Adapters in Transformers: a New Paradigm for Transfer Learning?...

Jonas Pfeiffer (main author)
Iryna Gurevych (PhD advisor)
What you will learn today
What you will learn today

1. What adapters (in transformers) are.
What **you** will learn today

1. What **adapters** (in transformers) are.
2. If adapters really are more **efficient** than normal fine-tuning.
What you will learn today

1. What adapters (in transformers) are.
2. If adapters really are more efficient than normal fine-tuning.
3. How to non-destructively compose tasks for transfer learning (AdapterFusion).
What you will learn today

1. What **adapters** (in transformers) are.
2. If adapters really are more **efficient** than normal fine-tuning.
3. How to non-destructively **compose tasks** for transfer learning (AdapterFusion).
4. How to **stack modular adapters** for zero-shot transfer to unseen and low-resource languages (MAD-X).
What you will learn today

1. What adapters (in transformers) are.
2. If adapters really are more efficient than normal fine-tuning.
3. How to non-destructively compose tasks for transfer learning (AdapterFusion).
4. How to stack modular adapters for zero-shot transfer to unseen and low-resource languages (MAD-X).
5. How to train adapters with the AdapterHub.ml framework
Some Basics...
Some Basics...

Everything I am about to cover involves:
Some Basics...

Everything I am about to cover involves:
- Transfer Learning in Natural Language Processing (NLP)
Some Basics...

Everything I am about to cover involves:
- Transfer Learning in Natural Language Processing (NLP)
- We **only** look at deep neural networks, specifically the **Transformer** architecture.
Some Basics...

Everything I am about to cover involves:
- Transfer Learning in Natural Language Processing (NLP)
- We only look at deep neural networks, specifically the Transformer architecture.
- We leverage pre-trained transformer-based models such as BERT/RoBERTa/XLM-R/mBERT.
Some Basics...

Everything I am about to cover involves:

- Transfer Learning in Natural Language Processing (NLP)
- We only look at deep neural networks, specifically the Transformer architecture.
- We leverage pre-trained transformer-based models such as BERT/RoBERTa/XLM-R/mBERT.
- These have been trained on massive amount of text data using Masked Language Modelling.
Some Basics...

Everything I am about to cover involves:

- Transfer Learning in Natural Language Processing (NLP)
- We only look at deep neural networks, specifically the Transformer architecture.
- We leverage pre-trained transformer-based models such as BERT/RoBERTa/XLM-R/mBERT.
- These have been trained on massive amount of text data using Masked Language Modelling.
- Transfer learning with these pre-trained models usually involves stacking a prediction head on top of the model.
Some Basics...

Everything I am about to cover involves:

- Transfer Learning in Natural Language Processing (NLP)
- We only look at deep neural networks, specifically the Transformer architecture.
- We leverage pre-trained transformer-based models such as BERT/RoBERTa/XLM-R/mBERT.
- These have been trained on massive amount of text data using Masked Language Modelling.
- Transfer learning with these pre-trained models usually involves stacking a prediction head on top of the model.
- Usually all parameters are fine-tuned on the downstream task (e.g. using cross-entropy loss).
Agenda

1. Adapters in Transformers
2. AdapterFusion
3. MAD-X
4. Efficiency of Adapters
5. AdapterHub.ml
Agenda

1. Adapters in Transformers
2. AdapterFusion
3. MAD-X
4. Efficiency of Adapters
5. AdapterHub.ml
What are Adapters?

= Parameters are frozen  = Parameters are fine-tuned
What are Adapters?

A single Transformer (encoder) layer

- Parameters are frozen
- Parameters are fine-tuned
What are Adapters?

A single Transformer (encoder) layer

= Parameters are frozen

= Parameters are fine-tuned
What are Adapters?

A single Transformer (encoder) layer

\[ \Theta \leftarrow \operatorname{argmin}_\Theta L(D_{\text{NLI}}; \Theta) \]

- \( D_{\text{NLI}} \) = NLI Dataset
- \( L \) = Loss function, e.g. cross entropy loss
- \( \Theta \) = Parameters of the model

= Parameters are frozen  = Parameters are fine-tuned
What are Adapters?

A single Transformer (encoder) layer

\[ \Theta \leftarrow \arg\min_{\Theta} L(D_{\text{NLI}}; \Theta) \]

- Parameters are frozen
- Parameters are fine-tuned

What are Adapters?

A single Transformer (encoder) layer

\[ \Theta \leftarrow \arg\min_{\Theta} L(D_{\text{NLI}}; \Theta) \]

= Parameters are frozen  = Parameters are fine-tuned

What are Adapters?

A single Transformer (encoder) layer

\[ \theta \leftarrow \arg\min_{\theta} L(D_{\text{NLI}}; \theta) \]

What are Adapters?

A single Transformer (encoder) layer

Adapter parameters $\phi$ are 
**encapsulated** between transformer layers with parameters $\Theta$

$$\Theta \leftarrow \text{argmin}_{\Theta} L(D_{\text{NLI}}; \Theta)$$

What are Adapters?

A single Transformer (encoder) layer

Adapter parameters $\phi$ are **encapsulated** between transformer layers with parameters $\Theta$ which are frozen

$$\phi \leftarrow \arg\min_{\phi} L(D_{\text{NLI}}; \Theta, \phi)$$

Encapsulated Adapters? 😐

- Adapters learn transformations that make the underlying model more suited to a task or language.
Encapsulated Adapters? 😐

- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.
MLM (English)

- Adapters learn transformations that make the underlying model more suited to a task or language.
- Using masked language modelling (MLM), we can learn language-specific transformations for e.g. English and Quechua.
Encapsulated Adapters?

- Adapters learn transformations that make the underlying model more suited to a task or language.
- Using masked language modelling (MLM), we can learn language-specific transformations for e.g. English and Quechua.
Encapsulated Adapters? 😐

- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.

MLM (English)  
MLM (Quechuan)
MLM (English)  MLM (Quechuan)

- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.
Encapsulated Adapters?

MLM (English)

MLM (Quechuan)

- Adapters learn transformations that make the underlying model more suited to a task or language.
- Using masked language modelling (MLM), we can learn language-specific transformations for e.g. English and Quechua.
- As long as the underlying model is kept fixed, these transformations are roughly interchangeable.
Encapsulated Adapters?

