CS 4803 / 7643: Deep Learning

Topics:
  – Low-label ML Formulations

Zsolt Kira
Georgia Tech
Administrativia

• Projects!

• Project Check-in due **April 11^{th}**
  – Will be graded pass/fail, if fail then you can address the issues
  – Counts for 5 points of project score

• Poster due date moved to **April 23^{rd}** (last day of class)
  – No presentations

• Final submission due date **April 30^{th}**
Types of Learning

• Important note:
  – Your project should include doing something beyond just downloading open-source code and tuning hyper-parameters.
  – This can include:
    • implementation of additional approaches (if leveraging open-source code),
    • theoretical analysis, or
    • a thorough investigation of some phenomena.

• When using external resources, provide references to anything you used in the write-up!
But wait, there’s more!

• **Transfer Learning**
• Domain adaptation
• Semi-supervised learning
• Zero-shot learning
• One/Few-shot learning
• Meta-Learning
• Continual / Lifelong-learning
• Multi-modal learning
• Multi-task learning
• Active learning
• …
Transfer Learning

Inductive Transfer Learning

Labeled data are available in a source domain
Labeled data are available in a target domain

Self-taught Learning

No labeled data in a source domain

Transductive Transfer Learning

Labeled data are available only in a source domain
Source and target tasks are learnt simultaneously

Multi-task Learning

Case 2

Domain Adaptation

Assumption: different domains but single task

Sample Selection Bias / Covariance Shift

Assumption: single domain and single task

Unsupervised Transfer Learning

No labeled data in both source and target domain

A Survey on Transfer Learning Sinno Jialin Pan and Qiang Yang Fellow, IEEE
Taskonomy

Builds graph of transferability between computer vision tasks:

1. Collect dataset of 4 million input images and labels for 26 vision tasks
   a. Surface normal, Depth estimation, Segmentation, 2D Keypoints, 3D pose estimation
2. Train convolutional autoencoder architecture for each tasks

http://taskonomy.stanford.edu/

Disentangling Task Transfer Learning, Amir R. Zamir, Alexander Sax*, William B. Shen*, Leonidas Guibas, Jitendra Malik, Silvio Savarese

Slide Credit: Camilo & Higuera
Taskonomy

Builds graph of transferability between computer vision tasks:

3. Transferability obtained by Analytic Hierarchy Process (from pairwise comparisons between all possible sources for each target task)
4. Final graph obtained by subgraph selection optimization (best performance from a limited set of source tasks): *transfer policy*

Empirical study on performance and data-efficiency gains from transfer using different datasets (Places and Imagenet)
Taskonomy
But wait, there’s more!

- Transfer Learning
- Domain adaptation
- Semi-supervised learning
- Zero-shot learning
- One/Few-shot learning
- Meta-Learning
- Continual / Lifelong-learning
- Multi-modal learning
- Multi-task learning
- Active learning
- …
Reducing Label Requirements

• Alternative solution to gathering more data: exploit other sources of data that are imperfect but plentiful
  – unlabeled data (unsupervised learning)
  – Multi-modal data (multimodal learning)
  – Multi-domain data (transfer learning, domain adaptation)
Few-Shot Learning

D_{train}  

D_{test}  

(C) Dhruv Batra & Zsolt Kira  
Slide Credit: Hugo Larochelle
Few-Shot Learning

People are good at it

Human-level concept learning through probabilistic program induction

Brenden M. Lake, Ruslan Salakhutdinov, Joshua B. Tenenbaum

Machines are getting better at it

(C) Dhruv Batra & Zsolt Kira

Slide Credit: Hugo Larochelle
Few-Shot Learning

• Let’s attack directly the problem of **few-shot learning**
  – we want to design a learning algorithm $A$ that outputs good parameters $\theta$ of a model $M$, when fed a small dataset $D_{\text{train}}=\{(x_i,y_i)\}_{i=1}$

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

• Rather than features, in few-shot learning, we aim at transferring the complete training of the model on new datasets (not just transferring the features or initialization)
  – ideally there should be no human involved in producing a model for new datasets
Prior Methods

