

A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT

Ce Zhou^{1*} Qian Li^{2*} Chen Li^{2*} Jun Yu^{3*} Yixin Liu^{3*} Guangjing Wang¹
 Kai Zhang³ Cheng Ji² Qiben Yan¹ Lifang He³ Hao Peng² Jianxin Li²
 Jia Wu⁴ Ziwei Liu⁵ Pengtao Xie⁶ Caiming Xiong⁷ Jian Pei⁸
 Philip S. Yu⁹ Lichao Sun³

¹Michigan State University, ²Beihang University, ³Lehigh University,

⁴Macquarie University, ⁵Nanyang Technological University, ⁶University of California San Diego,

⁷Salesforce AI Research,⁸Duke University, ⁹University of Illinois at Chicago

Abstract

Pretrained Foundation Models (PFMs) are regarded as the foundation for various downstream tasks with different data modalities. A PFM (e.g., BERT, ChatGPT, and GPT-4) is trained on large-scale data which provides a reasonable parameter initialization for a wide range of downstream applications. In contrast to earlier approaches that utilize convolution and recurrent modules to extract features, BERT learns bidirectional encoder representations from Transformers, which are trained on large datasets as contextual language models. Similarly, the Generative Pretrained Transformer (GPT) method employs Transformers as the feature extractor and is trained using an autoregressive paradigm on large datasets. Recently, ChatGPT shows promising success on large language models, which applies an autoregressive language model with zero shot or few shot prompting. The remarkable achievements of PFM have brought significant breakthroughs to various fields of AI in recent years. Numerous studies have proposed different methods, datasets, and evaluation metrics, raising the demand for an updated survey.

This study provides a comprehensive review of recent research advancements, challenges, and opportunities for PFMs in text, image, graph, as well as other data modalities. The review covers the basic components and existing pretraining methods used in natural language processing, computer vision, and graph learning. Additionally, it explores advanced PFMs used for different data modalities and unified PFMs that consider data quality and quantity. The review also discusses research related to the fundamentals of PFMs, such as model efficiency and compression, security, and privacy. Finally, the study provides key implications, future research directions, challenges, and open problems in the field of PFMs. Overall, this survey aims to shed light on the research of the PFMs on scalability, security, logical reasoning ability, cross-domain learning ability, and the user-friendly interactive ability for artificial general intelligence.

*The authors contributed equally to this research. Correspondence to Ce Zhou(zhouce@msu.edu) and Qian Li (liqian@act.buaa.edu.cn).

Contents

1	Introduction	4
1.1	PFMs and Pretraining	4
1.2	Contribution and Organization	5
2	Basic Components	6
2.1	Transformer for PFMs	6
2.2	Learning Mechanisms for PFMs	7
2.3	Pretraining Tasks for PFMs	8
2.3.1	Pretraining Tasks for NLP	9
2.3.2	Pretraining Tasks for CV	9
2.3.3	Pretraining Tasks for GL	10
3	PFMs for Natural Language Processing	10
3.1	Word Representations Methods	11
3.2	Model Architecture Designing Methods	13
3.3	Masking Designing Methods	13
3.4	Boosting Methods	14
3.5	Instruction-Aligning Methods	16
3.6	Summary	18
4	PFMs for Computer Vision	18
4.1	Learning by Specific Pretext Task	19
4.2	Learning by Frame Order	20
4.3	Learning by Generation	21
4.4	Learning by Reconstruction	21
4.5	Learning by Memory Bank	22
4.6	Learning by Sharing	23
4.7	Learning by Clustering	25
4.8	Summary	26
5	PFMs for Graph Learning	27
5.1	Learning by Graph Information Completion	27
5.2	Learning by Graph Consistency Analysis	28
5.3	Learning by Graph Property Prediction	30
5.4	Learning by Masked Autoencoder	31
5.5	Other Learning Strategies on Graph Data	31
5.6	Summary	31
6	PFMs for Other Data Modality	32
6.1	PFMs for Speech	33
6.2	PFMs for Video	33
6.3	PFMs for Multimodal	34
6.4	PFM for Code Generation	35
6.5	SOTA Unified PFMs	35
7	Other Advanced Topics on PFMs	37
7.1	Model Efficiency	37
7.2	Model Compression	38
7.3	Security and Privacy	38

8 Future Research Challenges and Open Problems	39
8.1 Challenges on Data	40
8.2 Challenges on Foundation	40
8.3 Challenges on Model Design	41
8.4 Challenges on Finetuning and Prompt	41
8.5 Open Problems for Future PFM	42
9 Conclusion	42
A Basic Components	43
A.1 Basic Components on NLP	43
A.1.1 Language Model	43
A.2 Basic Components on GL	44
A.2.1 Notations and Definitions of Graphs	45
A.2.2 Learning Settings on Graphs	45
B Traditional Learning Methods	47
B.1 Traditional Text Learning	47
B.2 Traditional Image Learning	47
B.2.1 Convolution-Based Networks	48
B.2.2 Recurrent neural networks	48
B.2.3 Generation-Based Networks	48
B.2.4 Attention-Based Networks	49
B.2.5 Transformer-Based Networks	49
B.3 Traditional Graph Learning	49
C PFM Theory	51
C.1 Different Perspectives	51
C.2 Different Categories	51
D Pretext Task Taxonomy on CV	52
E PFM for Reinforcement Learning	53
F Evaluation Metrics	54
G Datasets	57
G.1 Downstream Tasks and Datasets on NLP	57
G.2 Downstream Tasks and Datasets on CV	62
G.3 Downstream Tasks and Datasets on Graph	66

1 Introduction

Pretrained Foundation Models (PFMs) are regarded as essential and significant components of Artificial Intelligence (AI) in the era of big data. The foundation model is first named in [1], which means a broader class of models and their functions. PFMs are extensively studied in the three major AI fields: natural language processing (NLP) [2], computer vision (CV) [3] and graph learning (GL) [4]. PFMs are powerful general models that are effective in various fields or across fields. They have demonstrated great potential in learning feature representations in various learning tasks, such as text classification [5], text generation [6], image classification [7], object detection [8], and graph classification [9]. PFMs show superior performance for training on multiple tasks with large-scale corpus and fine-tuning it to similar small-scale tasks, making it possible to initiate rapid data processing.

1.1 PFMs and Pretraining

PFMs are built upon the pretraining technique, which aims to train a general model using large amounts of data and tasks that can be fine-tuned easily in different downstream applications. The idea of pretraining originates from transfer learning [10] in CV tasks. Recognizing the effectiveness of pretraining in the field of CV, people have begun to use pretraining technology to enhance model performance in other areas. When pretraining techniques are applied to the NLP domain, well-trained language models (LMs) can capture rich knowledge beneficial for downstream tasks, such as long-term dependencies, hierarchical relationships, etc. In addition, the significant advantage of pretraining in the NLP field is that training data can be derived from any unlabeled text corpus, that is, there is an unlimited amount of training data in the pretraining process. Early pretraining is a static technique, such as NNLM [11] and Word2vec [12], but static methods were difficult to adapt to different semantic environments. Therefore, dynamic pretraining techniques are proposed, such as BERT [13], XLNet [14], etc. Fig. 1 depicts the history and evolution of PFMs in the NLP, CV, and GL domains. The PFMs based on the pretraining technique use large corpora to learn generic semantic representations. With the introduction of these pioneering works, various PFMs have emerged and been applied to downstream tasks and applications.

A great example of PFM application is ChatGPT¹. ChatGPT is fine-tuned from the generative pretrained transformer GPT-3.5, which was trained on a blend of text and code [15, 16]. ChatGPT applies reinforcement learning from human feedback (RLHF) [17, 18], which has become a promising way to align large language models (LLMs) with a human’s intent [19]. The surprisingly superior performance of ChatGPT may lead to a tipping point for a shift of training paradigm for each type of PFMs – applying *instruction aligning* techniques, e.g., reinforcement learning (RL), prompt tuning [20, 21, 22], and chain-of-thought (COT) [23, 24], to move towards artificial general intelligence.

We focus on reviewing PFMs for text, image, and graph, which is a relatively mature research taxonomy. For text, it is a multi-purpose LM to predict the next word or character in a sequence. For example, PFMs can be used for machine translation, question-answering systems, topic modeling, sentiment analysis, etc. For image, it is similar to PFMs on text, which uses huge datasets to train a big model suitable for many CV tasks. For graphs, a similar pretraining idea is also applied to obtain PFMs, which are used for many downstream tasks. Apart from the PFMs for a specific data domain, we also review and state some other advanced PFMs, such as the PFMs for speech, video, and cross-domain data, and multimodal PFMs. An exemplary illustration is the GPT-4 model, as described by OpenAI [25], which is a massive multimodal language model that can process both text and image inputs and generate text outputs. GPT-4 has demonstrated human-level performance on various professional and academic evaluation tasks. Moreover, there

¹<https://openai.com/blog/chatgpt/>

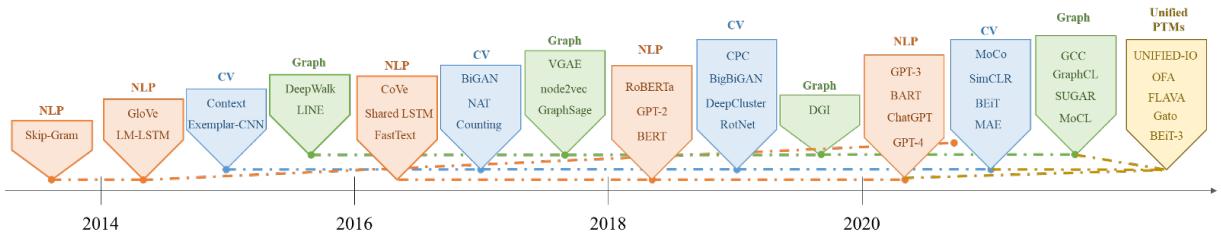


Figure 1: The history and evolution of PFMs.

is a growing trend in PFMs that deals with multimodal data, known as unified PFMs. This term refers to models that can handle different types of data such as text, images, and audio. In this regard, we provide a definition of unified PFMs and a review of the current state-of-the-art models in recent research. Notable examples include OFA [26], UNIFIED-IO [27], FLAVA [28], BEiT-3 [29], and others.

According to the features of existing PFMs, we conclude that the PFMs have the following two major advantages. First, minor fine-tuning is required to enhance the model performance on downstream tasks. Second, the PFMs have already been vetted on the quality aspect. Instead of building a model from scratch to solve a similar problem, we can apply PFMs to task-related datasets. The great promise of PFMs has inspired a wealth of related work to focus on the model efficiency [30], security [31, 32, 33, 34] and compression [35, 36].

