Meta Learning
A Brief Introduction

Xiachong Feng

Outline

- Introduction to Meta Learning
- Types of Meta-Learning Models
- Papers:
  - 《Optimization as a model for few-shot learning》*ICLR2017*
  - 《Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks》*ICML2017*
  - 《Meta-Learning for Low-Resource Neural Machine Translation》*EMNLP2018*
- Conclusion

Meta-learning

Machine Learning
复杂分类效果差

Reinforcement learning
对于序列决策问题，单一深度学习无法解决（结合DL+RL）

Deep Learning
结合表示学习，基本上解决了一对一映射的问题

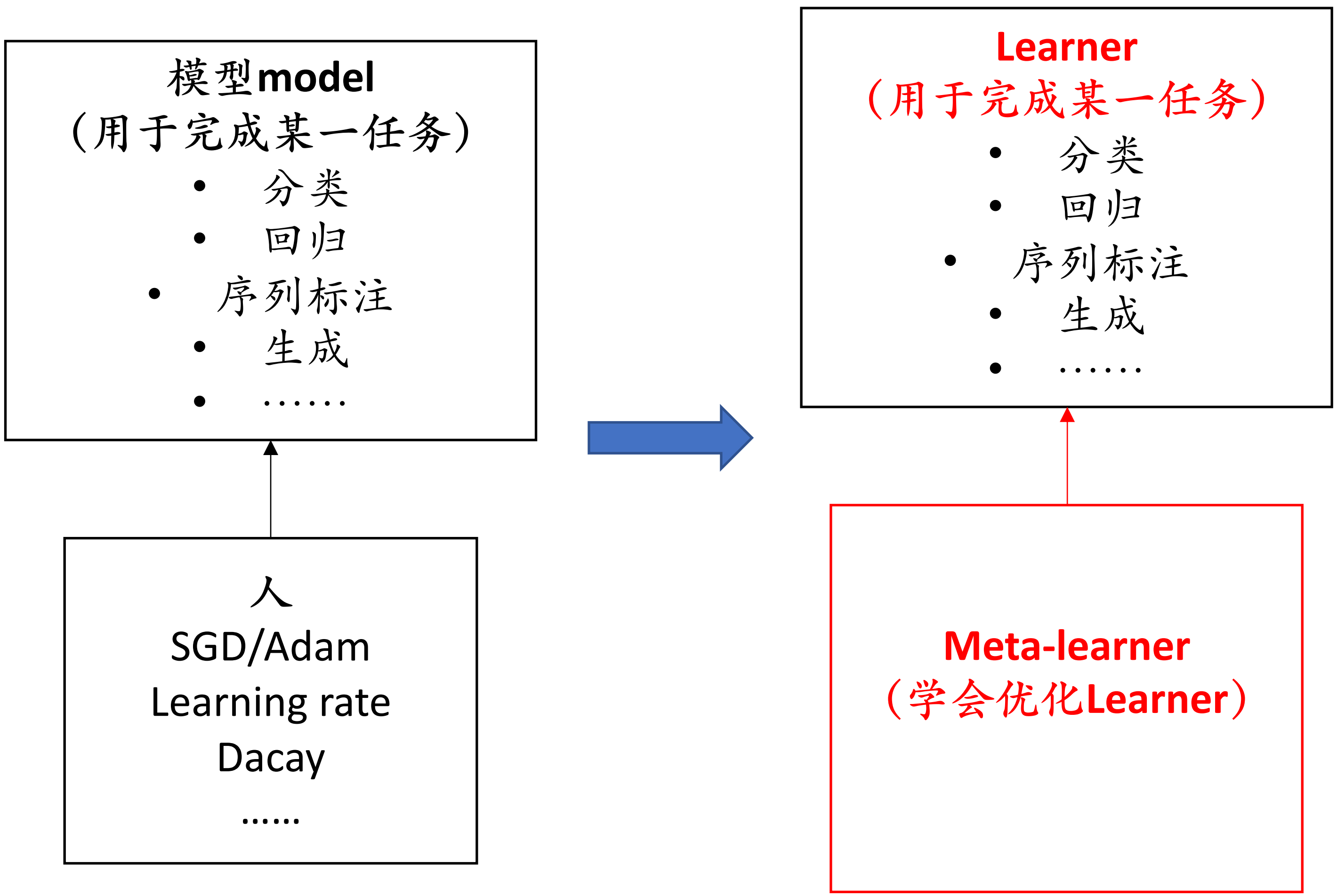
Meta Learning
之前依赖于巨量的训练，需要充分的利用以往的知识经验来指导新任务的学习

最前沿：百家争鸣的Meta Learning/Learning to learn https://zhuanlan.zhihu.com/p/28639662

Meta-learning

- Learning to learn（学会学习）
- 学会学习：拥有学习的能力。
- 举一个金庸武侠的例子：我们都知道，在金庸的武侠世界中，有各种各样的武功，不同的武功都不一样，有**内功**也有外功。那么里面的张无忌就特别厉害，因为他练成了**九阳神功**。有了九阳神功，张无忌**学习新的武功就特别快**，在电影倚天屠龙记之魔教教主中，张无忌分分钟学会了张三丰的太极拳打败了玄冥二老。**九阳神功就是一种学会学习的武功！**
- Meta learning就是AI中的九阳神功

Example



Machine or Deep learning

Meta learning

Types of Meta-Learning Models

- Humans learn following different methodologies tailored to specific circumstances.
- In the same way, not all meta-learning models follow the same techniques.
- **Types of Meta-Learning Models**
  1. Few Shots Meta-Learning
  2. Optimizer Meta-Learning
  3. Metric Meta-Learning
  4. Recurrent Model Meta-Learning
  5. Initializations Meta-Learning

Few Shots Meta-Learning

- Create models that can learn from minimalistic datasets mimicking --> *(learn from tiny data)*
- Papers
    - Optimization As A Model For Few Shot Learning （ICLR2017）
    - One-Shot Generalization in Deep Generative Models （ICML2016）
    - Meta-Learning with Memory-Augmented Neural Networks （ICML2016）

Optimizer Meta-Learning

- Task: Learning how to optimize a neural network to better accomplish a task.
- There is one network **(the meta-learner)** which learns to update another network **(the learner)** so that the learner effectively learns the task.
- Papers:
  - Learning to learn by gradient descent by gradient descent (NIPS 2016)
  - Learning to Optimize Neural Nets

Metric Meta-Learning

- To determine a metric space in which learning is particularly efficient. This approach can be seen as a subset of few shots meta-learning in which we used a learned metric space to evaluate the quality of learning with a few examples
- Papers:
  - Prototypical Networks for Few-shot Learning(NIPS2017)
  - Matching Networks for One Shot Learning(NIPS2016)
  - Siamese Neural Networks for One-shot Image Recognition
  - Learning to Learn: Meta-Critic Networks for Sample Efficient Learning

