TencentPretrain: A Scalable and Flexible Toolkit for Pre-training Models of Different Modalities

Zhe Zhao[1], Yudong Li[2], Cheng Hou[1], Jing Zhao[1], Rong Tian[1], Weijie Liu[1], Yiren Chen[1], Ningyuan Sun[1], Haoyan Liu[1], Weiquan Mao[1], Han Guo[1], Weigang Guo[1], Taiqiang Wu[1], Tao Zhu[1], Wenhang Shi[3], Chen Chen[1], Shan Huang[1], Sihong Chen[1], Liqun Liu[1], Feifei Li[1], Xiaoshuai Chen[1], Xingwu Sun[1], Zhanhui Kang[1], Xiaoyong Du[3], Linlin Shen[2], Kimmo Yan[1]

[1] Tencent Hunyuan AI Foundation Model Oteam

[2] School of Computer Science and Software Engineering, Shenzhen University

[3] School of Information and DEKE, MOE, Renmin University of China

Abstract

Recently, the success of pre-training in text domain has been fully extended to vision, audio, and cross-modal scenarios. The proposed pre-training models of different modalities are showing a rising trend of homogeneity in their model structures, which brings the opportunity to implement different pre-training models within a uniform framework. In this paper, we present TencentPretrain, a toolkit supporting pre-training models of different modalities. The core feature of TencentPretrain is the modular design. The toolkit uniformly divides pre-training models into 5 components: ***embedding***, ***encoder***, ***target embedding***, ***decoder***, and ***target***. As almost all of common modules are provided in each component, users can choose the desired modules from different components to build a complete pre-training model. The modular design enables users to efficiently reproduce existing pre-training models or build brand-new one. We test the toolkit on text, vision, and audio benchmarks and show that it can match the performance of the original implementations.

1 Introduction

Pre-training on large-scale data and then fine-tuning on downstream tasks has become a paradigm for text, vision, and audio tasks (Devlin et al., 2019; Bao et al., 2021; Baevski et al., 2020). In addition to the similarity in the pipeline paradigm, these pre-training models as well have close model structures: On one hand, most of them consist of the following components, ***embedding***, ***encoder***, ***target embedding***, ***decoder***, and ***target***, on the other hand, many modules in above components are shared among models of different modalities. For example, the transformer module (in encoder component) (Vaswani et al., 2017), which is successful in the field of text, is increasingly being applied to the vision and audio modalities. (Dosovitskiy et al., 2020; Gulati et al., 2020). Table 1 lists the commonly used pre-training models and their modules.

Homogeneity trend of pre-training models is becoming obvious, which makes it possible to integrate them into a uniform framework. An representative work in this direction is Huggingface Transformers (Wolf et al., 2020), which exploits a non-modular design mode. For each pre-training model in Huggingface Transformers, several separate classes are created, and the code is not refactored with additional abstractions. Users can develop their pre-training models independently which is useful to collaborative development in the community. However, in this design mode, users need to implement the model from scratch when adding a new pre-training model, requiring considerable code work. In addition, with the increased number of pre-training models, the number of classes and lines of code also increases linearly. Codes with the same function may be written many times, which degrades the readability and maintainability of the project.

In response to shortcomings of non-modular design mode, we introduce TencentPretrain, a modular toolkit specially designed for pre-training models of different modalities. As shown in Figure 1, TencentPretrain has five components, namely ***embedding***, ***encoder***, ***target embedding***, ***decoder***, and ***target***. Among them, ***target embedding*** and ***decoder*** components are optional, since the targets of many pre-training models do not involve sequence decoding (Zhang et al., 2020; Lewis et al., 2020). TencentPretrain is hierarchical modular designed with two degrees of freedom. At component level, users are free to combine modules within a component, for example, combining multiple modules in ***target*** to perform multi-task pre-training (Lan et al., 2019; Sun et al., 2020). At model level,

*Corresponding Author
E-mail: nlpzhezhao@tencent.com

Pre-training model	Modality	Embedding	Encoder	Target embedding	Decoder	Target
ELMo (Peters et al., 2018)	Text	word	bi-lstm	-	-	bilm
Infersent (Conneau et al., 2017)	Text	word	gru	-	-	cls
CoVe (McCann et al., 2017)	Text	word	lstm	word	lstm	lm
BERT (Devlin et al., 2019)	Text	word, pos, seg	transformer	-	-	mlm, sp
GPT-2 (Radford et al., 2019)	Text	word, pos	transformer	-	-	lm
T5 (Raffel et al., 2020)	Text	word	transformer	word	transformer	lm
ViT (Dosovitskiy et al., 2020)	Vision	patch, pos	transformer	-	-	cls
BEiT (Bao et al., 2021)	Vision	patch, pos	transformer	-	-	mlm
S2T (Wang et al., 2020)	Audio	speech, pos	transformer	word, pos	transformer	lm
ViLT (Kim et al., 2021)	Text-vision	word_patch, pos, seg	transformer	-	-	mlm, cls