1.2 Contribution and Organization

There are several survey studies [37, 8, 5, 6, 7, 1] that have reviewed the pretrained models for some specific areas such as text generation [6], visual transformer [7], objection detection [8].

Bommasani et.al. [1] summarize the opportunities and risks of the foundation model. However, existing works did not achieve a comprehensive review of PFMs in different areas (e.g., CV, NLP, GL, Speech, Video) and different aspects such as pretraining tasks, efficiency, efficacy, and privacy. In this survey, we specifically track the evolution of PFMs in the NLP domain, as well as how pretraining is transferred to and adopted by CV and GL. Compared with other surveys, there is no comprehensive introduction and analysis of existing PFMs from all three fields. Unlike reviews of previous pretrained models, we summarize existing models ranging from traditional models to PFMs with recent works in the three domains. Traditional models emphasize static feature learning. Dynamic PFMs give an introduction to structures, which is the mainstream research. We further present some other research for PFMs, including other advanced and unified PFMs, model efficiency and compression, security, and privacy. Finally, we summarize future research challenges and open problems in different domains. We also comprehensively present the related evaluation metrics and datasets in **Appendix F and G**. In summary, the main contributions are as follows:

- We present a solid and up-to-date review of the development of PFM in NLP, CV, and GL. Over the review, we discuss and provide insights about the generalized PFM design and pretraining methodology among the three major application domains.
- We summarize the development of PFMs in other multimedia areas such as speech and video. Besides, we discuss advanced topics about PFMs, including unified PFMs, model efficiency and compression, and security and privacy.

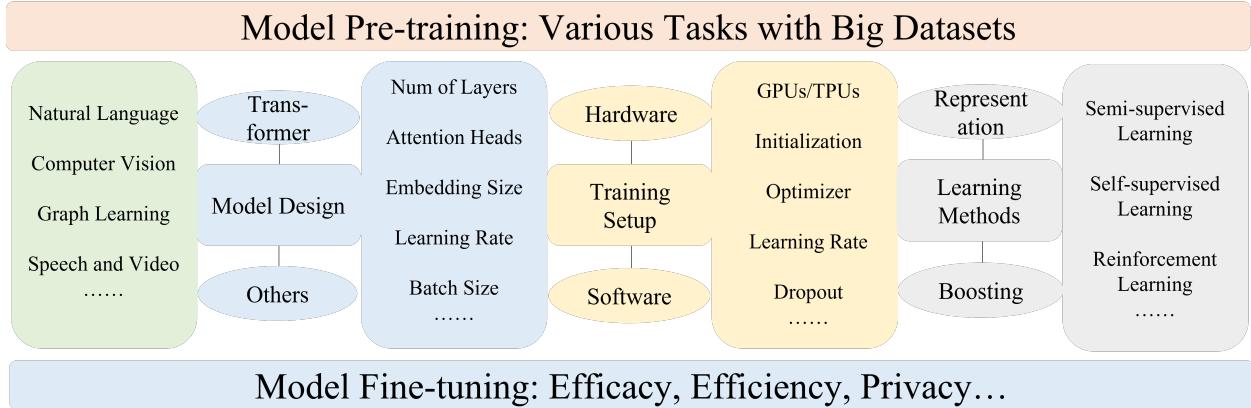


Figure 2: The general conceptual architecture of PFM: data, model, and system.

- Through the review of PFM in various modalities for different tasks, we discuss the main challenges and opportunities for future research of very large models in the big data era, which guides a new generation of collaborative and interactive intelligence based on PFM.

The rest of the survey is organized as follows. Section 2 introduces the basic components. Sections 3, 4 and 5 summarize the existing PFM in NLP, CV and GL, respectively. Sections 6, 7 introduce other advanced research for PFM, including advanced and unified PFM, model efficiency and compression, as well as security and privacy, respectively. Furthermore, we summarize the main challenges for PFM in Section 8 before concluding the survey in Section 9.

2 Basic Components

The general conceptual architecture of PFM is shown in Fig. 2. The PFM are huge neural network models, which are all about neural information processing. The specific designs of PFM vary according to the data modality and task requirements in different areas. Transformer is a mainstream model architecture design for PFM in many areas such as NLP and CV. Training large models need to have various datasets for model pretraining. After training the PFM, the model should be fine-tuned to satisfy downstream requirements such as efficacy, efficiency, and privacy. In this section, we introduce the basic model architectures, concepts, and settings of PFM in NLP, CV, and GL domains. For the introduction of a more detailed component, please refer to **Appendix A**.

2.1 Transformer for PFM

The Transformer [38] is an innovative architecture that facilitates the transfer of weighted representation knowledge between various neural units. It relies solely on attention mechanisms and doesn't use recurrent or convolutional architectures. The attention mechanism is a crucial component of the Transformer as it assigns weights to all the encoded input representations and learns the most important part of the input data. The output of the attention is obtained by taking the weighted sum of the values, and the weights are calculated using the compatibility function of the query with the corresponding key [38]. Numerous attention mechanisms [39] have been developed in large models. For instance, in natural language processing, self-attention is created to connect various positions in a single sequence for generating a representation

of the same sequence. Transformer leverages a mask matrix to provide an attention mechanism based on self-attention, in which the mask matrix specifies which words can “see” each other.

Transformer is an important structure for PFM in NLP, CV, and GL areas. For NLP, the Transformer can help solve the long-range dependency issues when processing sequential input data. For example, the GPT-3 [20] is a generative model based on the transformer. For CV, the Vision Transformer (ViT) [40] is proposed to represent an image to a series of image patches, which is similar to a series of word embeddings. For GL, the Graph Transformer Networks (GTN) [41] are employed to learn new graph structures and powerful node representations without domain knowledge. Transformers become scalable enough to drive ground-breaking capabilities for PFM thanks to the transformer structures to achieve higher parallelization. The ViT-22B model [42], for instance, has about 22B parameters, and the largest language models can have upwards of 100B parameters (e.g., GPT-3 has 175B and PaLM [43] has 540B parameters).

2.2 Learning Mechanisms for PFM

Deep learning models in CV have been shown a large margin to outperform traditional learning models in most tasks, including the common classification, recognition, detection, and segmentation tasks and the specific matching, tracking, and sequence prediction. These learning methods are not only available in CV, but also in NLP and GL.

Supervised Learning Suppose we are given a training dataset \mathbf{X} containing $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ to represent the original data in training dataset, where \mathbf{x}_i denotes the i -th training sample, and y_i denotes the corresponding label. The complete network is to learn a function $f(\mathbf{x}; \boldsymbol{\theta})$ by minimizing the objective function as follows.

$$\arg \min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \lambda \Omega(\boldsymbol{\theta}), \quad (1)$$

where \mathcal{L} and Ω represent the predefined loss function and a regularization term, respectively. The function f has a nested form like

$$\begin{aligned} \mathbf{h}_1(\mathbf{x}_i) &= g(\mathbf{x}_i^\top \boldsymbol{\omega}_1 + b_1), \\ \mathbf{h}_{l+1}(\mathbf{x}_i) &= g(\mathbf{h}_l(\mathbf{x}_i)^\top \boldsymbol{\omega}_l + b_l), l = 1, 2, \dots, N \end{aligned} \quad (2)$$

where l is the index of layer in deep learning model and N is the number of layers, which means that $\boldsymbol{\theta} = \{\boldsymbol{\omega}_l, b_l, l = 1, 2, \dots, N\}$.

Semi-Supervised Learning Assume we are given another unlabelled dataset $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^m$ in addition to the previous dataset with human labels. If we want to utilize both datasets to learn an ideal network, the learning process can be formulated as

$$\arg \min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \frac{1}{m} \sum_{i=1}^m \mathcal{L}'(f'(\mathbf{z}_i; \boldsymbol{\theta}'), R(\mathbf{z}_i, \mathbf{X})) + \lambda \Omega(\boldsymbol{\theta}), \quad (3)$$

where R is a relation function defining the targets for unlabelled data, and then these pseudo-labels are integrated into the end-to-end training process. f' is an encoder to learn a new representation for the original data in the dataset \mathbf{Z} . Specifically, if there is no label to any data in the training process, we can learn from the properties inside the data itself via the internal distance or the designed pretext tasks, which are known as unsupervised learning and self-supervised learning (SSL), respectively. The latter is our main focus discussed in detail in Section 4.3.

Weakly-Supervised Learning The weakly-supervised method is the balance between fully-supervised learning and SSL according to the dependence on human labels. The SSL designs special pretext tasks to serve as the supervised learning, but the fully supervised learning utilizes existing labels attached to the data. However, both of them can learn good visual features and perform well on specific downstream tasks. Suppose there are inaccurate K labels for the dataset, and any label can be attached to a data sample. Thus, we denote the true label of image \mathbf{x}_i as $\mathbf{y}_i \in \{0, 1\}^K$, $i = 1, 2, \dots, n$, and any entry of \mathbf{y}_i could be 0 or 1. Here we need to minimize the total nK loss terms , which are formulated as follows.

$$\arg \min_{\boldsymbol{\theta}} \frac{1}{nK} \sum_{i=1}^n \sum_{k=1}^K \mathcal{L}(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i^k) + \lambda \Omega(\boldsymbol{\theta}), \quad (4)$$

where $[y_i^1, y_i^2, \dots, y_i^K] = \mathbf{y}_i$, and \mathcal{L} could be a loss function suitable for binomial classification problem. For any entry in \mathbf{y}_i , computing the loss function of the one-versus-all binomial classification is needed.

Self-Supervised Learning SSL utilizes the information in the data itself to learn essential feature representations for different tasks. By applying the self-defined pseudo labels, it can avoid the cost of manually labeling large datasets for PFMs. In NLP, the language models can be trained by predicting masked characters, words, or sentences. Variational autoencoder (VAE) and generative adversarial network (GAN) are two types of generative SSL methods, which are to reconstruct the data itself. Besides, contrastive learning, as a type of discriminative SSL method, is widely applied in CV, NLP, and GL. The main idea of contrastive learning is to learn the prior knowledge distribution of the data itself with the aid of various methods such as data augmentation. In this way, contrastive learning can learn a model that makes similar instances closer in the projected space, and dissimilar instances farther apart in the projected space. Here we show a simple version of contrastive loss:

$$\mathcal{L}_c(\mathbf{x}_i, \mathbf{x}_j, \theta) = m \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2^2 + (1 - m) \max(0, \epsilon - \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2)^2 \quad (5)$$

where m is 1 if two samples have the same label, otherwise 0, and ϵ is the upper bound distance.

