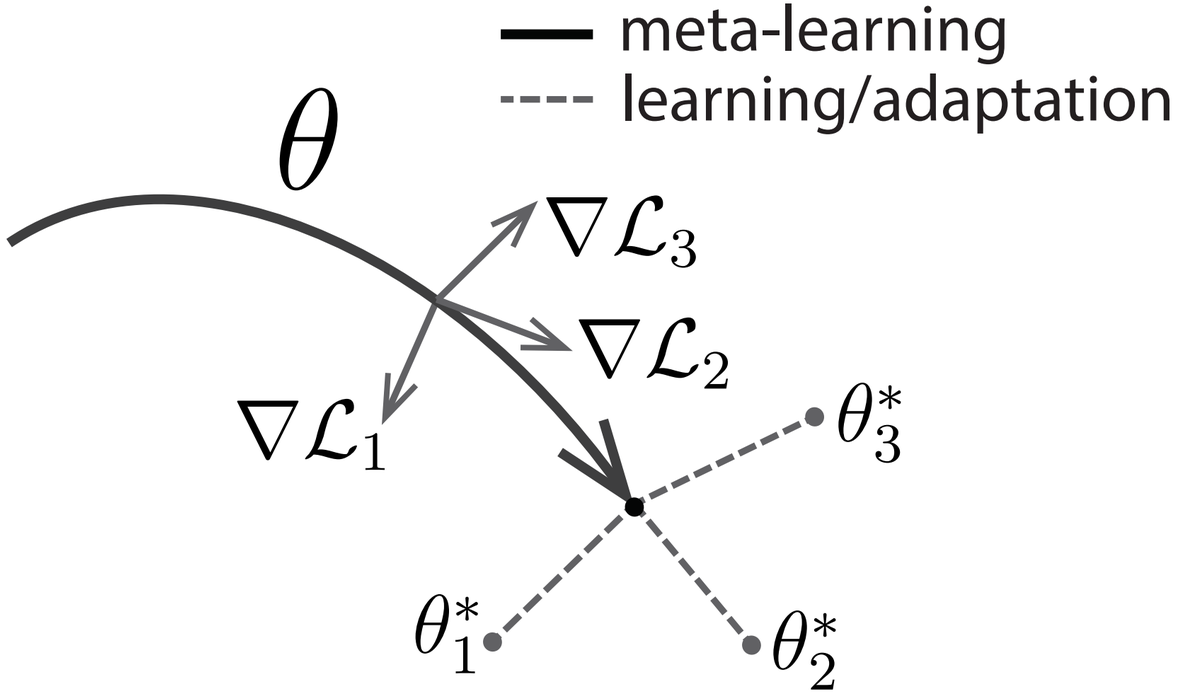
- Training a "**prototype extractor**" to optimize each episode's test set performance
  - no notion of learning an update rule



- Prototypical Networks for Few-shot Learning (2016)
  Jake Snell, Kevin Swersky and Richard Zemel