• One-shot learning has been studied before
  – One-Shot learning of object categories (2006) Fei-Fei Li, Rob Fergus and Pietro Perona
  – Knowledge transfer in learning to recognize visual objects classes (2004) Fei-Fei Li
  – Object classification from a single example utilizing class relevance pseudo-metrics (2004) Michael Fink

• These largely relied on hand-engineered features
  – with recent progress in end-to-end deep learning, we hope to learn a representation better suited for few-shot learning
Prior Meta-Learning Methods

• Early work on learning an update rule
  – Learning a synaptic learning rule (1990) *Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier*
  – The Evolution of Learning: An Experiment in Genetic Connectionism (1990) *David Chalmers*

• 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*
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: Meta-Learning

• Hyper-parameter optimization
  – idea of learning the learning rates and the initialization conditions

• Gradient-based hyperparameter optimization through reversible learning (2015) *Dougal Maclourin, David Duvenaud, and Ryan P. Adams*
Related Work: Meta-Learning

- AutoML (Bayesian optimization, reinforcement learning)

- Neural Architecture Search with Reinforcement Learning (2017) *Barret Zoph and Quoc Le*
Meta-Learning

• Learning algorithm $A$
  – *input*: training set $D_{train} = \{ (x_i, y_i) \}$
  – *output*: parameters $\theta$ model $M$ (the learner)
  – *objective*: good performance on test set $D_{test} = \{ (x'_i, y'_i) \}$

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

\( D_{\text{train}} : D_{\text{test}} \)

\( \text{episode} \)

\( D_{\text{train}} \) \( D_{\text{test}} \)

\( D_{\text{train}} \) \( D_{\text{test}} \)

\( D_{\text{train}} \) \( D_{\text{test}} \)

\( D_{\text{train}} \) \( D_{\text{test}} \)

(C) Dhruv Batra & Zsolt Kira

Slide Credit: Hugo Larochelle
Meta-Learning

\[ D_{\text{train}} \quad D_{\text{test}} \]

episode

\[ D_{\text{train}} \quad D_{\text{test}} \]

Meta-Train

\[ D_{\text{meta-train}} \]

Meta-learner (A)

Meta-Test

\[ D_{\text{meta-test}} \]
Meta-Learning
Meta-Learning
Meta-Learning
Meta-Learning
Meta-Learning Nomenclature

Episode

- Training set
- Test set
- Meta-training set
- Meta-test set

Support set
Query set
Training set
Test set
Meta-Learning Nomenclature

• Assuming a probabilistic model $M$ over labels, the cost per episode can become

$$C(\Theta; D_{train}, D_{test}) = \frac{1}{|D_{test}|} \sum_{(x'_i, y'_i) \in D_{test}} - \log p_{\Theta}(y'_i | x'_i, D_{train})$$

• Depending on the choice of meta-learner, $p_{\Theta}(y | x, D_{train})$ will take a different form
Meta-Learner

• 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), MAML (Finn et al. 2017)
  – Derive it from a black box neural network
    • MANN (Santoro et al. 2016)
    • SNAIL (Mishra et al. 2018)
Meta-Learner

• 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), MAML (Finn et al. 2017)
  – Derive it from a black box neural network
    • MANN (Santoro et al. 2016)
    • SNAIL (Mishra et al. 2018)
Matching Networks

• Training a “pattern matcher”

\[ \hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \]

\[ a(\hat{x}, x_i) = \frac{e^{c(f(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}} \]

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

• Training a "prototype extractor"

\[
p_\phi(y = k \mid x) = \frac{\exp(-d(f_\phi(x), c_k))}{\sum_k \exp(-d(f_\phi(x), c_k'))}
\]

\[
c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i)
\]

\[
S_k = \{(x_i, y_i) \mid y_i = k, (x_i, y_i) \in D_{train}\}
\]

\[\phi \equiv \Theta\]

• Prototypical Networks for Few-shot Learning (2017)
  Jake Snell, Kevin Swersky and Richard Zemel
Prototypical Networks

