Meta-learning and its applications to NLP

Katia Shutova

ILLC
University of Amsterdam

30 April 2020

Deep learning in NLP

Deep learning models have achieved much success in NLP, but...

- using large datasets for training
- the resulting models are not easily adaptive
- unrealistic to have such large datasets for every possible task, application scenario, domain or language

We need models that are adaptive and can learn from a few examples.

Self-supervised pre-training

- general-purpose word and sentence encoding models
- with self-supervised pre-training (e.g. BERT, GPT-2)
- provide a good starting point for task-specific fine-tuning

and yet...

- to perform well in a given task
- need to fine-tune on a large task-specific dataset

Do not enable few-shot learning or model adaptation.

Meta-learning

Meta-learning, aka "learning to learn"

- a framework to train models to perform fast adaptation from a few examples
- a different learning paradigm: episodic learning
- many promising results in computer vision
- still relatively new to NLP (but we have some initial positive results already!)

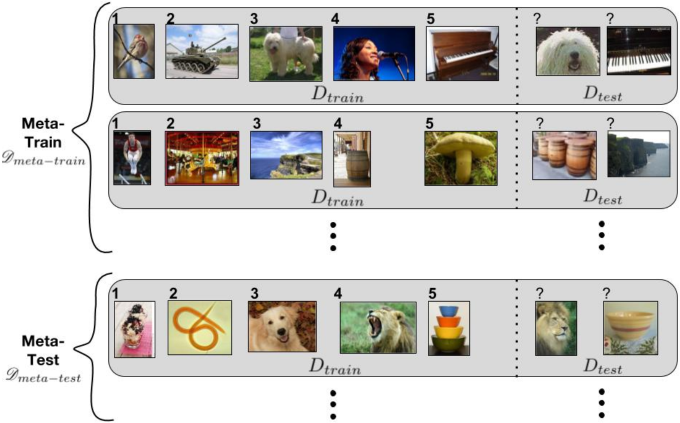
Episodic learning

Learning from a collection of few-shot tasks, called episodes

Each episode has its own

- training set = support set
- test set = query set

Meta-training and meta-test sets



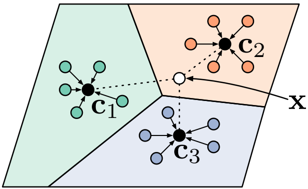
Meta-learning methods

1. **Metric-based**
    - embed examples in each episode using a neural network
    - compute probability distribution over labels for all query examples
    - based on their similarity with the support examples.
2. **Model-based**
    - achieve rapid learning directly through their architectures.
3. **Optimisation-based**
    - explicitly include generalizability in their objective function.

Metric-based method: Prototypical networks

Snell et al 2017. *Prototypical Networks for Few-shot Learning*. NIPS.

- use an embedding function f_θ to encode each input into a vector
- compute a prototype feature vector for every class k
- as the **mean vector** of the embedded **support examples** in this class.



$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\theta(x_i)$$

Prototypical networks

For a given query input x:

- compute the distance between its embedding and each of the prototype vectors
- pass through a **softmax**
- to get the **distribution over classes**

$$P(y = k|x) = softmax(-d_\phi(f_\theta(x), c_k)) = \frac{exp(-d_\phi(f_\theta(x), c_k))}{\sum_{k'} exp(-d_\phi(f_\theta(x), c_{k'}))}$$

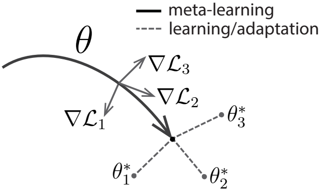
where d_ϕ is the distance function

- Snell et al. use squared Euclidean distance
- The loss function is the negative log-likelihood.

Optimisation-based method: Model-agnostic meta-learning

Finn et al. 2017. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. ICML.

- General and model-agnostic method
- applicable to **any learning problem**
- and **any model architecture** (trainable with gradient descent)



Model-agnostic meta-learning (MAML)

Key intuition:

- learn a good parameter initialisation
- such that the model has **maximal performance** on a new task
- after the parameters have been updated in a few gradient steps
- computed with **a small amount of data** from that new task.

Essentially, the goal is to learn internal representations that are broadly suitable for many tasks.

MAML overview

The learner model f_θ, parametrized by θ

- e.g. a sentence encoder, such as an LSTM or Transformer.

The meta-learning algorithm

1. **Adapt** to a new task $\mathcal{T}_i$, given the task objective
   - computing the loss on the **support set**
2. Perform **meta-optimisation** over a batch of tasks (episodes)
   - computing the loss on the **query sets**.