Reinforcement Learning RL is another type of learning paradigm that models the learning process as a sequential interaction between an agent and an environment, where a RL agent seeks to learn an optimal policy for sequential decision-making problems. Specifically, at each time interaction step t , the agent receives a state s_t in a state space \mathcal{S} , and selects an action a_t from an action space \mathcal{A} , following a policy $\pi_\theta(a_t | s_t) : \mathcal{A} \rightarrow \mathcal{S}$ parameterized by θ . Then the agent receives a scalar immediate reward $r_t = r(s_t, a_t)$ and the next state s_{t+1} according to the environment dynamics, where $r(s, a)$ is the reward function. For each episode, this process continues until the agent reaches a terminal state. After an episode is finished, the RL agent will restart to begin a new episode. The return for each state is discounted, accumulated reward with the discount factor $\gamma \in (0, 1]$, $R_t = R(s_t, a_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. The agent aims to maximize the expectation of such long-term return from each state,

$$\max_{\theta} \mathbb{E}_{s_t} [R_t | s_t, a_t = \pi_\theta(s_t)]. \quad (6)$$

2.3 Pretraining Tasks for PFMs

Pretraining is an initialization framework, which generally needs to be used in conjunction with fine-tuning downstream tasks. In the scheme of pretraining and finetuning, the parameters of the model are trained on pre-set tasks to capture specific attributes, structure, and community information. The pretrained features can assist downstream tasks, provide sufficient information, and speed up the convergence of the model.

2.3.1 Pretraining Tasks for NLP

The pretraining tasks can be divided into five categories according to the learning methods: Mask Language Modeling (MLM), Denoising AutoEncoder (DAE), Replaced Token Detection (RTD), Next Sentence Prediction (NSP), Sentence Order Prediction (SOP). RTD, NSP, and SOP are contrastive learning methods, which assume that the observed samples are more semantically similar than the random samples.

Mask Language Modeling (MLM). MLM erases some words randomly in the input sequence and then predicts these erased words during pretraining. Typical examples include BERT [13] and SpanBERT [44].

Denoising AutoEncoder (DAE). DAE is used to add noise to the original corpus and reconstruct the original input using the corpus containing noise. BART [45] is a representative example.

Replaced Token Detection (RTD). RTD is a discriminant task that determines whether the LM has replaced the current token. This task is introduced in ELECTRA [46]. By training the model to distinguish whether a token has been replaced or not, the model can acquire language knowledge.

Next Sentence Prediction (NSP). In order to make the model understand the correlation between the two sentences and capture sentence-level representations, a NSP task is introduced. The PFM inputs two sentences from different documents and checks whether the order of the sentences is correct. A typical example is BERT.

Sentence Order Prediction (SOP). Different from NSP, SOP uses two contiguous fragments from a document as positive samples and the exchange order of the two fragments as negative samples. The PFMs can better model the correlation between sentences, such as ALBERT [47].

2.3.2 Pretraining Tasks for CV

There are many pretraining tasks created for CV to learn the feature space, which is based on SSL. It utilizes pretext tasks that contain human-designed labels, like jigsaw puzzles or the comparison of various patches from images. This enables the generalization of learned representations to a range of downstream tasks.

Specific Pretext Task. A pretext task also referred to as a predefined task, is created for the encoder networks to perform during the pretraining phase. The network is trained by predicting the answer to a special pretext task. Based on particular features of the data, pseudo labels are generated for the fictitious task. Then, using guided learning techniques, the encoder networks are trained to solve the pretext task. For example, inpainting aims to pretrain models by predicting the missed center part.

Frame Order Learning Task. Learning frame order from videos involves frame processing through time steps, which can serve as the pretraining task for CV. This issue usually relates to completing pretextual exercises that can aid in the acquisition of visual temporal representations.

Data Generation Task. The representational capabilities within the generative adversarial networks (GANs) can also be used in the pretraining tasks. Projecting data back into the latent space, as demonstrated by BiGANs [48], is helpful for auxiliary supervised discrimination tasks by acting as feature representations.

Data Reconstruction Task. Since the images can be divided into patches inspired by the natural language, some pretraining tasks for NLP can also be used in CV, e.g., the autoencoder-based masked prediction. The original image is first divided into a few patches and discrete visual tokens are used to encode each patch. The visual tokens from the masked patches are outputted in the second stage to match the corresponding visual tokens from the fixed tokenizer.

Miscellaneous. To train the PFMs in CV, additional pretraining tasks are suggested. For instance,

based on contrastive learning, encoder networks are used for pretraining on various data augmentation. The parameters are trained by maximizing the distance between negative pairs (e.g., pairs with different labels) and minimizing the distance between positive pairs (e.g., pairs with the same labels). To pretrain the parameters of the backbone network, the DeepClustering [49] method divides the representations into various clusters and labels these clusters as supervised signals.

2.3.3 Pretraining Tasks for GL

The pre-set tasks in GL are similar to other pretext tasks. However, they can be supervised or unsupervised depending on the design. According to the pretraining purpose and potential motivation in GL, such tasks can be divided into the following categories:

Graph Information Completion. This task refers to firstly masking part of the information in the input graph, and then recovering the masked information based on the analysis of the remaining information distribution. Similar tasks also exist in CV and NLP, and their goals are to fill in hidden pixels or words, respectively.

Graph Property Prediction. Different from directly modeling the information of the input graph, this task aims to provide a variety of self-supervised signals by mining the potential properties of the input graph. Specifically, on the one hand, it considers node attributes, local substructure, and connectivity information to provide predictive regression tasks; on the other hand, it assigns pseudo-labels to nodes through information such as clusters, structure density, and attribute similarity to provide classification tasks.

Graph Consistency Analysis. The goal of this task is to maximize the consistency between samples with similar semantic information in the graph embedding and minimize the agreement between samples with unrelated semantic information. In the actual scenario, it can be divided into consistency analysis of context/self/cross-scale according to different model training strategies.

Miscellaneous. Compared with using only one pretext task, some methods have designed some integration mechanisms to incorporate the advantages of multiple pretext tasks into a unified framework. Besides, some graph data in specific fields have unique self-supervised signals with practical significance that can be used for pretraining under targeted design.

In summary, the transformer is an important component of the large model architecture, which helps learn the important features and mine intrinsic structure in data. Different learning mechanisms can be used for training PFM according to the datasets and specific tasks. Especially, SSL is a promising mechanism to learn knowledge embeddings from the data considering the large scale of unlabeled data in various areas. RL provides a new way to fine-tune the PFM for downstream tasks by optimizing a policy (model) against the reward model. How to design effective and efficient tasks for PFM to master the knowledge behind the data is an important research topic.

3 PFM for Natural Language Processing

NLP is a research field that integrates linguistics and computer science. Its main research tasks include part-of-speech tagging, named entity recognition, semantic role labeling, machine translation, question answering, sentiment analysis, text summarization, text classification, relationship extraction, event extraction, etc. The idea of PFM first gained popularity in NLP. Then CV and GL adopt the promising pretraining technology. The PFM trains on a large benchmark dataset and is fine-tuned on the primary task dataset to obtain a model which can solve new similar tasks. It models syntactic and semantic representations of words si-

multaneously and changes the representation of polysemous words dynamically according to different input contexts. PFM learns a rich knowledge of grammar and semantic reasoning with better results. Numerous PFMs have been proposed in the past few years, as shown in **Table 1**.

In this section, we first introduce word representation learning models including the autoregressive language model (LM), contextual LM, and permuted LM. Then, we present the neural network architectures for the PFM designing method and the masking designing method. Besides, we summarize boosting methods for enhancing model performance, multi-task learning, and different downstream tasks. Finally, we introduce the instruction-aligning methods, e.g. RLHF and Chain-of-Thoughts, which are applied in PFMs, such as ChatGPT, to provide outputs that more closely match human preferences and are less harmful.

3.1 Word Representations Methods

Many large-scale pretrained models have achieved better performance than humans in question answering, machine reading comprehension, and natural language reasoning, which indicates that the current construction approach of PFMs is practical. The existing pretraining LMs are mainly divided into three branches according to the word representations approach: (1) *autoregressive LM*, (2) *contextual LM*, and (3) *permuted LM*. The word prediction direction and contextual information are the most important factors among these three branches.

Autoregressive Language Model The autoregressive LM predicts the next possible word based on the preceding word or the last possible word based on the succeeding word. It is selected as a feature extractor and text representations are extracted from the former words. Thus, it has better performance in NLG tasks such as text summarization and machine translation. For a sequence, $T = [w_1, w_2, \dots, w_N]$, the probability of a given word calculated as follows:

$$p(w_1, w_2, \dots, w_N) = \prod_{i=1}^N p(w_i | w_1, w_2, \dots, w_{i-1}), \quad (7)$$

where $i > 1$ and N is the length of the input sequence.

The GPT [50] adopts a two-stage method of self-supervised pretraining and supervised fine-tuning and uses stacked Transformer [38] as its decoder. As a follow-up, the OpenAI team continues to expand GPT, proposes the GPT-2 [51] and increases the number of stacked Transformer layers to 48 layers. The total number of parameters reached 1.5 billion. GPT-2 also introduces multi-task learning [52]. The GPT-2 has a considerable model capacity and can be adjusted for different task models rather than fine-tuning them. However, GPT-2 also uses an autoregressive LM. Therefore, it improves the performance of the model without increasing the cost dramatically. Due to the lack of contextual modeling ability with a one-way Transformer, the main performance improvement of GPT-2 comes from the combined effect of multi-task pretraining, super-large datasets, and super-large models. Task-based datasets for fine-tuning are still needed for specific downstream tasks. Increasing the training scale of the LM can lead to a significant enhancement in task-independent performance. Hence, GPT-3 [20] was developed, which features a model size of 175 billion parameters and is trained with 45 Terabytes of data. This enables it to exhibit good performance without the need for fine-tuning for specific downstream tasks.