Table 1: Typical pre-training models and their modules. The above models use different variants of transformer. Due to the page limit, we do not list the details of transformer module in encoder component. In addition, abbreviations are used in embedding and target columns. pos and seg respectively stand for position and segment embeddings. bilm, cls, lm, mlm, sp respectively stand for bi-directional language model, classification, language model, masked language model, sentence prediction.

users can combine modules from different components to constitute a complete pre-training model.

Modularity in design makes TencentPretrain scalable with the increasing number of newly proposed pre-training models. Users are allowed to reuse existing modules with little efforts, avoiding repeated implementation of core functions. At the same time, TencentPretrain provides a robust and clear interface among different components. It brings flexibility, allowing users to build custom model structures through a configuration file without any code work.

TencentPretrain is implemented with PyTorch (Paszke et al., 2019), and it supports distributed training and DeepSpeed optimization library (Rasley et al., 2020). TencentPretrain is fully connected with Huggingface Transformers, providing comprehensive conversion scripts of pre-training models between the two frameworks. Users can switch between the two frameworks at low cost. TencentPretrain is tested on text, vision, and audio benchmarks and is able to reproduce the results of SOTA pre-training models. The TencentPretrain toolkit is publicly available at `https://github.com/Tencent/TencentPretrain`.

2 Related Work

2.1 Pre-training models

Pre-training models have been widely applied in text scenario. The success of pre-training is largely due to the powerful encoders for feature extraction (e.g., LSTM and Transformer), as well as the progress of pre-training target for learning knowledge from unsupervised corpus (Zhang et al., 2020; Lewis et al., 2020; Lan et al., 2019). More recently, the text pre-training paradigm has been replicated in other modalities. For example, Transformer encoder (and its variants) has been widely used in vision (Dosovitskiy et al., 2020), audio (Gulati et al., 2020; Chen et al., 2022), and vision-language tasks (Radford et al., 2021; Kim et al., 2021). Regarding pre-training target component, text models have inspired models of other modalities. Mirroring the idea of masked language modeling (MLM), MAE (He et al., 2022), BEiT (Bao et al., 2021), and SimMIM (Xie et al., 2022) use masked image modeling (MIM) for self-supervised vision pre-training. Speech model Wav2vec2.0 (Baevski et al., 2020) exploit negative sampling in pre-training target, which is previously used in word embedding (Mikolov et al., 2013) and sentence prediction models (Logeswaran and Lee, 2018; Devlin et al., 2019; Lan et al., 2019).

In addition to the sharing of modules, several works have recently shown the feasibility of using the same pre-trained weight to handle different modalities simultaneously. For example, ERNIE-ViLG (Zhang et al., 2021) and Talk2Face (Li et al., 2022) exploit prefix language model to achieve bi-directional text-and-image generation. PolyViT uses a single transformer model for image, video and audio classification (Likhosherstov et al., 2021).

It can be seen that homogeneity trend of pre-training models is becoming obvious, from sharing modules, to using the same network and parameters. This inspires us to build a unified framework that can implement various pre-training models efficiently.

2.2 Toolkits with modular design

Modular design regards a complex system as the combination of multiple modules, each of which

