Adapters in Transformers:

a New Paradigm for Transfer Learning?…

Jonas Pfeiffer (main author)
Iryna Gurevych (PhD advisor)

TECHNISCHE UNIVERSITÄT DARMSTADT

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**).

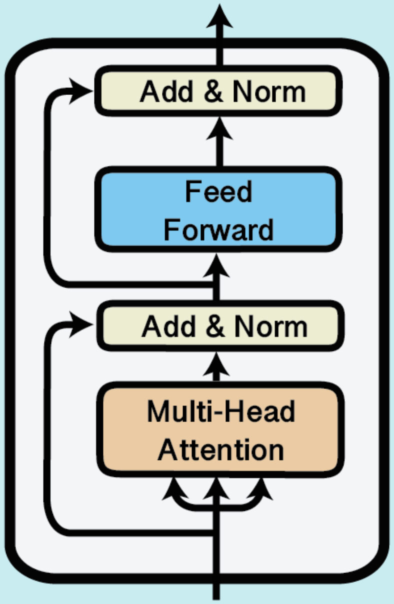
What **you** will learn today

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

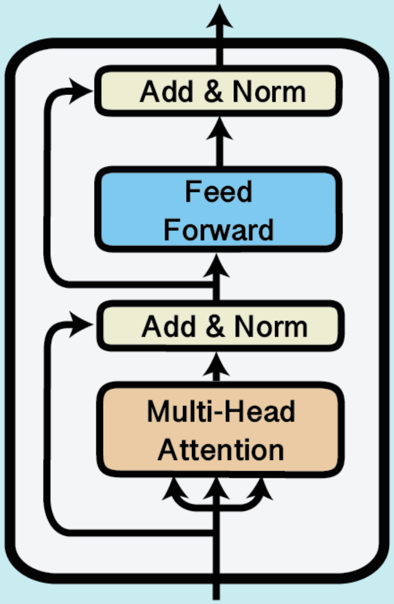
1. What **adapters** (in transformers) are.
2. If adapters really are more **efficient** than normal fine-tuning.
3. How to non-destructively **compose tasks** for transfer learning (**AdapterFusion**).
4. How to **stack modular adapters** for zero-shot transfer to unseen and low-resource languages (**MAD-X**).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

Some Basics...



Some Basics...

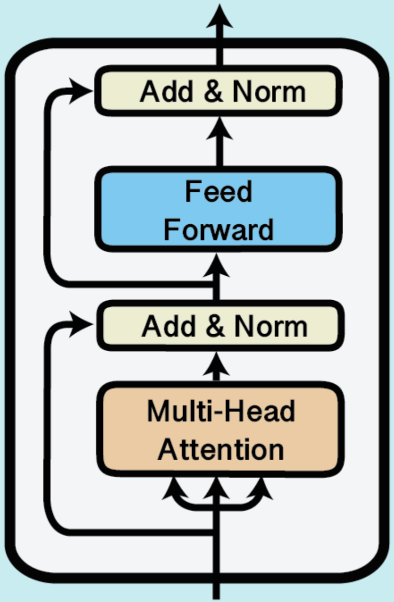
Everything I am about to cover involves:



Some Basics...

Everything I am about to cover involves:

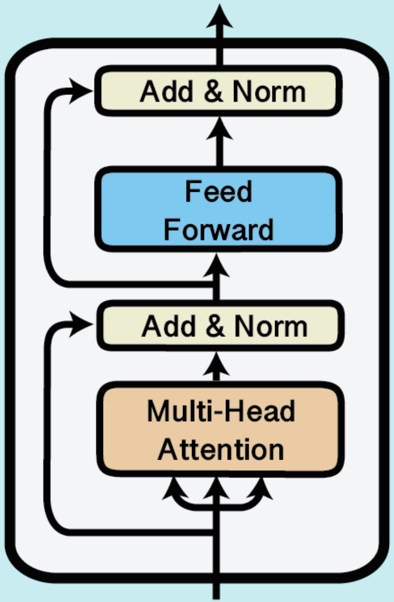
- Transfer Learning in Natural Language Processing (NLP)



Some Basics...

Everything I am about to cover involves:

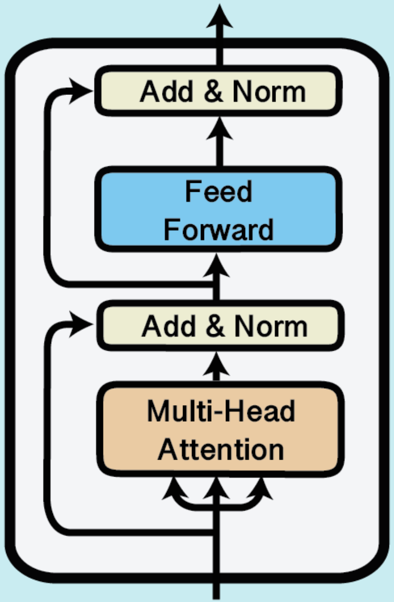
- Transfer Learning in Natural Language Processing (NLP)
- We **only** look at deep neural networks, specifically the **Transformer** architecture.



Some Basics...

Everything I am about to cover involves:

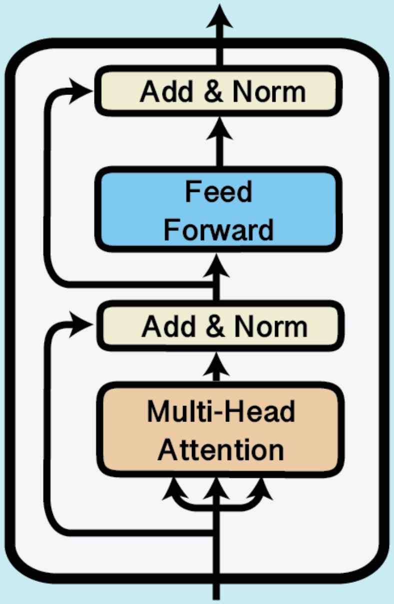
- Transfer Learning in Natural Language Processing (NLP)
- We **only** look at deep neural networks, specifically the **Transformer** architecture.
- We leverage pre-trained transformer-based models such as BERT/RoBERTa/XLM-R/mBERT.



Some Basics...

Everything I am about to cover involves:

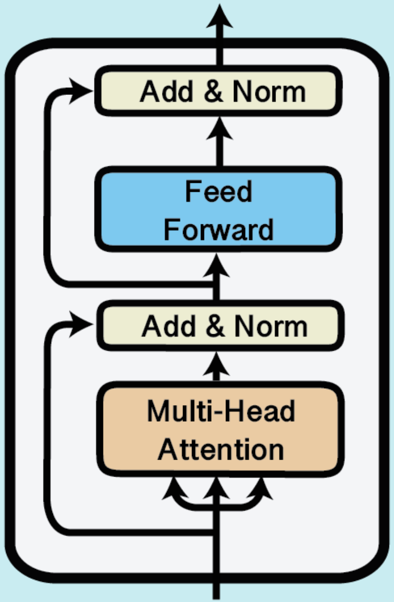
- Transfer Learning in Natural Language Processing (NLP)
- We **only** look at deep neural networks, specifically the **Transformer** architecture.
- We leverage pre-trained transformer-based models such as BERT/RoBERTa/XLM-R/mBERT.
- These have been trained on **massive** amount of text data using **Masked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



Some Basics...

Everything I am about to cover involves:

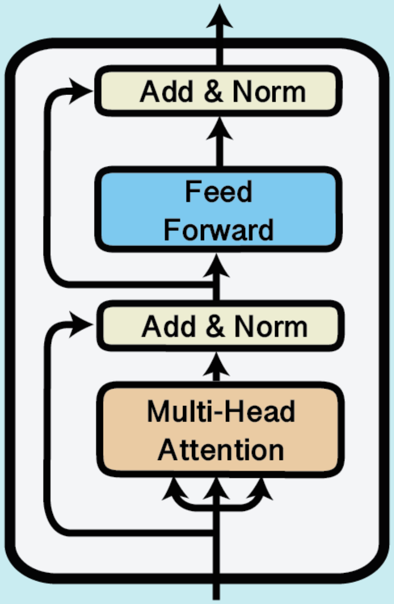
- Transfer Learning in Natural Language Processing (NLP)
- We **only** look at deep neural networks, specifically the **Transformer** architecture.
- We leverage pre-trained transformer-based models such as BERT/RoBERTa/XLM-R/mBERT.
- These have been trained on **massive** amount of text data using **Masked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

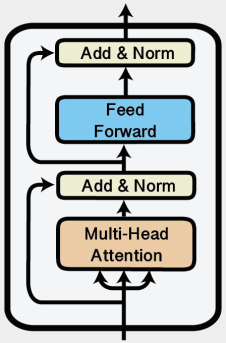
What are Adapters?

= Parameters are frozen

= Parameters are fine-tuned

What are Adapters?

A single Transformer (encoder) layer

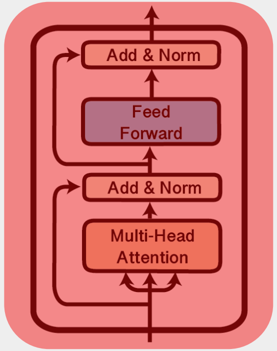


= Parameters are frozen

= Parameters are fine-tuned

What are Adapters?

A single Transformer (encoder) layer

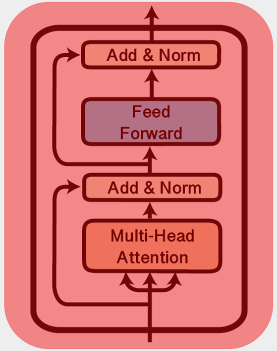


= Parameters are frozen

= Parameters are fine-tuned

What are Adapters?

A single Transformer (encoder) layer



$$\boldsymbol{\theta} \longleftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L(D_{\mathrm{NLI}}; \boldsymbol{\theta})$$

$\boldsymbol{\theta}$

D_{NLI} = NLI Dataset

L = Loss function, e.g. cross entropy loss

$\boldsymbol{\theta}$ = Parameters of the model

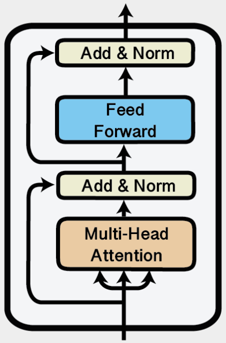
= Parameters are frozen

= Parameters are fine-tuned

What are Adapters?

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L(D_{\mathrm{NLI}}; \boldsymbol{\theta})$$

A single Transformer (encoder) layer



= Parameters are frozen

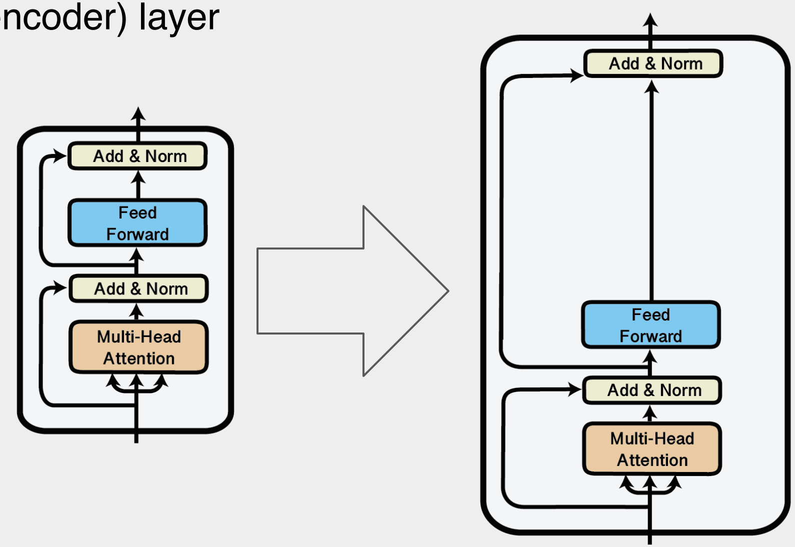
= Parameters are fine-tuned

Houlsby, Neil, et al. "Parameter-Efficient Transfer Learning for NLP." *International Conference on Machine Learning*. 2019.

What are Adapters?

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L(D_{\mathrm{NLI}};\, \boldsymbol{\theta})$$

A single Transformer (encoder) layer



= Parameters are frozen

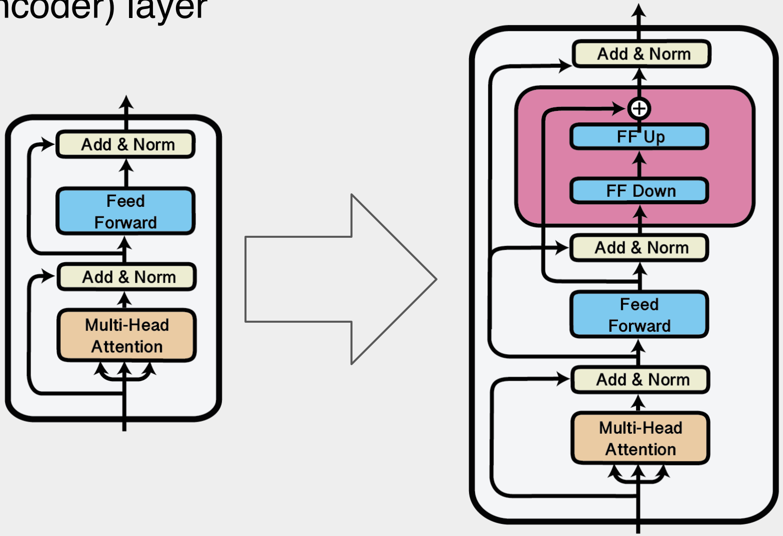
= Parameters are fine-tuned

Houlsby, Neil, et al. "Parameter-Efficient Transfer Learning for NLP." *International Conference on Machine Learning*. 2019.

What are Adapters?