Contextual Language Model The autoregressive LM only uses the information above or below and cannot use the information above and below at the same time. ELMO [53] only uses bi-directional Long

Short-Term Memory (LSTM), which is a concatenation of two unidirectional LSTMs in backward and forward. The contextual LM predictions are based on contextual words. It uses a Transformer encoder, and the upper and lower layers of the model are all directly connected to each other due to the self-attention mechanism. For a sequence of words T , the probability of a given word calculates as follows

$$p(w_1, w_2, \dots, w_N) = \prod_{i=1}^N p(w_i | w_1, w_2, \dots, w_N). \quad (8)$$

BERT [13] uses a stacked multi-layer bi-directional Transformer as the basic structure, and Word-Piece [54] as a word segmentation method. The model input consists of three parts: word embedding, segment embedding, and position embedding. It uses a bi-directional Transformer as a feature extractor, which offsets the defect of ELMO and GPT. However, the shortcomings of BERT are also not to be ignored. The bidirectional Transformer structure does not eliminate the constraints of the self-encoding model. Its vast number of model parameters are very unfriendly to devices with low computing resources and are challenging to deploy and apply. Furthermore, the hidden language modeling in pretraining will lead to inconsistencies with the input of the model in the fine-tuning stage. Most PFM need more training tasks and a larger corpus. Aiming at the problem of insufficient training, Liu et al. [55] propose the RoBERTa. It uses a larger batch size and unlabeled data. Furthermore, it trains the model for a longer time, removes the NSP task, and adds long sequence training. In processing text input, different from BERT, Byte Pair Encoding (BPE) [56] is adopted for word segmentation. BPE uses a different mask mode for each input sequence, even if the input sequence is the same.

Permuted Language Model The modeling method with a contextual LM can be regarded as the autoencoding model. However, due to the inconsistency in the training stage and fine-tuning stage, the performance of the autoencoding model is poor in the Natural Language Generation (NLG) task. Permuted LM aims to combine the advantages of the autoregressive LM and the autoencoder LM. It improves the defects of the two models to a great extent and can be used as a basic idea for the construction of future pretraining target tasks. For a given input sequence $T = [w_1, w_2, \dots, w_N]$, the formal representation of the target function of the permuted LM is as follows

$$\max_{\theta} \mathbb{E}_{z \sim Z_N} \left[\sum_{t=1}^N \log p_{\theta}(x_{z_{T=t}} | x_{z_{T < t}}) \right], \quad (9)$$

where θ is the shared parameter in all permutations, Z_N represents the set of all possible permutations of the input sequence T , and $z_{T=t}$ and $z_{T < t}$ represents the t -th element and the $[1, 2, \dots, t-1]$ elements of a permutation $z \in Z_N$.

MLM represented by BERT can implement bi-directional coding well. However, MLM uses the mask marking during pretraining but not during fine-tuning, which resulted in inconsistent data during pretraining and fine-tuning. To achieve bi-directional coding and avoid the problems of MLM, the permuted LM is proposed. permuted LM is based on the autoregressive LM, which avoids the influence of inconsistent data. However, unlike traditional autoregressive models, permuted LM no longer models sequences in order. It gives all possible permutations of sequences to maximize the expected logarithmic likelihood of the sequence. In this way, any position can take advantage of contextual information from all positions, making permuted LM implement bidirectional encoding. The most common permuted LM models are XLNET [14] and MPNet [57]. XLNET is a PFM based on a permuted language modeling approach, which incorporates two crucial techniques from Transformer-XL: relative positional encoding and the segment recurrence mechanism. In contrast, MPNet combines Masked Language Modeling (MLM) and permuted

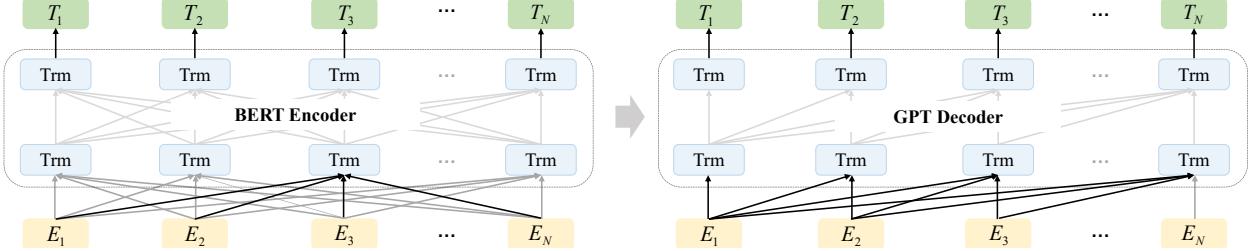


Figure 3: The architectures of BART [45]: generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder). An autoregressive decoder is used to determine the likelihood of the original document after the corrupted document (on the left) has been encoded using a bidirectional model.

language modeling to predict token dependencies, using auxiliary position information as input to enable the model to view a complete sentence and reduce position differences. These two models represent significant advancements in the field of PFM.

3.2 Model Architecture Designing Methods

ELMO adopts a multi-layer RNN structure. Each layer is a bi-directional LSTM structure composed of a forward and backward LM. The maximum likelihood of these two directions is taken as the objective function. Compared with the word vector method, ELMO introduces contextual information and improves the polysemy problem, but ELMO’s overall ability to extract linguistic features is weak.

The application research of PFM has two main directions. One is PFM with fine-tuning (e.g., BERT), and the other one is PFM with zero/few-shot prompts (e.g., GPT). **BERT** uses a bi-directional encoder in Transformer to predict which words are masked and determine whether two sentences are contextual. However, the document is encoded bidirectionally and missing tokens are predicted independently, which reduces the generation ability [45]. **GPT** uses an autoregressive decoder as a feature extractor to predict the next word based on the first few words and solve downstream tasks using fine-tuning, so it is more suitable for text-generation tasks. However, GPT only uses the former words for prediction, which cannot learn bidirectional interaction information.

Different from these models, **BART** [45] is a noise-reducing autoencoder built by seq2seq model adopting the encoder-decoder structure, as shown in Fig. 3 from [45]. Pretraining mainly includes using noise to destroy text and using the seq2seq model to rebuild the original text. The encoding layer adopts a bi-directional Transformer. It adopts five modes of adding noise: (1) single word mask; (2) word deletion; (3) span mask; (4) sentence rearrangement; (5) document rearrangement. In the encoder part, the sequence has been masked before inputting it into the encoder. Then, the decoder restores the original sequence according to the encoding representation output by the encoder and the sequence that has not been masked. The addition of a series of noise patterns makes the performance of BART in sequence generation and natural language reasoning tasks significantly improved.

3.3 Masking Designing Methods

The attention mechanism first aggregates essential words into sentence vectors, and vital sentence vectors into text vectors, which allows the model to pay different attention to different inputs [58]. For BERT, as a bidirectional encoding LM, any two words in an input sentence can see each other. However, it hinders the ability of BERT model to learn NLG tasks.

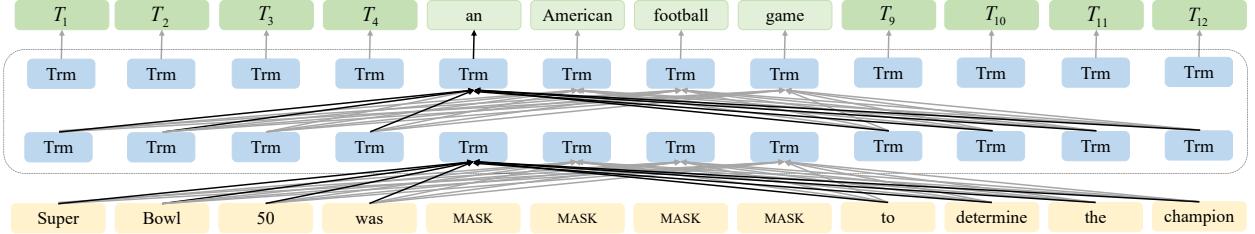


Figure 4: The architecture of SpanBERT [44].

Joshi et al. [44] propose SpanBERT based on RoBERTa, which adopts the idea of dynamic masking and single segment pretraining, as shown in Fig. 4 from [44]. The span mask and the Span Boundary Objective (SBO) are also proposed to mask words of a certain length. The target task of the span-boundary is to restore all the masked span (tokens) by the observed tokens at both ends. The training stage uses the dynamic mask strategy proposed in the RoBERTa, instead of the mask during the data preprocessing. Unlike BERT, SpanBERT randomly covers up a continuous text and adds the SBO training target. It predicts the span using the token closest to the span boundary and eliminates the NSP pretraining task.

The BERT and GPT can only separate the training encoder and decoder without joint training in the NLG task. Song et al. [59] propose the masked seq2seq pretraining model MASS. In the training stage, the input sequence of the encoder is randomly masked as a continuous segment of length k . The masked segment will be recovered through the MASS decoder. UniLM [60] completes the learning of the NLG model by designing a different mask for two sentences in the input data. For the first sentence, UniLM uses the same structure as the Transformer encoder making each word notice its preceding and following words. For the second sentence, each word can only notice all the words in the first sentence and the preceding words in the current sentence. Thus, the first and second sentences of the model input form the classic seq2seq pattern.

3.4 Boosting Methods

Boosting on Model Performance Most of the popular pretraining models need lots of pretraining data, which imposes huge requirements on the hardware, making it challenging to retrain, and only fine-tuning can be done to the model. To solve these problems, some models appear. For example, ERNIE Tiny released by Baidu is a miniaturized ERNIE [61], that reduces the number of layers and increases the prediction speed by 4.3 times with a slight decrease in accuracy. Lan et al. propose the ALBERT [47] to reduce memory consumption and training speed. However, it is undeniable that no matter what kind of compression is done for these large-scale models, the performance of the models in these tasks will deteriorate sharply. It requires paying attention to the efficient representation of high-level semantic and grammatical information and lossless compression in future works. By using word-embedded parameter factorization and hidden parameter sharing between layers, ALBERT significantly reduces the number of parameters of the model without performance loss. It proposes the training task of SOP, which predicts the order of the two sentences to improve the performance.