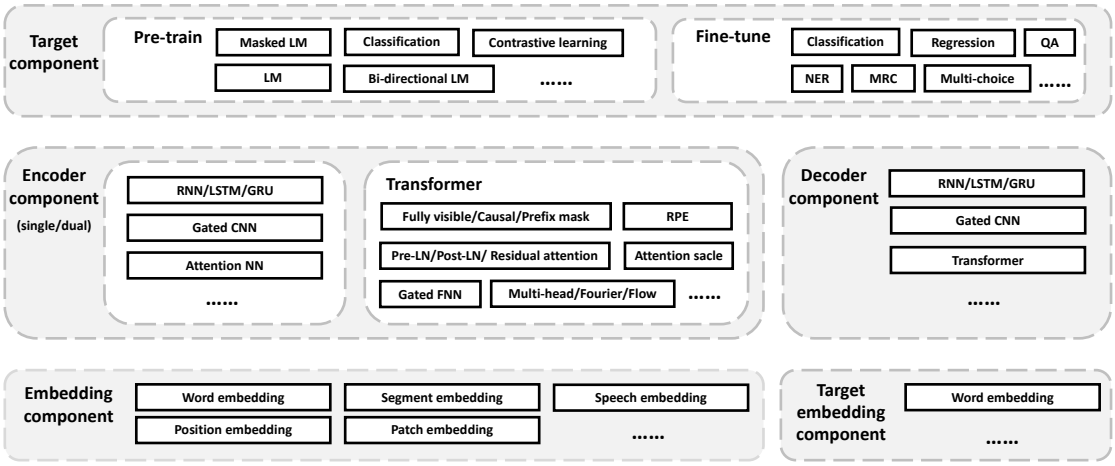

Figure 1: The architecture of TencentPretrain. Pre-training models are implemented through module combination. TencentPretrain encourages reusing the existing modules and writing code at the module granularity.

can be independently modified and replaced. In the field of artificial intelligence, a typical work with modular design is Keras (Chollet et al., 2015). The core data structure of Keras is layer. Keras allows building arbitrary graphs of layers to construct NN models. In the NLP field, modular toolkits are prevailing and they decompose models from different perspectives with different abstraction levels. For example, OpenNMT (Klein et al., 2017) is a modular toolkit designed for NMT. It builds an NMT model through the combination of encoder and decoder modules. Related NLP modular toolkits include OpenAttack (designed for text attack) (Zeng et al., 2021), Ngram2vec (designed for word embedding) (Zhao et al., 2017), TextFlint (designed for robustness evaluation) (Wang et al., 2021), NeuralClassifier (designed for text classification) (Liu et al., 2019a), and etc.

Inspired by the above-mentioned works, this paper proposes TencentPretrain, a modular designed toolkit for pre-training models of different modalities. Compared with Huggingface Transformers (Wolf et al., 2020), the most well-known pre-training toolkit, TencentPretrain provides additional abstractions on pre-training model implementations, splitting a complete model into multiple modules hierarchically. Pre-training weights between two toolkits can be switched easily. In fact, TencentPretrain can be regarded as the high-level encapsulation of Huggingface Transformers.

It is worth mentioning that TencentPretrain reuses part of the code in UER (Zhao et al., 2019), which is published in 2019 and supports several text pre-training models. Compared with UER, TencentPretrain is improved in three aspects: 1) It supports the modular design within components, providing a more scalable manner to build pre-training models; 2) The ***target embedding*** and ***decoder*** components are introduced to support sequence generation; 3) In addition to text, TencentPretrain supports vision, audio, and cross-modal pre-training models. Currently, TencentPretrain supports around 30 pre-training models.

3 Framework

The current mainstream pre-training models are basically similar in structure. In the embedding component, the data is mapped into an embedding matrix. And then the matrix is passed through the encoder. Finally the target layer performs pre-training tasks according to the output of the encoder layer. If the pre-training task requires sequence generation, the decoder is inserted between the encoder and the target.

Figure 1 demonstrates the overall framework of TencentPretrain. It divides a pre-training model into five components, and various modules are provided in each component. In practice, a user firstly selects one or multiple modules from each component (modularization within component), and then combine modules from different components to build a pre-training model (modularization cross components). In the rest of this section, we respectively introduce the above five components and modules included in them.

3.1 Embedding

In the ***embedding*** component, TencentPretrain converts text, image, and audio modal data into em-

bedding matrix. The matrix holds the low-level features as the input to the encoder.

TencentPretrain also contains auxiliary embedding modules, e.g., position embedding and segment embedding. The embedding of pre-training model is usually obtained by the addition of multiple modules. As shown in Table 1 (Embedding column), the addition of word, position, and segment embeddings constitutes the embedding layer of BERT; the addition of patch and position embeddings constitutes the embedding layer of ViT. TencentPretrain supports hierarchical modular design, enabling users to freely combine modules within ***embedding*** component to construct the desired embedding layer. This design greatly reduces code redundancy since different models often use similar, instead of identical combinations.

3.2 Encoder

TencentPretrain supports traditional encoders (e.g., LSTM and CNN) (Hochreiter and Schmidhuber, 1997; Kim, 2014), as well as transformer and its variants (e.g., different normalization (He et al., 2021), attention (Lee-Thorp et al., 2021), masking strategies (Dong et al., 2019)). Users can construct customized transformer encoder by combining related options.