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L(D_{\mathrm{NLI}};\ \boldsymbol{\theta})$$

A single Transformer (encoder) layer



= Parameters are frozen

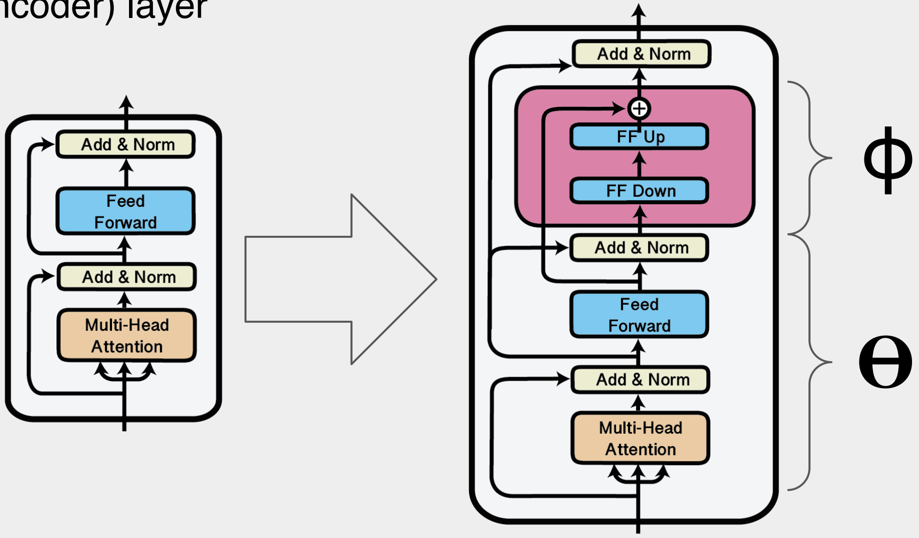
= Parameters are fine-tuned

Houlsby, Neil, et al. "Parameter-Efficient Transfer Learning for NLP." *International Conference on Machine Learning*. 2019.

What are Adapters?

$$\boldsymbol{\theta} \leftarrow \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L(D_{\mathrm{NLI}}; \boldsymbol{\theta})$$

A single Transformer (encoder) layer



Adapter parameters ϕ are **encapsulated** between transformer layers with parameters $\boldsymbol{\theta}$

= Parameters are frozen

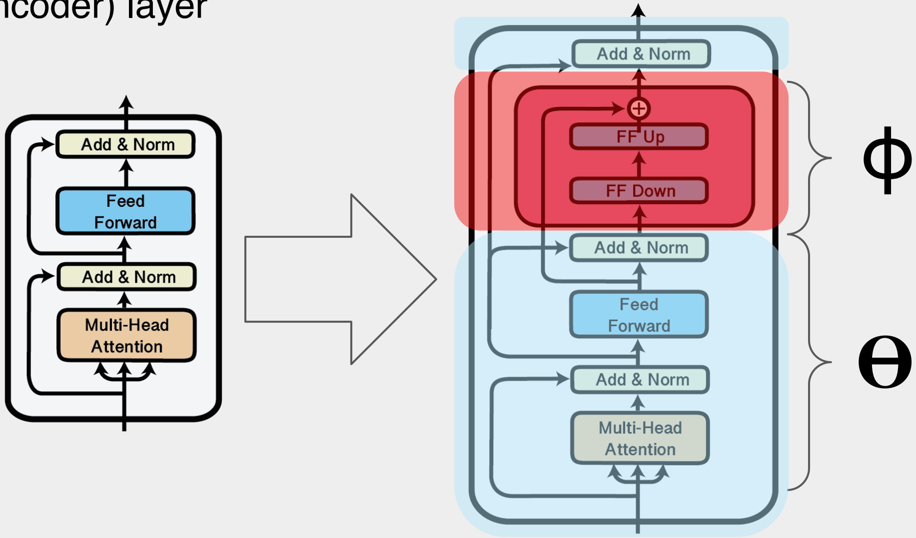
= Parameters are fine-tuned

Houlsby, Neil, et al. "Parameter-Efficient Transfer Learning for NLP." *International Conference on Machine Learning*. 2019.

What are Adapters?

$$\phi \leftarrow \underset{\phi}{\operatorname{argmin}}\, L(D_{\mathrm{NLI}};\, \boldsymbol{\theta},\phi)$$

A single Transformer (encoder) layer



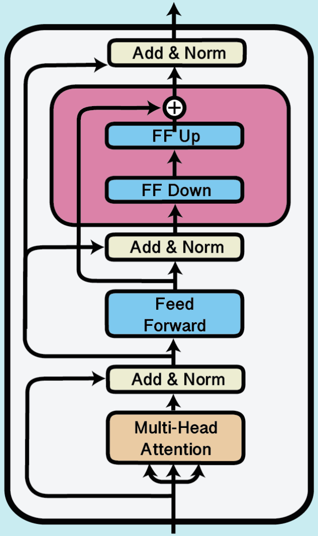
Adapter parameters ϕ are **encapsulated** between transformer layers with parameters $\boldsymbol{\theta}$ which are frozen

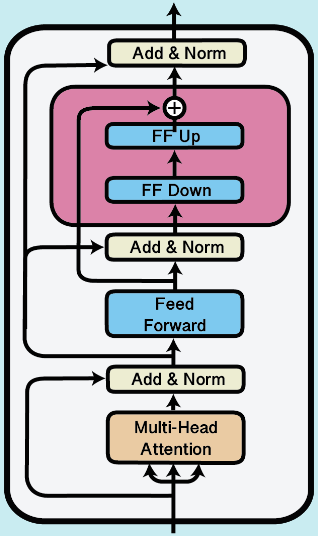
= Parameters are frozen

= Parameters are fine-tuned

Houlsby, Neil, et al. "Parameter-Efficient Transfer Learning for NLP." *International Conference on Machine Learning*. 2019.

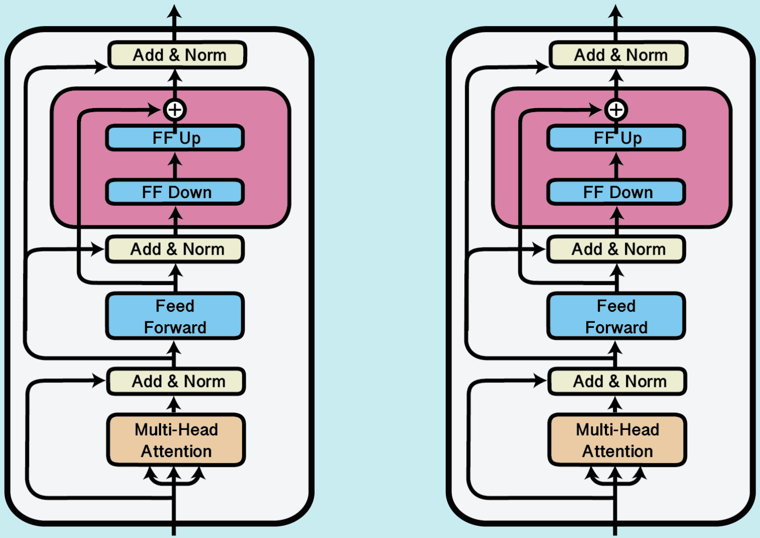
Encapsulated Adapters? 🤨





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

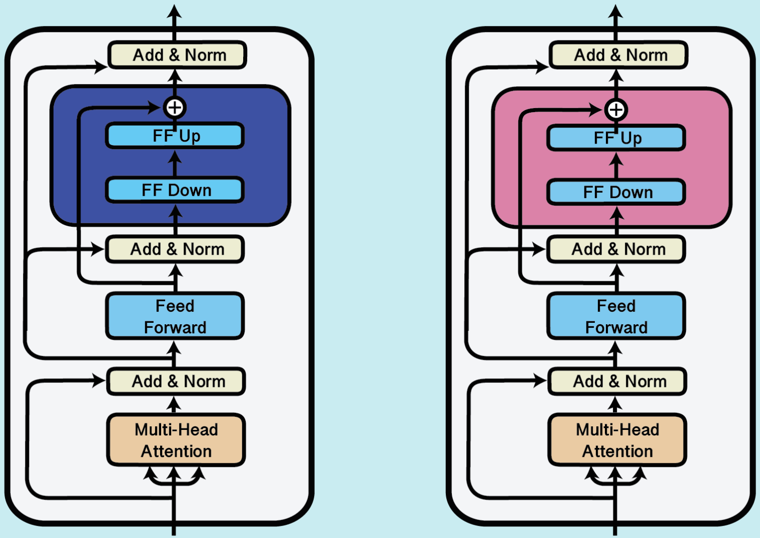
Encapsulated Adapters? 🤨



- 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**.

Encapsulated Adapters? 🤨

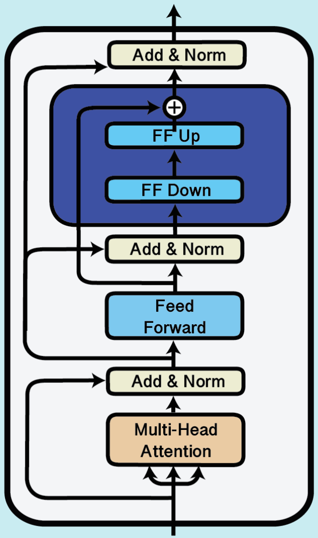
MLM (English)



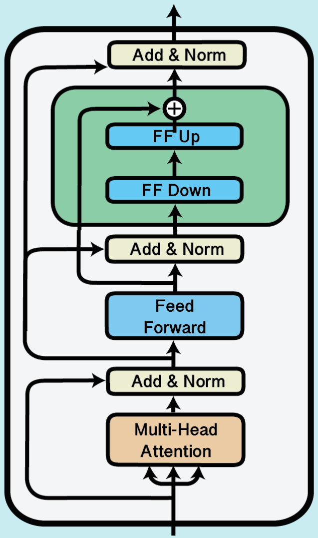
- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.

Encapsulated Adapters? 🤨

MLM (English)



MLM (Quechuan)

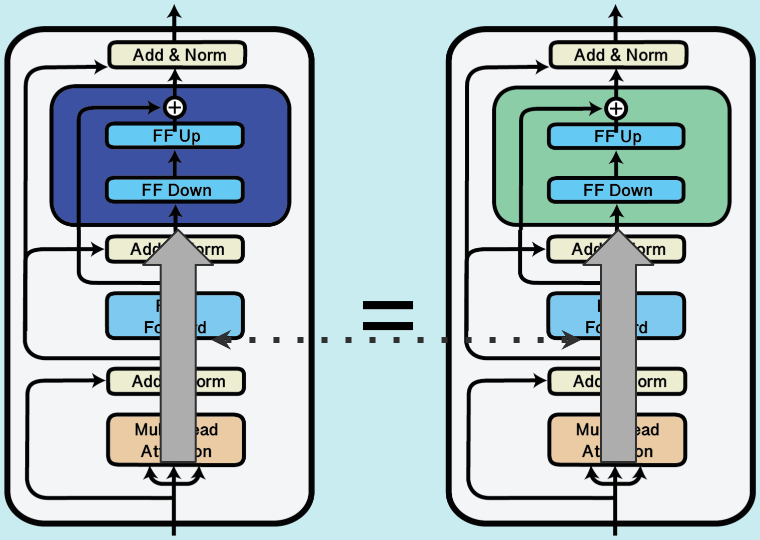


- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.

Encapsulated Adapters? 🤨

MLM (English)

MLM (Quechuan)

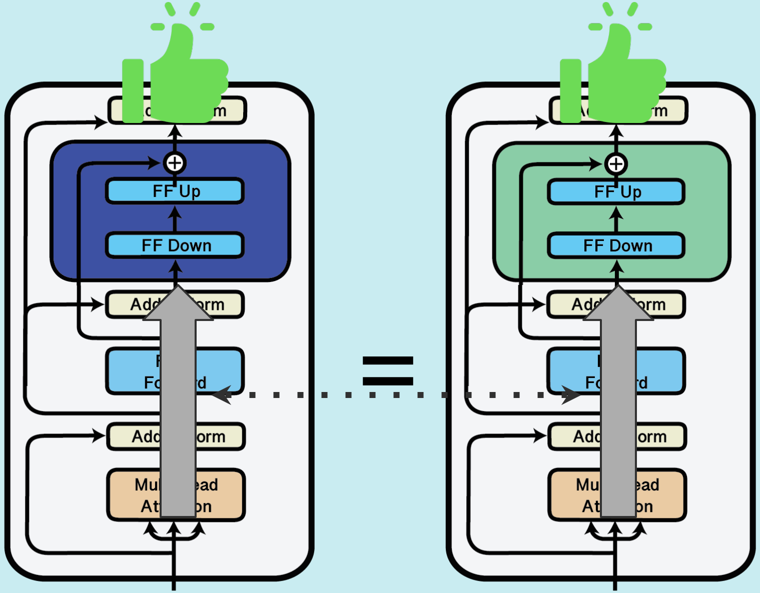


- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.

Encapsulated Adapters? 🤨

MLM (English)

MLM (Quechuan)

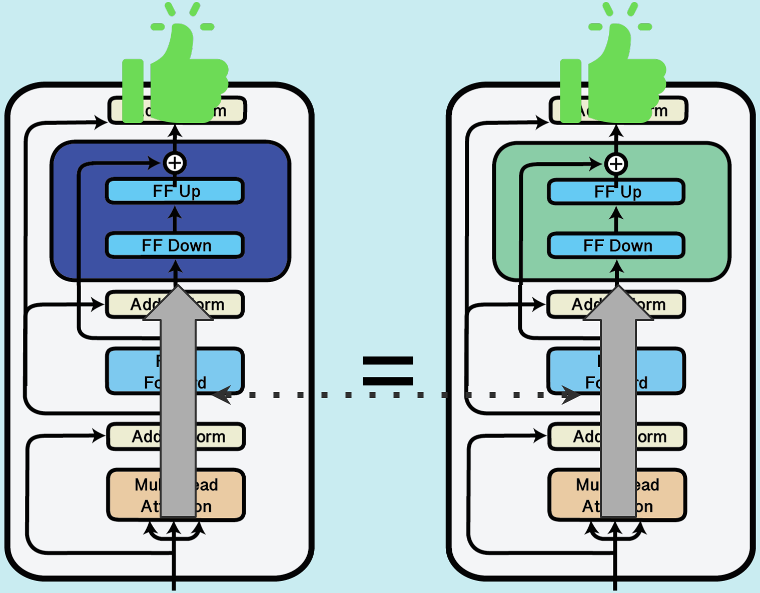


- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.