Recurrent Model Meta-Learning

- The meta-learner algorithm will train a RNN model will process a dataset sequentially and then process new inputs from the task
- Papers:
  - Meta-Learning with Memory-Augmented Neural Networks
  - Learning to reinforcement learn
  - RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning

Initializations Meta-Learning

- Optimized for an initial representation that can be effectively fine-tuned from a small number of examples
- Papers:
  - Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks （ICML 2017）
  - Meta-Learning for Low-Resource Neural Machine Translation （EMNLP2018）

Papers

Few Shots Meta-Learning、Recurrent Model Meta-Learning、Optimizer Meta-Learning、*Initializations Meta-Learning*、*Supervised Meta Learning*

Optimization As a Model For Few Shot Learning

(ICLR2017)

Modern Meta Learning

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

(ICML2017)

Meta Learning in NLP

Meta-Learning for Low-Resource Neural Machine Translation

(EMNLP2018)

Optimization As a Model For Few Shot Learning

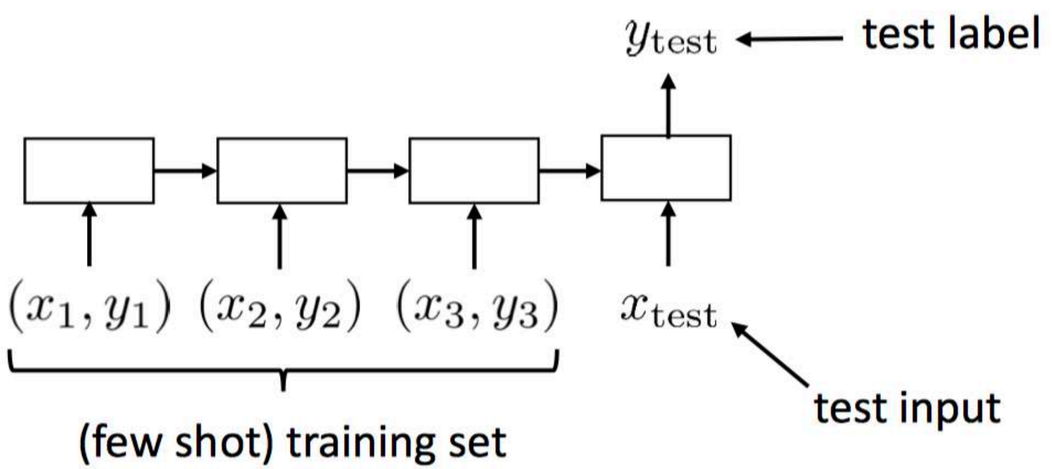
Twitter, Sachin Ravi, Hugo Larochelle

ICLR2017

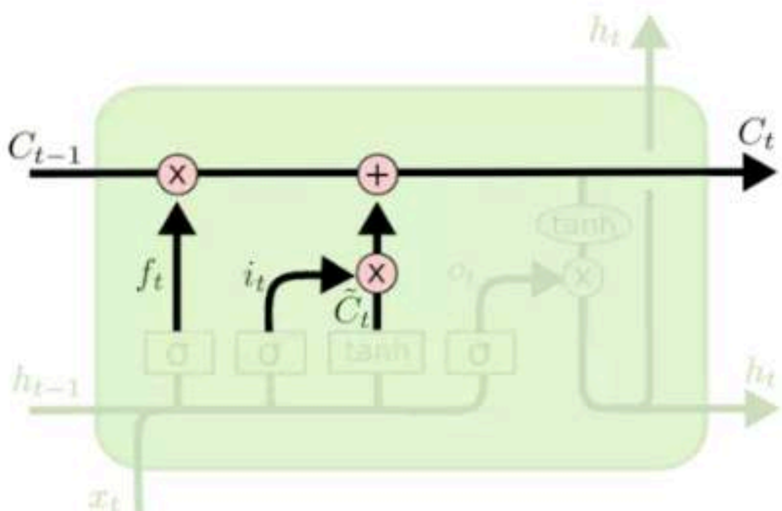
- ***Few Shots*** *Meta-Learning*
- ***Recurrent Model*** *Meta-Learning*
- ***Optimizer*** *Meta-Learning*
- ***Supervised*** *Meta Learning*
- ***Initializations*** *Meta-Learning*

Few Shots Learning

- Given a **tiny** labelled training set S, which has N examples, $S = \{(x_1, y_1), \dots (x_N, y_N),\}$
- **In classification problem:**
  - $K - shot$ Learning
  - N classes
  - K labelled examples(K is always less than 20)



LSTM-Cell state update



forgetting the things we decided to forget earlier

$$C_t = \overbrace{f_t * C_{t-1}} + \underbrace{i_t * \tilde{C}_t}$$

new cell state   old cell state

new candidate values

理解 LSTM 网络 https://www.jianshu.com/p/9dc9f41f0b29

Supervised learning

Optimizer
SGD
Adam
……

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

神经网络NN
（用于完成某一任务）
- 分类
- 回归
- 序列标注
- 生成
- ……

$$f(x) \rightarrow y$$

image　　label

Meta learning

- Meta-learning suggests framing the learning problem at two levels. *(Thrun, 1998; Schmidhuber et al., 1997)*
  - The first is quick acquisition of knowledge within each separate task presented. (Fast adaption)
  - This process is guided by the second, which involves slower extraction of information learned across all the tasks.(Learning)

Motivation

- Deep Learning has shown great success in a variety of tasks with large amounts of labeled data.
- Gradient-based optimization *(momentum, Adagrad, Adadelta and ADAM)* in high capacity classifiers requires many iterative steps over many examples to perform well.
- Start from a random initialization of its parameters.
- Perform poorly on few-shot learning tasks.

Is there an optimizer can finish the optimization task using just few examples?

Method

LSTM cell-state update :

$$\boxed{\begin{array}{l} c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ \\ \theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t \end{array}}$$