MAML algorithm

1. Adapt to a new task $\mathcal{T}_i$, given the task objective:
   - compute updated parameters θ_i' using the **support set**

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

2. Perform meta-optimisation over a batch of tasks (episodes)
   - minimise meta-objective across tasks, on the **query sets**:

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

   - perform a meta-update of shared parameters θ

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

MAML algorithm

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

First-order approximation of MAML

- Computing second-order gradients is computationally expensive
- Finn et al. proposed a first order approximation of MAML
- compute the gradients with respect to the updated parameters θ'_i rather than the initial parameters θ

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

Hybrid method: ProtoMAML

Triantafillou et al. 2020. *Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples*. ICLR.

- Prototypical networks with Euclidean distance are **equivalent to a linear model** with a particular parameterization

$$-||f_\theta(x) - c_k||^2 = -f_\theta(x)^T f_\theta(x) + 2{c_k}^T f_\theta(x) - {c_k}^T c_k$$

$f_\theta(x)^T f_\theta(x)$ is constant with respect to class k

$$2{c_k}^T f_\theta(x) - {c_k}^T c_k = {w_k}^T f_\theta(x) + b_k$$

w_k and b_k are the weights and biases for the output unit corresponding to class k.

ProtoMAML

Key idea:

- initialise the final layer of the learner classifier in each episode
- with prototypical network-equivalent weights and biases
- and continue to learn with MAML.

Benefits:

- combines the strength of prototypical networks and MAML
- extends MAML beyond N-way, K-shot scenario.

Meta-learning in NLP

1. Address one NLP task (e.g. focus on learning new classes)
   - **Tasks addressed**: relation classification, entity typing, text classification, word sense disambiguation
2. Apply meta-learning across multiple NLP tasks
   - Bansal et al. 2019 – to be discussed later in this session
3. Apply meta-learning across languages
   - **machine translation** for low-resource languages
   - **NLI** and **question answering** (Nooralahzadeh et al. 2020) – to be discussed next Thursday

Meta-learning in NLP: Methods

- Model architectures:
  - feed-forward networks
  - graph convolutional networks
  - recurrent networks (LSTM, GRU)
  - transformers
- Meta-learning methods:
  - First-order MAML (the most popular)
  - several extensions thereof proposed
  - Prototypical networks
  - ProtoMAML

Meta-learning for word sense disambiguation

Holla et al. 2020. *Learning to Learn to Disambiguate: Meta-Learning for Few-Shot Word Sense Disambiguation*. ArXiv.

WSD task: determine the sense of a word (e.g. WordNet sense)

*The children **ran** to the store*

*Service **runs** all the way to Cranbury*

*She is **running** a relief operation in Sudan*

*the story or argument **runs** as follows*

*Does this old car still **run** well?*

*Who's **running** for treasurer this year?*

Our goal: learn new word senses from a few examples

Challenges in WSD

- The nature of the learning problem
  - WSD exhibits inter-word dependencies within sentences
  - has a large number of classes
  - and dramatic class imbalances.
- Existing supervised approaches
  - learn a model per word
  - require very large training datasets
  - that are impossible to produce at a realistic scale.

A problem desperately in need of a few-shot learning approach!

But also presents new challenges compared to the controlled setup in most current meta-learning approaches (N-way, K-shot classification).

Task definition and episode generation

- Classify word use with respect to a predefined sense inventory
- typically treated as a **sequence labelling task**
- convert it to a **"word in context" classification task**.

*She is **running** a relief operation in Sudan.*

- Divide words into meta-training and meta-test splits
- Meta-training: 4 words per episode (with multiple senses)
- Meta-test: 1 word per episode (with multiple senses)
- experiment with support sets of 8, 16 and 32.