- Training a "**prototype extractor**"
  - distance function $d(\cdot, \cdot)$ can be anything (euclidean squared, negative cosine)
  - if distance is euclidean squared, equivalent to learning an embedding network $f_\phi(\cdot)$ such that a Gaussian classifier works well
  - prototype vectors are equivalent to output weights of a neural network
  - Snell et al. find that using more classes in the meta-training episodes compared to meta-testing works better

- Prototypical Networks for Few-shot Learning (2017)
  *Jake Snell, Kevin Swersky and Richard Zemel*
Meta-Learner LSTM

- Training a "gradient descent procedure" applied on some learner $M$
  - gradient descent starts from some initial parameters $\theta_0$ 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*
Meta-Learner LSTM

- Training a "gradient descent procedure" applied on some learner \( M \)
  
  - gradient descent starts from some initial parameters \( \theta_0 \) 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 \( \theta_t \)
  - state update \( \tilde{c}_t \) is the negative gradient \(-\nabla_{\theta_{t-1}} \mathcal{L}_t \)
  - \( f_t \) and \( i_t \) are LSTM gates:

  \[
  i_t = \sigma \left( W_I \cdot \left[ \nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1} \right] + b_I \right) 
  \]

  \[
  f_t = \sigma \left( W_F \cdot \left[ \nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1} \right] + b_F \right) 
  \]

- Optimization as a Model for Few-Shot Learning (2017)

  *Sachin Ravi and Hugo Larochelle*
Meta-Learner LSTM

• Training a “gradient descent procedure” applied on some learner $M$
  
  ‣ gradient descent starts from some initial parameters $\theta_0$ and then performs the following updates:

  $$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} L_t$$

  ‣ 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 $\theta_t$ — $c_0$ becomes a learned initialization
  - state update $\tilde{c}_t$ is the negative gradient $-\nabla_{\theta_{t-1}} L_t$

  - $f_t$ and $i_t$ are LSTM gates:

    $$i_t = \sigma \left( W_I \cdot [\nabla_{\theta_{t-1}} L_t, L_t, \theta_{t-1}, i_{t-1}] + b_I \right)$$  \text{adaptive learning rate}

    $$f_t = \sigma \left( W_F \cdot [\nabla_{\theta_{t-1}} L_t, L_t, \theta_{t-1}, f_{t-1}] + b_F \right)$$  \text{adaptive weight decay}

• Optimization as a Model for Few-Shot Learning (2017)
  
  Sachin Ravi and 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-Learning Algorithm

Algorithm 1 Train Meta-Learner

Input: Meta-training set $\mathcal{D}_{meta\text{--}train}$, Learner $M$ with parameters $\theta$, Meta-Learner $R$ with parameters $\Theta$.

1: $\Theta_0 \leftarrow$ random initialization
2: 
3: for $d = 1, n$ do
4:     $D_{train}, D_{test} \leftarrow$ random dataset from $\mathcal{D}_{meta\text{--}train}$  
5:     $\theta_0 \leftarrow c_0$  
6:     $\Theta_d \leftarrow \Theta_{d-1}$  
7:     for $t = 1, T$ do
8:         $X_t, Y_t \leftarrow$ random batch from $D_{train}$  
9:         $\mathcal{L}_t \leftarrow \mathcal{L}(M(X_t; \theta_{t-1}), Y_t)$  
10:        $c_t \leftarrow R((\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t); \Theta_{d-1})$  
11:        $\theta_t \leftarrow c_t$  
12:     end for
13: 
14:     $X, Y \leftarrow D_{test}$
15:     $\mathcal{L}_{test} \leftarrow \mathcal{L}(M(X; \theta_T), Y)$  
16:     Update $\Theta_d$ using $\nabla_{\Theta_{d-1}} \mathcal{L}_{test}$  
17: 
18: end for
Meta-Learner LSTM

• Training a "gradient descent procedure" applied on some learner $M$
  
  ‣ LSTM parameters are shared across $M$’s parameters (i.e. treated like a large minibatch)
  
  ‣ can ignore (stop) gradients through the inputs of the LSTM
  
  ‣ gradient (and loss) inputs to the Meta-LSTM preprocessed as proposed by Andrychowicz et al. (2016)

\[
\nabla^k \rightarrow \begin{cases} 
\left(\frac{\log(|\nabla|)}{p}, \text{sgn}(\nabla)\right) & \text{if } |\nabla| \geq e^{-p} \\
(-1, e^p \nabla) & \text{otherwise}
\end{cases}
\]