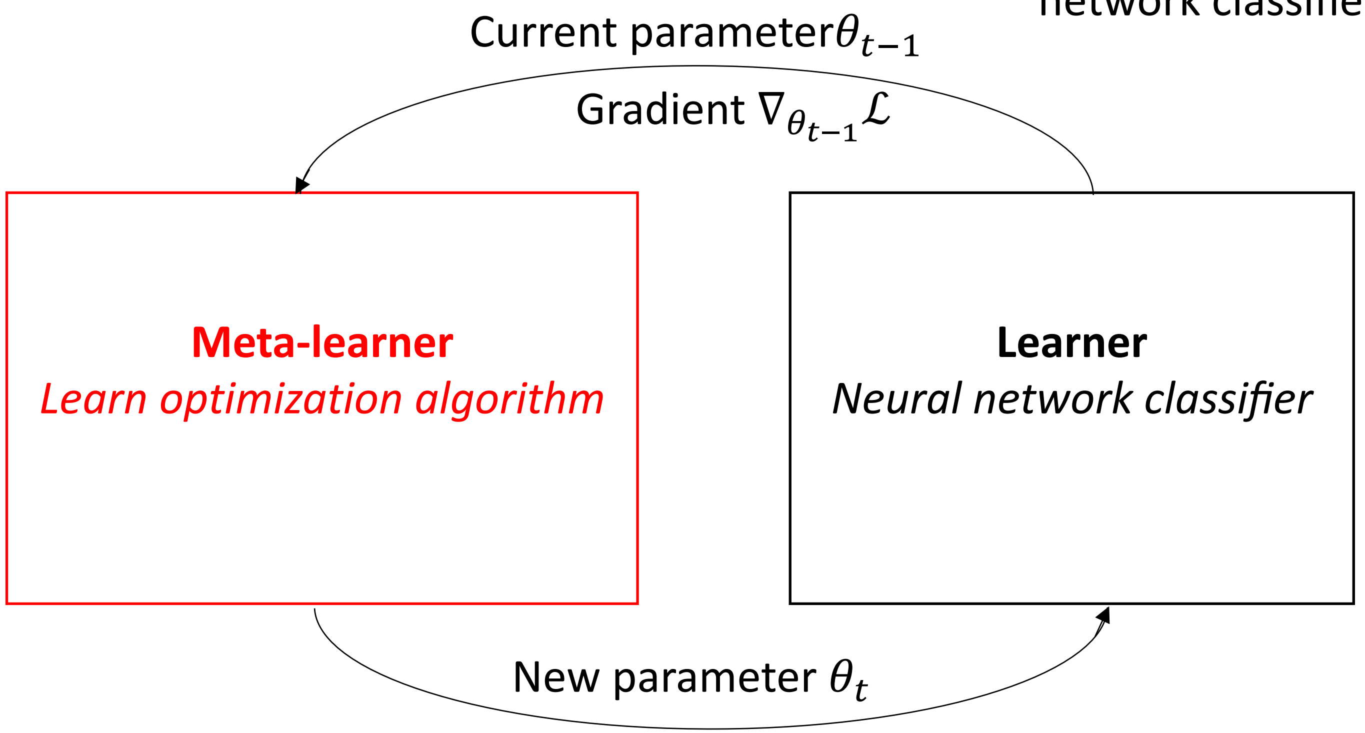
Gradient based update :

$$f_t = 1, c_{t-1} = \theta_{t-1}, i_t = \alpha_t,$$
$$\tilde{c}_t = -\nabla_{\theta_{t-1}} \mathcal{L}_t$$

Propose an **LSTM based meta-learner** model to learn the exact <u>optimization algorithm</u> used to train another **learner** neural network classifier in the few-shot regime.

Method

LSTM-based meta-learner optimizer that is trained to optimize a learner neural network classifier.



Gradient-based optimization: $\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t$

Meta-learner optimization: $\theta_t = \mathrm{metalearner}(\theta_{t-1}, \nabla_{\theta_{t-1}}\mathcal{L})$

knowing how to quickly optim the parameters

Model

$$\boxed{\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ \theta_t &= \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t \end{aligned}}$$

$$f_t = 1, c_{t-1} = \theta_{t-1}, i_t = \alpha_t,$$
$$\tilde{c}_t = -\nabla_{\theta_{t-1}} \mathcal{L}_t$$

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

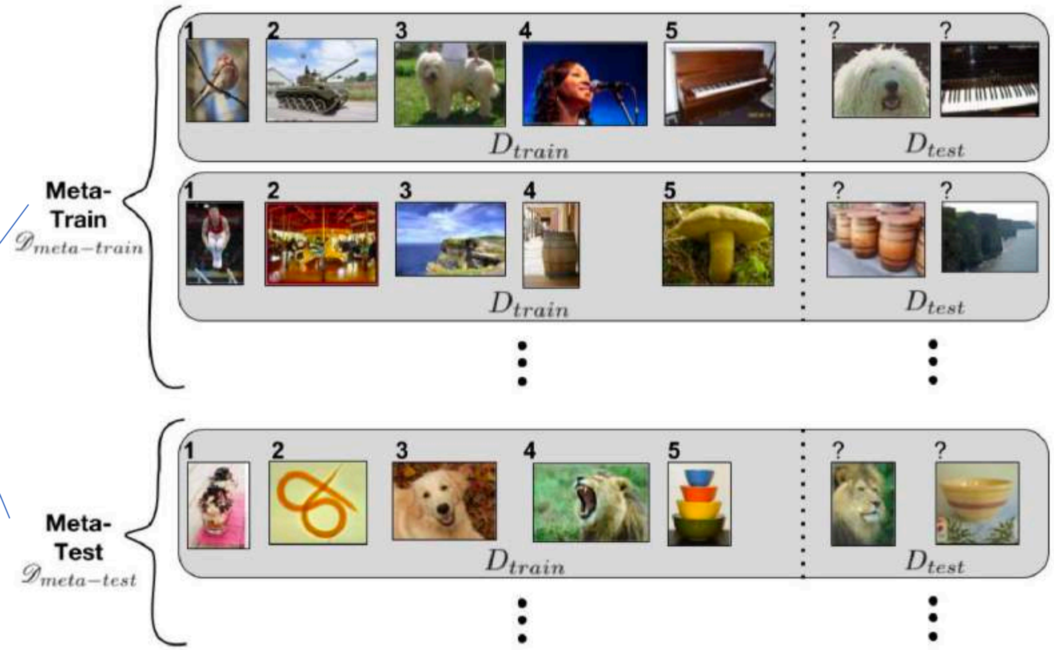
Given by learner

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

Given by learner

Task Description

episode

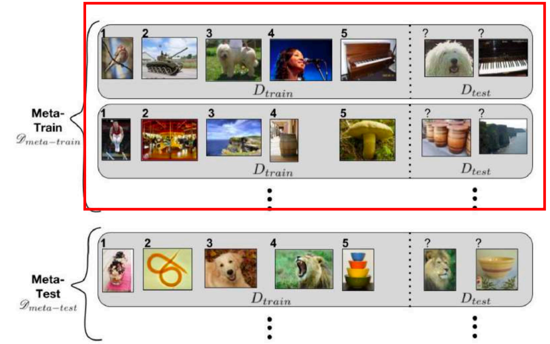


Used to train **learner**

Used to train **meta-learner**

Training

- Example: 5 classes, 1 shot learning
- $\mathcal{D}_{train}, \mathcal{D}_{test}$ ←Random dataset from $\mathcal{D}_{meta-train}$



Learner
Neural network classifier (θ_{t-1})

Loss $\mathcal{L}$
$\mathcal{L}_t \leftarrow \mathcal{L}(M(\mathbf{X}_t; \theta_{t-1}), \mathbf{Y}_t)$

Gradient $\nabla_{\theta_{t-1}}\mathcal{L}$

Loss $\mathcal{L}$
Current param θ_{t-1}
Gradient $\nabla_{\theta_{t-1}}\mathcal{L}$

Meta-learner
Learn optimization algorithm (Θ_{d-1})