Encapsulated Adapters? 🤨

MLM (English)

MLM (Quechuan)

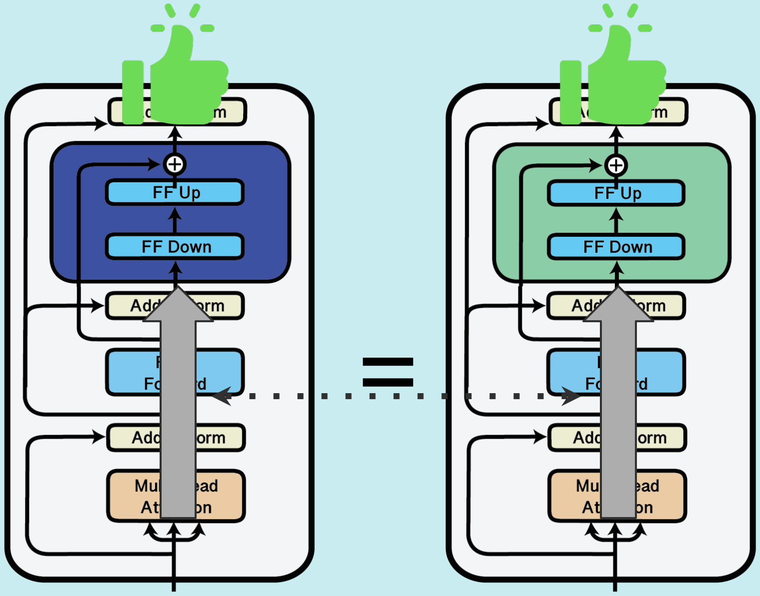


- Adapters **learn transformations** that make the underlying model **more suited** to a task or language.
- Using masked language modelling (MLM), we can learn **language-specific transformations** for e.g. **English** and **Quechua**.
- As long as the underlying model is kept fixed, these transformations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

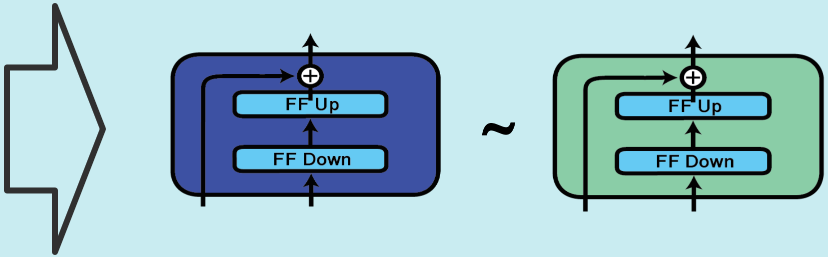
Encapsulated Adapters? 🤨

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**.



Parameter Efficiency of Adapters in Transformers

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

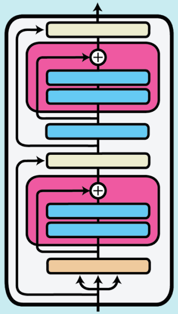
Adapters are smaller in size than training the full model.

Performance on GLUE tasks

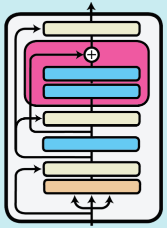
	Full	Pfeif.	Houl.
RTE (Wang et al., 2018)	66.2	70.8	69.8
MRPC (Dolan and Brockett, 2005)	90.5	89.7	91.5
STS-B (Cer et al., 2017)	88.8	89.0	89.2
CoLA (Warstadt et al., 2019)	59.5	58.9	59.1
SST-2 (Socher et al., 2013)	92.6	92.2	92.8
QNLI (Rajpurkar et al., 2016)	91.3	91.3	91.2
MNLI (Williams et al., 2018)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

Number of newly introduced Parameters

	Base		**Large**	
CRate	#Params	Size	#Params	Size
64	0.2M	0.9Mb	0.8M	3.2Mb
16	0.9M	3.5Mb	3.1M	13Mb
2	7.1M	28Mb	25.2M	97Mb



Houlsby et al., 2019



Pfeiffer et al., 2020a

Houlsby, Neil, et al. "Parameter-Efficient Transfer Learning for NLP." *International Conference on Machine Learning*. 2019.
Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *arXiv preprint* (2020a).
Pfeiffer, Jonas, et al. "Adapterhub: A framework for adapting transformers." *Proceedings of EMNLP: Systems Demonstrations* (2020b)

Agenda

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

site code paper

AdapterFusion: Non-Destructive Task Composition for Transfer Learning

Proceedings of EACL 2021

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, Iryna Gurevych

TECHNISCHE UNIVERSITÄT DARMSTADT

NYU

Problems of Multi-Task and Transfer Learning

Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

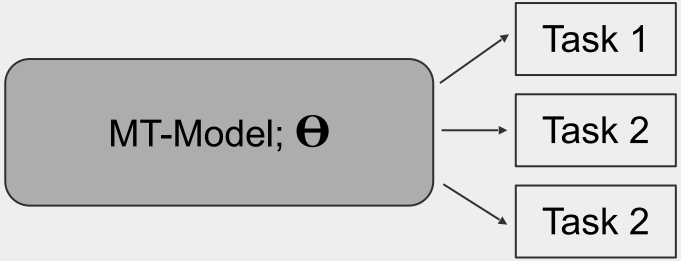
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

MT-Model; $\boldsymbol{\theta}$

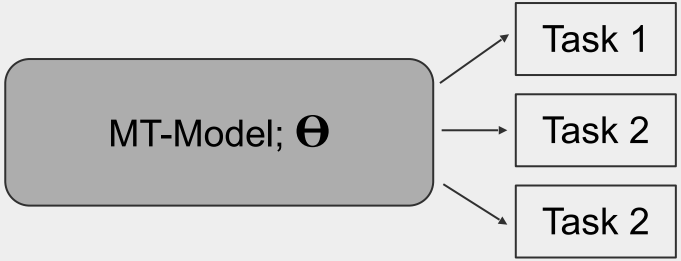
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:



Problems of Multi-Task and Transfer Learning

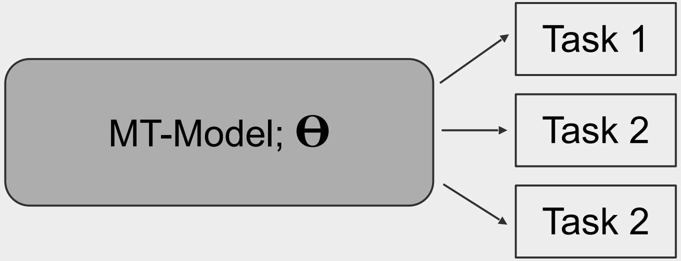
Multi-Task Learning:



Catastrophic Interference: Sharing all parameters $\boldsymbol{\theta}$ between tasks results in deterioration of performance for a subset of tasks.

Problems of Multi-Task and Transfer Learning

Multi-Task Learning:

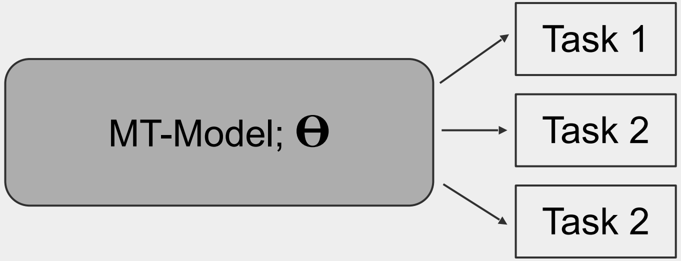


Catastrophic Interference: Sharing all parameters θ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:

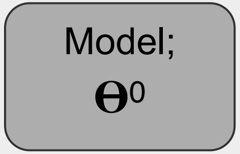
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:



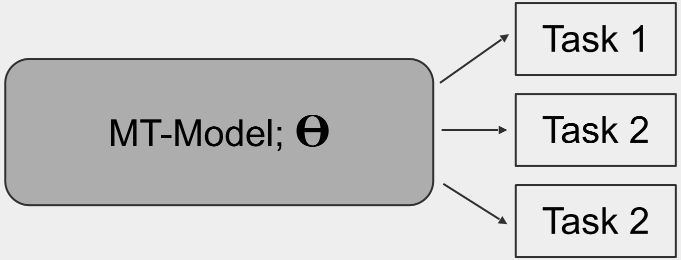
Catastrophic Interference: Sharing all parameters Θ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:



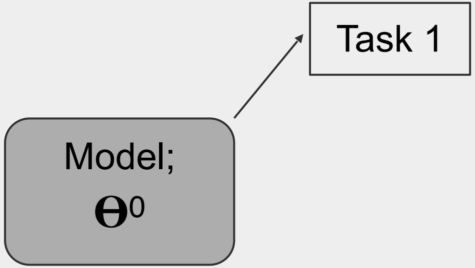
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:



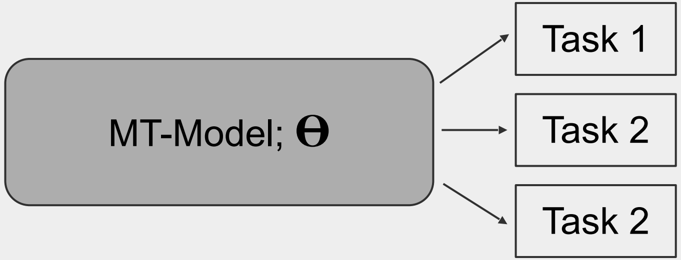
Catastrophic Interference: Sharing all parameters ϴ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:



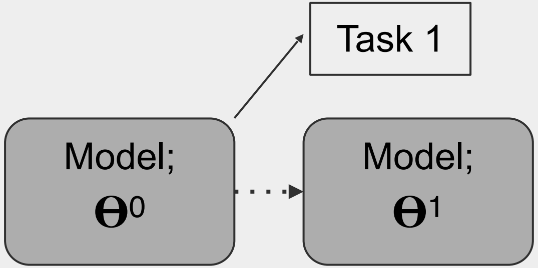
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:



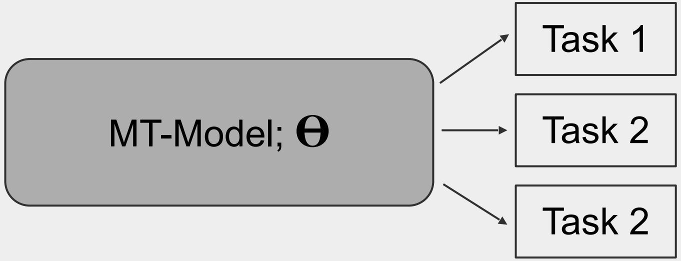
Catastrophic Interference: Sharing all parameters Θ between tasks results in deterioration of performance for a subset of tasks.

Sequential Fine-Tuning:

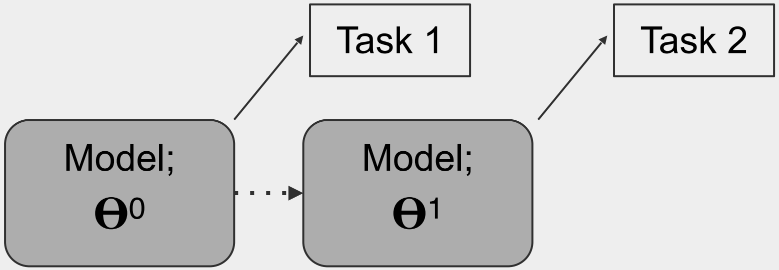


Problems of Multi-Task and Transfer Learning

Multi-Task Learning:



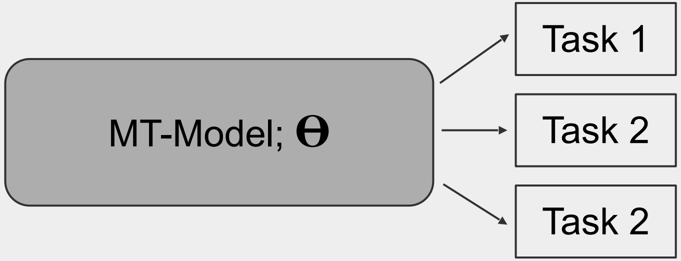
Sequential Fine-Tuning:



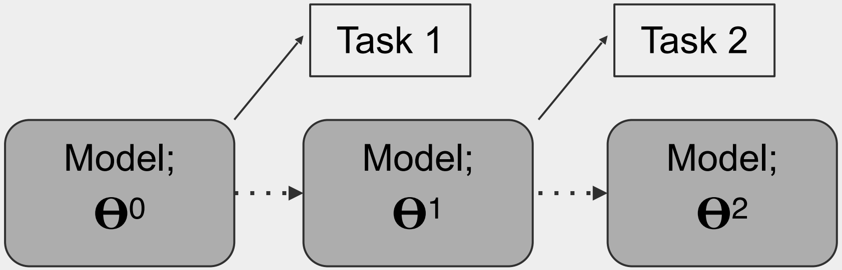
Catastrophic Interference: Sharing all parameters θ between tasks results in deterioration of performance for a subset of tasks.