Methods

- Model architectures:
  - Glove + GRU
  - ELMo + MLP
  - fine-tuning BERT base.
- Meta-learning methods:
  - First- and second-order MAML
  - Prototypical networks
  - ProtoMAML (and its second-order variant)

Results

Embedding/ Encoder	**Method**	**Average macro F1 score** $\|S\| = 8$	$\|S\| = 16$	$\|S\| = 32$
-	MajoritySenseBaseline	0.259	0.264	0.261
GloVe+GRU	NearestNeighbor	–	–	–
	NE-Baseline	0.507 ± 0.005	0.479 ± 0.004	0.451 ± 0.009
	EF-ProtoNet	0.539 ± 0.009	0.538 ± 0.003	0.562 ± 0.005
	EF-FOMAML	0.341 ± 0.002	0.321 ± 0.004	0.303 ± 0.005
	EF-ProtoFOMAML	0.529 ± 0.010	0.540 ± 0.004	0.553 ± 0.009
	ProtoNet	$\mathbf{0.601 \pm 0.003}$	$\mathbf{0.633 \pm 0.008}$	$\mathbf{0.654 \pm 0.004}$
	FOMAML	0.418 ± 0.005	0.392 ± 0.007	0.375 ± 0.005
	ProtoFOMAML	0.599 ± 0.005	0.617 ± 0.004	0.627 ± 0.004
ELMo+MLP	NearestNeighbor	0.641	0.645	0.654
	NE-Baseline	0.640 ± 0.012	0.633 ± 0.001	0.614 ± 0.008
	EF-ProtoNet	0.635 ± 0.004	0.661 ± 0.004	0.683 ± 0.003
	EF-FOMAML	0.414 ± 0.006	0.383 ± 0.003	0.352 ± 0.003
	EF-ProtoFOMAML	0.621 ± 0.004	0.623 ± 0.008	0.611 ± 0.005
	ProtoNet	0.688 ± 0.004	0.709 ± 0.006	$\mathbf{0.731 \pm 0.006}$
	FOMAML	0.589 ± 0.010	0.587 ± 0.012	0.575 ± 0.016
	ProtoFOMAML	$\mathbf{0.689 \pm 0.007}$	$\mathbf{0.711 \pm 0.004}$	0.726 ± 0.004
BERT	NearestNeighbor	0.704	0.716	0.741
	NE-Baseline	0.599 ± 0.023	0.539 ± 0.025	0.473 ± 0.015
	EF-ProtoNet	0.655 ± 0.004	0.682 ± 0.005	0.721 ± 0.009
	EF-FOMAML	0.522 ± 0.007	0.450 ± 0.008	0.393 ± 0.002
	EF-ProtoFOMAML	0.662 ± 0.006	0.654 ± 0.009	0.665 ± 0.009
	ProtoNet	$\mathbf{0.750 \pm 0.008}$	$\mathbf{0.755 \pm 0.002}$	$\mathbf{0.766 \pm 0.003}$
	FOMAML	0.550 ± 0.011	0.476 ± 0.010	0.436 ± 0.014
	ProtoFOMAML	0.731 ± 0.004	0.739 ± 0.008	0.744 ± 0.005

Acknowledgement

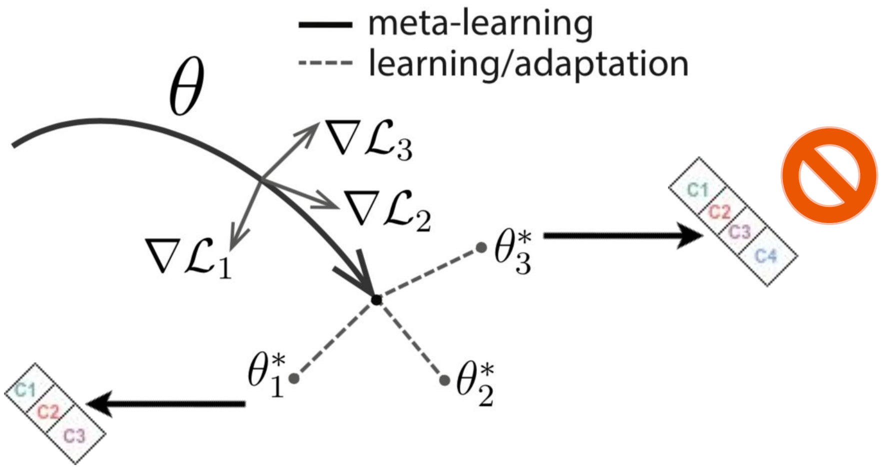
Some images were adapted from Hugo Larochelle

Few-Shot Learn Across Diverse NLP Classification Tasks

Authors: Trapit Bansal, Rishikesh Jha, Andrew McCallum
Presented by: Aman Hussain & Albert Harkema