$c_t \leftarrow R((\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t); \Theta_{d-1})$

Output of meta learner c_t

Learner Update

Output of meta learner c_t

Update learner
$\theta_t \leftarrow c_t$

Learner
Neural network classifier (θ_t)

Meta-Learner Update

Learner
Neural network classifier (θ_t)

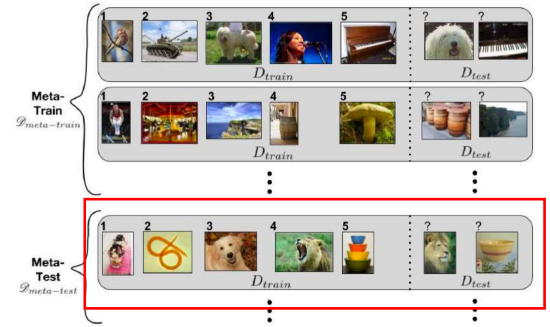
Loss $\mathcal{L}_{test}$

$$\Theta_d = \Theta_{d-1} - \alpha \nabla_{\Theta_{d-1}} \mathcal{L}_{test}$$

Initializations Meta-Learning

- Initial value of the cell state C_0
- Initial weights of the classifier θ_0
- $C_0 = \theta_0$
- Learning this initial value lets the meta-learner determine the optimal initial weights of the learner

Testing



- Example: 5 classes, 1 shot learning
- $\mathcal{D}_{train}, \mathcal{D}_{test}$ ←Random dataset from $\mathcal{D}_{meta-test}$

Learner
(Init with θ_0, Current θ_{t-1})

Loss $\mathcal{L}$
$\mathcal{L}_t \leftarrow \mathcal{L}(M(\mathbf{X}_t; \theta_{t-1}), \mathbf{Y}_t)$

Gradient $\nabla_{\theta_{t-1}}\mathcal{L}$

Loss $\mathcal{L}$
Current param θ_{t-1}
Gradient $\nabla_{\theta_{t-1}}\mathcal{L}$

Meta-learner
learn optimization algorithm(Θ)

$c_t \leftarrow R((\nabla_{\theta_{t-1}}\mathcal{L}_t, \mathcal{L}_t); \Theta_{d-1})$

Output of meta learner c_t

Learner Update

Output of meta learner c_t

Update learner
$\theta_t \leftarrow c_t$

Learner
Neural network classifier(θ_t)

Testing

Learner
Neural network classifier

Metric

Training

Learner Update

Meta-Learner Update

Algorithm 1 Train Meta-Learner

Input: Meta-training set $\mathscr{D}_{meta-train}$, Learner M with parameters θ, Meta-Learner R with parameters Θ.

$\Theta_0 \leftarrow$ random initialization

for $d = 1, n$ **do**

  $D_{train}, D_{test} \leftarrow$ random dataset from $\mathscr{D}_{meta-train}$

  $\theta_0 \leftarrow c_0$ ▷ Intialize learner parameters

  for $t = 1, T$ **do**

    $\mathbf{X}_t, \mathbf{Y}_t \leftarrow$ random batch from D_{train}

    $\mathcal{L}_t \leftarrow \mathcal{L}(M(\mathbf{X}_t; \theta_{t-1}), \mathbf{Y}_t)$ ▷ Get loss of learner on train batch

    $c_t \leftarrow R((\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t); \Theta_{d-1})$ ▷ Get output of meta-learner using Equation 2

    $\theta_t \leftarrow c_t$ ▷ Update learner parameters

  end for

  $\mathbf{X}, \mathbf{Y} \leftarrow D_{test}$

  $\mathcal{L}_{test} \leftarrow \mathcal{L}(M(\mathbf{X}; \theta_T), \mathbf{Y})$ ▷ Get loss of learner on test batch

  Update Θ_d using $\nabla_{\Theta_{d-1}} \mathcal{L}_{test}$ ▷ Update meta-learner parameters

end for

Trick

- **Parameter Sharing**
  - meta-learner to produce updates for deep neural networks, which consist of tens of thousands of parameters, to prevent an explosion of meta-learner parameters we need to employ some sort of **parameter sharing.**
- **Batch Normalization**
  - Speed up learning of deep neural networks by reducing internal covariate shift within the learner's hidden layers.

About this paper

- Few Shots Meta-Learning
  - K-shot image classification
- Recurrent Model Meta-Learning
  - Use LSTM cell state as optimizer
- Optimizer Meta-Learning
  - Meta-learner is an optimizer
- Supervised Meta Learning
  - Image classification task
- Initializations Meta-Learning
  - Learning this initial value lets the meta-learner determine the optimal initial weights of the learner

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

University of California, Berkeley
Chelsea Finn, Pieter Abbeel, Sergey Levine

ICML 2017

- ***Few Shots*** *Meta-Learning*
- ***Supervised*** *Meta Learning*
- ***Reinforcement*** *Meta Learning*
- ***Initializations*** *Meta-Learning*

Problem

- Prior meta-learning methods that **learn an update function or learning rule**
- **Expand the number of learned parameters**
- Place constraints on the model architecture
  - Recurrent model
  - Siamese network（孪生网络）

Motivation

- **Model-agnostic**
  - any model trained with gradient descent
  - a variety of different learning problems,
    - classification, regression, reinforcement learning.
- If the **internal representation** is suitable to many tasks, simply fine-tuning the parameters slightly can produce good results.
- Learning process can be viewed as **maximizing the sensitivity** of the loss functions of new tasks with respect to the parameters: when the sensitivity is high, <u>small local changes to the parameters can lead to large improvements in the task loss</u>.

Few shots meta learning

- The goal of few-shot meta-learning is to train a model that can **quickly adapt to a new task** using only a few data points and training iterations.
- The goal of meta-learning is to train a model on a variety of learning tasks, such that it can solve new learning tasks using only a small number of training samples.
- Method:
  - Train the model's **initial parameters**

Task description

- Model: $f(x) \rightarrow a$
- Task:

transition distribution

$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t), H\}$$

loss function

distribution over initial observations

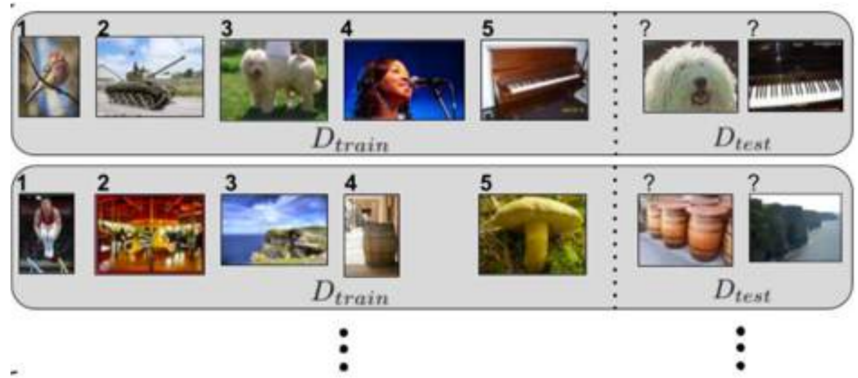
an episode length

- Supervised learning problem: $H = 1$
- Loss: $\mathcal{L}(\mathbf{x}_1, \mathbf{a}_1, \ldots, \mathbf{x}_H, \mathbf{a}_H) \rightarrow \mathbb{R}$