Problems of Multi-Task and Transfer Learning

Multi-Task Learning:



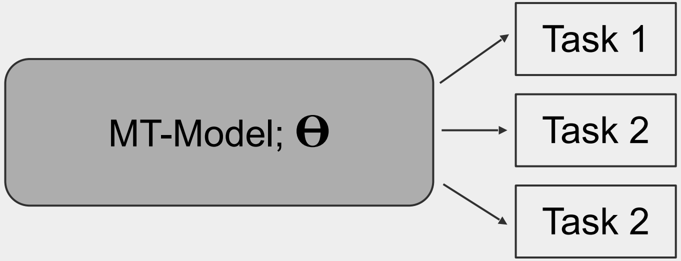
Sequential Fine-Tuning:



Catastrophic Interference: Sharing all parameters $\boldsymbol{\theta}$ between tasks results in deterioration of performance for a subset of tasks.

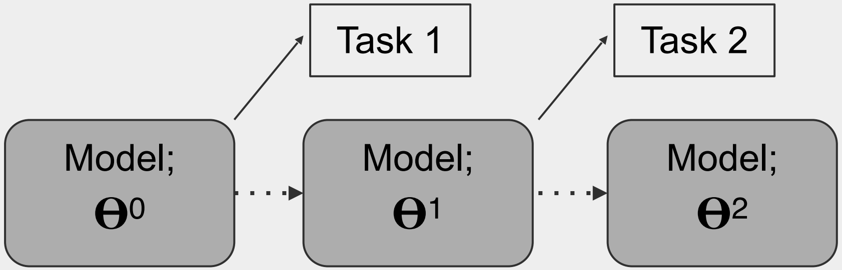
Problems of Multi-Task and Transfer Learning

Multi-Task Learning:



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

Sequential Fine-Tuning:



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?

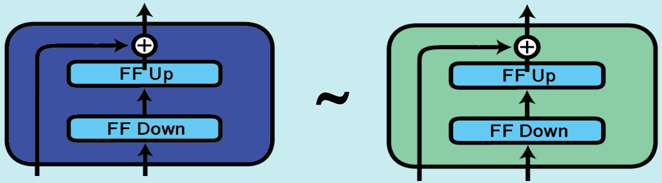
1. Train **task-specific weights** (adapters) for each task.

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

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

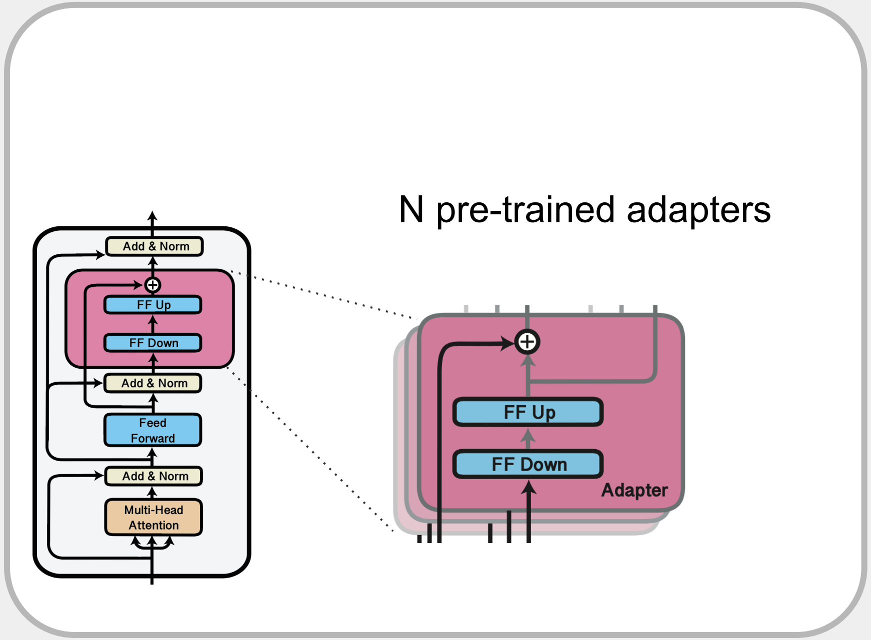
1. **Combine** the representations **subsequently**.

Assuming...



Sharing Information across multiple tasks

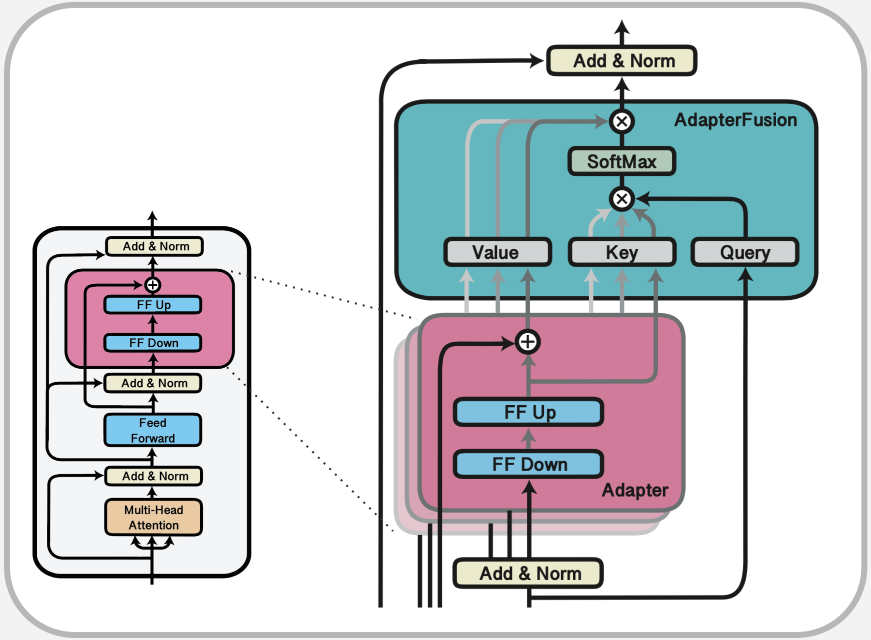
Given a pool of adapters, we want to leverage the stored information to solve a new task:



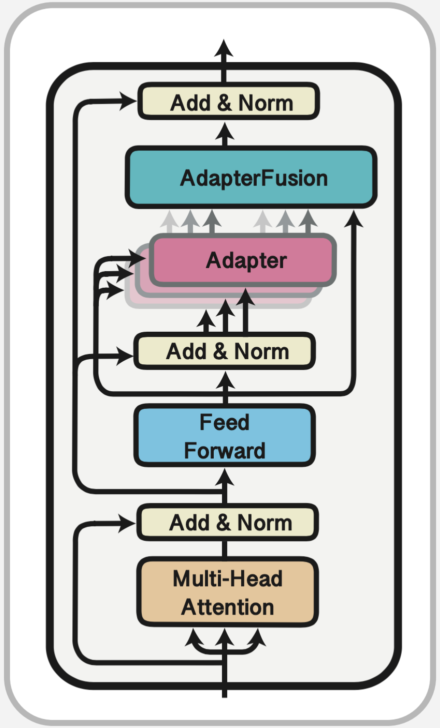
Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *EACL* (2021).

Sharing Information across multiple tasks

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

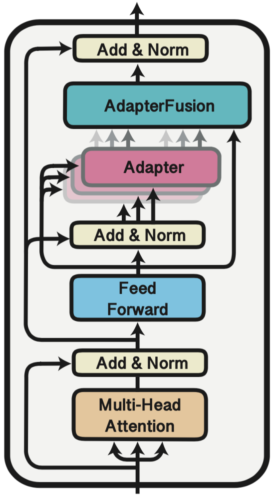


Learn dynamic attention **weighting** on a **target task** given the representations of the given **Adapters**

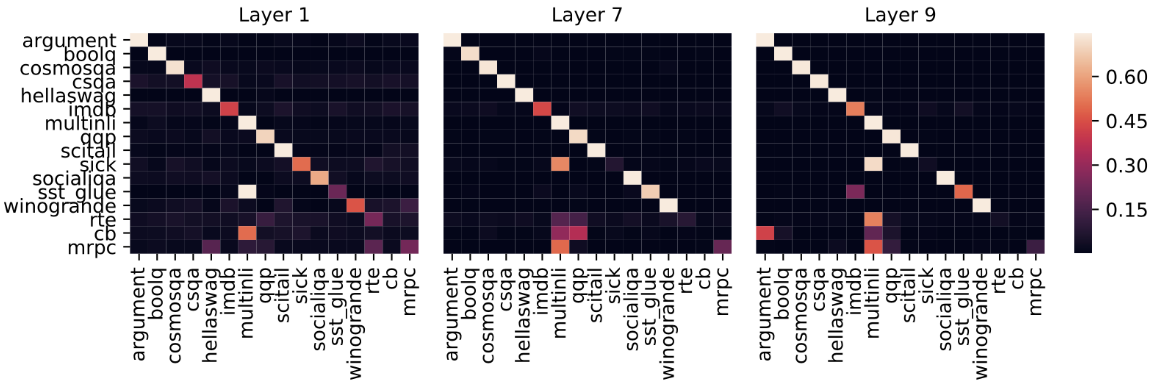


Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *EACL*(2021).

"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.

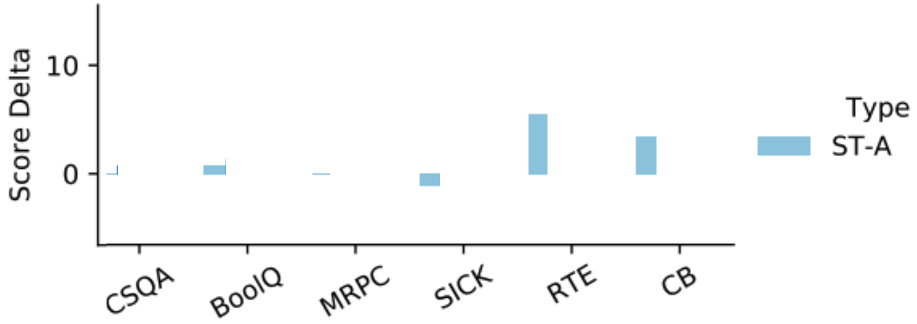


Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *arXiv preprint* (2020a).

Performance of AdapterFusion

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

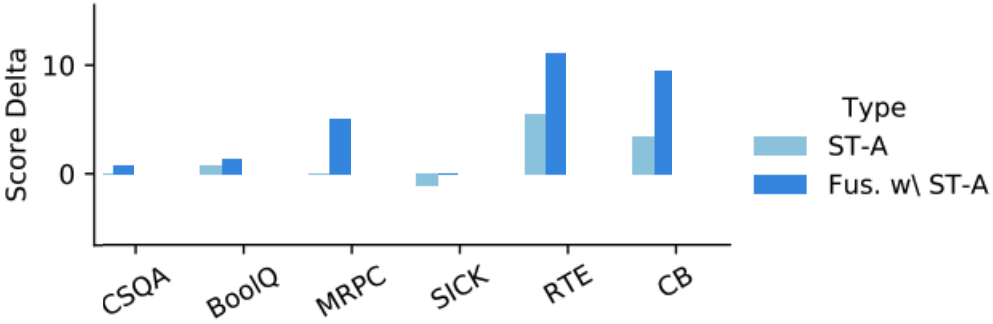
We find that **AdapterFusion** performs well for lower resource datasets where less than **10k** examples exist.



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

Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *EACL* (2021).

Performance of AdapterFusion

Performance of **Full** *finetuning* vs. **S***ingle* **T***ask* **A***dapters* vs. **F***usion* **w***ith* **S***ingle* **T***ask* **A***dapters*.

high resource → low resource

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

Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *EACL* (2021).

Performance of AdapterFusion

Performance of **Full** *finetuning* vs. **S***ingle* **T***ask* **A***dapters* vs. **F***usion* **w***ith* **S***ingle* **T***ask* **A***dapters*.

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

high resource → low resource

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

Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *EACL* (2021).

Performance of AdapterFusion

Performance of **Full** *finetuning* vs. **S***ingle* **T***ask* **A***dapters* vs. **F***usion* **w***ith* **S***ingle* **T***ask* **A***dapters*.

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.

high resource

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

low resource

Pfeiffer, Jonas, et al. "AdapterFusion: Non-destructive task composition for transfer learning." *EACL* (2021).

Agenda

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

site code paper

MAD-X: An Adapter-based Framework for Multi-task Cross-lingual Transfer

Proceedings of EMNLP 2020

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, Sebastian Ruder

TECHNISCHE UNIVERSITÄT DARMSTADT

UNIVERSITY OF CAMBRIDGE

DeepMind

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

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

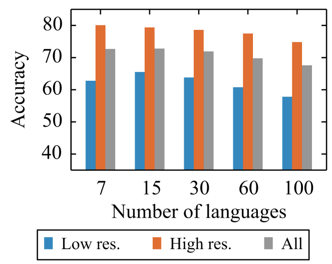
Related Work & Baselines

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

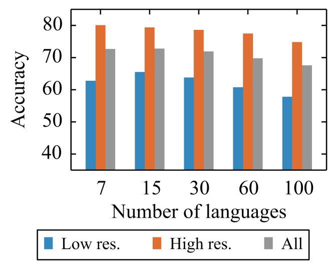
- Suffer from "the curse of multilinguality" (Conneau et al. 2020)



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



<u>**BUT**</u>

- 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** (<u>seen</u> e.g. English, Chinese, and <u>unseen</u>, 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

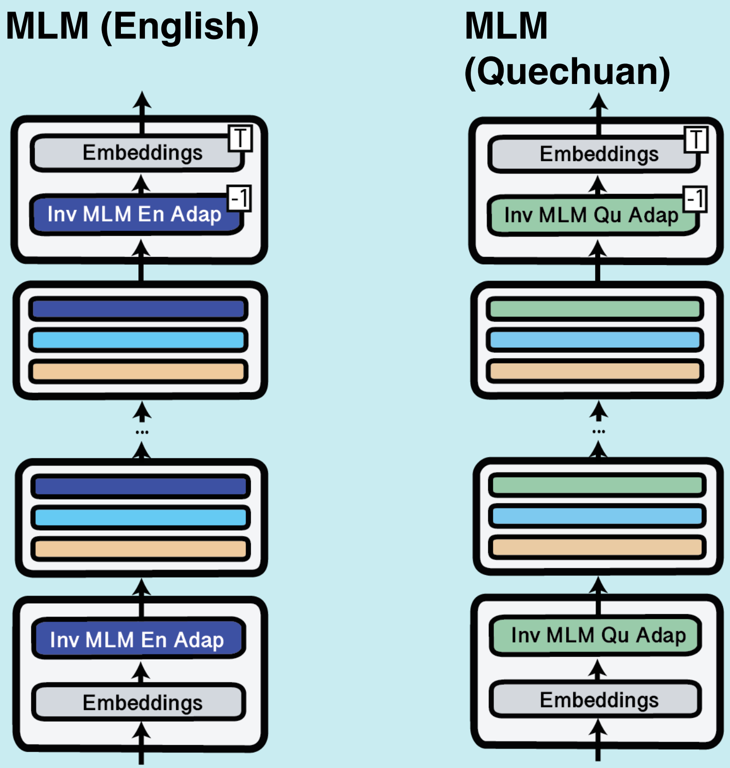
- **Languages** (<u>seen</u> e.g. English, Chinese, and <u>unseen</u>, Quechuan, Guarani)
- **Tasks** (e.g. NER, COPA, SQuAD)

“<u>Language agnostic</u>” **task**-adapters are **stacked** on top of language adapters for zero-shot transfer to unseen languages.