- Adapters learn transformations that make the underlying model more suited to a task or language.
- Using masked language modelling (MLM), we can learn language-specific transformations for e.g. English and Quechua.
- As long as the underlying model is kept fixed, these transformations are roughly interchangeable.
Parameter Efficiency of Adapters in Transformers

Training adapters instead of full model fine-tuning achieves similar results.

Adapters are smaller in size than training the full model.

Performance on GLUE tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Full</th>
<th>Pfeif.</th>
<th>Houlsby</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTE (Wang et al., 2018)</td>
<td>66.2</td>
<td>70.8</td>
<td>69.8</td>
</tr>
<tr>
<td>MRPC (Dolan and Brockett, 2005)</td>
<td>90.5</td>
<td>89.7</td>
<td>91.5</td>
</tr>
<tr>
<td>STS-B (Cer et al., 2017)</td>
<td>88.8</td>
<td>89.0</td>
<td>89.2</td>
</tr>
<tr>
<td>CoLA (Warstadt et al., 2019)</td>
<td>59.5</td>
<td>58.9</td>
<td>59.1</td>
</tr>
<tr>
<td>SST-2 (Socher et al., 2013)</td>
<td>92.6</td>
<td>92.2</td>
<td>92.8</td>
</tr>
<tr>
<td>QNLI (Rajpurkar et al., 2016)</td>
<td>91.3</td>
<td>91.3</td>
<td>91.2</td>
</tr>
<tr>
<td>MNLI (Williams et al., 2018)</td>
<td>84.1</td>
<td>84.1</td>
<td>84.1</td>
</tr>
<tr>
<td>QQP (Iyer et al., 2017)</td>
<td>91.4</td>
<td>90.5</td>
<td>90.8</td>
</tr>
</tbody>
</table>

Number of newly introduced parameters

<table>
<thead>
<tr>
<th>CRate</th>
<th>Base #Params</th>
<th>Base Size</th>
<th>Large #Params</th>
<th>Large Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.2M</td>
<td>0.9Mb</td>
<td>0.8M</td>
<td>3.2Mb</td>
</tr>
<tr>
<td>16</td>
<td>0.9M</td>
<td>3.5Mb</td>
<td>3.1M</td>
<td>13Mb</td>
</tr>
<tr>
<td>2</td>
<td>7.1M</td>
<td>28Mb</td>
<td>25.2M</td>
<td>97Mb</td>
</tr>
</tbody>
</table>

Agenda

1. Adapters in Transformers
2. AdapterFusion
3. MAD-X
4. Efficiency of Adapters
5. AdapterHub.ml
AdapterFusion: Non-Destructive Task Composition for Transfer Learning

Proceedings of EACL 2021

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, Iryna Gurevych
Problems of Multi-Task and Transfer Learning
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$

Task 1
Task 2
Task 2
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$

Task 1

Task 2

Catastrophic Interference: Sharing all parameters $\Theta$ between tasks results in deterioration of performance for a subset of tasks.
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$

Task 1

Task 2

Catastrophic Interference: Sharing all parameters $\Theta$ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$

- Task 1
- Task 2

Catastrophic Interference: Sharing all parameters $\Theta$ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:

Model; $\Theta_0$
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$

Task 1
Task 2

Catastrophic Interference: Sharing all parameters $\Theta$ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:

Model; $\Theta^0$

Task 1
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$

Task 1
Task 2
Task 2

Catastrophic Interference: Sharing all parameters $\Theta$ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:

Model; $\Theta^0$

Task 1

Model; $\Theta^1$
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

- **MT-Model; \( \Theta \)**
  - Task 1
  - Task 2

  **Catastrophic Interference**: Sharing all parameters \( \Theta \) between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:

- **Model; \( \Theta^0 \)**
- **Model; \( \Theta^1 \)**
  - Task 1
  - Task 2
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

**Catastrophic Interference**: Sharing all parameters $\Theta$ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\Theta$

- Task 1
- Task 2

Catastrophic Interference: Sharing all parameters $\Theta$ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:

Model; $\Theta^0$
→ Model; $\Theta^1$
→ Model; $\Theta^2$

- Task 1
- Task 2

Catastrophic Forgetting: Sequential fine-tuning on tasks results in forgetting information learned in earlier stages of transfer learning.
Problems of Multi-Task and Transfer Learning
Problems of Multi-Task and Transfer Learning

How to mitigate?
Problems of Multi-Task and Transfer Learning

How to mitigate?

1. Train **task-specific weights** (adapters) for each task.
Problems of Multi-Task and Transfer Learning

How to mitigate?

1. Train task-specific weights (adapters) for each task.

=> No information can be “forgotten” as pre-trained weights are not overwritten.
Problems of Multi-Task and Transfer Learning

How to mitigate?

1. Train **task-specific weights** (adapters) for each task.

   => No information can be “forgotten” as pre-trained weights are not overwritten.

   => Tasks do not “interfere”, as they have designated parameters.
Problems of Multi-Task and Transfer Learning

How to mitigate?

1. Train task-specific weights (adapters) for each task.

   => No information can be “forgotten” as pre-trained weights are not overwritten.

   => Tasks do not “interfere”, as they have designated parameters.

1. Combine the representations subsequently.
Assuming...
Sharing Information across multiple tasks

Given a pool of adapters, we want to leverage the stored information to solve a new task:

Sharing Information across multiple tasks

Given a pool of adapters, we want to leverage the stored information to solve a new task:

Learn dynamic attention weighting on a target task given the representations of the given Adapters

“Solving” Catastrophic interference and forgetting

Because of **task specific weights** and **residual connections** the model can opt-in and opt-out of leveraging information stored within adapters.

Performance of **AdapterFusion**

Performance of **Full finetuning vs.** Single Task Adapters (**ST-A**) vs. Fusion with Single Task Adapters (**F. w/ ST-A**).

We find that **AdapterFusion** performs well for lower resource datasets where less than **10k** examples exist.
Performance of AdapterFusion

Performance of **Full finetuning vs.**
Single Task Adapters (**ST-A**) **vs.**
Fusion with Single Task Adapters (**F. w/ ST-A**).

