# *Adapters*: A Unified Library for Parameter-Efficient and Modular Transfer Learning

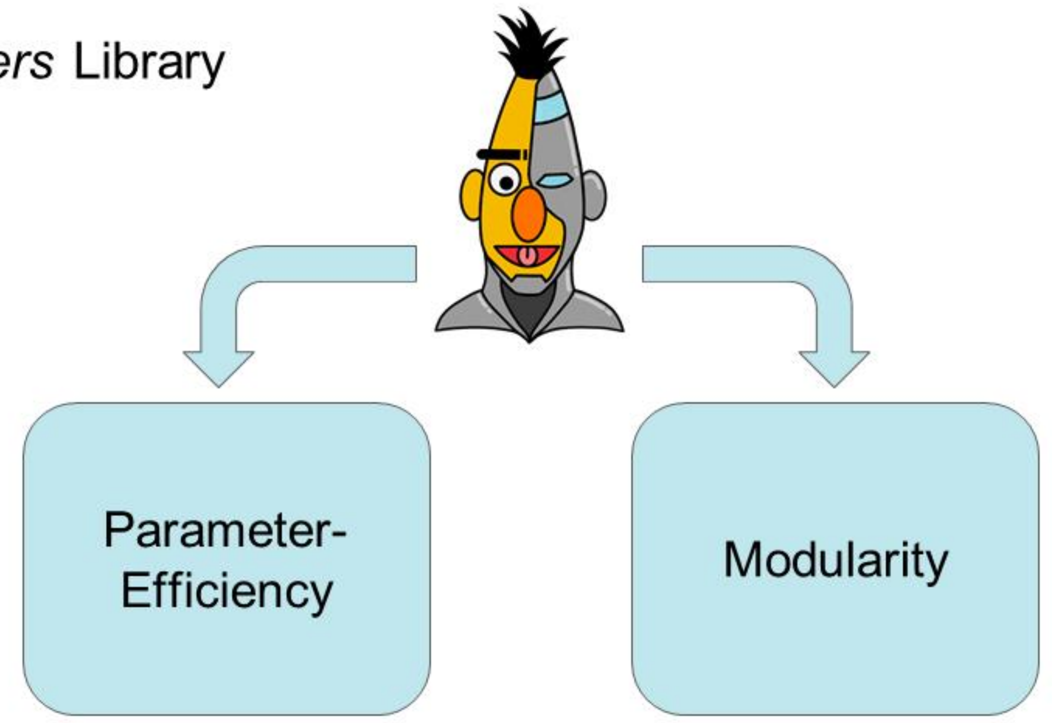
Clifton Poth*, Hannah Sterz*, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulic, Sebastian Ruder, Iryna Gurevych, Jonas Pfeiffer