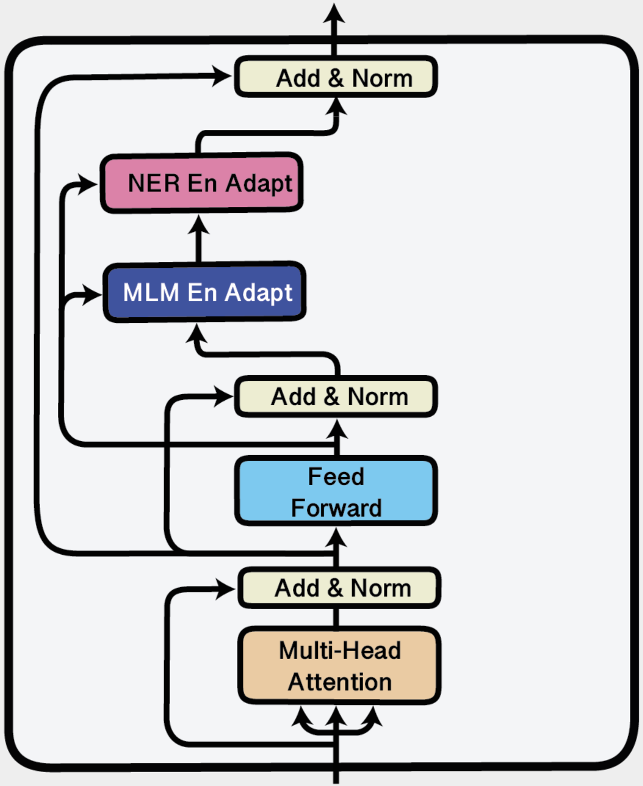
MAD-X

Step 1: Train Language Adapters

We train **language adapters** for the **source language** and the **target language** with masked language modelling on Wikipedia.



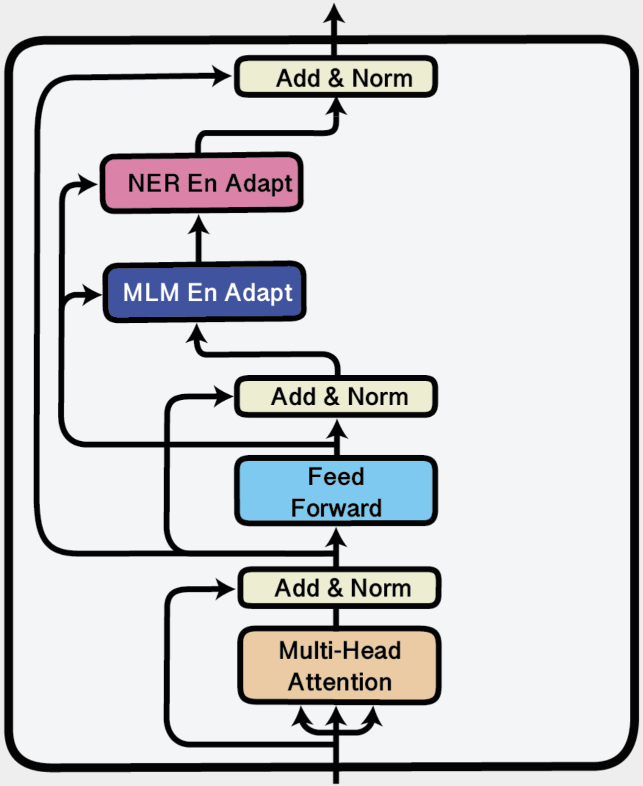
MAD-X



Add & Norm
NER En Adapt
MLM En Adapt
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention

MAD-X

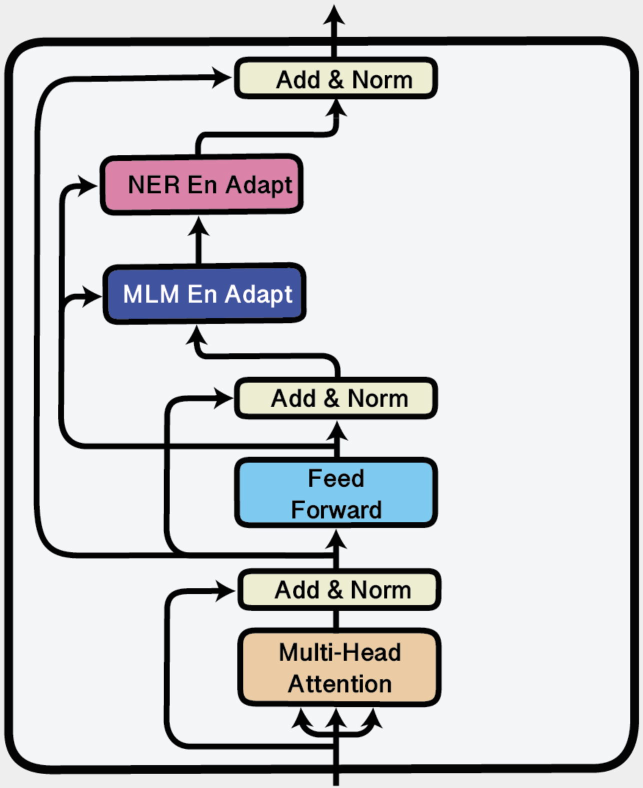
Step 2: Train a Task Adapter



MAD-X

Step 2: Train a Task Adapter

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

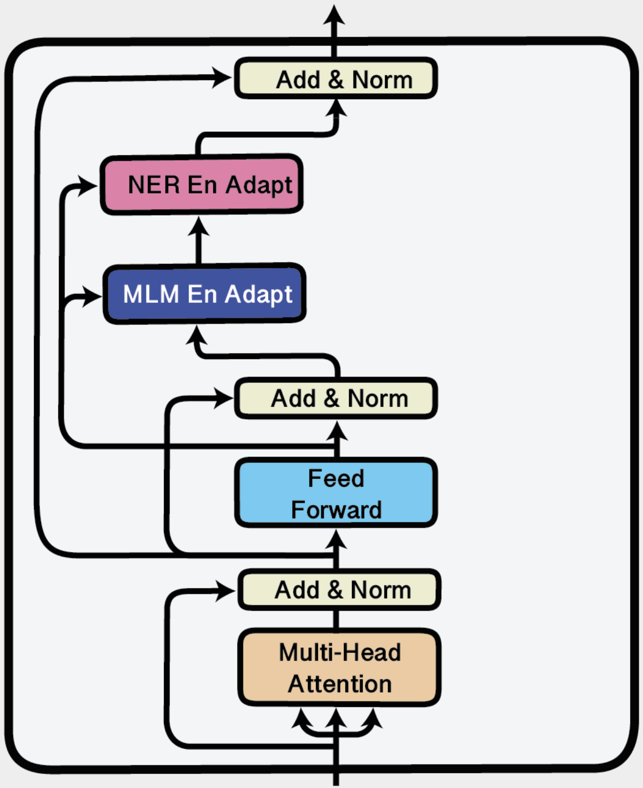


MAD-X

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 ϕ_l as well as the transformer weights Θ are frozen while only the task adapter parameter ϕ_t are trained.

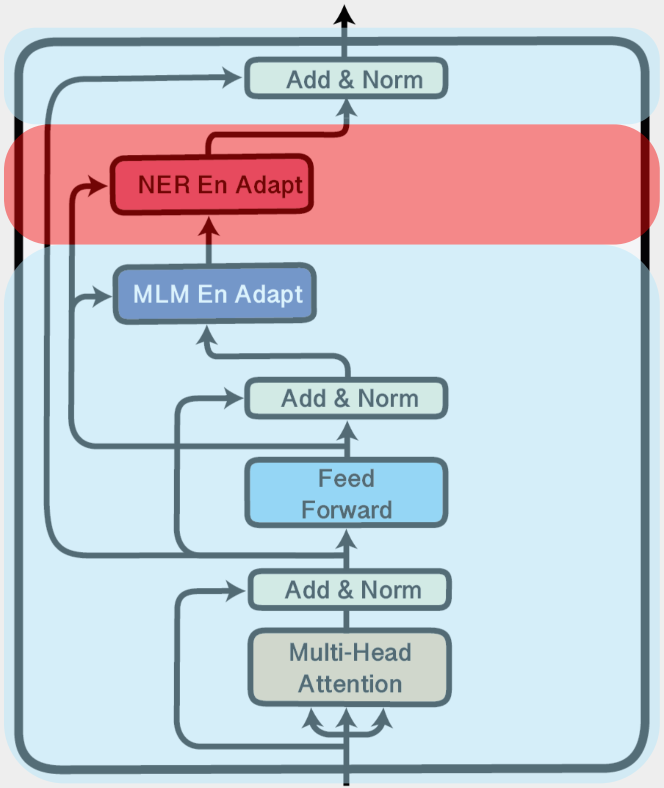


MAD-X

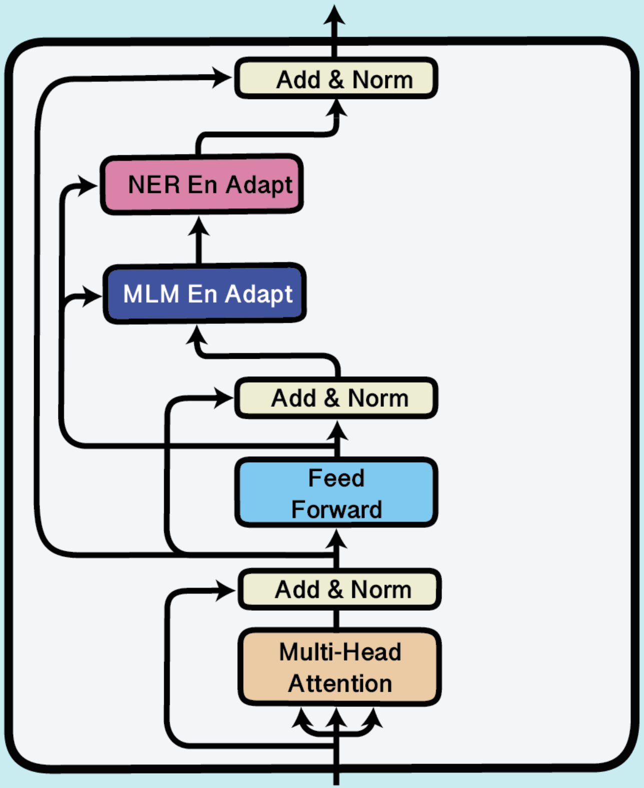
Step 2: Train a Task Adapter

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

The language adapter ϕ_l as well as the transformer weights Θ are frozen while only the task adapter parameter ϕ_t are trained.