We find that **AdapterFusion** performs well for lower resource datasets where less than **10k** examples exist.
Performance of AdapterFusion

Performance of **Full finetuning vs. Single Task Adapters vs. Fusion with Single Task Adapters.**


<table>
<thead>
<tr>
<th>Dataset</th>
<th>Full</th>
<th>ST-A</th>
<th>F. w/ ST-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNLI</td>
<td>83.17</td>
<td>84.32</td>
<td>84.28</td>
</tr>
<tr>
<td>QQP</td>
<td>90.87</td>
<td>90.59</td>
<td>90.71</td>
</tr>
<tr>
<td>SST</td>
<td>92.39 ± 0.22</td>
<td>91.85 ± 0.41</td>
<td>92.20 ± 0.18</td>
</tr>
<tr>
<td>WGrande</td>
<td>60.01 ± 0.08</td>
<td>61.09 ± 0.11</td>
<td>60.23 ± 0.31</td>
</tr>
<tr>
<td>IMDB</td>
<td>94.05 ± 0.21</td>
<td>93.85 ± 0.07</td>
<td>93.82 ± 0.39</td>
</tr>
<tr>
<td>HSswag</td>
<td>39.25 ± 0.76</td>
<td>38.11 ± 0.14</td>
<td>37.98 ± 0.01</td>
</tr>
<tr>
<td>SocIQA</td>
<td>62.05 ± 0.04</td>
<td>62.41 ± 0.11</td>
<td>63.16 ± 0.24</td>
</tr>
<tr>
<td>CosQA</td>
<td>60.28 ± 0.40</td>
<td>60.01 ± 0.02</td>
<td>60.65 ± 0.55</td>
</tr>
<tr>
<td>SciTail</td>
<td>94.32 ± 0.11</td>
<td>93.90 ± 0.16</td>
<td>94.04 ± 0.23</td>
</tr>
<tr>
<td>Argument</td>
<td>76.87 ± 0.32</td>
<td>77.65 ± 0.34</td>
<td>77.65 ± 0.21</td>
</tr>
<tr>
<td>CSQA</td>
<td>58.88 ± 0.40</td>
<td>58.91 ± 0.57</td>
<td>59.73 ± 0.54</td>
</tr>
<tr>
<td>BoolQ</td>
<td>74.84 ± 0.24</td>
<td>75.66 ± 1.25</td>
<td>76.25 ± 0.19</td>
</tr>
<tr>
<td>MRPC</td>
<td>85.14 ± 0.45</td>
<td>85.16 ± 0.52</td>
<td>90.29 ± 0.84</td>
</tr>
<tr>
<td>SICK</td>
<td>87.30 ± 0.42</td>
<td>86.20 ± 0.00</td>
<td>87.28 ± 0.99</td>
</tr>
<tr>
<td>RTE</td>
<td>65.41 ± 0.90</td>
<td>71.04 ± 1.62</td>
<td>76.82 ± 1.68</td>
</tr>
<tr>
<td>CB</td>
<td>82.49 ± 2.33</td>
<td>86.07 ± 3.87</td>
<td>92.14 ± 0.97</td>
</tr>
</tbody>
</table>

Mean: 75.46, 76.05, 77.33
Performance of **Full finetuning vs. Single Task Adapters vs. Fusion with Single Task Adapters.**

We find that **AdapterFusion** performs well for lower resource datasets where less than 10k examples exist.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Full</th>
<th>ST-A</th>
<th>F. w/ ST-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNLI</td>
<td>83.17</td>
<td>84.32</td>
<td>84.28</td>
</tr>
<tr>
<td>QQP</td>
<td>90.87</td>
<td>90.59</td>
<td>90.71</td>
</tr>
<tr>
<td>SST</td>
<td>92.39 ±0.22</td>
<td>91.85 ±0.41</td>
<td>92.20 ±0.18</td>
</tr>
<tr>
<td>WGrande</td>
<td>60.01 ±0.08</td>
<td>61.09 ±0.11</td>
<td>60.23 ±0.31</td>
</tr>
<tr>
<td>IMDB</td>
<td>94.05 ±0.21</td>
<td>93.85 ±0.07</td>
<td>93.82 ±0.39</td>
</tr>
<tr>
<td>HSswag</td>
<td>39.25 ±0.76</td>
<td>38.11 ±0.14</td>
<td>37.98 ±0.01</td>
</tr>
<tr>
<td>SocIQA</td>
<td>62.05 ±0.04</td>
<td>62.41 ±0.11</td>
<td>63.16 ±0.24</td>
</tr>
<tr>
<td>CosQA</td>
<td>60.28 ±0.40</td>
<td>60.01 ±0.02</td>
<td>60.65 ±0.55</td>
</tr>
<tr>
<td>SciTail</td>
<td>94.32 ±0.11</td>
<td>93.90 ±0.16</td>
<td>94.04 ±0.23</td>
</tr>
<tr>
<td>Argument</td>
<td>76.87 ±0.32</td>
<td>77.65 ±0.34</td>
<td>77.65 ±0.21</td>
</tr>
<tr>
<td>CSQA</td>
<td>58.88 ±0.40</td>
<td>58.91 ±0.57</td>
<td>59.73 ±0.54</td>
</tr>
<tr>
<td>BoolQ</td>
<td>74.84 ±0.24</td>
<td>75.66 ±1.25</td>
<td>76.25 ±0.19</td>
</tr>
<tr>
<td>MRPC</td>
<td>85.14 ±0.45</td>
<td>85.16 ±0.52</td>
<td>90.29 ±0.84</td>
</tr>
<tr>
<td>SICK</td>
<td>87.30 ±0.42</td>
<td>86.20 ±0.00</td>
<td>87.28 ±0.99</td>
</tr>
<tr>
<td>RTE</td>
<td>65.41 ±0.90</td>
<td>71.04 ±1.62</td>
<td>76.82 ±1.68</td>
</tr>
<tr>
<td>CB</td>
<td>82.49 ±2.33</td>
<td>86.07 ±3.87</td>
<td>92.14 ±0.97</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>75.46</td>
<td>76.05</td>
<td>77.33</td>
</tr>
</tbody>
</table>

Performance of AdapterFusion

Performance of **Full finetuning vs. Single Task Adapters vs. Fusion with Single Task Adapters.**

We find that AdapterFusion performs well for lower resource datasets where less than 10k examples exist.