https://adapterhub.ml

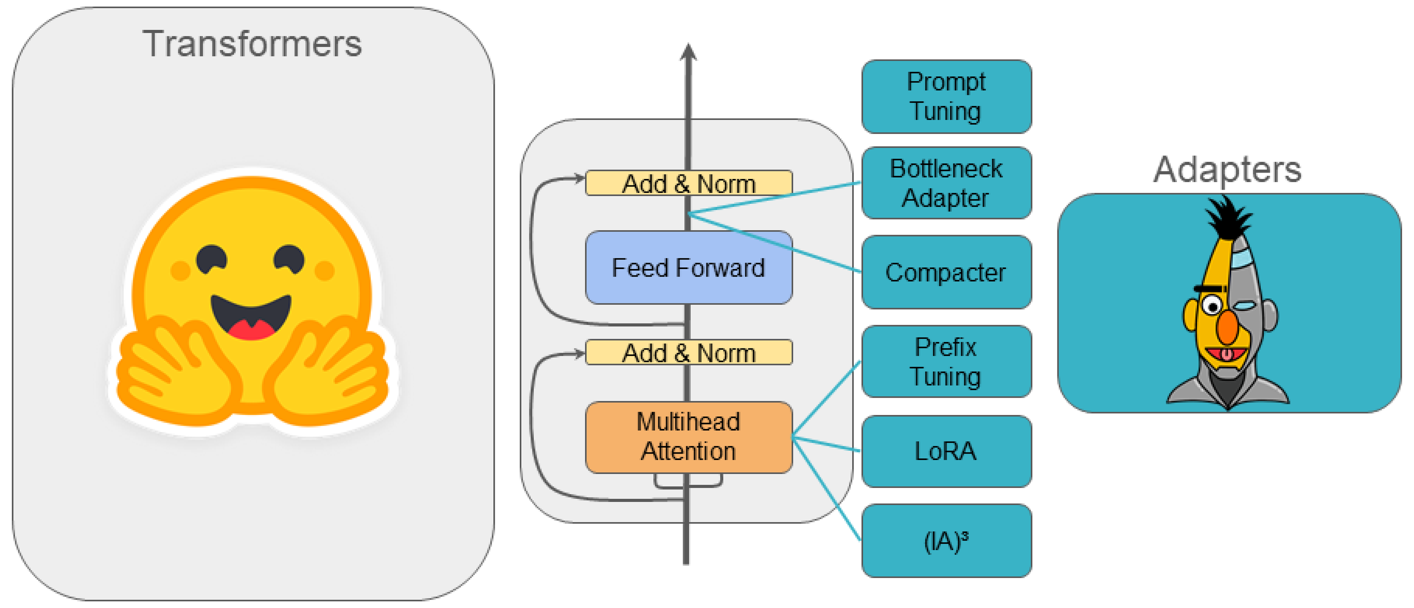
https://github.com/adapter-hub/adapters

`pip install adapters`

## *Adapters* Library



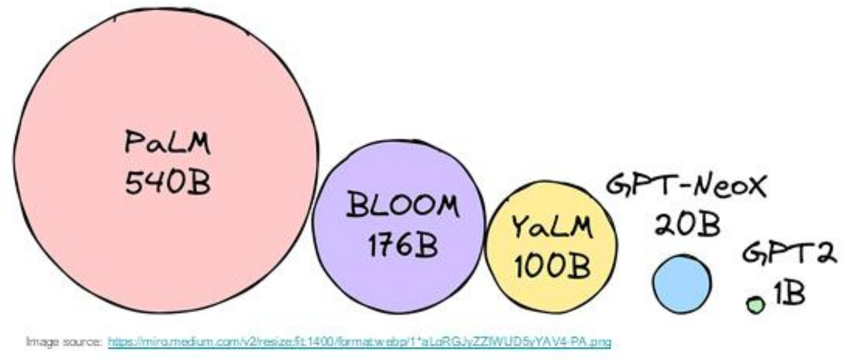

## *Adapters* is an add-on to Hugging Face's Transformers




## Why Parameter-Efficiency?

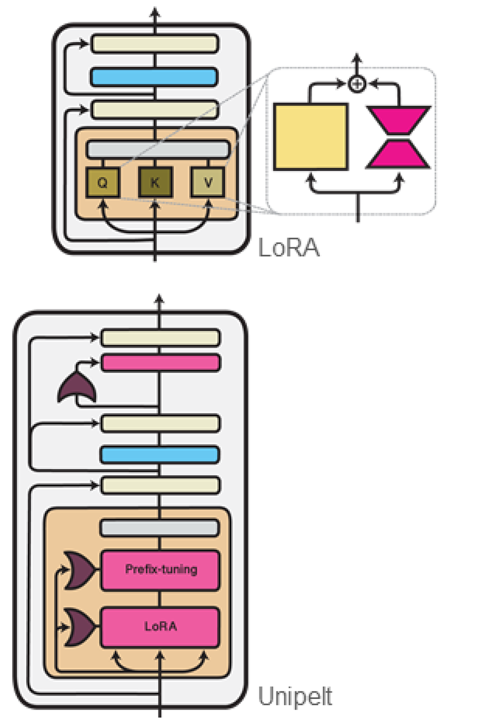
As LM sizes grow, full model fine-tuning becomes expensive