Limitations of MAML

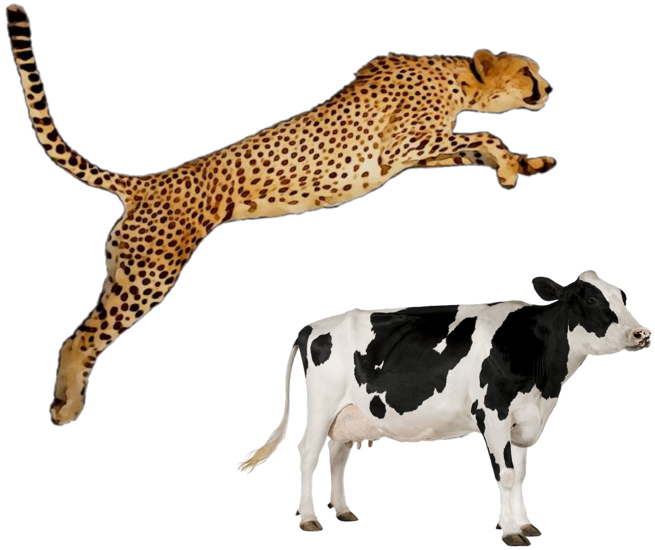
Requires fixed number of classes across different tasks



LEOPARD

- **Parameter Generator:**
  Initializes **task-dependent** softmax parameters

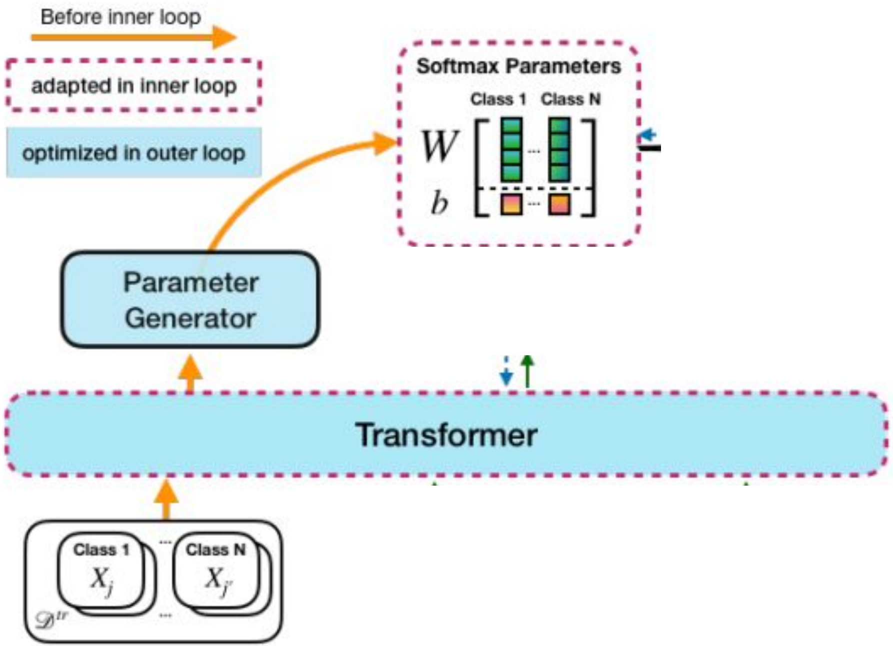
- **Parameter Efficient Training:**
  Adapt efficiently across **diverse** tasks



LEOPARD architecture

Parameter Generator
N-way task conditioned for on meta-training data

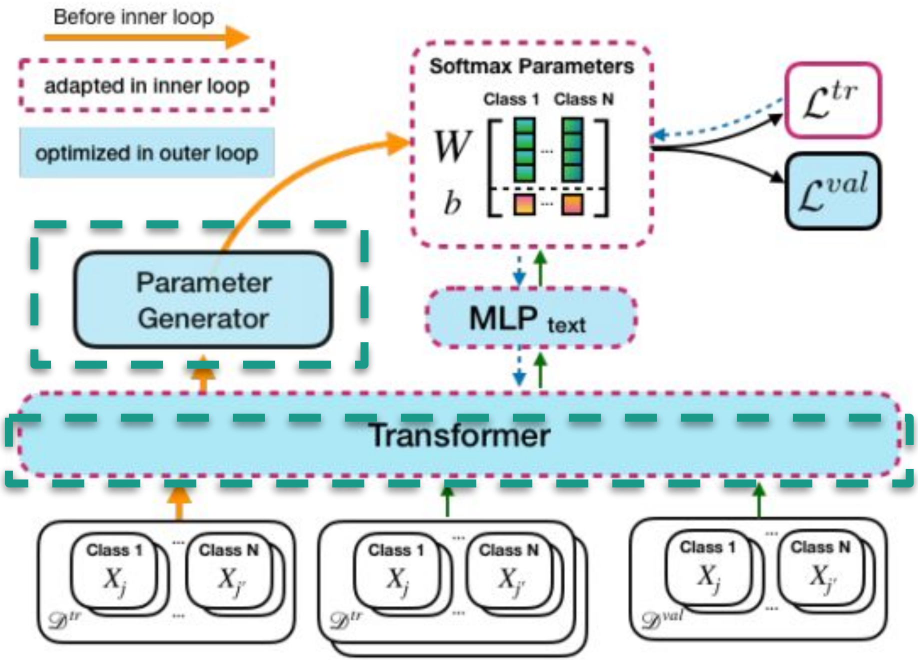
$$w_i^n, b_i^n = \frac{1}{|C_i^n|} \sum_{x_j \in C_i^n} g_\psi(f_\theta(\mathbf{x}_j))$$