AdapterFusion is able to maintain performance for high resource datasets. It learns to activate only its own adapter.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Full</th>
<th>ST-A</th>
<th>F. w/ ST-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNLI</td>
<td>83.17</td>
<td>84.32</td>
<td>84.28</td>
</tr>
<tr>
<td>QQP</td>
<td>90.87</td>
<td>90.59</td>
<td>90.71</td>
</tr>
<tr>
<td>SST</td>
<td>92.39 ±0.22</td>
<td>91.85 ±0.41</td>
<td>92.20 ±0.18</td>
</tr>
<tr>
<td>WGrande</td>
<td>60.01 ±0.08</td>
<td>61.09 ±0.11</td>
<td>60.23 ±0.31</td>
</tr>
<tr>
<td>IMDB</td>
<td>94.05 ±0.21</td>
<td>93.85 ±0.07</td>
<td>93.82 ±0.39</td>
</tr>
<tr>
<td>HSwag</td>
<td>39.25 ±0.76</td>
<td>38.11 ±0.14</td>
<td>37.98 ±0.01</td>
</tr>
<tr>
<td>SocIQA</td>
<td>62.05 ±0.04</td>
<td>62.41 ±0.11</td>
<td>63.16 ±0.24</td>
</tr>
<tr>
<td>CosQA</td>
<td>60.28 ±0.40</td>
<td>60.01 ±0.02</td>
<td>60.65 ±0.55</td>
</tr>
<tr>
<td>SciTail</td>
<td>94.32 ±0.11</td>
<td>93.90 ±0.16</td>
<td>94.04 ±0.23</td>
</tr>
<tr>
<td>Argument</td>
<td>76.87 ±0.32</td>
<td>77.65 ±0.34</td>
<td>77.65 ±0.21</td>
</tr>
<tr>
<td>CSQA</td>
<td>58.88 ±0.40</td>
<td>58.91 ±0.57</td>
<td>59.73 ±0.54</td>
</tr>
<tr>
<td>BoolQ</td>
<td>74.84 ±0.24</td>
<td>75.66 ±1.25</td>
<td>76.25 ±0.19</td>
</tr>
<tr>
<td>MRPC</td>
<td>85.14 ±0.45</td>
<td>85.16 ±0.52</td>
<td>90.29 ±0.84</td>
</tr>
<tr>
<td>SICK</td>
<td>87.30 ±0.42</td>
<td>86.20 ±0.00</td>
<td>87.28 ±0.99</td>
</tr>
<tr>
<td>RTE</td>
<td>65.41 ±0.90</td>
<td>71.04 ±1.62</td>
<td>76.82 ±1.68</td>
</tr>
<tr>
<td>CB</td>
<td>82.49 ±2.33</td>
<td>86.07 ±3.87</td>
<td><strong>92.14 ±0.97</strong></td>
</tr>
</tbody>
</table>

Mean        | 75.46 | 76.05 | **77.33**

Agenda

1. Adapters in Transformers
2. AdapterFusion
3. MAD-X
4. Efficiency of Adapters
5. AdapterHub.ml
MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer

Proceedings of EMNLP 2020

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, Sebastian Ruder
Task: Zero-shot transfer to low-resource languages
Task: Zero-shot transfer to low-resource languages

Step 1:

Train a multilingual model.
Task: Zero-shot transfer to low-resource languages

Step 1:

**Train** a multilingual model.

Step 2:

**Fine-tune** model on a **task** in a high resource **source language**.
Task: Zero-shot transfer to low-resource languages

Step 1:
Train a multilingual model.

Step 2:
Fine-tune model on a task in a high resource source language.

Step 3:
Transfer and evaluate the model on a low resource target language.
Task: Zero-shot transfer to low-resource languages

Step 1:

Train a multilingual model.

Step 2:

Fine-tune model on a task in a high resource source language.

Step 3:

Transfer and evaluate the model on a low resource target language.

Why?

Training data is expensive and not available for many languages, especially ones that are considered “low-resource”.
Related Work & Baselines
Related Work & Baselines

Deep massively multilingual models such as **multilingual-BERT** (mBERT; Devlin et al. 2019) or **XLM-RoBERTa** (XLM-R; Conneau et al. 2020) achieve
Related Work & Baselines

Deep massively multilingual models such as **multilingual-BERT** (mBERT; Devlin et al. 2019) or **XLM-RoBERTa** (XLM-R; Conneau et al. 2020) achieve

+ SotA results on cross-lingual transfer
Deep massively multilingual models such as multilingual-BERT (mBERT; Devlin et al. 2019) or XLM-RoBERTa (XLM-R; Conneau et al. 2020) achieve

+ SotA results on cross-lingual transfer

BUT
Related Work & Baselines

Deep massively multilingual models such as **multilingual-BERT** (mBERT; Devlin et al. 2019) or **XLM-RoBERTa** (XLM-R; Conneau et al. 2020) achieve

+ SotA results on cross-lingual transfer

**BUT**

- Suffer from “the curse of multilinguality” (Conneau et al. 2020)
Related Work & Baselines

Deep massively multilingual models such as multilingual-BERT (mBERT; Devlin et al. 2019) or XLM-RoBERTa (XLM-R; Conneau et al. 2020) achieve

+ SotA results on cross-lingual transfer

BUT

- Suffer from “the curse of multilinguality” (Conneau et al. 2020)
  - and cannot represent all (7000+) languages in a single model.
- performance especially deteriorates for low resource languages not covered in the training data. (Ponti et al. 2020)
Our Approach
Our Approach

Assumption:
Our Approach

Assumption:

Massive multilinguality of mBERT and XLM-R
Our Approach

Assumption:

Massive multilinguality of mBERT and XLM-R

=> perfect for transfer learning to unseen languages.
Our Approach

Assumption:

Massive multilinguality of mBERT and XLM-R

=> perfect for transfer learning to unseen languages.

We propose MAD-X, that incorporates Adapters (Houlsby et al. 2018) for
Our Approach

Assumption:

Massive multilinguality of mBERT and XLM-R

=> perfect for transfer learning to unseen languages.

We propose MAD-X, that incorporates Adapters (Houlsby et al. 2018) for

- Languages (seen e.g. English, Chinese, and unseen, Quechuan, Guarani)
Our Approach

Assumption:

Massive multilinguality of mBERT and XLM-R

=> perfect for transfer learning to unseen languages.

We propose MAD-X, that incorporates Adapters (Houlsby et al. 2018) for

- Languages (seen e.g. English, Chinese, and unseen, Quechuan, Guarani)
- Tasks (e.g. NER, COPA, SQuAD)
Our Approach

Assumption:

**Massive multilinguality** of mBERT and XLM-R

=> perfect for transfer learning to unseen languages.