Model

- We want to learn the new task T_{new}

Sample tasks from $p(T)$

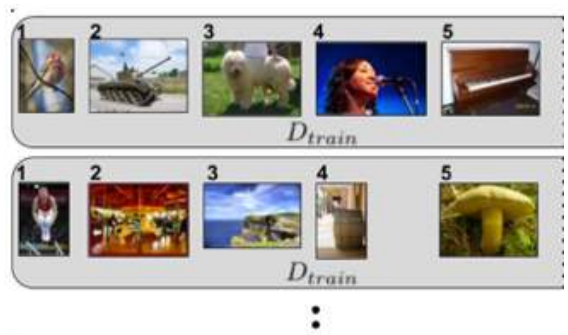


T_1

T_2

Train f_θ on the $\mathcal{D}_{train}$ using gradient based method

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta)$$



T_1: $\theta \longrightarrow \theta_1'$

T_2: $\theta \longrightarrow \theta_2'$

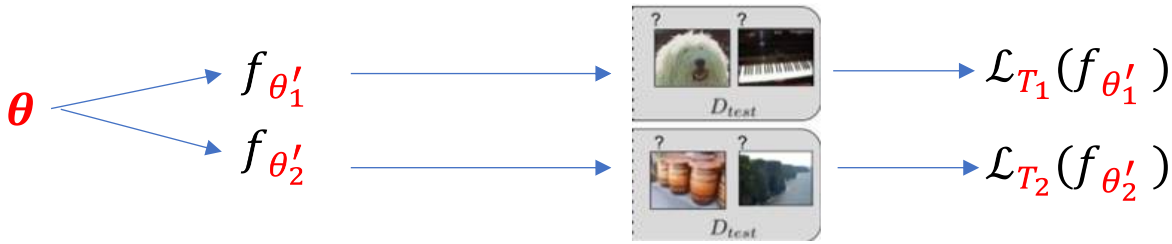
$f_{\theta_1'} \longrightarrow$

$f_{\theta_2'} \longrightarrow$

$\longrightarrow \mathcal{L}_{T_1}(f_{\theta_1'})$

$\longrightarrow \mathcal{L}_{T_2}(f_{\theta_2'})$

Model



Object function

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right)$$

θ is easy to fine-tune

Update $\boldsymbol{\theta}$ by:

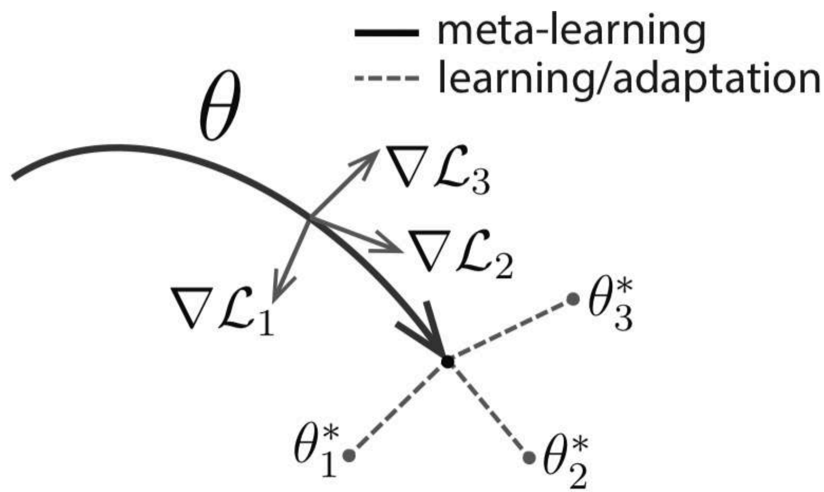
$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$$

Algorithm

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```



About this paper

- This work is a simple **model and task-agnostic** algorithm for meta-learning that trains a model's **parameters** such that a small number of gradient updates will lead to fast learning on a new task.
- A variety of different learning problems,
  - Classification
  - Regression
  - Reinforcement learning

Meta-Learning for Low-Resource Neural Machine Translation

Jiatao Gu , Yong Wang , Yun Chen , Kyunghyun Cho and Victor O.K. Li

The University of Hong Kong

New York University

Author

- The 4th year Ph.D. student at the **University of Hong Kong**
- Former visiting scholar at the CILVR lab**, New York University**
- Received Bachelor's Degree
  - **Tsinghua University** in 2014
- Research interests
  - Machine Translation
  - Natural Language Processing
  - Deep Learning
- **2018 Papers**
  - NAACL(1) AAAI(2) ICLR(1) EMNLP(1)

Meta learning

- Meta-learning tries to solve the problem of **"fast adaptation on new training data."**
- One of the most successful applications of meta-learning has been on **few-shot (or one-shot) learning.**
- **Two categories of meta-learning**
  - learning a **meta-policy** for updating model parameters
  - learning a **good parameter initialization** for fast adaptation

MAML

- Extend the recently introduced **model-agnostic meta-learning algorithm** for low resource neural machine translation (NMT).
- **Task:**
  - **viewing language pairs as separate tasks**.
  - use MAML to find the initialization of model parameters that facilitate fast adaptation for a new language pair with a minimal amount of training examples.

Meta learning for LR-NMT

Source Tasks:

$\{T^1, T^2, \dots, T^K\}$ →

T^1 : German→English
T^2 : Frence→English
.....
T^k : Dutch→English
.....
T^K : Polish→English

Target Tasks: T^0 →

T^0 :Turkish→English

Meta learn

Sample one Task T^k

from Source Tasks:

$\{T^1, T^2, \dots, T^K\}$

T^1 : German→English
T^2 : Frence→English
.....
$\boldsymbol{T^k}$ **: Dutch→English**
.....
T^K : Polish→English

$\boldsymbol{T^k}$ **: Dutch→English**

Sample train dataset D_{T^k}
and test dataset D'_{T^k}

Train D_{T^k}
1:Dutch→English
2:Dutch→English
......

Test D'_{T^k}
1:Dutch→English
2:Dutch→English
......