LEOPARD architecture

Parameter Efficient Training

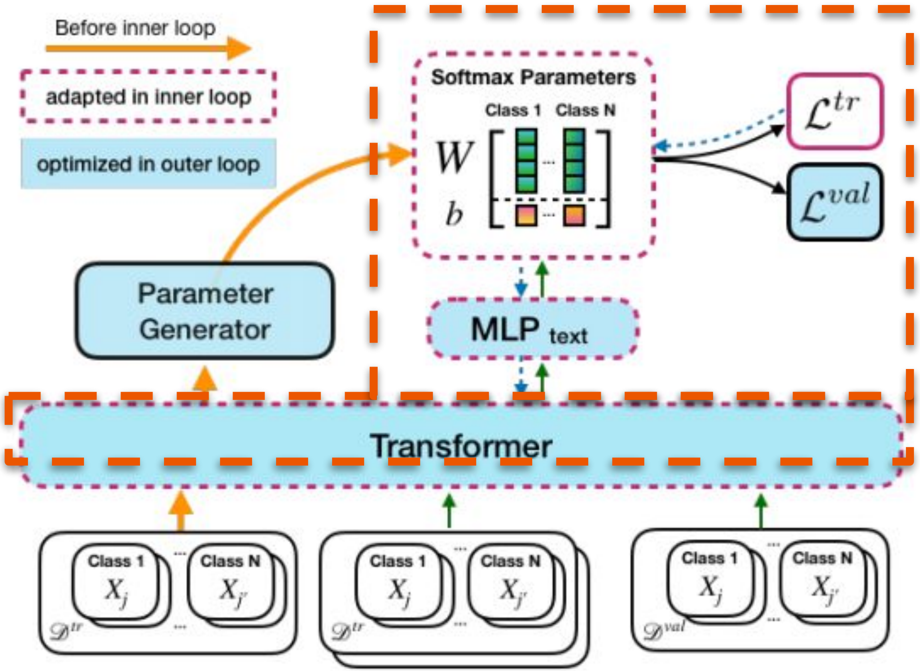
1. Task agnostic
2. Task specific



LEOPARD architecture

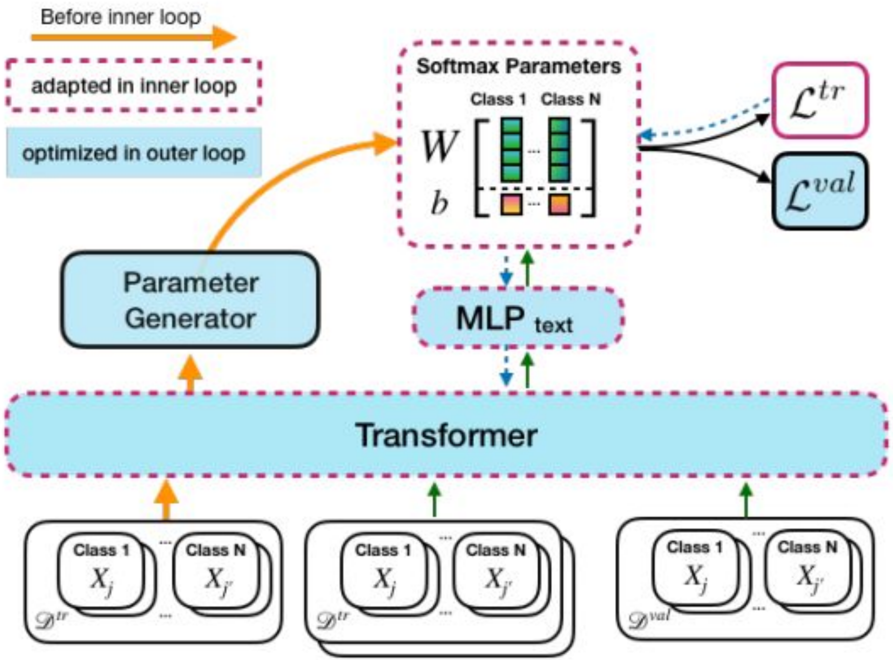
Parameter Efficient Training

1. Task agnostic
2. Task specific



Experiment Setup

- Per-layer learning rate for inner loop
- Pre-trained BERT
- Hyperparameter: task specific no. of layers