We propose **MAD-X**, that incorporates Adapters (Houlsby et al. 2018) for

- **Languages** (seen e.g. English, Chinese, and unseen, Quechuan, Guarani)
- **Tasks** (e.g. NER, COPA, SQuAD)

“Language agnostic” task-adapters are stacked on top of language adapters for zero-shot transfer to unseen languages.
MAD-X

Step 1: Train Language Adapters

We train language adapters for the source language and the target language with masked language modelling on Wikipedia.
MAD-X

Step 2: Train a Task Adapter
MAD-X

Step 2: Train a Task Adapter

We train task adapters in the source language stacked on top of the source language adapter.
MAD-X

Step 2: Train a Task Adapter

We train **task adapters** in the source language **stacked** on top of the source **language adapter**.

The **language adapter** $\phi_I$ as well as the transformer weights $\Theta$ are frozen while only the **task adapter** parameter $\phi_t$ are trained.
MAD-X

Step 2: Train a Task Adapter

We train task adapters in the source language stacked on top of the source language adapter.

The language adapter $\phi_l$ as well as the transformer weights $\Theta$ are frozen while only the task adapter parameter $\phi_t$ are trained.
MAD-X

Step 3: Zero-Shot transfer to unseen language
MAD-X

Step 3: Zero-Shot transfer to unseen language

We replace the **source** language adapter with the **target** language adapter, while keeping the “language agnostic” **task adapter**.
MAD-X

Step 3: Zero-Shot transfer to unseen language

We replace the source language adapter with the target language adapter, while keeping the “language agnostic” task adapter.
Datasets

NER: WikiAnn Dataset (Pan et al. 2017, Rahimi et al. 2019). We chose a diverse set of languages from different language families.

XQuAD (Cross-lingual Question Answering Dataset) (Artetxe et al. 2020)

XCOPA (Ponti et al. 2020b)

<table>
<thead>
<tr>
<th>Language</th>
<th>ISO code</th>
<th>Language family</th>
<th># of Wiki articles</th>
<th>Covered by SOTA?</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>en</td>
<td>Indo-European</td>
<td>6.0M</td>
<td>✓</td>
</tr>
<tr>
<td>Japanese</td>
<td>ja</td>
<td>Japonic</td>
<td>1.2M</td>
<td>✓</td>
</tr>
<tr>
<td>Chinese</td>
<td>zh</td>
<td>Sino-Tibetan</td>
<td>1.1M</td>
<td>✓</td>
</tr>
<tr>
<td>Arabic</td>
<td>ar</td>
<td>Afro-Asiatic</td>
<td>1.0M</td>
<td>✓</td>
</tr>
<tr>
<td>Javanese</td>
<td>jv</td>
<td>Austronesian</td>
<td>57k</td>
<td>✓</td>
</tr>
<tr>
<td>Swahili</td>
<td>sw</td>
<td>Niger-Congo</td>
<td>56k</td>
<td>✓</td>
</tr>
<tr>
<td>Icelandic</td>
<td>is</td>
<td>Indo-European</td>
<td>49k</td>
<td>✓</td>
</tr>
<tr>
<td>Burmese</td>
<td>my</td>
<td>Sino-Tibetan</td>
<td>45k</td>
<td>✓</td>
</tr>
<tr>
<td>Quechua</td>
<td>qu</td>
<td>Quechua</td>
<td>22k</td>
<td>✓</td>
</tr>
<tr>
<td>Min Dong</td>
<td>cdo</td>
<td>Sino-Tibetan</td>
<td>15k</td>
<td>✓</td>
</tr>
<tr>
<td>Ilokano</td>
<td>ilo</td>
<td>Austronesian</td>
<td>14k</td>
<td>✓</td>
</tr>
<tr>
<td>Mingrelian</td>
<td>xmf</td>
<td>Kartvelian</td>
<td>13k</td>
<td>✓</td>
</tr>
<tr>
<td>Meadow Mari</td>
<td>mhr</td>
<td>Uralic</td>
<td>10k</td>
<td></td>
</tr>
<tr>
<td>Maori</td>
<td>mi</td>
<td>Austronesian</td>
<td>7k</td>
<td></td>
</tr>
<tr>
<td>Turkmen</td>
<td>tk</td>
<td>Turkic</td>
<td>6k</td>
<td></td>
</tr>
<tr>
<td>Guarani</td>
<td>gn</td>
<td>Tupian</td>
<td>4k</td>
<td></td>
</tr>
</tbody>
</table>
Baselines

Standard Zero-Shot
Baselines

Standard Zero-Shot

mBERT
Baselines

Standard Zero-Shot

1. **Fine-tune** a multilingual model on the task in the *source* language.
Baselines

Standard Zero-Shot

1. **Fine-tune** a multilingual model on the task in the **source** language.

2. **Evaluate** on the **target** language.

![Diagram showing mBERT connected to English NER and Quechua NER]
Baselines

Standard Zero-Shot

1. Fine-tune a multilingual model on the task in the source language.

2. Evaluate on the target language.

Target Language Fine-Tuning
Baselines

Standard Zero-Shot

1. **Fine-tune** a multilingual model on the task in the **source** language.
2. Evaluate on the **target** language.

Target Language Fine-Tuning

1. **Fine-tune** a multilingual model using **MLM** on a corpus of the **target** language.
Baselines

**Standard Zero-Shot**

1. **Fine-tune** a multilingual model on the task in the *source* language.

2. **Evaluate** on the *target* language.

**Target Language Fine-Tuning**

1. **Fine-tune** a multilingual model using **MLM** on a corpus of the *target* language.

2. **Fine-tune** a multilingual model on the task in the *source* language.
Baselines

Standard Zero-Shot

1. **Fine-tune** a multilingual model on the task in the *source* language.
2. **Evaluate** on the *target* language.