Meta learn

Train $D_{\mathcal{T}^k}$

1:Dutch→English

2:Dutch→English

……

NMT(θ)

MAML： $\boxed{\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}$

$\boxed{\theta'_k} = \text{Learn}(D_{\mathcal{T}^k};\theta) = \theta - \eta \nabla_\theta \mathcal{L}^{D_{\mathcal{T}^k}}(\theta)$

Test $D'_{\mathcal{T}^k}$

1:Dutch→English

2:Dutch→English

……

NMT(θ'_k)

MAML： $\boxed{\mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})}$

$\mathcal{L}^{D'_{\mathcal{T}^k}}(\theta'_k)$

Meta learn

T^1 : German→English $\longrightarrow$ $\mathcal{L}^{D'}{}_{\mathcal{T}^k}(\theta'_k)$

T^2 : Frence→English $\longrightarrow$ $\mathcal{L}^{D'}{}_{\mathcal{T}^k}(\theta'_k)$

....

T^k : Dutch→English $\longrightarrow$ $\mathcal{L}^{D'}{}_{\mathcal{T}^k}(\theta'_k)$

....

T^K : Polish→English $\longrightarrow$ $\mathcal{L}^{D'}{}_{\mathcal{T}^k}(\theta'_k)$

$$\theta \leftarrow \theta - \eta' \sum_k \nabla_\theta \mathcal{L}^{D'}{}_{\mathcal{T}^k}(\theta'_k),$$

MAML：

$$\boxed{\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})}$$

Learn

Given

Target Tasks: T^0 → T^0 :Turkish→English

initial parameters → θ^0

Objective function

$$\mathrm{Learn}(D_{\mathcal{T}};\theta^0) = \arg\max_{\theta} \mathcal{L}^{D_{\mathcal{T}}}(\theta)$$

$$= \arg\max_{\theta} \sum_{(X,Y)\in D_{\mathcal{T}}} \log p(Y|X,\theta) - \beta\|\theta - \theta^0\|^2,$$

maximum likelihood criterion often used for training a usual NMT system

discourages the newly learned model from deviating too much from the initial parameters

Meta learning for LR-NMT

Source Tasks:

$\{T^1, T^2, \ldots, T^K\}$ →

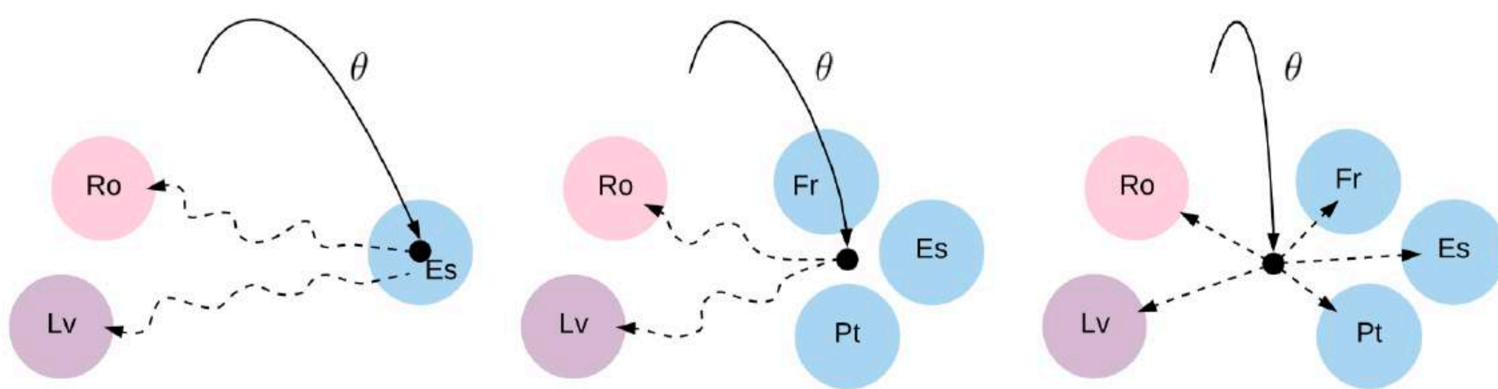
T^1 : German→English
T^2 : Frence→English
.....
T^k : Dutch→English
.....
T^K : Polish→English

Target Tasks: T^0 → T^0 :Turkish→English

$$\theta^* = \text{Learn}(\mathcal{T}^0; \text{MetaLearn}(\mathcal{T}^1, \ldots, \mathcal{T}^K)).$$

Transfer vs Multilingual vs Meta

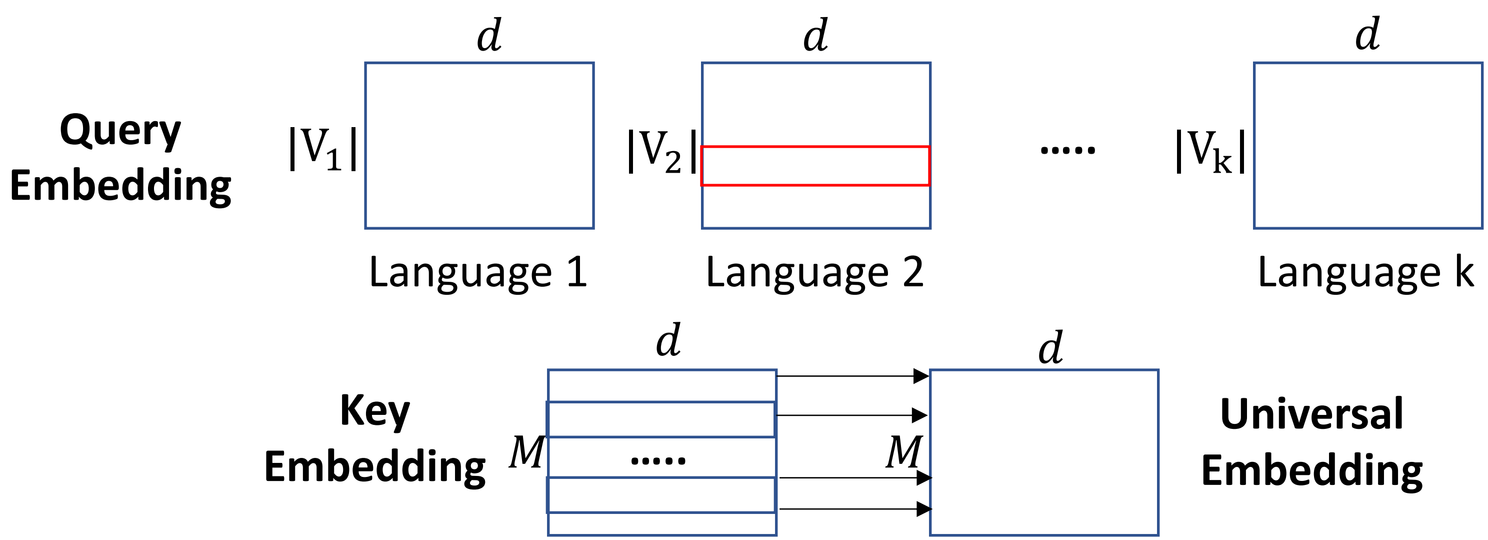
- **Transfer learning**
  - trains an NMT system specifically for a source language pair (Es-En) and finetunes the system for each target language pair (RoEn, Lv-En).
- **Multilingual learning**
  - trains a single NMT system that can handle many different language pairs (Fr-En, Pt-En, Es-En)
- **Meta learning**
  - trains the NMT system to be useful for fine-tuning on various tasks including the source and target tasks.