→ Fine-tuning a smaller set of parameters can be more time and memory efficient




## Supported Adapter Methods

- Single Methods
  - Implemented: Bottleneck, Compacter, LoRA, (IA)³, Prefix Tuning, Prompt Tuning, Invertible Adapters
- Complex Methods
  - Flexible combination of single methods in joint adapters setups
  - Examples: Mix-and-Match adapters or Unipelt




## Code Demo: Configure adapters

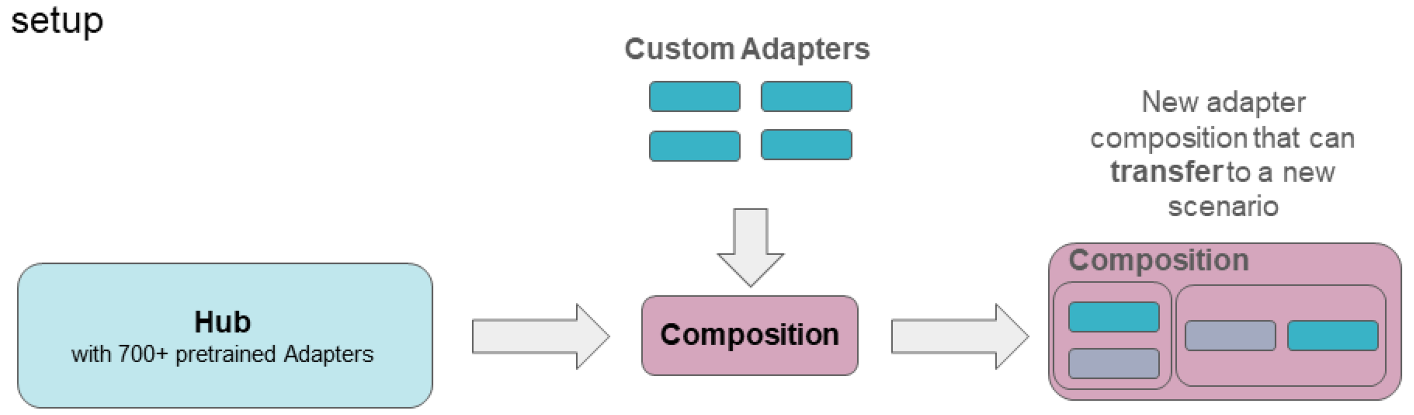
```python
import adapters
from adapters import ConfigUnion, PrefixTuningConfig, ParBnConfig
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained("microsoft/deberta-v3-base")

adapters.init(model)

adapter_config = ConfigUnion(
    PrefixTuningConfig(prefix_length=20),
    ParBnConfig(reduction_factor=4),
)
model.add_adapter("my_adapter", config=adapter_config, set_active=True)
```

1. Standard Transformers model
2. Add all adapter functionality
3. Define a complex adapter config
4. Add & enable new adapter