Target Language Fine-Tuning

1. **Fine-tune** a multilingual model using **MLM** on a corpus of the *target* language.
2. **Fine-tune** a multilingual model on the task in the *source* language.
3. **Evaluate** on the *target* language.
Results NER

NER F1 scores averaged over all 16 target languages when transferring from each source language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.
Results NER

NER F1 scores averaged over all 16 target languages when transferring from each source language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.
Results NER

NER F1 scores averaged over all 16 target languages when transferring from each source language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.
Results NER

NER F1 scores averaged over all 16 target languages when transferring from each source language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.
Results NER

NER F1 scores averaged over all 16 target languages when transferring from each source language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.
Results NER

NER F1 scores averaged over all 16 target languages when transferring from each source language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.
Languages are more low-resource or unseen during pre-training

<table>
<thead>
<tr>
<th></th>
<th>en</th>
<th>ja</th>
<th>zh</th>
<th>ar</th>
<th>jv</th>
<th>sw</th>
<th>is</th>
<th>my</th>
<th>qu</th>
<th>cdo</th>
<th>ilo</th>
<th>xmf</th>
<th>mi</th>
<th>mhr</th>
<th>tk</th>
<th>gn</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>en</strong></td>
<td>0.2</td>
<td>-0.7</td>
<td>0.3</td>
<td>11.8</td>
<td>5.8</td>
<td>7.7</td>
<td>4.7</td>
<td>5.9</td>
<td>19.7</td>
<td>26.0</td>
<td>8.1</td>
<td>15.9</td>
<td>10.2</td>
<td>12.8</td>
<td>15.5</td>
<td>7.3</td>
</tr>
<tr>
<td><strong>ja</strong></td>
<td>-0.3</td>
<td>0.8</td>
<td>4.0</td>
<td>-4.4</td>
<td>2.0</td>
<td>4.2</td>
<td>0.5</td>
<td>6.3</td>
<td>14.6</td>
<td>37.4</td>
<td>-3.6</td>
<td>16.3</td>
<td>23.7</td>
<td>2.9</td>
<td>2.2</td>
<td>2.8</td>
</tr>
<tr>
<td><strong>zh</strong></td>
<td>4.7</td>
<td>3.4</td>
<td>-0.1</td>
<td>2.6</td>
<td>4.7</td>
<td>6.7</td>
<td>1.3</td>
<td>21.2</td>
<td>36.6</td>
<td>3.2</td>
<td>13.0</td>
<td>26.8</td>
<td>15.8</td>
<td>-0.7</td>
<td>8.7</td>
<td></td>
</tr>
<tr>
<td><strong>ar</strong></td>
<td>7.4</td>
<td>-1.7</td>
<td>-1.6</td>
<td>0.7</td>
<td>9.8</td>
<td>12.1</td>
<td>9.8</td>
<td>-4.3</td>
<td>24.5</td>
<td>44.8</td>
<td>26.9</td>
<td>19.2</td>
<td>20.7</td>
<td>21.7</td>
<td>20.1</td>
<td>4.4</td>
</tr>
<tr>
<td><strong>jv</strong></td>
<td>-8.4</td>
<td>-3.5</td>
<td>-5.0</td>
<td>-5.7</td>
<td><strong>2.9</strong></td>
<td>-2.2</td>
<td>0.5</td>
<td>-1.0</td>
<td>3.7</td>
<td>18.3</td>
<td>4.1</td>
<td>-3.1</td>
<td>8.2</td>
<td>6.5</td>
<td>7.5</td>
<td>-3.9</td>
</tr>
<tr>
<td><strong>sw</strong></td>
<td>-1.4</td>
<td>-4.4</td>
<td>-8.3</td>
<td>-2.9</td>
<td>5.1</td>
<td><strong>2.2</strong></td>
<td>3.8</td>
<td>1.4</td>
<td>17.1</td>
<td>28.6</td>
<td>16.7</td>
<td>11.2</td>
<td>9.3</td>
<td>13.8</td>
<td>14.4</td>
<td>-1.0</td>
</tr>
<tr>
<td><strong>is</strong></td>
<td>-3.2</td>
<td>-4.4</td>
<td>-7.7</td>
<td>-10.8</td>
<td>9.8</td>
<td>-7.4</td>
<td><strong>1.6</strong></td>
<td>0.6</td>
<td>7.3</td>
<td>27.8</td>
<td>3.4</td>
<td>7.7</td>
<td>12.6</td>
<td>14.2</td>
<td>10.4</td>
<td>10.5</td>
</tr>
<tr>
<td><strong>my</strong></td>
<td>-7.5</td>
<td>-1.7</td>
<td>-3.1</td>
<td>-9.4</td>
<td>5.3</td>
<td>-3.7</td>
<td>-2.8</td>
<td><strong>10.6</strong></td>
<td>-3.3</td>
<td>15.9</td>
<td>-3.9</td>
<td>-0.5</td>
<td>-4.5</td>
<td>2.2</td>
<td>2.6</td>
<td>-1.3</td>
</tr>
<tr>
<td><strong>qu</strong></td>
<td>-4.5</td>
<td>-0.2</td>
<td>0.2</td>
<td>-5.8</td>
<td>4.7</td>
<td>-0.1</td>
<td>12.1</td>
<td>0.4</td>
<td><strong>9.0</strong></td>
<td>25.3</td>
<td>-0.3</td>
<td>18.4</td>
<td>22.1</td>
<td>9.1</td>
<td>16.4</td>
<td>13.1</td>
</tr>
<tr>
<td><strong>cdo</strong></td>
<td>-13.7</td>
<td>-0.0</td>
<td>-1.3</td>
<td>7.2</td>
<td>15.4</td>
<td>5.9</td>
<td>22.3</td>
<td>15.7</td>
<td>5.9</td>
<td><strong>16.3</strong></td>
<td>-0.5</td>
<td>22.7</td>
<td>18.0</td>
<td>9.4</td>
<td>16.6</td>
<td>15.3</td>
</tr>
<tr>
<td><strong>ilo</strong></td>
<td>6.0</td>
<td>-2.9</td>
<td>-5.0</td>
<td>5.2</td>
<td>14.9</td>
<td>12.5</td>
<td>14.6</td>
<td>-0.3</td>
<td>11.6</td>
<td>16.6</td>
<td><strong>12.0</strong></td>
<td>12.3</td>
<td>7.9</td>
<td>14.7</td>
<td>6.1</td>
<td>4.6</td>
</tr>
<tr>
<td><strong>xmf</strong></td>
<td>-4.5</td>
<td>-0.2</td>
<td>1.7</td>
<td>-9.2</td>
<td>-0.3</td>
<td>-5.9</td>
<td>-3.0</td>
<td>-5.7</td>
<td>3.0</td>
<td>24.3</td>
<td>-5.8</td>
<td><strong>0.1</strong></td>
<td>-6.0</td>
<td>5.8</td>
<td>4.9</td>
<td>-4.2</td>
</tr>
<tr>
<td><strong>mi</strong></td>
<td>-1.0</td>
<td>-0.1</td>
<td>-0.4</td>
<td>-1.4</td>
<td>4.0</td>
<td>-1.8</td>
<td>8.5</td>
<td>2.3</td>
<td>5.0</td>
<td>10.9</td>
<td>-3.5</td>
<td>19.8</td>
<td><strong>5.2</strong></td>
<td>9.0</td>
<td>8.5</td>
<td>6.1</td>
</tr>
<tr>
<td><strong>mhr</strong></td>
<td>-5.6</td>
<td>-2.5</td>
<td>-4.8</td>
<td>-7.0</td>
<td>-2.0</td>
<td>0.7</td>
<td>2.6</td>
<td>-0.3</td>
<td>3.8</td>
<td>10.0</td>
<td>-10.3</td>
<td>-0.4</td>
<td>9.0</td>
<td><strong>7.6</strong></td>
<td>5.1</td>
<td>-0.3</td>
</tr>
<tr>
<td><strong>tk</strong></td>
<td>-2.7</td>
<td>-1.7</td>
<td>-3.8</td>
<td>-5.5</td>
<td>6.3</td>
<td>-9.5</td>
<td>3.0</td>
<td>-3.5</td>
<td>2.7</td>
<td>14.4</td>
<td>11.2</td>
<td>6.7</td>
<td>3.2</td>
<td><strong>11.2</strong></td>
<td><strong>12.5</strong></td>
<td>-1.9</td>
</tr>
<tr>
<td><strong>gn</strong></td>
<td>-16.1</td>
<td>-1.6</td>
<td>-3.6</td>
<td>14.5</td>
<td>13.8</td>
<td>11.6</td>
<td>9.6</td>
<td>-6.2</td>
<td>-9.4</td>
<td>8.7</td>
<td>14.6</td>
<td>-9.9</td>
<td>-5.4</td>
<td>-4.2</td>
<td>-8.5</td>
<td><strong>1.9</strong></td>
</tr>
</tbody>
</table>
Relative F1 improvement of MAD-X<sub>Large</sub> over XLM-R<sub>Large</sub> in cross-lingual NER transfer.