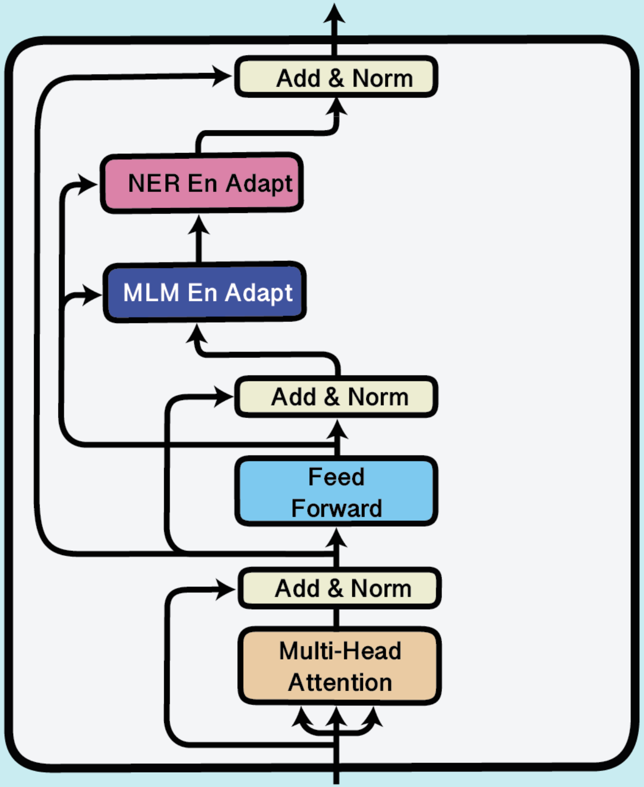


MAD-X


Add & Norm
NER En Adapt
MLM En Adapt
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention

MAD-X

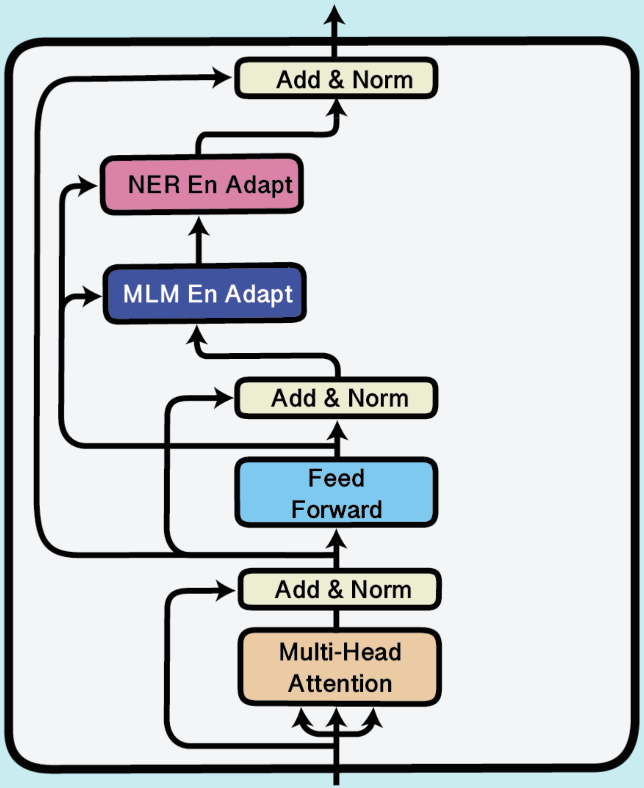
Step 3: Zero-Shot transfer to unseen language



MAD-X

Step 3: Zero-Shot transfer to unseen language

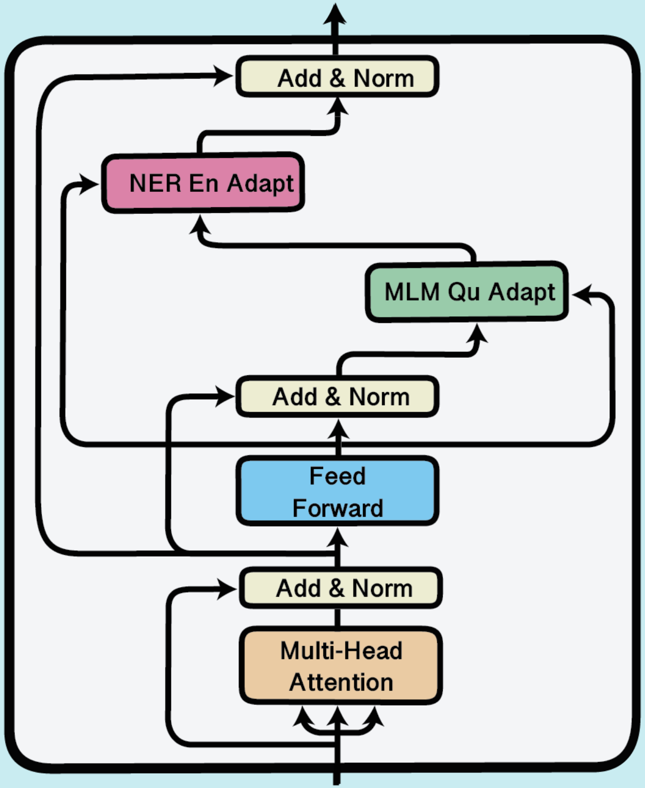
We **replace** the **source** language adapter with the **target** language adapter, while **keeping** the “language agnostic” **task adapter**.



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)

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

Baselines

Standard Zero-Shot

Baselines

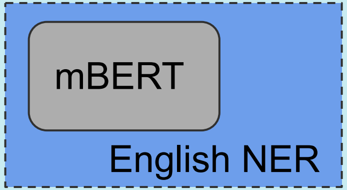
Standard Zero-Shot

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.

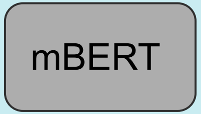
2. **Evaluate** on the **target** language.

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

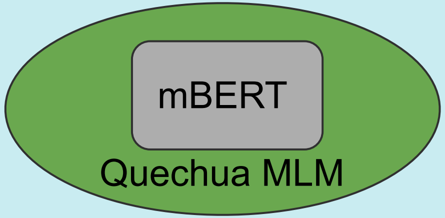
Standard Zero-Shot

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

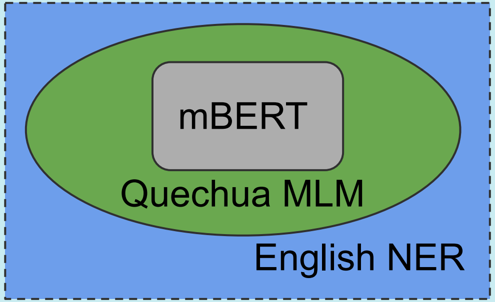
Standard Zero-Shot

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

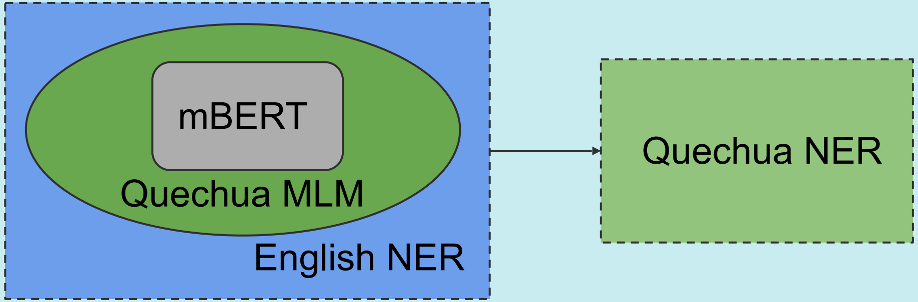
Standard Zero-Shot

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.

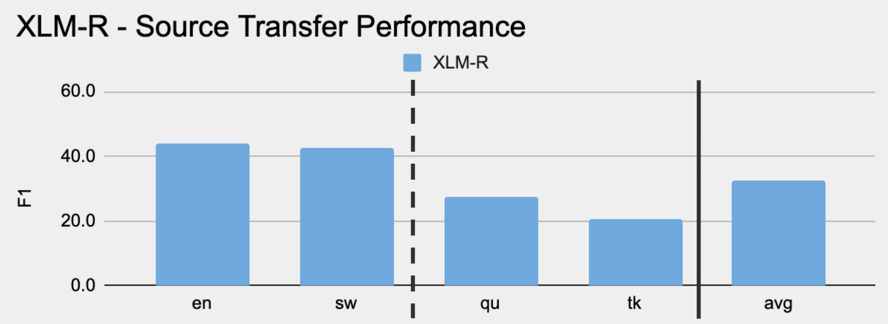


Results NER

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.

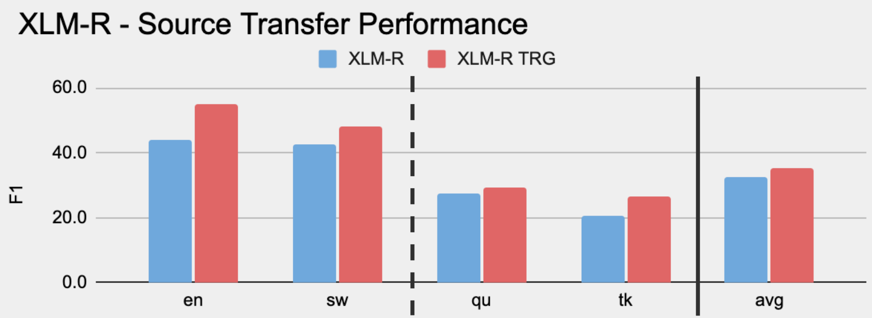
Results NER

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.



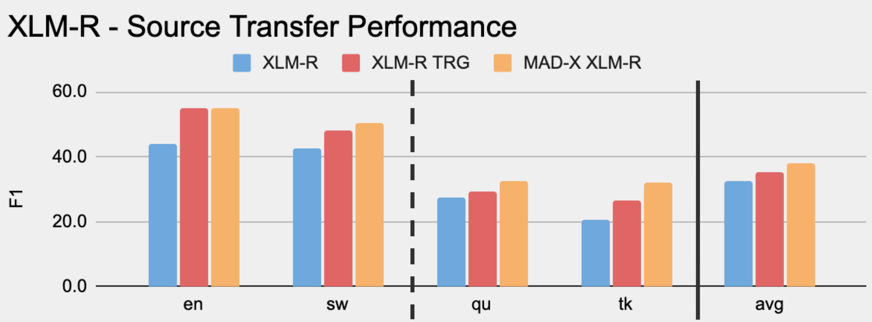
Results NER

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.



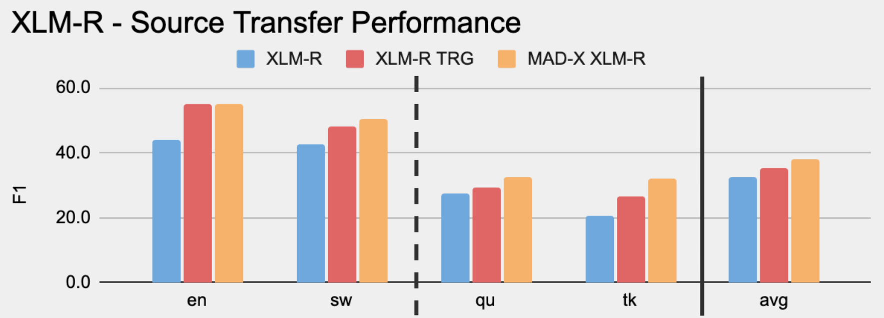
Results NER

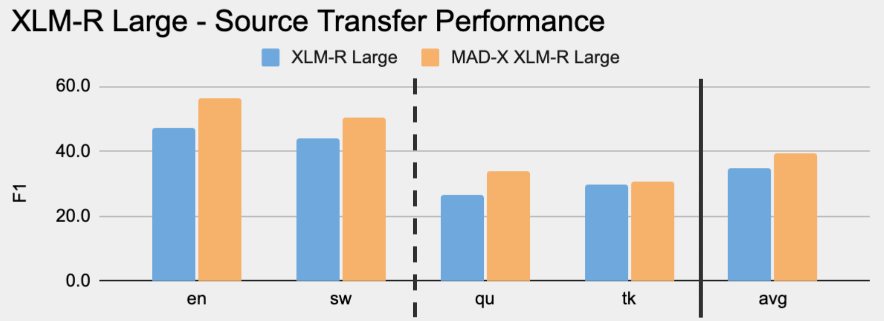
NER **F1 scores averaged** over all 16 **target languages** when transferring from each **source** language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.



Results NER

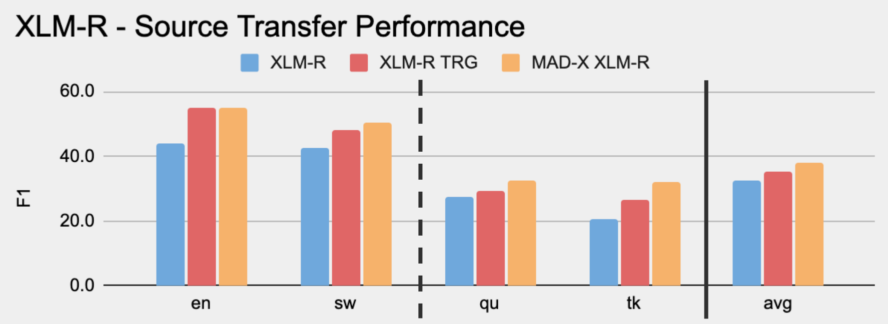
NER **F1 scores averaged** over all 16 **target languages** when transferring from each **source** language (i.e., the columns refer to source languages). The vertical dashed line distinguishes between languages seen in multilingual pretraining and the unseen ones.

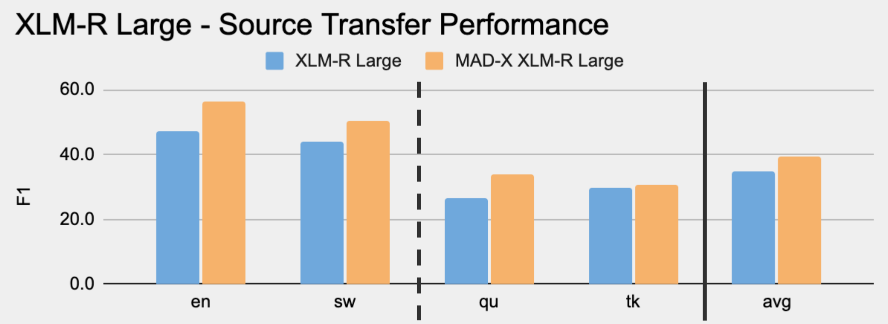


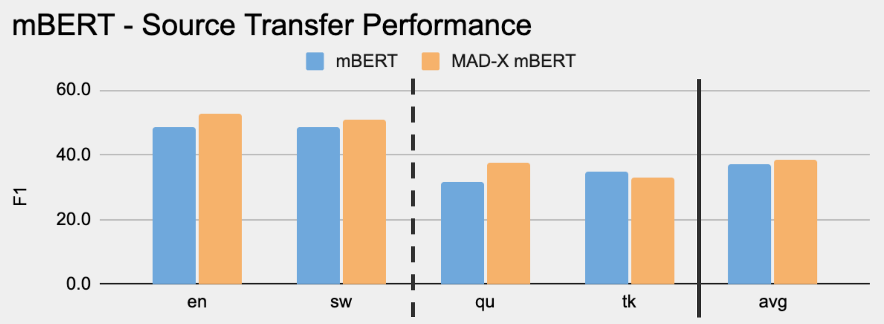


Results NER

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.