Experiments

Training Tasks

GLUE: 8 tasks focus on sentence-level classification (without WNLI & STS-B)

During Meta-Training: classify between every pair of labels

Experiments

Evaluation and Baselines

Samples: for every k ∊ {4, 8, 16} sample 10 training datasets

Validation Task: SciTail

Models: BERTbase, Multi-task BERT, Prototypical BERT

Evaluation: 17 target NLP tasks

BERTBASE

Results

Unseen Tasks

- Relative gain in accuracy:
    - **14.5%** (k=4)
    - 10.75% (k=8)
    - 10.9% (k=16)
- Outperforms baselines for **never seen tasks:** entity typing, rating classification, text classification
- Prototypical networks **worse** than fine-tuning methods for **never seen tasks**

	N	k	$\text{BERT}_{\text{base}}$	$\text{MT-BERT}_{\text{softmax}}$	MT-BERT	Proto-BERT	LEOPARD
Entity Typing							
CoNLL	4	4	50.44 ± 08.57	52.28 ± 4.06	**55.63** ± 4.99	32.23 ± 5.10	54.16 ± 6.32
		8	50.06 ± 11.30	65.34 ± 7.12	58.32 ± 3.77	34.49 ± 5.15	**67.38** ± 4.33
		16	74.47 ± 03.10	71.67 ± 3.03	71.29 ± 3.30	33.75 ± 6.05	**76.37** ± 3.08
MITR	8	4	49.37 ± 4.28	45.52 ± 5.90	**50.49** ± 4.40	17.36 ± 2.75	49.84 ± 3.31
		8	49.38 ± 7.76	58.19 ± 2.65	58.01 ± 3.54	18.70 ± 2.38	**62.99** ± 3.28
		16	69.24 ± 3.68	66.09 ± 2.24	66.16 ± 3.46	16.41 ± 1.87	**70.44** ± 2.89
Text Classification							
Airline	3	4	42.76 ± 13.50	43.73 ± 7.86	46.29 ± 12.26	40.27 ± 8.19	**54.95** ± 11.81
		8	38.00 ± 17.06	52.39 ± 3.97	49.81 ± 10.86	51.16 ± 7.60	**61.44** ± 03.90
		16	58.01 ± 08.23	58.79 ± 2.97	57.25 ± 09.90	48.73 ± 6.79	**62.15** ± 05.56
Disaster	2	4	**55.73** ± 10.29	52.87 ± 6.16	50.61 ± 8.33	50.87 ± 1.12	51.45 ± 4.25
		8	**56.31** ± 09.57	56.08 ± 7.48	54.93 ± 7.88	51.30 ± 2.30	55.96 ± 3.58
		16	64.52 ± 08.93	**65.83** ± 4.19	60.70 ± 6.05	52.76 ± 2.92	61.32 ± 2.83
Emotion	13	4	09.20 ± 3.22	09.41 ± 2.10	09.84 ± 2.14	09.18 ± 3.14	**11.71** ± 2.16
		8	08.21 ± 2.12	11.61 ± 2.34	11.21 ± 2.11	11.18 ± 2.95	**12.90** ± 1.63
		16	13.43 ± 2.51	**13.82** ± 2.02	12.75 ± 2.04	12.32 ± 3.73	13.38 ± 2.20
Political Bias	2	4	54.57 ± 5.02	54.32 ± 3.90	54.66 ± 3.74	56.33 ± 4.37	**60.49** ± 6.66
		8	56.15 ± 3.75	57.36 ± 4.32	54.79 ± 4.19	58.87 ± 3.79	**61.74** ± 6.73
		16	60.96 ± 4.25	59.24 ± 4.25	60.30 ± 3.26	57.01 ± 4.44	**65.08** ± 2.14
Political Audience	2	4	**51.02** ± 1.23	50.45 ± 1.01	50.96 ± 1.72	49.55 ± 1.98	50.84 ± 1.33
		8	50.87 ± 1.88	51.63 ± 1.81	50.36 ± 1.53	50.62 ± 1.35	**51.74** ± 1.37
		16	**53.09** ± 1.93	52.41 ± 1.25	51.24 ± 2.18	50.92 ± 1.56	51.90 ± 1.43
Political Message	9	4	15.64 ± 2.73	13.71 ± 1.10	14.49 ± 1.75	14.22 ± 1.25	**15.69** ± 1.57
		8	13.38 ± 1.74	14.33 ± 1.32	15.24 ± 2.81	15.67 ± 1.96	**18.02** ± 2.32
		16	**20.67** ± 3.89	18.11 ± 1.48	19.20 ± 2.20	16.49 ± 1.96	18.07 ± 2.41
Rating Books	3	4	39.42 ± 07.22	44.82 ± 9.00	38.97 ± 13.27	48.44 ± 7.43	**54.92** ± 6.18
		8	39.55 ± 10.01	51.14 ± 6.78	46.77 ± 14.12	52.13 ± 4.79	**59.16** ± 4.13
		16	43.08 ± 11.78	54.61 ± 6.79	51.68 ± 11.27	57.28 ± 4.57	**61.02** ± 4.19
Rating DVD	3	4	32.22 ± 08.72	45.94 ± 7.48	41.23 ± 10.98	47.73 ± 6.20	**49.76** ± 9.80
		8	36.35 ± 12.50	46.23 ± 6.03	45.24 ± 9.76	47.11 ± 4.00	**53.28** ± 4.66
		16	42.79 ± 10.18	49.23 ± 6.68	45.19 ± 11.56	48.39 ± 3.74	**53.52** ± 4.77
Rating Electronics	3	4	39.27 ± 10.15	39.89 ± 5.83	41.20 ± 10.69	37.40 ± 3.72	**51.71** ± 7.20
		8	28.74 ± 08.22	46.53 ± 5.44	45.41 ± 09.49	43.64 ± 7.31	**54.78** ± 6.48
		16	45.48 ± 06.13	48.71 ± 6.16	47.29 ± 10.55	44.83 ± 5.96	**58.69** ± 2.41
Rating Kitchen	3	4	34.76 ± 11.20	40.41 ± 5.33	36.77 ± 10.62	44.72 ± 9.13	**50.21** ± 09.63
		8	34.49 ± 08.72	48.35 ± 7.87	47.98 ± 09.73	46.03 ± 8.57	**53.72** ± 10.31
		16	47.94 ± 08.28	52.94 ± 7.14	53.79 ± 09.47	49.85 ± 9.31	**57.00** ± 08.69
Overall Average		4	38.06	40.04	40.05	36.13	**45.84**
		8	36.83	45.73	43.92	39.05	**50.65**
		16	48.10	49.60	48.74	39.63	**55.02**