Languages are more low-resource or unseen during pre-training.

<table>
<thead>
<tr>
<th>Source Language</th>
<th>Target Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>-0.2 -0.7 0.3 -11.8 5.8 7.7 4.7 5.9 19.7 26.0 8.1 15.9 10.2 12.8 15.5 7.3</td>
</tr>
<tr>
<td>ja</td>
<td>-0.3 0.8 4.0 -4.4 2.0 4.2 0.5 6.3 14.6 37.4 -3.6 16.3 23.7 2.9 2.2 2.8</td>
</tr>
<tr>
<td>zh</td>
<td>-4.7 3.4 -0.1 -0.1 2.6 4.7 6.7 1.3 21.2 36.6 3.2 13.0 26.8 15.8 -0.7 8.7</td>
</tr>
<tr>
<td>ar</td>
<td>-7.4 -1.7 -1.6 0.7 9.8 12.1 9.8 -4.3 24.5 44.8 26.9 19.2 20.7 21.7 20.1 4.4</td>
</tr>
<tr>
<td>jv</td>
<td>-8.4 -3.5 -5.0 -5.7 2.9 -2.2 0.5 -1.0 3.7 18.3 4.1 -3.1 8.2 6.5 7.5 -3.9</td>
</tr>
<tr>
<td>sw</td>
<td>-1.4 -4.4 -8.3 -2.9 5.1 2.2 3.8 1.4 17.1 28.6 16.7 11.2 9.3 13.8 14.4 -1.0</td>
</tr>
<tr>
<td>ls</td>
<td>-3.2 -4.4 -7.7 -10.8 9.8 -7.4 1.6 0.6 7.3 27.8 3.4 7.7 12.6 14.2 10.4 10.5</td>
</tr>
<tr>
<td>my</td>
<td>-7.5 -1.7 -3.1 -9.4 5.3 -3.7 -2.8 10.6 -3.3 15.9 -3.9 -0.5 -4.5 2.2 2.6 -1.3</td>
</tr>
<tr>
<td>qu</td>
<td>-4.5 -0.2 0.2 -5.8 4.7 -0.1 12.1 0.4 9.0 25.3 -0.3 18.4 22.1 9.1 16.4 13.1</td>
</tr>
<tr>
<td>cdo</td>
<td>-13.7 -0.0 -1.3 7.2 15.4 5.9 22.3 15.7 5.9 16.3 -0.5 22.7 18.0 9.4 16.6 15.3</td>
</tr>
<tr>
<td>ilo</td>
<td>-6.0 -2.9 -5.0 5.2 14.9 12.5 14.6 -0.3 11.6 16.6 12.0 12.3 7.9 14.7 6.1 4.6</td>
</tr>
<tr>
<td>xmf</td>
<td>-4.5 -0.2 1.7 -9.2 -0.3 -5.9 -3.0 -5.7 3.0 24.3 -5.8 0.1 -6.0 5.8 4.9 -4.2</td>
</tr>
<tr>
<td>mi</td>
<td>-1.0 -0.1 -0.4 -1.4 4.0 -1.8 8.5 2.3 5.0 10.9 -3.5 19.8 5.2 9.0 8.5 6.1</td>
</tr>
<tr>
<td>mhr</td>
<td>-5.6 -2.5 -4.8 -7.0 -2.0 0.7 2.6 -0.3 3.8 10.0 -10.3 -0.4 9.0 7.6 5.1 -0.3</td>
</tr>
<tr>
<td>tk</td>
<td>-2.7 -1.7 -3.8 -5.5 6.3 -9.5 3.0 -3.5 2.7 14.4 -11.2 6.7 3.2 11.2 12.5 -1.9</td>
</tr>
<tr>
<td>gn</td>
<td>-16.1 -1.6 -3.6 14.5 -13.8 -11.6 -9.6 -6.2 -9.4 8.7 14.6 -9.9 -5.4 -4.2 -8.5 1.9</td>
</tr>
</tbody>
</table>
Relative **F1 improvement** of MAD-X\textsuperscript{Large} over XLM-R\textsuperscript{Large} in cross-lingual NER transfer.