Boosting for Multi-task Learning ERNIE(Baidu) [61] is mainly composed of two parts, the Transformer encoder and task embedding. In the Transformer encoder, the self-attention mechanism is used to capture the context information of each token and generate context representation embedding. Task embedding is a technique that applies different characteristics to a task. ERNIE 2.0 [62] introduces multi-task learning to

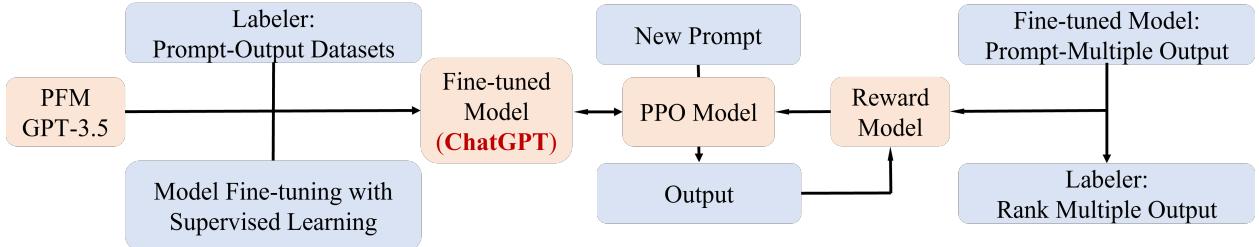


Figure 5: Boosting GPT-3.5 to ChatGPT using Reinforcement Learning from Human Feedback.

realize the pretraining of lexical, grammar, and semantics. ERNIE 2.0 uses seven different pretraining tasks, covering three aspects: word level, sentence level, and semantic level. It uses continual learning, making the knowledge in the previous training task retained and enabling the model to acquire long-distance memory. It uses a Transformer encoder and introduces task embedding, enabling the model to distinguish different tasks in the continual learning process. UniLM [60] uses three pretraining tasks: unidirectional LM, bidirectional LM, and encoder-decoder LM. It can simultaneously complete three kinds of target tasks in the pretraining stage through the self-attention layer mask mechanism. In the training stage, UniLM adopts the small-segment mask strategy proposed by SpanBERT, and the loss function is composed of the loss functions of the above three pretraining tasks. To maintain the contribution consistency on all loss functions, the three pretraining tasks are trained simultaneously. Modeling and parameter sharing of multiple tasks make LMs achieve good generalization ability in Natural Language Understanding (NLU) and NLG tasks.

Boosting for Different Downstream Tasks The pretraining models tend to be large-sized, so how to match different downstream tasks is equally important. Some pretraining models that are trained on specialized corpora have appeared [63, 64, 65]. Cui et al. [63] propose the BERT-whole word masking model (BERT-WWM). They directly use BERT in Chinese to be masked randomly according to the original MLM training, resulting in the loss of semantic information. Since there is no explicit language boundary in Chinese, it is easy to lose significant meaning. ZEN [64] is a text encoder based on BERT, which adopts N-gram to enhance performance and effectively integrates considerable granular text information with fast convergence speed and good performance. Tsai et al. [65] propose an oriented multilingual sequence labeling model for sequence labeling tasks. The knowledge distillation method is adopted to achieve better performance in the two tasks: part of speech labeling and morphological attribute prediction for multiple low-resource languages. The inference time is shortened by 27 times.

Examples: ChatGPT and Bard As shown in Fig. 5, ChatGPT is fine-tuned based on the PFM GPT-3.5 using RLHF. ChatGPT uses a different data collection setup compared to InstructGPT. First, a large dataset with prompts and the desired output behaviors is collected. The dataset is used to fine-tune GPT-3.5 with supervised learning. Second, given the fine-tuned model and a prompt, the model will generate several model outputs. A labeler gives the desired score and ranks the output to compose a comparison dataset, which is used to train the reward model. Finally, the fine-tuned model (ChatGPT) is optimized against the reward model using the Proximal Policy Optimization (PPO)[66] RL algorithm.

Another experimental conversational PFM, the Bard², is developed by Google. Bard is based on the LM for Dialogue Applications (LaMDA). LaMDA [67] is built upon the Transformer, which is pretrained on 1.56T words of dialog data and web text. Safety and factual grounding are two main challenges for

²<https://blog.google/technology/ai/bard-google-ai-search-updates/>

conversational AI, LaMDA applies the approaches that fine-tuning with high-quality annotated data and external knowledge sources to improve model performance.

3.5 Instruction-Aligning Methods

Instruction-aligning methods aim to let the LM follow human intents and generate meaningful outputs. The general approach is fine-tuning the pretrained LM with high-quality corpus in a supervised manner. To further improve the usefulness and harmlessness of LMs, some works introduce RL into the fine-tuning procedure so that LMs could revise their responses according to human or AI feedback. Both supervised and RL approaches can leverage chain-of-thought [24] style reasoning to improve the human-judged performance and transparency of AI decision-making.

Supervised Fine-Tuning (SFT) SFT is a well-established technique to unlock knowledge and apply it to specific real-world, even unseen tasks. The template for SFT is composed of input-output pairs and an instruction [113]. For example, given the instruction “Translate this sentence to Spanish:” and an input “The new office building was built in less than three months.”, we want the LM to generate the target “El nuevo edificio de oficinas se construyó en tres meses.”. The template is commonly humanmade including unnatural instructions [114] and natural instructions [115, 116], or bootstrap based on a seed corpus [117]. Ethical and social risks of harm from LMs are significant concerns in SFT [118]. LaMDA, the largest LM to date, thus relies on crowdworker annotated data for providing a safety assessment of any generated LaMDA response in three conversation categories: natural, sensitive, and adversarial. The list of rules serves further safety fine-tuning and evaluation purposes.

Reinforcement Learning from Feedback RL has been applied to enhance various models in NLP tasks such as machine translation [119], summarization [18], dialogue generation [120], image captioning [121], question generation [122], text-games [123], and more [124, 125, 126]. RL is a helpful method for optimizing non-differentiable objectives in language generation tasks by treating them as sequential decision-making problems. However, there is a risk of overfitting to metrics that use neural networks, leading to nonsensical samples that score well on the metrics [127]. RL is also used to align LMs with human preferences [128, 129, 130].

InstructGPT proposes to fine-tune large models with PPO against a trained reward model to align LMs with human preference [19], which is the same method applied by ChatGPT named RLHF. Specifically, the reward model is trained with comparison data of human labelers’ manual rankings of outputs. For each of them, the reward model or machine labeler calculates a reward, which is used to update the LM using PPO. More details are illustrated in Fig. 5.

One of the recent breakthroughs in PFM technology is GPT-4 [25], which follows a pretraining approach to predict the subsequent token in a document and then undergoes RLHF fine-tuning. As the task complexity increases, GPT-4 outperforms GPT-3.5 in terms of reliability, creativity, and capability to handle more nuanced instructions.

Sparrow [130], developed by DeepMind, also utilizes RLHF that reduces the risk of unsafe and inappropriate answers. Despite some promising results using RLHF by incorporating fluency, progress in this field is impeded by a lack of publicly available benchmarks and implementation resources, resulting in a perception that RL is a difficult approach for NLP. Therefore, an open-source library named RL4LMs [127] is introduced recently, which consists of building blocks for fine-tuning and evaluating RL algorithms on LM-based generation.

Table 1: Summary of PFM_s in NLP. The pretraining task includes language model (LM), masked LM (MLM), permuted LM (PLM), denoising autoencoder (DAE), knowledge graphs (KG), and knowledge embedding (KE).