(a) Transfer Learning (b) Multilingual Transfer Learning (c) Meta Learning

Unified Lexical Representation

- **Problem**
  - vocabulary mismatch across different languages
- **Method**
  - *Universal Neural Machine Translation for Extremely Low Resource Languages NAACL 2018*



$$\epsilon^0[x] = \sum_{i=1}^{M} \alpha_i \epsilon_u[i],$$

Experiment

- **Dataset** ***(all to English)***
  - **Source Tasks(18)**
    - Bulgarian (Bg), Czech (Cs), Danish (Da), German (De), Greek (El), Spanish (Es), Estonian (Et), French (Fr), Hungarian (Hu), Italian (It), Lithuanian (Lt), Dutch (Nl), Polish (Pl), Portuguese (Pt), Slovak (Sk), Slovene (Sl) and Swedish (Sv), Russian (Ru)
  - **Target Tasks(5)**
    - **Romanian (Ro)** from WMT'16
    - **Latvian (Lv), Finnish (Fi), Turkish (Tr)** from WMT'17
    - **Korean (Ko)** from Korean Parallel Dataset.
- **Validation (Dev)**
  - Either **Ro-En or Lv-En** as a validation set for meta-learning

Model

- **Transformer**
  - d_model = d_hidden = 512
  - N_layer = 6
  - N_head = 8
  - N_batch = 4000
  - T_warmup = 16000
- **Universal lexical representation（ULR）**

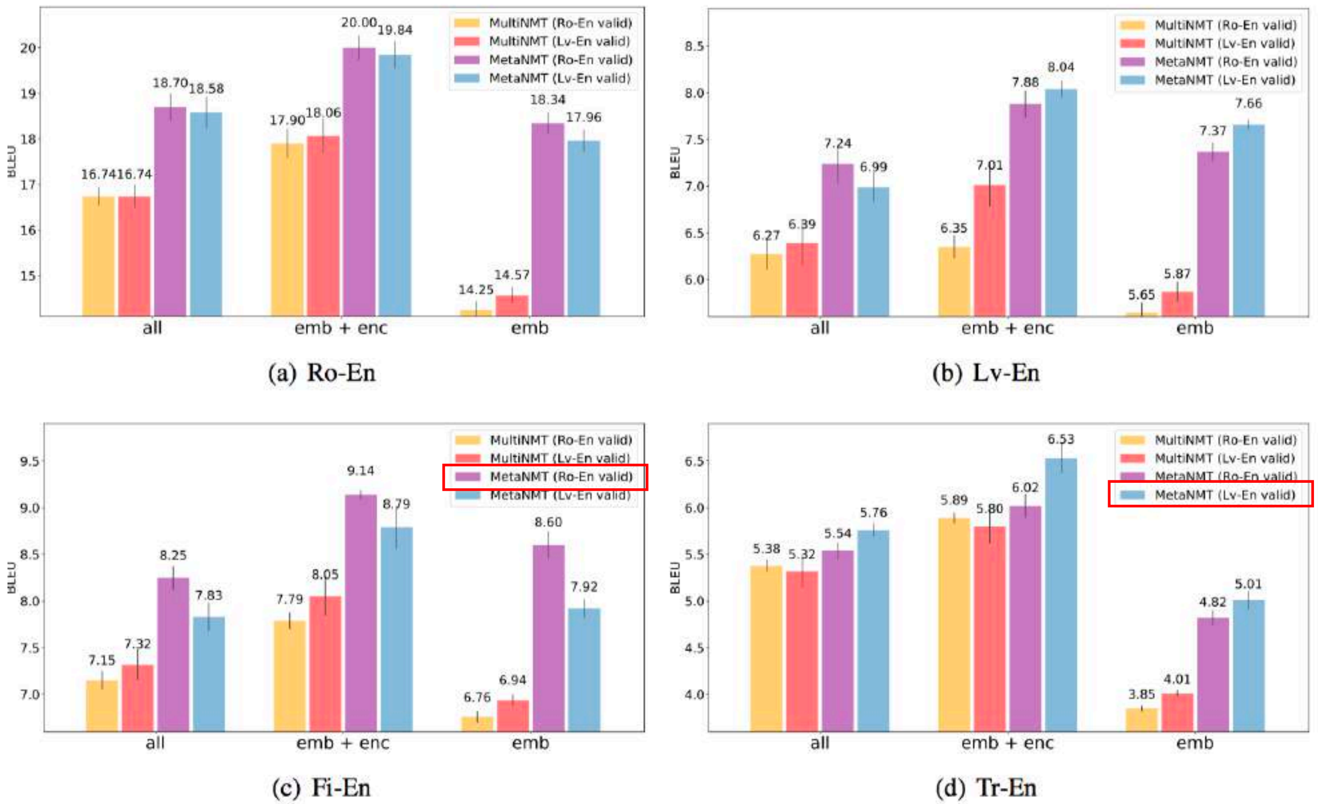
Learning

- **Single** gradient step of language-specific learning with **Adam.**
- For each target task, we sample training examples to form a low-resource task.
- Build tasks of **4k, 16k, 40k and 160k** English tokens for each language.
- Randomly sample the training set **five times** for each experiment and report the **average score**
- Each fine-tuning is done on a **training set**, early-stopped on a **validation set** and evaluated on a **test set**.

Fine-tuning Strategies

- Update all three modules during **meta learning**
- Fine tuning
  - Fine-tuning all the modules (all)
  - Fine-tuning the embedding and encoder, but freezing the parameters of the decoder (emb+enc)
  - Fine-tuning the embedding only (emb)