RELATED WORK: FEW-SHOT LEARNING

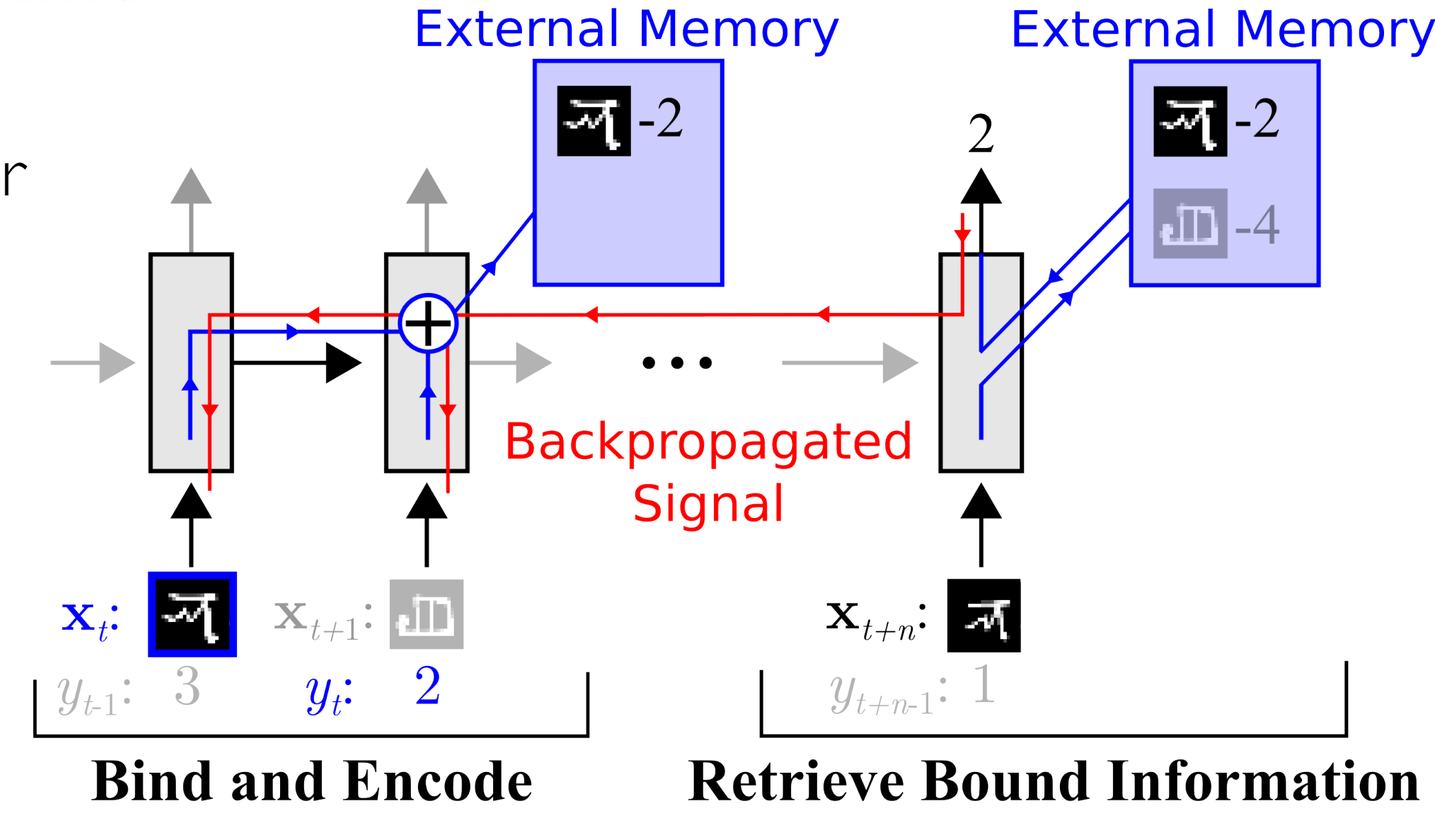
- Training a "**initialization+fine-tuning**" procedure that's based on a known update (e.g. ADAM)
  - much simpler than a meta-LSTM, yet works quite well!