Top right corner represent the realistic scenario of transferring from high resource to low resource Languages are more low-resource or unseen during pre-training.
Results NER

**Relative F1 improvement of MAD-X\textsuperscript{Large} over XLM-R\textsuperscript{Large} in cross-lingual NER transfer.**

Top right corner represent the realistic scenario of transferring from high resource to low resource.
Agenda

1. Adapters in Transformers
2. AdapterFusion
3. MAD-X
4. Efficiency of Adapters
5. AdapterHub.ml
AdapterDrop: On the Efficiency of Adapters in Transformers

arxiv 2020

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, Iryna Gurevych
AdapterDrop: Training/Inference Efficiency of Adapters

**AdapterDrop: Training/Inference Efficiency of Adapters**

**Training:** Adapters are faster to train because we do not backpropagate through the entire network.

**Inference:** Adapters are slightly slower because of added components.

**AdapterDrop: Training/Inference Efficiency of Adapters**

**Training:** Adapters are faster to train because we do not backpropagate through the entire network.

**Inference:** Adapters are slightly slower because of added components.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Adapter</th>
<th>Relative speed (for Seq.Len./Batch)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>128/16</td>
</tr>
<tr>
<td>Training</td>
<td>Houlsby</td>
<td>1.48</td>
</tr>
<tr>
<td></td>
<td>Pfeiffer</td>
<td>1.57</td>
</tr>
<tr>
<td>Inference</td>
<td>Houlsby</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Pfeiffer</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 1: Relative speed of adapters compared to fully fine-tuned models. For example, 1.6 for training with the Pfeiffer adapter means that we can perform 1.6 training steps with this adapter in the time of one update step with full model fine-tuning.

AdapterDrop: Training/Inference Efficiency of Adapters

Training: Adapters are faster to train because we do not backpropagate through the entire network.

Inference: Adapters are slightly slower because of added components.

AdapterDrop: We can drop adapters at earlier layers. This increases inference speed when predicting multiple tasks simultaneously.

AdapterDrop: Training/Inference Efficiency of Adapters

**Training**: Adapters are faster to train because we do not backpropagate through the entire network.

**Inference**: Adapters are slightly slower because of added components.

**AdapterDrop**: We can drop adapters at earlier layers. This increases inference speed when predicting multiple tasks simultaneously.

---


---

Table 1: Relative speed of adapters compared to fully fine-tuned models. For example, 1.6 for training with the Pfeiffer adapter means that we can perform 1.6 training steps with this adapter in the time of one update step with full model fine-tuning.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Adapter</th>
<th>Relative speed (for Seq.Len./Batch)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>128/16</td>
</tr>
<tr>
<td>Training</td>
<td>Houlsby</td>
<td>1.48</td>
</tr>
<tr>
<td></td>
<td>Pfeiffer</td>
<td>1.57</td>
</tr>
<tr>
<td>Inference</td>
<td>Houlsby</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Pfeiffer</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 2: Speedup for each shared transformer layer when performing inference for multiple tasks simultaneously (details are given in Appendix G.2)
**AdapterDrop**: Training/Inference Efficiency of Adapters

**Training**: Adapters are faster to train because we do not backpropagate through the entire network.

**Inference**: Adapters are slightly slower because of added components.

**AdapterDrop**: We can drop adapters at earlier layers. This increases inference speed when predicting multiple tasks simultaneously.


---

Table 1: Relative speed of adapters compared to fully fine-tuned models. For example, 1.6 for training with the Pfeiffer adapter means that we can perform 1.6 training steps with this adapter in the time of one update step with full model fine-tuning.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Adapter</th>
<th>Relative speed (for Seq.Len./Batch)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>128/16</td>
</tr>
<tr>
<td>Training</td>
<td>Houlsby</td>
<td>1.48</td>
</tr>
<tr>
<td></td>
<td>Pfeiffer</td>
<td>1.57</td>
</tr>
<tr>
<td>Inference</td>
<td>Houlsby</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>Pfeiffer</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 2: Speedup for each shared transformer layer when performing inference for multiple tasks simultaneously (details are given in Appendix G.2)

<table>
<thead>
<tr>
<th>Simultaneous Tasks</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup (each layer)</td>
<td>4.3%</td>
<td>6.6%</td>
<td>7.8%</td>
<td>8.4%</td>
</tr>
</tbody>
</table>

---

[Diagram showing the efficiency of adapters in training and inference, with specific speedup metrics for different tasks and layers.]
Agenda

1. Adapters in Transformers
2. AdapterFusion
3. MAD-X
4. Efficiency of Adapters
5. AdapterHub.ml
AdapterHub: A Framework for Adapting Transformers

Proceedings of EMNLP 2020: Systems Demonstrations

A central repository for pre-trained adapter modules

182 adapters  24 text tasks  32 languages

pip install adapter-transformers

Adapters are Lightweight 😱

"Adapter" refers to a set of newly introduced weights, typically within the layers of a transformer model. Adapters provide an alternative to fully fine-tuning the model for each downstream task, while maintaining performance. They also have the added benefit of requiring as little as 1MB of storage space per task!

Modular, Composable, and Extensible 🎨

Adapters, being self-contained modular units, allow for easy extension and composition. This opens up opportunities to compose adapters to solve new tasks.

Built on HuggingFace 🚀 Transformers

AdapterHub builds on the HuggingFace transformers framework requiring as little as two additional lines of code to train adapters for a downstream task.
Outlook

● Many **alternative** Adapter Approaches
  ○ Diff-Pruning (Guo et al. 2020)
  ○ BitFit (Ben-Zaken et al. 2020)
  ○ Prefix-Tuning (Li et al. 2021)

● How to best **compose** Adapters?

● **Domain** Adapters?

● **Hypernetworks**/CPGs for Adapters (i.e. UDapter (Uestuen et al. 2020))

● Increase the **modularity** of Adapters?

● ...


Key Takeaways
Key Takeaways

- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning.
Key Takeaways

- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning.
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
Key Takeaways

- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning.
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
Key Takeaways

- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning.
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
- **Specializing** the **vocabulary** to the target language, and **leveraging** pre-trained **knowledge** generally **improves** performance.
Key Takeaways

- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning.
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
- **Specializing** the **vocabulary** to the target language, and **leveraging** pre-trained **knowledge** generally **improves** performance.
- Adapters can also be **composed** to combine **information** from multiple tasks for **non-destructive** transfer learning.
Key Takeaways

- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning.
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
- **Specializing** the **vocabulary** to the target language, and leveraging pre-trained **knowledge** generally **improves** performance.
- Adapters can also be **composed** to combine **information** from multiple tasks for **non-destructive** transfer learning.
Key Takeaways

- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning.
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
- **Specializing** the **vocabulary** to the target language, and **leveraging** pre-trained **knowledge** generally **improves** performance.
- Adapters can also be **composed** to combine **information** from multiple tasks for **non-destructive** transfer learning.
References


References