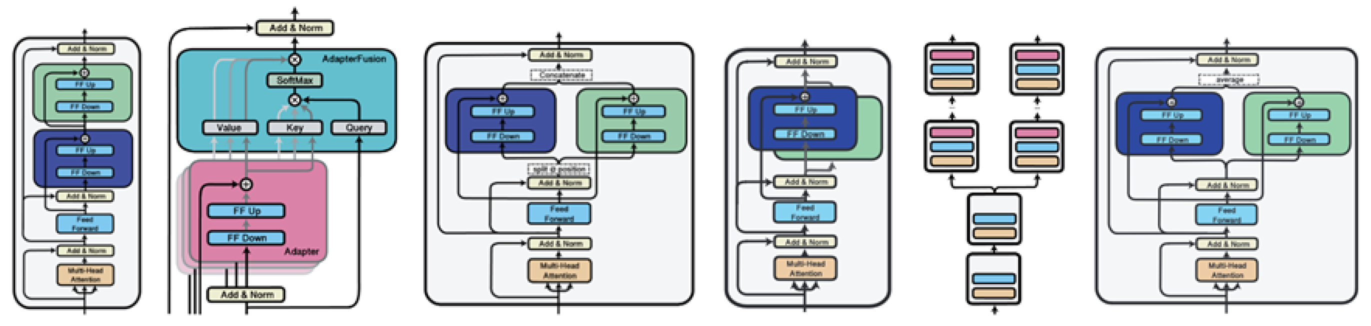
## Why Modularity?

- It's infeasible to have a specialized model for each use case.
- There are a lot of low resource use cases with insufficient training data

→ Modular composition enables transfer to new scenarios in a zero- or few-shot setup

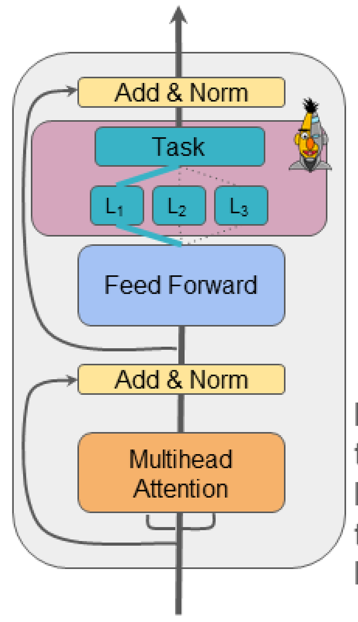



## *Adapters* uses composition blocks to enable modularity

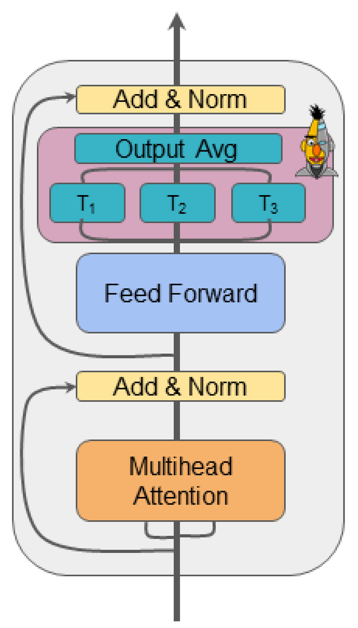


(a) Stack (b) Fuse (c) Split (d) BatchSplit (e) Parallel (f) Average

## Example Compositions




**MAD-X Setup**
transfer to new languages by routing through corresponding language adapter




**Ensemble Setup**
Weighted average the output of multiple task adapters

## Code Demo: Composing pre-trained adapters

```python
from adapters import AdapterSetup, AutoAdapterModel
import adapters.composition as ac
from transformers import AutoTokenizer


model = AutoAdapterModel.from_pretrained("roberta-base")
tokenizer = AutoTokenizer.from_pretrained("roberta-base")


qc = model.load_adapter("AdapterHub/roberta-base-pf-trec")
sent = model.load_adapter("AdapterHub/roberta-base-pf-imdb")


with AdapterSetup(ac.Parallel(qc, sent)):
    print(model(**tokenizer("What is AdapterHub?", return_tensors="pt")))
```