Year	Conference	Model	Architecture	Embedding	Training method	Code
2013	NeurIPS	Skip-Gram [68]	Word2Vec	Probabilistic	-	https://github.com/.../models
2014	EMNLP	GloVe [69]	Word2Vec	Probabilistic	-	-
2015	NeurIPS	LM-LSTM [70]	LSTM	Probabilistic	LM	https://github.com/.../GloVe
2016	IJCAI	Shared LSTM [71]	LSTM	Probabilistic	LM	https://github.com/.../adversarial_text
2017	TACL	FastText [72]	Word2Vec	Probabilistic	-	https://github.com/.../fastText
2017	NeurIPS	CoVe [73]	LSTM+Seq2Seq	Probabilistic	-	https://github.com/.../cove
2018	NAACL-HLT	ELMO [53]	LSTM	Contextual	LM	https://allennlp.org/elmo
2018	NAACL-HLT	BERT [13]	Transformer Encoder	Contextual	MLM	https://github.com/.../bert
2018		OpenAI GPT [50]	Transformer Decoder	Autoregressive	LM	https://github.com/.../transformer-lm
2019	ACL	ERNIE(THU)	Transformer Encoder	Contextual	MLM	https://github.com/.../ERNIE
2019	ACL	Transformer-XL [74]	Transformer-XL	Contextual	-	https://github.com/.../transformer-xl
2019	ICLR	InfoWord [75]	Transformer Encoder	Contextual	MLM	-
2019	ICLR	StructBERT [76]	Transformer Encoder	Contextual	MLM	-
2019	ICLR	ALBERT [47]	Transformer Encoder	Contextual	MLM	https://github.com/.../ALBERT
2019	ICLR	WKLM [77]	Transformer Encoder	Contextual	MLM	-
2019	ICML	MASS [59]	Transformer	Contextual	MLM(Seq2Seq)	https://github.com/.../MASS
2019	EMNLP-IJCNLP	KnowBERT [78]	Transformer Encoder	Contextual	MLM	https://github.com/.../kb
2019	EMNLP-IJCNLP	Unicoder [79]	Transformer Encoder	Contextual	MLM+TLM	-
2019	EMNLP-IJCNLP	MultiFit [80]	QRNN	Probabilistic	LM	https://github.com/.../multifit
2019	EMNLP-IJCNLP	SciBERT [81]	Transformer Encoder	Contextual	MLM	https://github.com/.../scibert
2019	EMNLP-IJCNLP	BERT-PKD [82]	Transformer Encoder	Contextual	MLM	https://github.com/.../Compression
2019	NeurIPS	Xlnet [14]	Transformer-XL Encoder	Permutation	PLM	https://github.com/.../xlnet
2019	NeurIPS	UNILM [60]	LSTM + Transformer	Contextual	LM + MLM	https://github.com/.../unilm
2019	NeurIPS	XLM [83]	Transformer Encoder	Contextual	MLM+CLM+TLM	https://github.com/.../XLM
2019	OpenAI Blog	GPT-2 [51]	Transformer Decoder	Autoregressive	LM	https://github.com/.../gpt-2
2019	arXiv	RoBERTa [55]	Transformer Encoder	Contextual	MLM	https://github.com/.../faiseq
2019	arXiv	ERNIE(Baidu) [61]	Transformer Encoder	Contextual	MLM+DLM	https://github.com/.../ERNIE
2019	EMC2@NeurIPS	Q8BERT [84]	Transformer Encoder	Contextual	MLM	https://github.com/.../quantized_bert.py
2019	arXiv	DistilBERT [85]	Transformer Encoder	Contextual	MLM	https://github.com/.../distillation
2020	ACL	fastBERT [86]	Transformer Encoder	Contextual	MLM	https://github.com/.../FastBERT
2020	ACL	SpanBERT [44]	Transformer Encoder	Contextual	MLM	https://github.com/.../SpanBERT
2020	ACL	BART [45]	Transformer	En: Contextual De: Autoregressive	DAE	https://github.com/.../transformers
2020	ACL	CamemBERT [87]	Transformer Encoder	Contextual	MLM(WWM)	https://camembert-model.fr
2020	ACL	XLM-R [88]	Transformer Encoder	Contextual	MLM	https://github.com/.../XLM
2020	ICLR	Reformer [89]	Reformer	Permutation	-	https://github.com/.../reformer
2020	ICLR	ELECTRA [46]	Transformer Encoder	Contextual	MLM	https://github.com/.../electra
2020	AAAI	Q-BERT [90]	Transformer Encoder	Contextual	MLM	-
2020	AAAI	XNLG [91]	Transformer	Contextual	MLM+DAE	https://github.com/.../xnlg
2020	AAAI	K-BERT [92]	Transformer Encoder	Contextual	MLM	https://github.com/.../K-BERT
2020	AAAI	ERNIE 2.0 [62]	Transformer Encoder	Contextual	MLM	https://github.com/.../ERNIE
2020	NeurIPS	GPT-3 [20]	Transformer Decoder	Autoregressive	LM	https://github.com/.../gpt-3
2020	NeurIPS	MPNet [57]	Transformer Encoder	Permutation	MLM+PLM	https://github.com/.../MPNet
2020	NeurIPS	ConvBERT [93]	Mixed Attention	Contextual	-	https://github.com/.../ConvBert
2020	NeurIPS	MiniLM [94]	Transformer Encoder	Contextual	MLM	https://github.com/.../minilm
2020	TACL	mBART [95]	Transformer	Contextual	DAE	https://github.com/.../mbart
2020	COLING	CoLAKE [96]	Transformer Encoder	Contextual	MLM+KE	https://github.com/.../CoLAKE
2020	LREC	FlauBERT [97]	Transformer Encoder	Contextual	MLM	https://github.com/.../Flaubert
2020	EMNLP	GLM [98]	Transformer Encoder	Contextual	MLM+KG	https://github.com/.../GLM
2020	EMNLP (Findings)	TinyBERT [99]	Transformer	Contextual	MLM	https://github.com/.../TinyBERT
2020	EMNLP (Findings)	RobBERT [100]	Transformer Encoder	Contextual	MLM	https://github.com/.../RobBERT
2020	EMNLP (Findings)	ZEN [64]	Transformer Encoder	Contextual	MLM	https://github.com/.../ZEN
2020	EMNLP (Findings)	BERT-MK [101]	KG-Transformer Encoder	Contextual	MLM	-
2020	Rep4NLP@ACL	CompressingBERT [35]	Transformer Encoder	Contextual	MLM(Pruning)	https://github.com/.../bert-prune
2020	JMLR	T5 [102]	Transformer	Contextual	MLM(Seq2Seq)	https://github.com/.../transformer
2021	T-ASL	BERT-www-Chinese [63]	Transformer Encoder	Contextual	MLM	https://github.com/.../BERT-www
2021	EACL	PET [103]	Transformer Encoder	Contextual	MLM	https://github.com/.../pet
2021	TACL	KEPLER [104]	Transformer Encoder	Contextual	MLM+KE	https://github.com/.../KEPLER
2021	EMNLP	SimCSE [105]	Transformer Encoder	Contextual	MLM+KE	https://github.com/.../SimCSE
2021	ICML	GLaM [106]	Transformer	Autoregressive	LM	-
2021	arXiv	XLM-E [107]	Transformer	Contextual	MLM	-
2021	arXiv	T0 [108]	Transformer	Contextual	MLM	https://github.com/.../T0
2021	arXiv	Gopher [109]	Transformer	Autoregressive	LM	-
2022	arXiv	MT-NLG [110]	Transformer	Contextual	MLM	-
2022	arXiv	LaMDA [67]	Transformer Decoder	Autoregressive	LM	https://github.com/.../LaMDA
2022	arXiv	Chinchilla [111]	Transformer	Autoregressive	LM	-
2022	arXiv	PaLM [43]	Transformer	Autoregressive	LM	https://github.com/.../PaLM
2022	arXiv	OPT [112]	Transformer Decoder	Autoregressive	LM	https://github.com/.../MetaSeq

Besides human feedback, one of the latest dialogue agents – Claude favors Constitutional AI [131] where the reward model is learned via RL from AI Feedback (RLAIF). Both the critiques and the AI feedback are steered by a small set of principles drawn from a ‘constitution’, the specification of a short list of principles or instructions, which is the only thing provided by humans in Claude. The AI feedback focuses on controlling the outputs to be less harmful by explaining its objections to dangerous queries.

Chain-of-Thoughts Chain-of-thought (CoT) prompting is a technique for improving the reasoning ability of LLMs by prompting them to generate a series of intermediate steps that lead to the final answer of a multi-step problem. The CoT is a series of intermediate reasoning steps, which can significantly improve the ability of LLMs to perform complex reasoning [24, 132, 133]. Besides, fine-tuning with CoT shows slightly more harmless compared to without CoT [131]. CoT prompting is an emergent property of model scale, meaning it works better with larger and more powerful language models. It is also possible to fine-tune models on CoT reasoning datasets to enhance this capability further and stimulate better interpretability.

In a CoT prompting experiment, a prompt is provided to the model that outlines a multi-step problem. The prompt might pose a question such as “After selling 30 out of his 100 chickens and 10 out of his 20 pigs, how many animals does a farmer have left?” The model then generates a sequence of intermediate reasoning steps, for example, “The farmer has $100-30=70$ chickens remaining” and “The farmer has $20-10=10$ pigs remaining,” before generating the final answer, such as “The farmer has $70+10=80$ animals remaining.” CoT prompting has demonstrated its efficacy in improving the performance of LLMs on various reasoning tasks, such as arithmetic, symbolic reasoning, and common sense. It is a promising technique that can enhance the ability of language models to reason about complicated problems.

3.6 Summary

The neural probabilistic LM uses a neural network to estimate the parameters of the probabilistic LM, which reduces the size of the model parameters while enlarging the number of context windows. With the help of a neural network, the LM does not need to improve the smoothing algorithm to alleviate the performance bottleneck continuously. Since the training target is unsupervised, a corpus with a large amount of data is enough for training. The negative sampling technique in the training process provides a new idea for the follow-up study of the target task in the LM. Furthermore, the neural probabilistic LM promotes the further development of downstream task research because of its good representation capability and training efficiency. After the pretraining LM, especially the BERT model, is proposed, the research in language modeling has entered a new phase. The bidirectional LM, the hidden LM, and the sorted LM adopted by the bidirectional LM have successfully modeled the grammatical and semantic information in natural language at a deeper level. ChatGPT is another milestone work in PFM using RL. The presentation ability of PFM is qualitatively better than that of the neural probabilistic LM. It even exceeds that of humans in some tasks.

4 PFM for Computer Vision

With the popularity of PFM used in NLP, it motivates researchers to start exploring PFM in CV. The term “pretraining” has not been clearly defined within the realm of deep learning research in CV. This word is first used in convolution-based networks when we adjust the parameters on a more general dataset such as ImageNet, which can make other tasks train to start with a warm-up initialization and thus converge with faster speed. In contrast to early CNN-based transfer learning techniques that rely on pretrained datasets with supervised signals, our examination of PFM centers on SSL which utilizes human-designed labels,

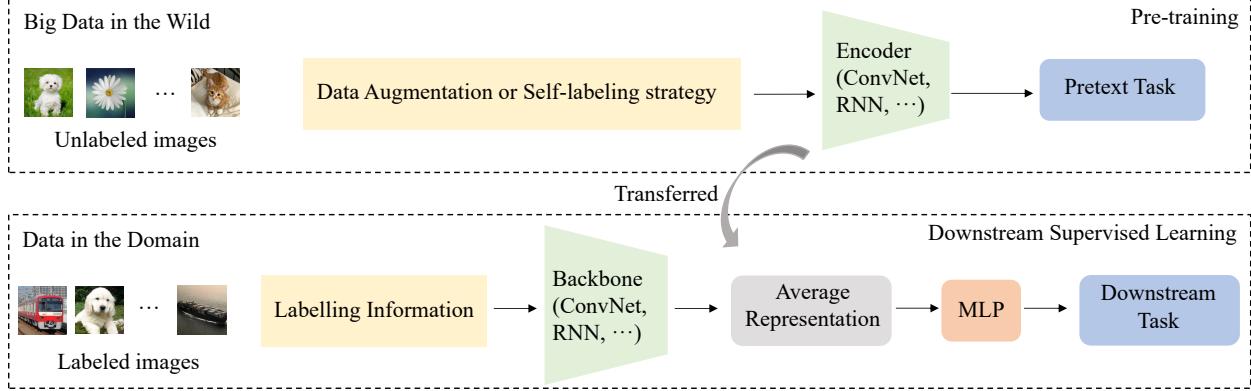


Figure 6: The general pipeline for SSL. The top part represents the pretraining, and the bottom stream obtains transferred parameters from above to learn downstream supervised tasks.

such as Jigsaw puzzles, or the comparison of different patches from images as pretext tasks. This allows for learned representations to be generalized to various downstream tasks, including classification, detection, recognition, segmentation, etc.

However, it is costly to rely on data annotations when the learning tasks become more complicated, making the labeling process more arduous and time-consuming than the actual learning. This is where SSL is urgently needed and how it can further fuel the progress of deep learning methods. To reduce the dependency on data labeling, unlabeled data are trained with self-supervision by matching, contrasting, or generating in SSL.

The general pipeline of SSL is shown in Fig. 6. During the pretraining stage, a pretext task is designed for the encoder networks to solve. The artificial labels for this pretext task are automatically generated based on specific attributes of the data, such as image patches from the same origin being labeled as “positive” and those from different origins as “negative”. Then, the encoder networks are trained to solve the pretext task by supervised learning methods. Since shallow layers extract fine-grained details such as edges, angles, and textures, while deeper layers capture task-related high-level features such as semantic information or image contents, learned encoders on pretext tasks can be transferred to downstream supervised tasks. During this stage, the parameters of the backbone are fixed, and only a simple classifier, such as a two-layer Multi-Layer Perceptron (MLP), needs to be learned. Considering the limited workload in the downstream training stage, this learning process is commonly referred to as fine-tuning. In summary, the representations learned during the pretraining stage in SSL can be reused on other downstream tasks and achieve comparable results.