Results

Domain Adaptation

- **LEOPARD** outperforms on **multi-domain** sentiment classification
- **MT-BERT** performs better on **Scitail** since it is trained on many **related** NLI datasets

	k	$\text{BERT}_{\text{base}}$	$\text{MT-BERT}_{\text{softmax}}$	MT-BERT	$\text{MT-BERT}_{\text{reuse}}$	Proto-BERT	LEOPARD
Natural Language Inference							
Scitail	4	58.53 ± 09.74	74.35 ± 5.86	63.97 ± 14.36	**76.65 ± 2.45**	76.27 ± 4.26	69.50 ± 9.56
	8	57.93 ± 10.70	**79.11 ± 3.11**	68.24 ± 10.33	76.86 ± 2.09	78.27 ± 0.98	75.00 ± 2.42
	16	65.66 ± 06.82	**79.60 ± 2.31**	75.35 ± 04.80	79.53 ± 2.17	78.59 ± 0.48	77.03 ± 1.82
Amazon Review Sentiment Classification							
Books	4	54.81 ± 3.75	68.69 ± 5.21	64.93 ± 8.65	74.79 ± 6.91	73.15 ± 5.85	**82.54 ± 1.33**
	8	53.54 ± 5.17	74.86 ± 2.17	67.38 ± 9.78	78.21 ± 3.49	75.46 ± 6.87	**83.03 ± 1.28**
	16	65.56 ± 4.12	74.88 ± 4.34	69.65 ± 8.94	78.87 ± 3.32	77.26 ± 3.27	**83.33 ± 0.79**
Kitchen	4	56.93 ± 7.10	63.07 ± 7.80	60.53 ± 9.25	75.40 ± 6.27	62.71 ± 9.53	**78.35 ± 18.36**
	8	57.13 ± 6.60	68.38 ± 4.47	69.66 ± 8.05	75.13 ± 7.22	70.19 ± 6.42	**84.88 ± 01.12**
	16	68.88 ± 3.39	75.17 ± 4.57	77.37 ± 6.74	80.88 ± 1.60	71.83 ± 5.94	**85.27 ± 01.31**