In addition, TencentPretrain supports dual-stream encoder, with which the users specify two encoder modules separately. Dual-stream encoder is usually used by models related with semantic search, such as text pair model SBERT (Reimers and Gurevych, 2019) and text-image pair model CLIP (Radford et al., 2021).

3.3 Target embedding and decoder (optional)

The pre-training tasks of some models involve sequence generation. These models require modules in ***target embedding*** component and ***decoder*** component. The modules in these two components are identical with the modules in ***embedding*** component and ***encoder*** component respectively.

3.4 Target

The module in ***target*** component receives high-level features obtained from encoder (or decoder) and then uses the features to perform pre-training tasks. Specifically, the target estimates gradients by objectives and updates the network weights. The target is of vital importance to the performance and has been extensively investigated in the pre-training field (Devlin et al., 2019; Lan et al., 2019; Sun et al., 2020). TencentPretrain supports comprehensive target modules, including language model (Radford et al., 2019), classification (Conneau et al., 2017), contrastive learning (Radford et al., 2021), etc.

Sometimes pre-training models use multiple tasks, e.g., predicting word and sentence relationship simultaneously in BERT and ALBERT. And multi-task is especially common in cross-modal scenario (Kim et al., 2021; Lu et al., 2019; Qi et al., 2020) since pre-training models have to deal with supervision signals from different modalities. The model can learn knowledge from different perspectives through multiple tasks. With the characteristic of hierarchical modular design, TencentPretrain facilitates the implementation of multi-task pre-training models. One can introduce multiple tasks by combining different modules in ***target*** component. The pre-training task can be easily added, modified, and replaced.

3.5 Downstream task fine-tuning

TencentPretrain supports comprehensive downstream tasks, including classification, regression, sequence labeling, reading comprehension, question answering, automated speech recognition, etc. As shown in Figure 1, the downstream task model can be constructed by replacing the pre-training target with specific task. In evaluation section, we show the performances of TencentPretrain on a range of benchmarks.

4 Usage

This section provides examples of building pre-training models with TencentPretrain. The modular design enables the users to quickly build the pre-training model through the combination of modules. Modules used in pre-training models are specified in configuration files and the examples are shown as follows:

```
# BERT implementation
{
   "embedding": ["word", "pos", "seg"],
   "encoder": "transformer",
   "target": ["mlm", "sp"]
}

# T5 implementation
{
   "embedding": ["word"]
   "encoder": "transformer"
   "tgt_embedding": ["word"]
```

Due to the page limit, we do not list entire configuration files. More details (e.g., Transformer encoder options) can be found in TencentPretrain project.

```
    "decoder": "transformer"
    "target": ["lm"]
}

# ViLT implementation
{
"embedding": ["patch_word", "pos", "seg"]
"encoder": "transformer"
"pooling": "first"
"target": ["cls", "mlm"]
}

# CLIP implementation
{
    "stream_0":{
        "embedding": ["word", "pos"],
        "encoder": "transformer",
        "pooling": "first"
    }
    "stream_1":{
        "embedding": ["patch", "pos"],
        "encoder": "transformer"
        "pooling": "first"
    }
    "target": ["clr"]
}
```

- BERT configuration file provides modules in ***embedding***, ***encoder***, and ***target*** components. Since BERT has two pre-training tasks, its target is the combination of masked language model (mlm) and sentence prediction (sp).
- T5 involves text generation. Its configuration file specifies modules used in ***target embedding*** and ***decoder*** components.
- ViLT, an image-text pre-training model, is basically similar with text pre-training model BERT. The main difference is that an image-text embedding module is used in ***embedding*** component.
- CLIP is a dual-stream model. The modules in stream0 process text and the modules in stream1 process image. Contrastive learning (clr) module is used in ***target*** component.

If the desired pre-training model cannot be built by the combination of existing modules, TencentPretrain encourages users to develop a new module, and combine it with existing modules. We take the implementation of ASR model S2T (Wang et al., 2020) as an example. Most modules required by S2T are available and we only need to implement a new module, speech embedding, which greatly speeds up the implementation process.

TencentPretrain and Huggingface Transformers are interoperable. The conversion scripts are publicly available, and the weights of different pre-training models can be converted between the two frameworks. In practical use, users are free to switch between these two frameworks.