In this section, we introduce different tasks for pretraining PFM in CV. The PFM can be trained by specific pretext tasks, frame order, generation, reconstruction, memory bank, sharing, clustering and so on. We summarize the PFM proposed in CV in **Table 2**.

4.1 Learning by Specific Pretext Task

In the early stage of unsupervised learning, the network is trained by designing a special pretext task and predicting the answer to this task. Dosovitskiy et al. [134, 135] pretrain the Exemplar CNN to discriminate the different patches from the unlabelled data. The experiments prove the designs can learn useful representations transferred to the standard recognition assignments. In the method based on context prediction [136], a handcrafted supervised signal about the position information serves as the label for the pair classification. Inpainting [137] aims to pretrain models by predicting the missed center part. Because inpainting is a

semantic-based prediction, another decoder is linked to the context encoder in this manner. Furthermore, the standard pixel-by-pixel reconstruction process of the decoder can be transferred to any other downstream inpainting tasks. Specifically, Colorization [138] is a method that evaluates how colorization as a pretext task can help to learn semantic representation for downstream tasks. It is also known as the *cross-channel encoding* since different image channels serve as input and the output is discriminated. Similarly, Split-Brain Autoencoder [139] also learns representations in a self-supervised way by forcing the network to solve cross-channel prediction tasks. Jigsaw [140] is proposed to pretrain the designed Context-Free Network (CFN) in a self-supervised manner by first designing the Jigsaw puzzle as a pretext task. Completing Damaged Jigsaw Puzzles (CDJP) [141] learns image representation by complicating pretext tasks furthermore, in which puzzles miss one piece and the other pieces contain incomplete color. Following the idea of designing efficient and effective pretext tasks, Noroozi et al. [142] use counting visual primitives as a special pretext task and outperform previous SOTA models on regular benchmarks. NAT [143] learns representation by aligning the output of backbone CNN to low-dimensional noise. RotNet [144] is designed to predict different rotations of images.

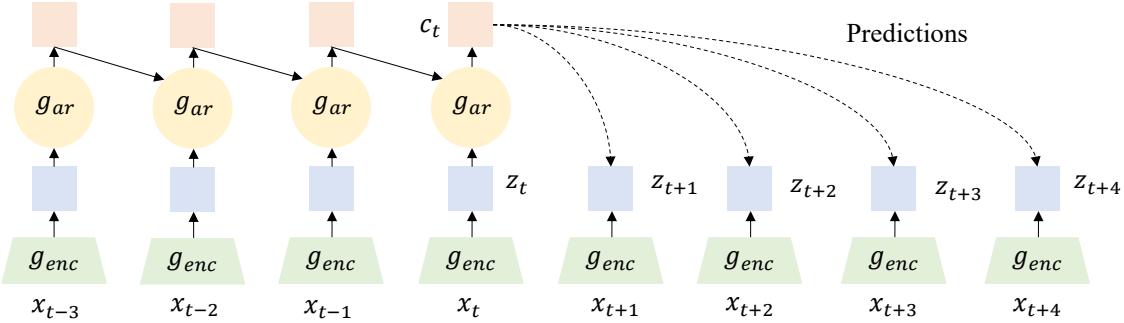


Figure 7: Contrastive Predictive Coding [145]. The input sequence can represent both images and videos.

4.2 Learning by Frame Order

The learning of sequence data such as videos always involves frame processing through time steps. This problem often connects with solving pretext tasks that can help to learn visual temporal representations. Contrastive Predictive Coding (CPC) [145] is the first model to learn data representations by predicting the future in latent space. This model can be fed with data in any modalities, like speech, images, text, etc. The components of CPC are shown in Fig. 7 from [145], where the x_t represents the input sequence of observations, z_t is a sequence of latent representations after the encoder g_{enc} , and c_t is a context latent representation that summarizes all the latent sequence $z_{\leq t}$ after an autoregressive model g_{ar} . Unlike the traditional model predicts future frames x_{t+k} by a generative model $p_k(x_{t+k}|c_t)$, CPC models a "density ratio" f_k to represent the mutual information between the context latent representation c_t and future frame x_{t+k} :

$$f_k(x_{t+k}, c_t) \propto p(x_{t+k}|c_t)/x_{t+k}. \quad (10)$$

After the encoding of recurrent neural networks, z_t and c_t can both be chosen for the downstream tasks as needed. The encoder and autoregressive model are trained by InfoNCE [145] as follows

$$\mathcal{L} = -\mathbb{E}_X[\log f_k(x_{t+k}, c_t) / \sum_{x_j \in X} f_k(x_j, c_t)], \quad (11)$$

where X denotes the training dataset containing both positive and negative samples. The density ratio f_k can be estimated by optimizing \mathcal{L} . CPC v2 revisits and improves CPC [146] by pretraining on unsupervised representations, and its representation generality can be transferred to data-efficient downstream tasks.

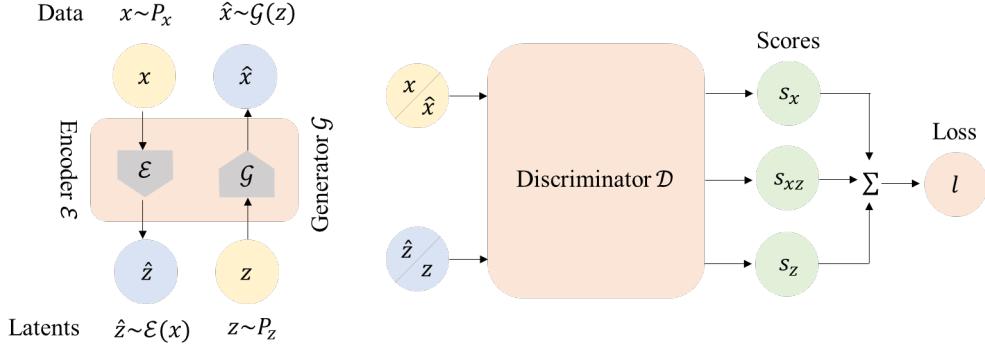


Figure 8: The structure of the BigBiGAN framework [147].

4.3 Learning by Generation

Although many existing applications are popular after the development of the GAN-based approach, the representation abilities inside the GANs are not entirely exploited due to the absence of a feature encoder. Thus, Bidirectional Generative Adversarial Networks (BiGANs) [48] is proposed to project data back into the latent space, which is useful for auxiliary supervised discrimination tasks via serving as feature representations.

Based on BiGANs, BigBiGAN [147] first achieves the SOTA in unsupervised representation learning on ImageNet by adding an encoder and modifying the discriminator. As shown in Fig. 8 from [147], the traditional components of GANs (encoder \mathcal{E} and generator \mathcal{G}) are used to produce data-latent pairs, denoted as $(\mathbf{x} \sim P_x, \hat{\mathbf{z}} \sim \mathcal{E}(\mathbf{x}))$ and $(\hat{\mathbf{x}} \sim \mathcal{G}(\mathbf{z}), \mathbf{z} \sim P_z)$. The final loss ℓ is defined as the sum of data-specific term s_x, s_z and data-joint term s_{xz} . The introduced discriminator \mathcal{D} (Adversarially Learned Inference (ALI) [148], or BiGAN [48]) learns to discriminate between pairs from the raw data, latent distribution and encoded vector.

4.4 Learning by Reconstruction

The iGPT [149] and ViT [40] models have demonstrated the feasibility of adapting the pretext task of masked prediction using auto-encoder from language to image data. BEiT [150] is the first to demonstrate that autoencoder-based masked prediction can outperform DINO [151], a conventional SOTA method without pretraining techniques. Specifically, BEiT consists of two stages: token embedding with discrete variational autoencoder (dVAE) [152], and tokenizer training with masked image prediction. In the first stage, the original image is split into some patches and encoded using discrete tokens, which is different from BERT since image patches don't have off-the-shelf tokens as words in NLP. In the second stage, the BEiT encoder takes a corrupted image containing unmasked and masked patches, and then the visual tokens of the masked patches are outputted to match the corresponding visual tokens from the fixed tokenizer. Despite its success, the separation between masked prediction and autoencoder training induces that the whole framework is not end-to-end and hinders learning effectiveness and efficiency.

To migrate this issue, MAE [154] proposes an end-to-end simple solution by predicting the masked patches directly from the unmasked ones with the Mean Squared Error (MSE) loss. It's worth noting that MAE uses a masking ratio of 75%, which is significantly higher than that of BERT (typically 15%). Ablation study suggests that higher masking ratios are beneficial for both fine-tuning and linear probing. Concurrently, SimMIM [155] proposes a similar autoencoder-based solution as MAE, in which they also confirm

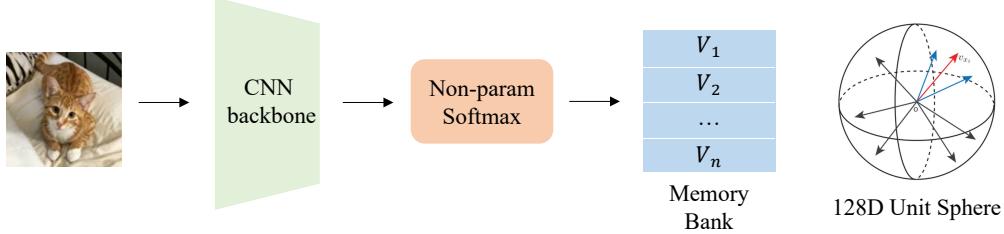


Figure 9: The general pipeline for the Memory Bank Method [153].

that a higher marking ratio and leveraging random masking strategy helps improve performance. The major difference is how they partition the responsibility of representation encoding and pretext prediction in the autoencoder. Since the decoder of SimMIM is simple, the encoder of SimMIM synchronously conducts both of them. On the contrary, the encoder in MAE solely undertakes the role of representation encoding, and the decoder is responsible for pretext prediction. Recently, Meta AI announces the Segment Anything Model (SAM) [156] which prompts users to specify what to segment in an image, allowing for a wide range of segmentation tasks without the need for additional training. SAM employs an MAE pretrained ViT-H [40] image encoder that runs once per image and produces an image embedding, as well as a prompt encoder that embeds input prompts such as clicks or boxes. Following that, a lightweight transformer-based mask decoder predicts object masks from image and prompt embeddings. The results show that SAM can generate high-quality masks from a single foreground point that are typically just modestly inferior to the manually annotated ground truth. It routinely achieves strong quantitative and qualitative outcomes on a wide range of downstream tasks using a zero-shot transfer approach and prompt engineering.