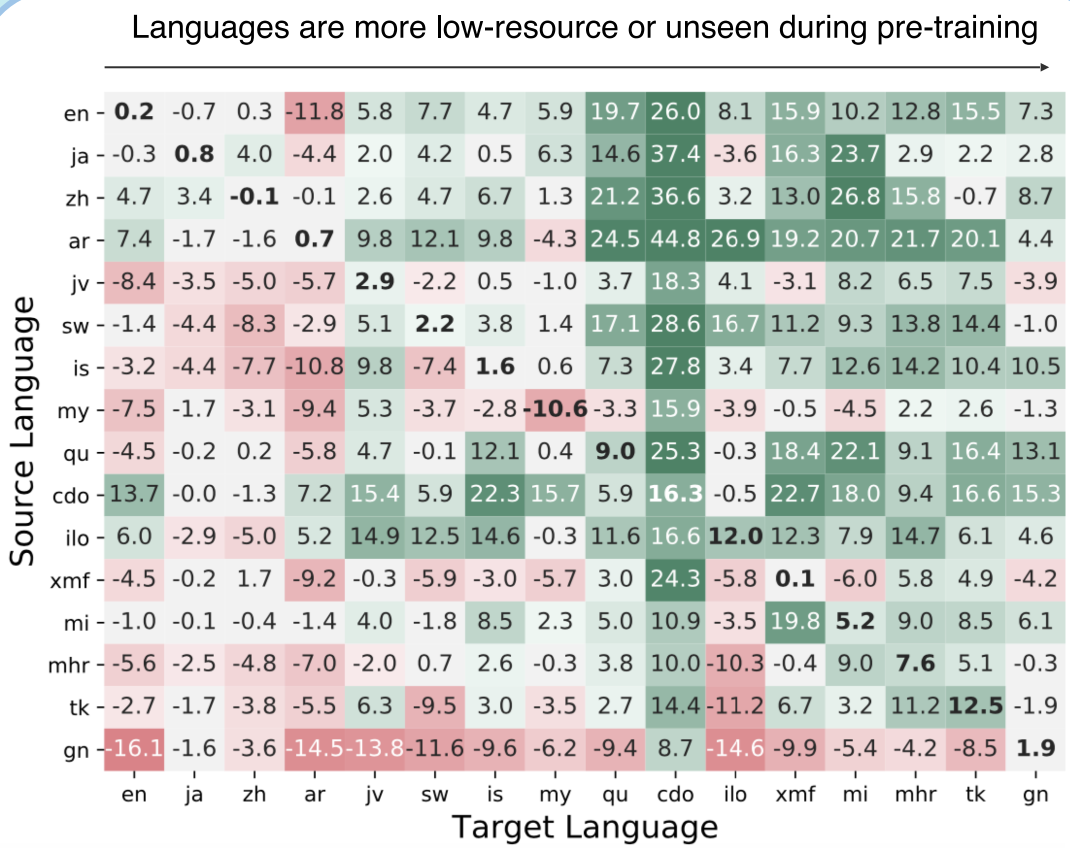
Results NER

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

Source Language \ Target Language	en	ja	zh	ar	jv	sw	is	my	qu	cdo	ilo	xmf	mi	mhr	tk	gn
en	**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
ja	-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
zh	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
ar	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
jv	-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
sw	-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
is	-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
my	-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
qu	-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
cdo	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
ilo	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
xmf	-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
mi	-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
mhr	-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
tk	-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
gn	-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**

Results NER

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



Results NER

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

Top right corner represent the **realistic scenario** of **transfering from high resource to low resource**

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

Source Language \ Target Language	en	ja	zh	ar	jv	sw	is	my	qu	cdo	ilo	xmf	mi	mhr	tk	gn
en	**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
ja	-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
zh	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
ar	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
jv	-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
sw	-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
is	-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
my	-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
qu	-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
cdo	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
ilo	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
xmf	-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
mi	-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
mhr	-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
tk	-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
gn	-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**

Results NER

Relative **F1 improvement** of **MAD-XLarge over XLM-RLarge** in cross-lingual NER transfer.

Top right corner represent the **realistic scenario** of **transfering from high resource to low resource**

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

Source Language \ Target Language	en	ja	zh	ar	jv	sw	is	my	qu	cdo	ilo	xmf	mi	mhr	tk	gn
en	**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
ja	-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
zh	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
ar	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
jv	-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
sw	-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
is	-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
my	-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
qu	-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
cdo	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
ilo	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
xmf	-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
mi	-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
mhr	-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
tk	-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
gn	-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**

Agenda

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

site code paper

AdapterDrop: On the Efficiency of Adapters in Transformers

arxiv 2020

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, **Jonas Pfeiffer**, Nils Reimers, Iryna Gurevych

TECHNISCHE UNIVERSITÄT DARMSTADT

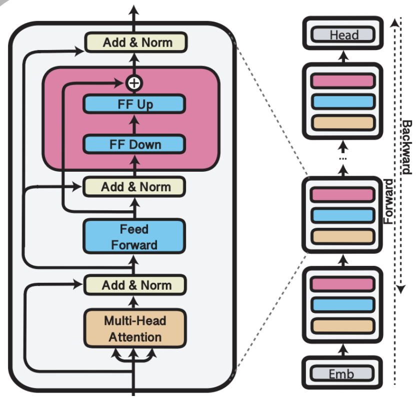
AdapterDrop: Training/Inference Efficiency of Adapters

Rücklé, Andreas, et al. "AdapterDrop: On the Efficiency of Adapters in Transformers." *arXiv preprint*. 2020.

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.

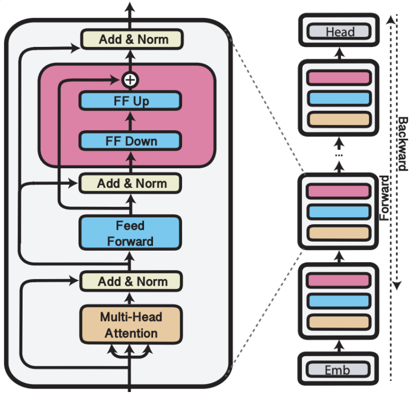


Rücklé, Andreas, et al. "AdapterDrop: On the Efficiency of Adapters in Transformers." *arXiv preprint*. 2020.

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.



Setting	Adapter	Relative speed *(for Seq.Len./Batch)*			
		128/16	128/32	512/16	512/32
Training	Houlsby	1.48	1.53	1.36	1.33
	Pfeiffer	1.57	1.60	1.41	1.37
Inference	Houlsby	0.94	0.94	0.96	0.96
	Pfeiffer	0.95	0.95	0.96	0.96

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.

Rücklé, Andreas, et al. "AdapterDrop: On the Efficiency of Adapters in Transformers." *arXiv preprint*. 2020.

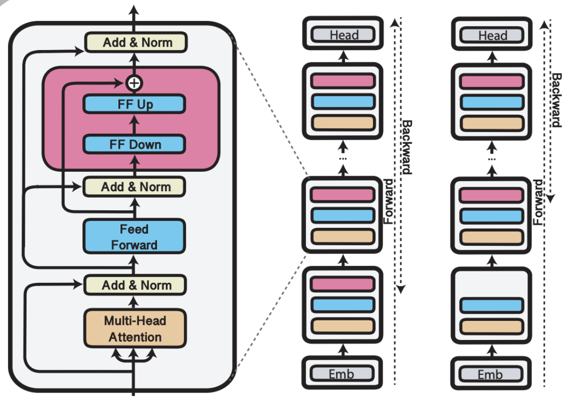
AdapterDrop: Training/Inference Efficiency of Adapters

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.

Rücklé, Andreas, et al. "AdapterDrop: On the Efficiency of Adapters in Transformers." *arXiv preprint*. 2020.



Setting	Adapter	Relative speed *(for Seq.Len./Batch)*			
		128/16	128/32	512/16	512/32
Training	Houlsby	1.48	1.53	1.36	1.33
	Pfeiffer	1.57	1.60	1.41	1.37
Inference	Houlsby	0.94	0.94	0.96	0.96
	Pfeiffer	0.95	0.95	0.96	0.96

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.

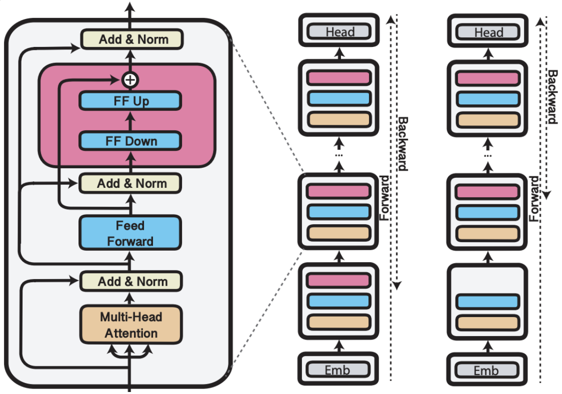
AdapterDrop: Training/Inference Efficiency of Adapters

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.

Rücklé, Andreas, et al. "AdapterDrop: On the Efficiency of Adapters in Transformers." *arXiv preprint*. 2020.



Setting	**Adapter**	**Relative speed** *(for Seq.Len./Batch)*			
		128/16	128/32	512/16	512/32
Training	Houlsby	1.48	1.53	1.36	1.33
	Pfeiffer	1.57	1.60	1.41	1.37
Inference	Houlsby	0.94	0.94	0.96	0.96
	Pfeiffer	0.95	0.95	0.96	0.96

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.

Simultaneous Tasks	2	4	8	16
Speedup (each layer)	4.3%	6.6%	7.8%	8.4%

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

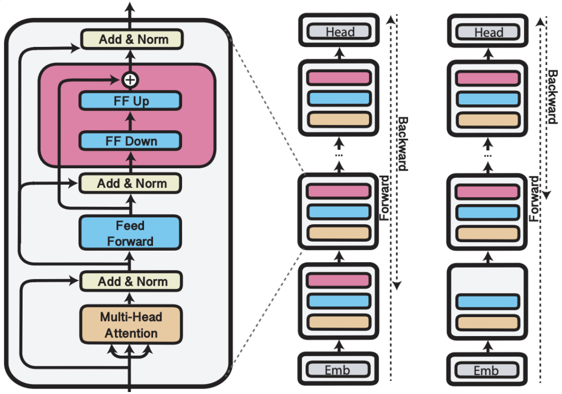
AdapterDrop: Training/Inference Efficiency of Adapters

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.

Rücklé, Andreas, et al. "AdapterDrop: On the Efficiency of Adapters in Transformers." *arXiv preprint*. 2020.

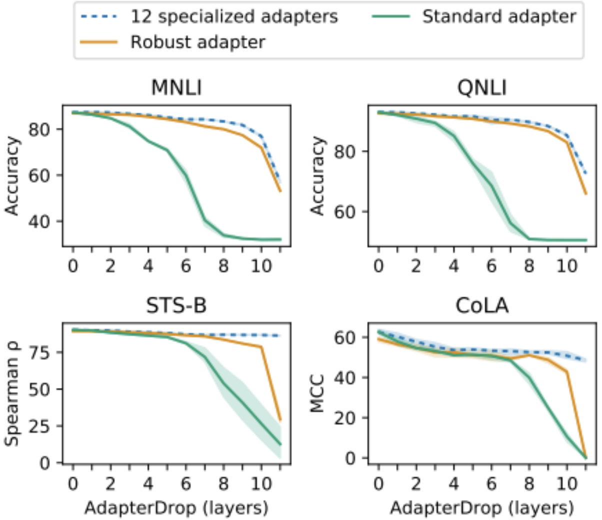


Setting	**Adapter**	**Relative speed** *(for Seq.Len./Batch)*			
		128/16	128/32	512/16	512/32
Training	Houlsby	1.48	1.53	1.36	1.33
	Pfeiffer	1.57	1.60	1.41	1.37
Inference	Houlsby	0.94	0.94	0.96	0.96
	Pfeiffer	0.95	0.95	0.96	0.96

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.

Simultaneous Tasks	2	4	8	16
Speedup (each layer)	4.3%	6.6%	7.8%	8.4%

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



Agenda

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

site code paper

AdapterHub: A Framework for Adapting Transformers

Proceedings of EMNLP 2020: Systems Demonstrations

Jonas Pfeiffer*, Andreas Rücklé*, Clifton Poth*, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, Iryna Gurevych

TECHNISCHE UNIVERSITÄT DARMSTADT

UNIVERSITY OF CAMBRIDGE

NYU

DeepMind

AdapterHub

Explore Upload Docs

A **central repository** for pre-trained **adapter modules**

182 adapters | **24** text tasks | **32** languages

```
pip install adapter-transformers
```

Explore | Upload | Docs | GitHub | Paper

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 moduar units, allow for easy extension and composition. This opens up opportunities to compose adapters to solve new tasks.

Built on HuggingFace 🤗 Transformers 🚀

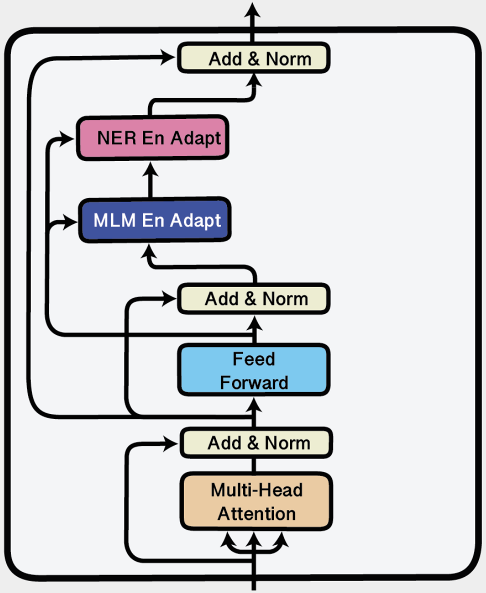
AdapterHub builds on the HuggingFace transformers framework requiring as little as two additional lines of code to train adapters for a downstream task.