Ablation Study

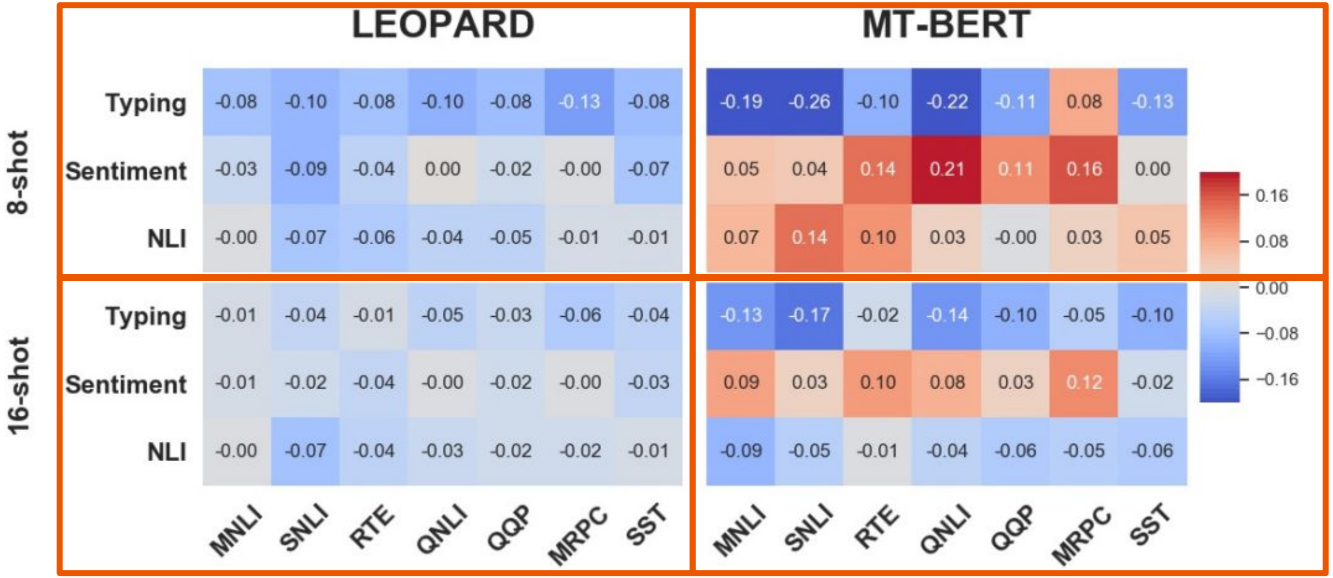
Parameter Generator: Removing generator and using zero-initialized softmax performs worse

Parameter Efficient Training: For all tasks, except NLI (Scitail), adapting all parameters is better

k	Model	Entity Typing	Sentiment Classification	NLI
16	LEOPARD $_{10}$	37.62 $\pm$ 7.37	58.10 $\pm$ 5.40	78.53 $\pm$ 1.55
	LEOPARD $_5$	62.49 $\pm$ 4.23	71.50 $\pm$ 5.93	73.27 $\pm$ 2.63
	LEOPARD	69.00 $\pm$ 4.76	76.65 $\pm$ 2.47	76.10 $\pm$ 2.21
	LEOPARD-ZERO	44.79 $\pm$ 9.34	74.45 $\pm$ 3.34	74.36 $\pm$ 6.67

Ablation Study

Training Task Selection: LEOPARD's performance is more consistent compared to MT-BERT



Discussion

- Include other baselines (e.g. single task / Ceiling [human baselines])
- MT-BERT outperforms on Entity Typing for k=4 (not discussed in the paper)
- MAML-related approaches effective and gaining popularity
- Is LEOPARD-like meta-learning the way forward to solving general linguistics in AI?

Our Opinion

4.5 ★★★★½

“Extensive experiments!”

- *Aman & Albert*

- Natural extension of MAML
- Extensive Experiments
- Ablation Study
- No interpretable baseline