With TencentPretrain, we build a pre-trained weight model zoo. Each pre-trained weight has two versions which can be loaded by either TencentPretrain or Huggingface Transformers. Currently, the TencentPretrain model zoo includes over 50 pre-trained weights. We provide pre-training data as well as training details, allowing users to reproduce results with less effort. The weights (pre-trained by TencentPretrain) are currently downloaded over 500 thousand times per month.

Model	HF	UER	TP
Transformer	1135	749	795
+BERT	+822	+130 (+word_pos_seg, bert)	+149 (+pos,seg, mlm,sp)
+RoBERTa	+696	+92 (+word_pos,mlm)	+0
+GPT-2	+688	+0	+0
+T5	+1008	+17(+word)	+0
+ViT	+493	-	+59(+patch)
+S2T	+824	-	+51(+speech)
+ViLT	+618	-	+15 (+word_patch)

Table 2: The number of code lines required for implementing a new pre-training model. The comment line in code is not counted. For UER and TencentPretrain, the added modules are also listed. Green and violet are used to denote embedding and target modules. Since UER does not support modularization within component, it has to introduce more modules (classes), e.g., word_pos_seg embedding and bert target, which are decomposed into multiple modules in TencentPretrain.

5 Evaluation

This section evaluates TencentPretrain framework quantitatively. Firstly, we compare TencentPretrain with non-modular framework in terms of implementation cost. Then we show that TencentPretrain can reproduce the results of SOTA models on a range of benchmarks.

5.1 Implementation cost

The number of code lines is used to estimate the implementation cost. We only count the code lines in classes inheriting *nn.Module*. We compare three frameworks, Huggingface Transformers

https://github.com/Tencent/TencentPretrain/tree/main/scripts
https://huggingface.co/uer
For Huggingface account, we inherit UER account instead of using TencentPretrain account.

Model	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	AVG
BERT-base (Ori.) (Devlin et al., 2019)	83.9	90.7	90.7	65.7	92.3	88.9	56.5	88.6	82.2
BERT-base (DistilBERT) (Sanh et al., 2019)	86.7	91.8	89.6	69.3	92.7	88.6	56.3	89.0	83.0
BERT-base (DynaBERT) (Hou et al., 2020)	84.8	92.0	90.9	71.1	92.9	87.7	58.1	89.8	83.4
BERT-base (Metadistil) (Zhou et al., 2022)	84.6	91.2	91.4	71.4	93.0	87.6	58.9	90.2	83.5
BERT-base (Ours)	83.4	91.1	91.2	67.9	92.4	86.5	59.6	89.1	82.6
RoBERTa-large (Ori.) (Liu et al., 2019b)	90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4	88.9
RoBERTa-large (Ours)	90.4	94.7	92.1	86.6	96.4	90.2	67.0	92.5	88.7

Table 3: The comparison of TencentPretrain with other implementations on GLUE benchmark. We pre-train from scratch and then fine-tune on a range of datasets

(HF), UER, and TencentPretrain (TP). Huggingface Transformers exploits non-modular design. UER exploits semi-modular design, which doesn't support modularization within component.

When we continue to add new pre-training models (as shown in Table 2 from top to bottom), the number of code lines required by the TencentPretrain is less than the other two toolkits. Take RoBERTa as an example, TencentPretrain does not require any code work since it reuses modules for BERT. UER needs to add word_pos module in ***embedding*** component and mlm module in ***target*** component. Huggingface Transformers builds a series of classes specific to RoBERTa, such as RoBERTaModel, RoBERTaEmbeddings, RoBERTaEncoder, RoBERTaPooler, which greatly increases the number of code lines. For other pre-training models, the conclusions are similar. The homogeneity among pre-training models makes the modular design much more advantageous.

5.2 Reproducibility

In this section, we follow the experimental settings of original papers. The scripts for running models on benchmarks are organized here, and users can easily reproduce the results in Table 3 and 4.

For text modality, we use GLUE benchmark to test TencentPretrain's performance. BERT-base and RoBERTa-large are used as test models. The results of BERT-base are listed in the first five rows in Table 3. As shown in AVG column, our result is 82.6, which falls into the range of 82.2-83.5 (the lowest and highest results reported by other papers). The average scores reported by DynaBERT and Metadistil are slightly higher than our result. One of the reasons is that development set of RTE only includes 277 instances, which leads to large fluctuations. The RTE results reported by DynaBERT and Metadistil are 3 point higher than our implementation. For RoBERTa-large, we can observe that our implementation results are close to the results reported by original RoBERTa paper.