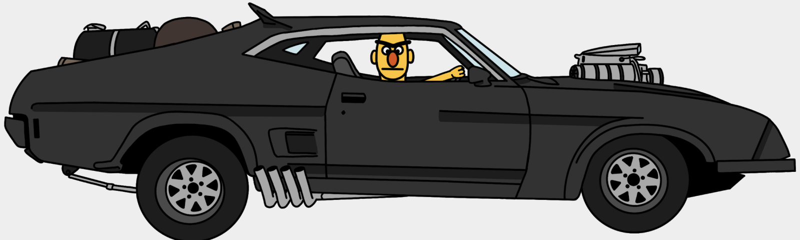
link

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?
- ...

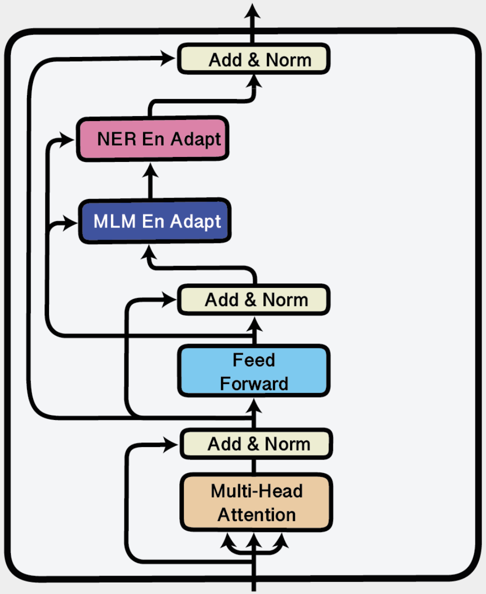
Key Takeaways

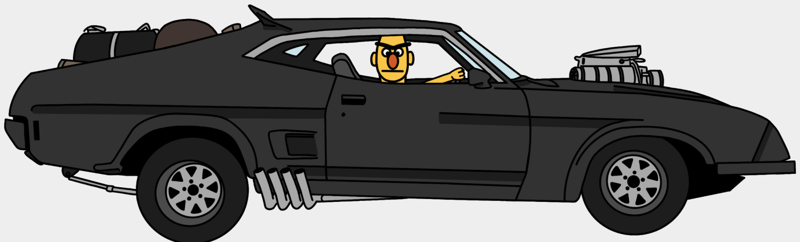




Key Takeaways

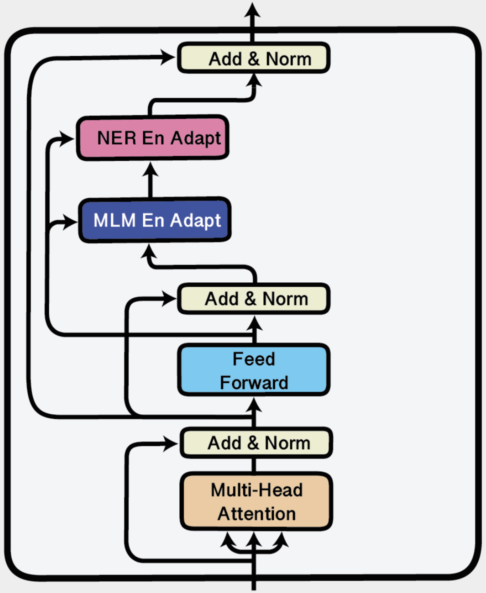
- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning

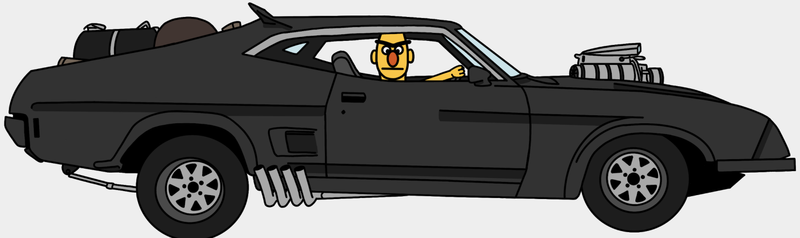




Key Takeaways

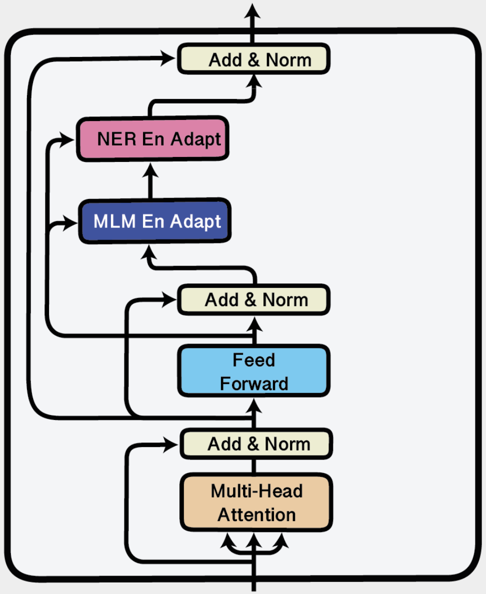
- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.

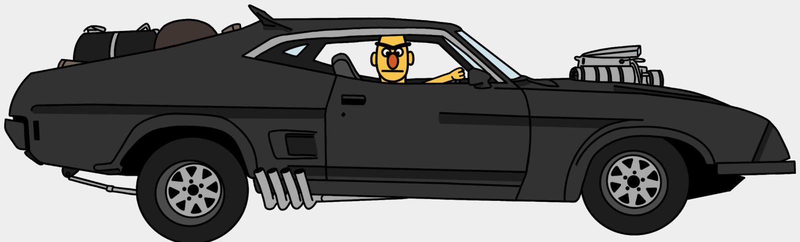




Key Takeaways

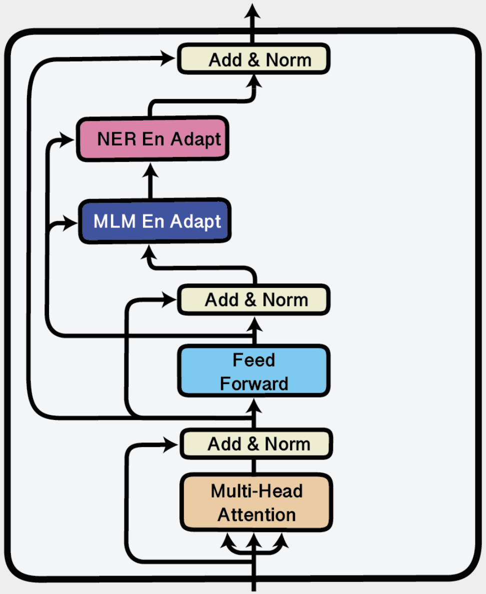
- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.

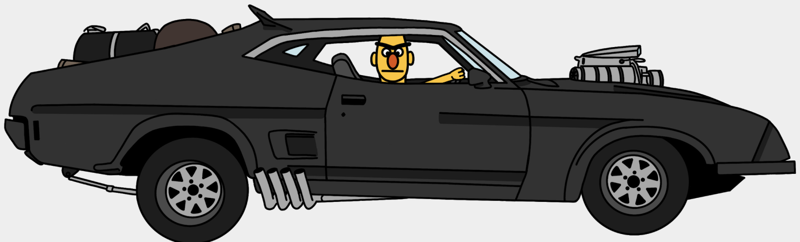




Key Takeaways

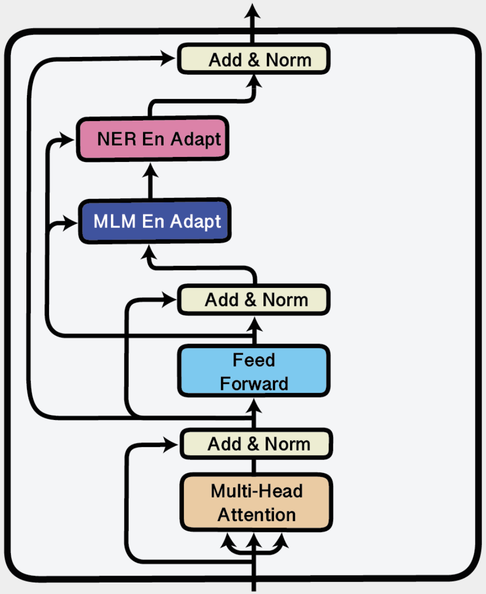
- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
- **Specializing** the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

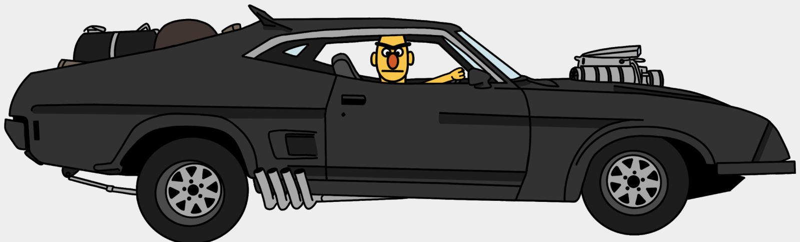




Key Takeaways

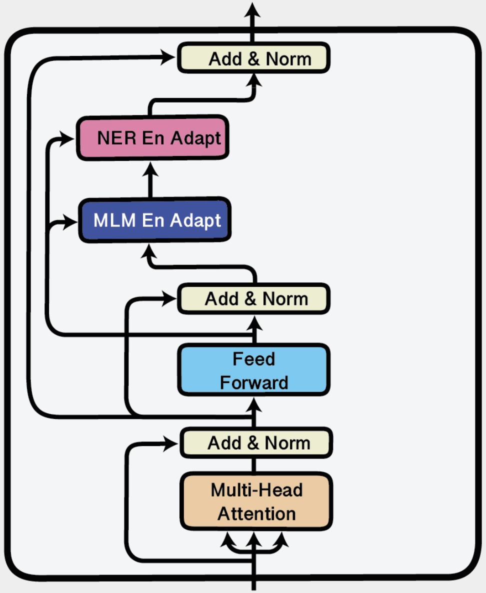
- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
- **Specializing** the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

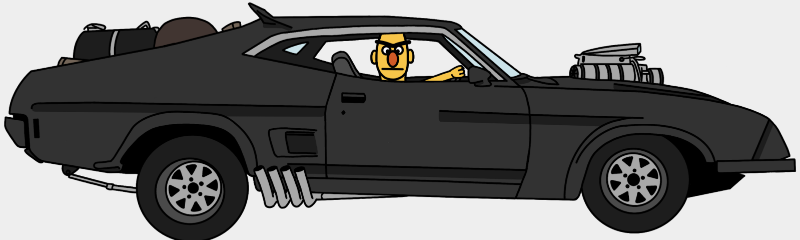




Key Takeaways

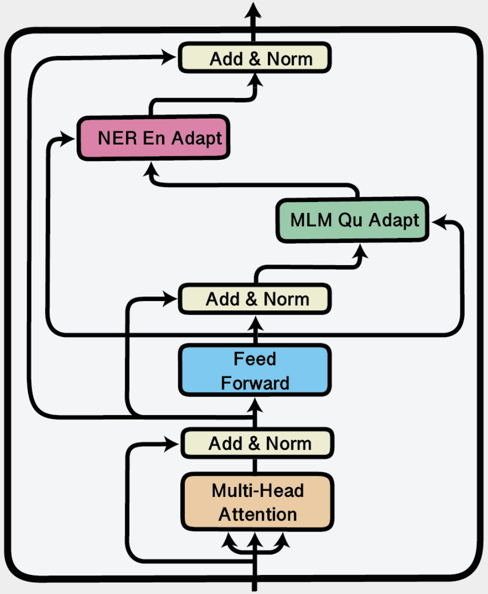
- **Adapters** train faster than normal fine-tuning, while maintaining the performance of full fine-tuning
- Adapters are **modular** units which can be **stacked** and fine-tuned **sequentially**.
- This is especially **helpful** for **zero-shot transfer** to unseen languages.
- **Specializing** the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

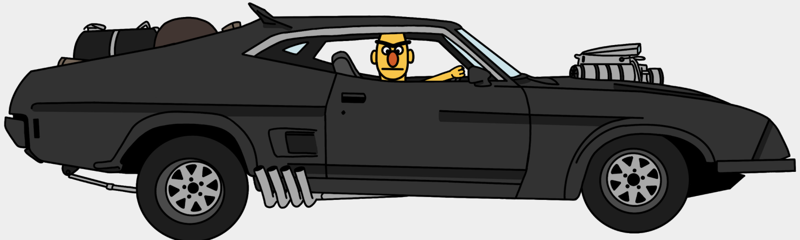




Key Takeaways

- **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

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of mono- lingual representations. In *Proceedings of the 58th Conference of the Association for Computational Linguistics, ACL 2020, Virtual Conference, July 6-8, 2020*.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 7057–7067.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Laurent Dinh, David Krueger, and Yoshua Bengio. 2015. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2790–2799.

References

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross- lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1946–1958.

Edoardo Maria Ponti, Ivan Vulić, Ryan Cotterell, Marinela Parovic, Roi Reichart, and Anna Korhonen. 2020. Parameter space factorization for zero-shot learning across tasks and languages. *arXiv preprint*.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020b. XCOPA: A Multilingual Dataset for Causal Commonsense Reasoning. *arXiv preprint arXiv:2005.00333*.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 151–164.

Melissa Roemmele, Cosmin Adrian Bejan, and An- drew S. Gordon. 2011. Choice of plausible alter- natives: An evaluation of commonsense causal rea- soning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Sym- posium, Technical Report SS-11-06, Stanford, Cali- fornia, USA, March 21-23, 2011*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Com- monsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.