1. Load 🤗 model with AdapterModel classes
2. Load adapters from  or 🤗
3. Dynamically activate compositions

## Adapter methods can match full fine-tuning performance

| Method | Sequence Classification: CoLA Dev MCC | MNLI Dev Acc. | MRPC Dev F1 | QNLI Dev Acc. | QQP Dev F1 | RTE Dev Acc. | SST-2 Dev Acc. | Regression: STS-B Dev PCC | Mult. Choice: Cosmos QA Dev Acc. | Extract. QA: SQuAD v2 Dev F1 | Tagging: CoNLL-2003 Test F1 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| double_seq_bn | 63.58 (±19.19) | 87.61 (±26.41) | **93.31** (±4.52) | 92.84 (±17.17) | 91.58 (±36.83) | **80.87** (±11.09) | 94.73 (±17.51) | 90.85 (±27.16) | 70.99 (±16.87) | **84.89** (±5.52) | **96.24** (±17.65) | **86.14** (±18.17) |
| seq_bn | **71.22** (±23.40) | 87.5 (±20.39) | 92.91 (±4.54) | **93.15** (±15.83) | 89.69 (±21.31) | 79.42 (±9.81) | 95.18 (±13.26) | 89.44 (±20.33) | 69.68 (±16.44) | 82.88 (±**1.04**) | 96.21 (±11.48) | 86.12 (±14.34) |
| par_bn | 63.95 (±23.72) | 87.44 (±21.66) | 93.24 (±4.65) | 93.04 (±17.26) | 88.32 (±33.14) | 77.98 (±10.95) | 94.95 (±16.97) | 90.33 (±**5.64**) | **80.10** (±18.47) | 82.56 (±6.70) | 91.95 (±27.60) | 85.81 (±16.98) |
| compacter | 55.52 (±**13.67**) | 86.10 (±**1.99**) | 90.43 (±**3.58**) | 92.42 (±**2.68**) | 86.68 (±**2.14**) | 68.59 (±**4.91**) | 94.15 (±**0.81**) | 90.06 (±23.27) | 67.91 (±10.42) | 79.20 (±8.87) | 91.27 (±8.58) | 82.03 (±**7.36**) |
| prefix_tuning | 61.62 (±4.93) | 86.98 (±18.91) | 91.06 (±4.09) | 92.46 (±9.55) | 87.07 (±15.58) | 71.12 (±6.06) | **95.18** (±0.54) | 90.13 (±29.23) | 66.13 (±**3.44**) | 78.16 (±2.41) | 95.15 (±**2.49**) | 83.19 (±8.84) |
| lora | 63.99 (±20.64) | 87.59 (±4.29) | 92.60 (±4.39) | 93.11 (±3.77) | 88.48 (±2.57) | 80.26 (±9.28) | 94.99 (±8.48) | 90.72 (±19.31) | 70.63 (±8.65) | 82.46 (±8.86) | 91.85 (±21.68) | 85.15 (±10.17) |
| ia3 | 63.03 (±21.39) | 86.19 (±5.08) | 92.32 (±3.94) | 91.88 (±3.73) | 86.41 (±13.46) | 76.89 (±7.17) | 94.42 (±2.13) | 90.65 (±29.16) | 66.85 (±9.69) | 78.52 (±10.11) | 91.56 (±21.94) | 83.52 (±11.62) |
| **Full Fine-tuning** | 63.66 | **87.63** | 90.20 | 92.81 | **91.92** | 78.77 | 94.81 | **91.20** | 68.87 | 82.91 | 95.23 | 85.27 |

Using roberta-base as base Transformer model

## Differences from AdapterHub v1

| | AdapterHub v1 | *Adapters* |
|---|---|---|
| Design | Fork of *Transformers* | Self-contained add-on |
| Adapter methods | 2 | 10 |
| Complex configurations | ✗ | ☑ |
| Composition blocks | ✗ | ☑ (6) |
| Model architectures | 3 | 20 |
| AdapterHub.ml/HFHub integration | ☑ / ✗ | ☑ / ☑ |

## Summary

*Adapters*…

- is an add-on to *Transformers* for easy parameter-efficient fine tuning and modular transfer
- supports 10 different adapter methods for 20 different models
- comes with 6 composition blocks for easy transfer
- provides access to 700+ pretrained adapters