  
  ‣ we are careful to avoid “leakage” from batchnorm statistics between meta-train / meta-test sets
  (sometimes referred to as the “transductive setting”)

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

- **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
  - better results are also reported by the so-called *bias transformation* architecture (One-Shot Visual Imitation Learning via Meta-Learning, Finn et al. 2017)
    - concatenates to one of the layers a trainable parameter vector, for instance to the input layer $[\mathbf{x}_i, \theta_b]$
    - decouples the updates of the bias and weights of that layer
    - with it, can be shown that even a single gradient descent update yields a universal approximator over functions mapping $D_{\text{train}}$ and $\mathbf{x}$ to any label $y$, for a sufficiently deep ReLU network and certain losses (Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm, Finn and Levine, 2018)

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

A general recipe:

\[
\theta \leftarrow \theta - \beta \sum_i \nabla_{\theta} L(\theta - \alpha \nabla_{\theta} L(\theta, D_{\text{train}}^i, D_{\text{test}}^i), D_{\text{test}}^i)
\]

"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”
Model-Agnostic Meta-Learning (MAML)

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

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

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

$$f_{\text{MAML}}(D_{\text{train}}, x) = f_{\theta'}(x)$$

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

Just another computation graph...
Can implement with any autodiff package (e.g., TensorFlow)
Comparison

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...)

(C) Dhruv Batra & Zsolt Kira
Slide Credit: Sergey Levine
Meta-Learner

• 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), MAML (Finn et al. 2017)
  – Derive it from a black box neural network
    • MANN (Santoro et al. 2016)
    • SNAIL (Mishra et al. 2018)
Black-Box Meta-Learner

- Frame meta-learning as sequence labeling with correct labels as delayed inputs

![Diagram showing sequence labeling with delayed inputs and reset memory.](Figure from Santoro et al. 2016)

- Learning to learn using gradient descent (2001)
  Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell
Memory-Augmented Neural Network

- Training a **neural Turing machine** to learn a learning algorithm

![Diagram of Memory-Augmented Neural Network]

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

- Mini-ImageNet (split used in Ravi & Larochelle, 2017)
  - 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

<table>
<thead>
<tr>
<th>Model</th>
<th>5-class</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>Baseline-finetune</td>
<td>28.86 ± 0.54%</td>
<td>49.79 ± 0.79%</td>
</tr>
<tr>
<td>Baseline-nearest-neighbor</td>
<td>41.08 ± 0.70%</td>
<td>51.04 ± 0.65%</td>
</tr>
<tr>
<td>Matching Network</td>
<td>43.40 ± 0.78%</td>
<td>51.09 ± 0.71%</td>
</tr>
<tr>
<td>Matching Network FCE</td>
<td>43.56% ± 0.84%</td>
<td>55.31% ± 0.73%</td>
</tr>
<tr>
<td>Meta-Learner LSTM (OURS)</td>
<td>43.44% ± 0.77%</td>
<td>60.60% ± 0.71%</td>
</tr>
</tbody>
</table>
Experiments

- **Mini-ImageNet** (split used in Ravi & Larochelle, 2017)
  - 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

<table>
<thead>
<tr>
<th>Model</th>
<th>5-class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-shot</td>
</tr>
<tr>
<td><strong>Prototypical Nets</strong> (Snell et al.)</td>
<td>49.42% ± 0.78%</td>
</tr>
<tr>
<td>MAML (Finn et al.)</td>
<td>48.70% ± 1.84%</td>
</tr>
<tr>
<td>SNAIL (Mishra et al.)</td>
<td>55.71% ± 0.99%</td>
</tr>
<tr>
<td>Matching Network FCE</td>
<td>43.56% ± 0.84%</td>
</tr>
<tr>
<td>Meta-Learner LSTM (OURS)</td>
<td>43.44% ± 0.77%</td>
</tr>
</tbody>
</table>
Extensions and Variations