Table 4 provides the results on vision and audio tasks. ViT (Dosovitskiy et al., 2020) and BEiT (Bao et al., 2021) are used as test models for vision datasets. Top1 accuracy on vision datasets is reported. The original paper of BEiT only reports results on ImageNet. For audio dataset, we report the Automatic Speech Recognition (ASR) results on LibriSpeech with S2T (Wang et al., 2020). Word Error Rate (WER) is shown in Table 4 (bottom). We can observe that the results of TencentPretrain are close to the results reported by original papers.

Model	CIFAR10	CIFAR100	ImageNet 1000
ViT-base	98.95	91.67	83.97
ViT-base(Ours)	98.73	92.12	83.97
BEiT-large	-	-	87.30
BEiT-large(Ours)	-	-	87.24

Model	devclean	devother	testclean	testother
S2T	3.8	8.9	4.4	9.0
S2T(Ours)	3.8	9.2	4.1	9.0

Table 4: The comparison of TencentPretrain with original implementations on datasets of vision and audio modalities.

6 Conclusion

This paper presents TencentPretrain, a pre-training toolkit characterized by modular design and multi-modal support. In TencentPretrain, pre-training models of different modalities are regarded as the combination of multiple modules, which is easy to configure and extensible. Furthermore, we quantitatively demonstrate that TencentPretrain facilitates the users to reuse existing modules and decreases the cost of model development. At last, we test TencentPretrain on a range of datasets and show that it can reproduce the SOTA results.

https://github.com/Tencent/TencentPretrain/wiki/Competition-solutions

Acknowledgements

The work was supported by the National Natural Science Foundation of China under Grant No. 62072458. We are grateful to Lusheng Zhang, Xiayu Li, Jian Kang, Jinwen Luo, Weidong Guo, Jiachi Liu, Jianwei Cui, Dongxiao Huang, Xingyu Bai, Yang Xu, Huanqin Wu, Tongwen Huang, Peng Meng, Yanming Xu, Chunquan Chen, Xuefeng Yang, Qi Ju for code contribution. We also received helpful ideas and feedback from members of the TencentNLP Oteam and Institute of Computer Vision, Shenzhen University.

References

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2021. Beit: Bert pre-training of image transformers. In *International Conference on Learning Representations*.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. 2022. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*.

François Chollet et al. 2015. keras.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in Neural Information Processing Systems*, 32.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

Ruining He, Anirudh Ravula, Bhargav Kanagal, and Joshua Ainslie. 2021. Realformer: Transformer likes residual attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 929–943.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793.

Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *International Conference on Machine Learning*, pages 5583–5594. PMLR.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *EMNLP*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Yudong Li, Xianxu Hou, Zhe Zhao, Linlin Shen, Xuefeng Yang, and Kimmo Yan. 2022. Talk2face: A unified sequence-based framework for diverse face generation and analysis tasks. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4594–4604.

Valerii Likhosherstov, Anurag Arnab, Krzysztof Choromanski, Mario Lucic, Yi Tay, Adrian Weller, and Mostafa Dehghani. 2021. Polyvit: Co-training vision transformers on images, videos and audio. *arXiv preprint arXiv:2111.12993*.

Liqun Liu, Funan Mu, Pengyu Li, Xin Mu, Jing Tang, Xingsheng Ai, Ran Fu, Lifeng Wang, and Xing Zhou. 2019a. Neuralclassifier: an open-source neural hierarchical multi-label text classification toolkit. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–92.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, 30.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Di Qi, Lin Su, Jia Song, Edward Cui, Taroon Bharti, and Arun Sacheti. 2020. Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *arXiv preprint arXiv:2001.07966*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq s2t: Fast speech-to-text modeling with fairseq. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 33–39.

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng,

Zexiong Pang, et al. 2021. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 347–355.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. 2022. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371.

Han Zhang, Weichong Yin, Yewei Fang, Lanxin Li, Boqiang Duan, Zhihua Wu, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie-vilg: Unified generative pre-training for bidirectional vision-language generation. *arXiv preprint arXiv:2112.15283*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Zhe Zhao, Hui Chen, Jinbin Zhang, Wayne Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju, and Xiaoyong Du. 2019. Uer: An open-source toolkit for pre-training models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 241–246.

Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 244–253.

Wangchunshu Zhou, Canwen Xu, and Julian McAuley. 2022. Bert learns to teach: Knowledge distillation with meta learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7037–7049.