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

RELATED WORK: FEW-SHOT LEARNING

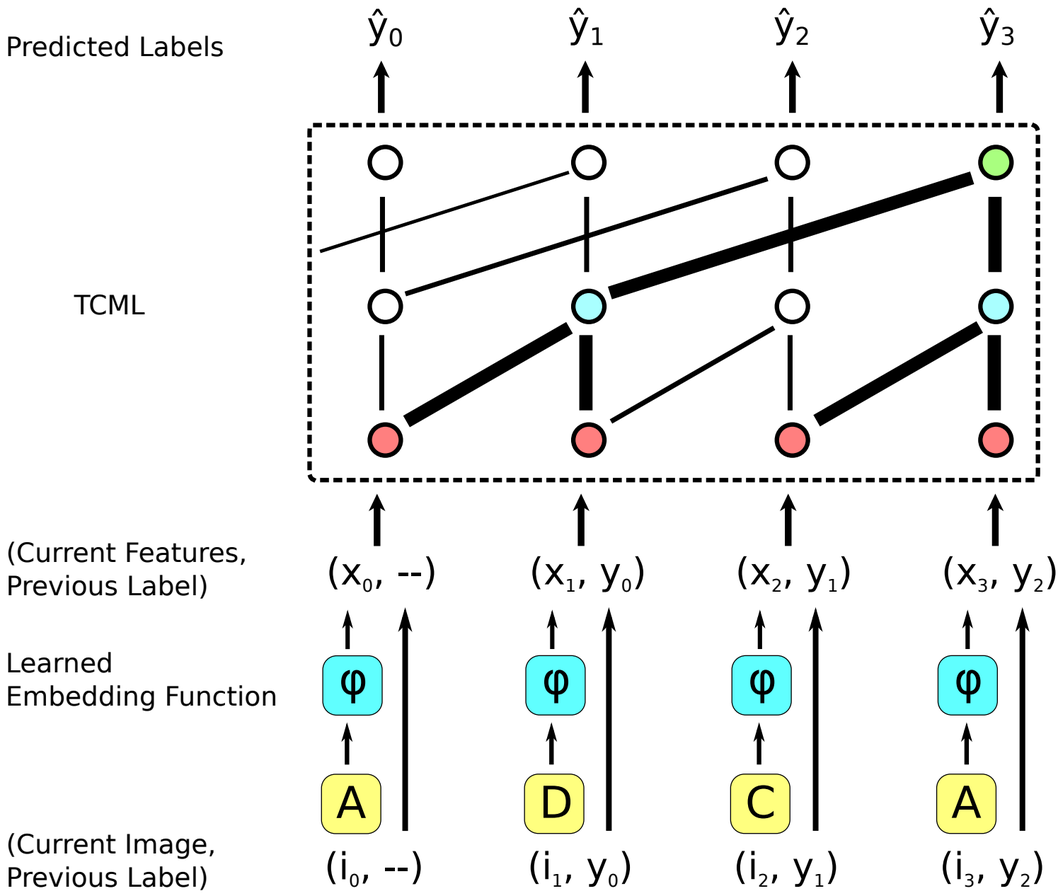
- Training a **neural Turing machine** to learn
  - no notion of gradient on learner



- One-shot learning with memory-augmented neural networks (2016)
  Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap

RELATED WORK: FEW-SHOT LEARNING

- Training a **convolutional network** to learn



- Meta-Learning with Temporal Convolutions (2017)
  Nikhil Mishra, Mostafa Rohaninejad, Xi Chen and Pieter Abbeel

EXPERIMENT

- Mini-ImageNet
  - random subset of 100 classes (64 training, 16 validation, 20 testing)
  - random sets D_{train} are generated by randomly picking 5 classes from class subset
  - model M is a small 4-layers CNN, meta-learner LSTM has 2 layers

Model	**5-class**	
	1-shot	5-shot
Baseline-finetune	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
Baseline-nearest-neighbor	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
Matching Network	$\mathbf{43.40 \pm 0.78\%}$	$51.09 \pm 0.71\%$
Matching Network FCE	**43.56% ± 0.84%**	$55.31\% \pm 0.73\%$
Meta-Learner LSTM (OURS)	**43.44% ± 0.77%**	**60.60% ± 0.71%**

EXPERIMENT
(updated)

- Mini-ImageNet
  - random subset of 100 classes (64 training, 16 validation, 20 testing)
  - random sets D_{train} are generated by randomly picking 5 classes from class subset
  - model M is a small 4-layers CNN, meta-learner LSTM has 2 layers

Model	**5-class**	
	1-shot	5-shot
Prototypical Nets (Snell et al.)	49.42% ± 0.78%	**68.20% ± 0.66%**
MAML (Finn et al.)	48.70% ± 1.84%	63.10% ± 0.92%
TCML (Mishra et al.)	**56.48% ± 0.99%**	61.22% ± 0.98%
Matching Network FCE	43.56% ± 0.84%	55.31% ± 0.73%
Meta-Learner LSTM (OURS)	43.44% ± 0.77%	60.60% ± 0.71%

DISCUSSION

- How to scale up to a variable number of classes / examples
  - we need an "ImageNet transposed"
- How best to characterize / parametrize learning algorithms (i.e. meta-models)
  - inspiration from other optimization algorithms? other learning algorithms?
- How to apply beyond supervised learning
  - unsupervised learning, semi-supervised learning, active learning, domain adaptation?
- … meta-meta-learning ?

MERCI !