- Semi-supervised learning (with distractors)
  - assign soft-labels to unlabeled examples
  - use soft-labels to refine prototypes

\[
\tilde{p}_c = \frac{\sum_i h(x_i) z_{i,c} + \sum_j h(\tilde{x}_j) \tilde{z}_{j,c} m_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c} m_{j,c}}
\]

  Ren, Triantafillou, Ravi, Snell, Swersky, Tenenbaum, Larochelle and Zemel
But beware

A Closer Look at Few-shot Classification,
Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, Jia-Bin Huang
Table 2: Few-shot classification results for both the mini-ImageNet and CUB datasets. The Baseline++ consistently improves the Baseline model by a large margin and is competitive with the state-of-the-art meta-learning methods. All experiments are from 5-way classification with a Conv-4 backbone and data augmentation.

<table>
<thead>
<tr>
<th>Method</th>
<th>CUB 1-shot</th>
<th>CUB 5-shot</th>
<th>mini-ImageNet 1-shot</th>
<th>mini-ImageNet 5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>47.12 ± 0.74</td>
<td>64.16 ± 0.71</td>
<td>42.11 ± 0.71</td>
<td>62.53 ± 0.69</td>
</tr>
<tr>
<td>Baseline++</td>
<td>60.53 ± 0.83</td>
<td>79.34 ± 0.61</td>
<td>48.24 ± 0.75</td>
<td>66.43 ± 0.63</td>
</tr>
<tr>
<td>MatchingNet Vinyals et al. (2016)</td>
<td>61.16 ± 0.89</td>
<td>72.86 ± 0.70</td>
<td>48.14 ± 0.78</td>
<td>63.48 ± 0.66</td>
</tr>
<tr>
<td>ProtoNet Snell et al. (2017)</td>
<td>51.31 ± 0.91</td>
<td>70.77 ± 0.69</td>
<td>44.42 ± 0.84</td>
<td>64.24 ± 0.72</td>
</tr>
<tr>
<td>MAML Finn et al. (2017)</td>
<td>55.92 ± 0.95</td>
<td>72.09 ± 0.76</td>
<td>46.47 ± 0.82</td>
<td>62.71 ± 0.71</td>
</tr>
<tr>
<td>RelationNet Sung et al. (2018)</td>
<td>62.45 ± 0.98</td>
<td>76.11 ± 0.69</td>
<td>49.31 ± 0.85</td>
<td>66.60 ± 0.69</td>
</tr>
</tbody>
</table>
Distribution Shift

• What if there is a distribution shift (cross-domain)?

• **Lesson**: Methods that are successful *within-domain* might be worse *across domains*!

<table>
<thead>
<tr>
<th>mini-ImageNet → CUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
</tr>
<tr>
<td>Baseline++</td>
</tr>
<tr>
<td>MatchingNet</td>
</tr>
<tr>
<td>ProtoNet</td>
</tr>
<tr>
<td>MAML</td>
</tr>
<tr>
<td>RelationNet</td>
</tr>
</tbody>
</table>

Table 3: **5-shot accuracy under the cross-domain scenario with a ResNet-18 backbone.** Baseline outperforms all other methods under this scenario.
Distribution Shift

Let’s Talk about Meta-Overfitting

• Meta learning requires task distributions
• When there are too few meta-training tasks, we can meta-overfit
•Specifying task distributions is hard, especially for meta-RL!
• Can we propose tasks automatically?
Random Task Proposals

- Use randomly initialize discriminators for reward functions
  \[ R(s, z) = \log p_D(z|s) \]
  - \( D \) \( \rightarrow \) randomly initialized network

- Important: Random functions over state space, **not** random policies
Does it Work?

2D Navigation

Cheetah

Ant

Meta-test performance with rewards

Discussions

• What is the right definition of distributions over problems?
  – varying number of classes / examples per class (meta-training vs. meta-testing) ?
  – semantic differences between meta-training vs. meta-testing classes ?
  – overlap in meta-training vs. meta-testing classes (see recent “low-shot” literature) ?

• Move from static to interactive learning
  – how should this impact how we generate episodes ?
  – meta-active learning ? (few successes so far)