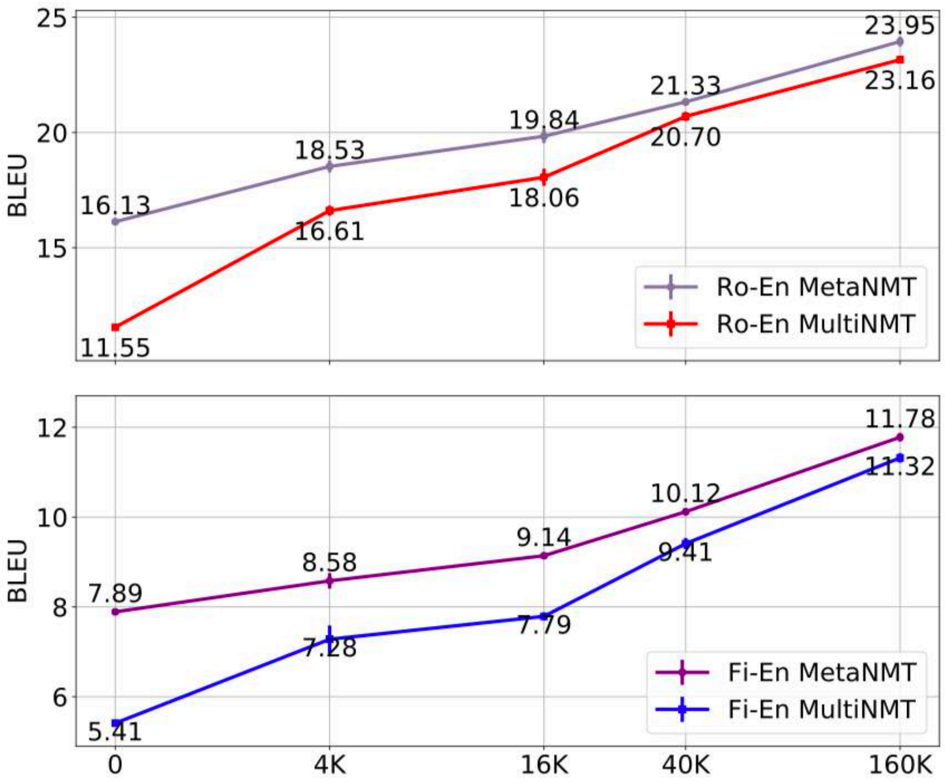
vs. Multilingual Transfer Learning



(a) Ro-En (b) Lv-En (c) Fi-En (d) Tr-En

- Significantly outperforms the multilingual, transfer learning strategy across all the target tasks regardless of which target task was used for early stopping
- The **emb+enc strategy** is most effective for both meta-learning and transfer learning approaches.
- **Choice of a validation task** has non-negligible impact on the final performance

Training Set Size



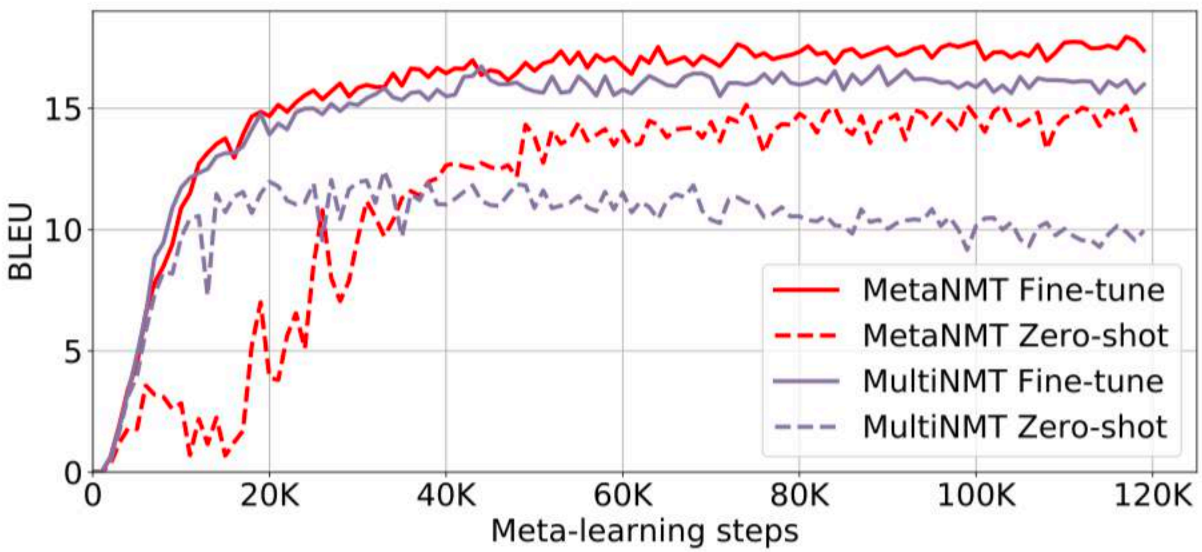
- Meta-learning approach is more **robust** to the drop in the size of the target task's training set

Impact of Source Tasks

Meta-Train	Ro-En		Lv-En		Fi-En		Tr-En		Ko-En	
	zero	finetune	zero	finetune	zero	finetune	zero	finetune	zero	finetune
–		$00.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$		$0.00 \pm .00$
Es	9.20	$15.71 \pm .22$	2.23	$4.65 \pm .12$	2.73	$5.55 \pm .08$	1.56	$4.14 \pm .03$	0.63	$1.40 \pm .09$
Es Fr	12.35	$17.46 \pm .41$	2.86	$5.05 \pm .04$	3.71	$6.08 \pm .01$	2.17	$4.56 \pm .20$	0.61	$1.70 \pm .14$
Es Fr It Pt	13.88	$18.54 \pm .19$	3.88	$5.63 \pm .11$	4.93	$6.80 \pm .04$	2.49	$4.82 \pm .10$	0.82	$1.90 \pm .07$
De Ru	10.60	$16.05 \pm .31$	5.15	$7.19 \pm .17$	6.62	$7.98 \pm .22$	3.20	$6.02 \pm .11$	1.19	$2.16 \pm .09$
Es Fr It Pt De Ru	15.93	$20.00 \pm .27$	6.33	$7.88 \pm .14$	7.89	$9.14 \pm .05$	3.72	$6.02 \pm .13$	1.28	$2.44 \pm .11$
All	18.12	$\mathbf{22.04 \pm .23}$	9.58	$\mathbf{10.44 \pm .17}$	11.39	$\mathbf{12.63 \pm .22}$	5.34	$\mathbf{8.97 \pm .08}$	1.96	$\mathbf{3.97 \pm .10}$
Full Supervised	31.76		15.15		20.20		13.74		5.97	

- Beneficial to use more source tasks
- The choice of source languages has different implications for different target languages

Training Curves



- Multilingual transfer learning rapidly saturates（饱和）and eventually degrades, as the model overfits to the source tasks.

Sample Translations

Source (Tr)	google mülteciler için 11 milyon dolar toplamak üzere bağış eşleştirme kampanyasını başlattı .
Target	google launches donation-matching campaign to raise $ 11 million for refugees .
Meta-0	google refugee fund for usd 11 million has launched a campaign for donation .
Meta-16k	google has launched a campaign to collect $ 11 million for refugees .
Source (Ko)	이번에 체포되어 기소된 사람들 중에는 퇴역한 군 고위관리 , 언론인 , 정치인 , 경제인 등이 포함됐다
Target	among the suspects are retired military officials , journalists , politicians , businessmen and others .
Meta-0	last year , convicted people , among other people , of a high-ranking army of journalists in economic and economic policies , were included .
Meta-16k	the arrested persons were included in the charge , including the military officials , journalists , politicians and economists .

Conclusion

- **Types of Meta-Learning Models**
  1. Few Shots Meta-Learning
  2. Optimizer Meta-Learning
  3. Metric Meta-Learning
  4. Recurrent Model Meta-Learning
  5. Initializations Meta-Learning

- **Two categories of meta-learning**
  - learning a meta-policy for updating model parameters
  - learning a good parameter initialization for fast adaptation

Thanks!