Leveraging ViT in MAE poses a serious inefficiency issue, where decreasing the patch size results in a quadratic increase in computing resources. To address the problem, there are two important solutions: (1) hierarchical ViT and (2) local attention. In the first direction, hierarchical ViT (hViT) was introduced, which utilizes a shrinking pyramid structure and techniques like shifted windows [157] to reduce computational demands. Unfortunately, hViT cannot be directly applied to enable MAE pretraining because the local window attention used in hViT makes it difficult to handle randomly masked patches as in MAE. Recently, Uniform Masking MAE (UM-MAE) [158] is proposed to empower MAE with hViTs, which introduces a two-stage pipeline: sampling and masking. It starts by randomly sampling a portion of patches (25% reported in the paper) from each block, and then follows by masking additional patches on top of the sampled ones. The first step helps to maintain common elements across different local windows, while the second step prevents shortcuts for pixel reconstruction from nearby low-level features, making the task more difficult. Another direction to improve efficiency focuses on reducing the input size by putting the attention of the network into some local small windows of the image. Motivated by the observation that local knowledge is sufficient for reconstructing masked patches, Local masked reconstruction (LoMaR) [159] was proposed. Rather than using the entire image for mask reconstruction, LoMaR samples a number of small windows and focuses attention on local regions, which outperforms MAE on downstream tasks in terms of learning efficiency.

4.5 Learning by Memory Bank

Non-Parametric Instance Discrimination (NPID) [153] is the first method that utilizes the instances to learn representations for downstream tasks. The detailed pipeline is shown in Fig. 9. The feature representations are stored in the memory bank for the convenience of computation because the instance-level classification objective needs all images in the training dataset. For any image x with feature representation $\mathbf{v} = f_\theta(x)$,

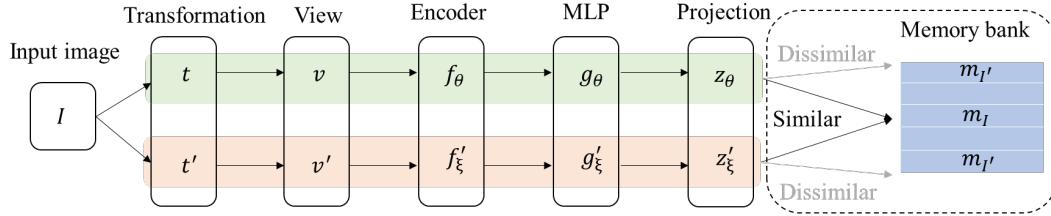


Figure 10: Summary of all two-stream models, including contrastive learning and memory-bank-based methods.

its probability of being recognized as i -th example is:

$$P(i|\mathbf{v}) = \exp(\mathbf{v}_i^\top \mathbf{v} / \tau) / \sum_{j=1}^n \exp(\mathbf{v}_j^\top \mathbf{v} / \tau), \quad (12)$$

where \mathbf{v}_i or \mathbf{v}_j is the representation of i -th or j -th sample, which serves as a substitute for the parametric class prototype (i.e., weights of a classifier). Additionally, τ is the temperature parameter borrowed from the knowledge distillation [160].

Local Aggregation (LA) [161] is another method that trains a CNN encoder to embed raw images into a lower dimension space – embedding space. When a metric of local aggregation is maximized, similar data instances move together in the embedding space while dissimilar instances move apart.

Based on NPID, Pretext Invariant Representation Learning (PIRL, pronounced as “pearl”) [162] is proposed to argue that semantic representations are invariant under pretext transformation tasks. Suppose the original view and transformed view of images are denoted as I and I^t , respectively. These sample views are fed into a CNN encoder, and the total empirical loss on the training dataset \mathcal{D} can be defined as:

$$\mathcal{L}_{total}(\theta; \mathcal{D}) = \mathbb{E}_{t \sim \mathcal{T}} \left[\frac{1}{|\mathcal{D}|} \sum_{I \in \mathcal{D}} \mathcal{L}(\mathbf{V}_I, \mathbf{V}_{I^t}) \right], \quad (13)$$

where \mathcal{T} denotes the different transformations of images. The loss encourages the representation of image I to be similar to that of I^t , and the representation of I^t to be dissimilar to that of different images I' , as shown in the dotted box of Fig. 10. Therefore, more negative sample pairs contribute to improving the scalability of the gradient and lead to the final learned encoder with stronger representation ability. That is the reason why the memory bank is introduced to store more previous representations for subsequent comparison.

4.6 Learning by Sharing

SSL prefers using two encoder networks for the different data augmentation, and then pretrains the parameters by maximizing the distance between negative pairs or minimizing the distance between positive pairs. Fig. 10 shows the two-stream models for all contrastive learning frameworks. The transformation t on the original input image I generates the view v , similarly, its counterpart t' generates v' . In general, two different or same encoders f_θ and f'_ξ are used to extract contrastive representations. The subsequent MLP heads g_θ and g'_ξ are used to learn more combinations that are beneficial to the contrastive loss. It is noticed that MLP and memory bank could be removed or preserved under different settings. In terms of the shared encoder, SSL can be divided into two categories: 1) Soft Sharing that two encoders share with similar but different parameters ($f_\theta \neq f'_\xi$); 2) Hard Sharing that two encoders maintain the same architectures and parameters ($f_\theta = f'_\xi$).

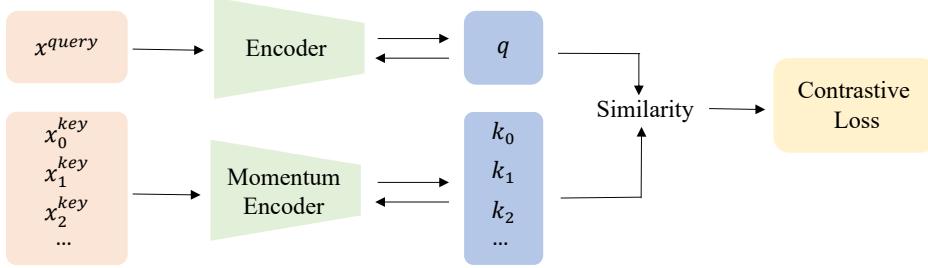


Figure 11: The general pipeline of MoCo [163], which is also a two-stream framework with different parameters.

Soft Sharing. Facebook AI Research (FAIR) presents Momentum Contrast (MoCo) [163] by using momentum to control the slight difference between two encoders. As shown in Fig. 11, one of the encoders is served as a dictionary look-up task that generates a queue of encoded data samples $\{k_0, k_1, \dots\}$. Another encoder generates encoded query $\{q_0, q_1, \dots\}$ with the training batch updated. The similarity is measured by the dot product of the new coming encoded query q and the encoded keys stored in the dictionary queue. Suppose there are K keys stored in the queue before the new key comes. The K keys are treated as negative samples to the query of the new key. To combine the contrastive loss on both negative and positive samples, InfoNCE Loss [145] is used for the pretraining in MoCo. The key design in MoCo for soft parameter sharing is called momentum update. He et al. [163] suggest that the direct parameter change of key encoder (i.e., momentum encoder) to query encoder loses the necessary consistency and yields poor results. The momentum encoder parameter θ_k is updated as:

$$\theta_k = m\theta_k + (1 - m)\theta_q, \quad (14)$$

where the query encoder parameter θ_q is learned directly from the gradients of new coming instance, and $m \in [0, 1]$ is a hyper-parameter that controls the consistency (θ_k is more consistent if m is closer to 1).

Inspired by the design of SimCLR [164], in MoCo v2 [164], the FAIR team introduces an MLP projection head after encoders and utilizes more data augmentation techniques to improve the performance. The further improvements are from that: 1) embedded linear classifier bridges the gap between unsupervised and supervised pretraining representations; 2) more contrastive samples are feasible from both the larger training batch and stronger data augmentation.

DeepMind proposed Bootstrap Your Own Latent (BYOL) [165] that contains representation, projection, and discrimination stages to achieve a new SOTA without using negative samples. They understand the discrimination between different views of raw images as necessary prevention from collapse during the pretraining. However, they argue that many negative samples are not indispensable to prevent this collapse. As shown in the left part of Fig. 10, there are two streams in BYOL with different parameters. The online network (top green) updates parameters by comparing the prediction generated itself and the regression target provided by the target network. Then the parameters of the target model (bottom red) are updated the same as Eq. (14), *i.e.*, $\xi \leftarrow \tau\xi + (1 - \tau)\theta$, where τ is the target decay rate to control the degree of parameter changing in the target network. Therefore, the target network can also be understood as a momentum encoder. Here, ξ in the target model is the parameter θ_k in momentum encoder, and θ in the online network denotes the parameter θ_q in the query encoder.

Hard Sharing. SimCLR [166] is proposed by Brain Team in Google Research which utilizes the hard parameter-sharing architecture. This simple framework can also be concluded in Fig. 10, in which we can

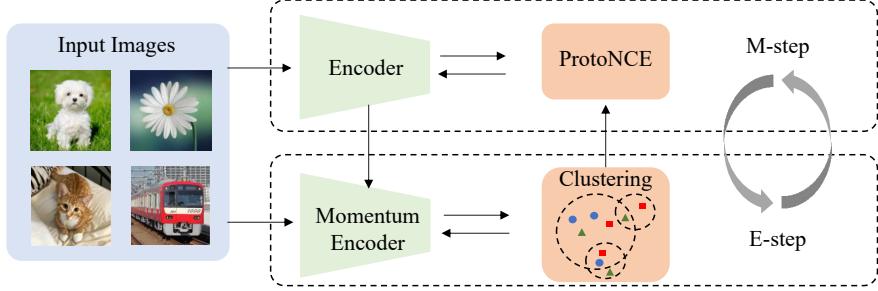


Figure 12: The key pipeline for the DeepCluster model [49].

see that representations of different views of the same image are learned in the network $f(\cdot)$. This base encoder shares the parameters with each other. Thus, memory bank and momentum setting to learn key and query encoders are not necessary, which contributes to a simpler backbone architecture and easier learning strategy. The loss function to maximize the similarity between different views of the same image (positive pairs) is defined as

$$\ell_{i,j} = -\log \exp(sim(z_i, z_j)/\tau) / \sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(sim(z_i, z_k)/\tau), \quad (15)$$

where (i, j) is a pair of positive samples, τ is an introduced hyper-parameter called temperature parameter [153], and $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function to control the denominator containing only negative pairs.

To avoid the dependence on a large number of explicit pairwise feature comparisons, Swapping Assignments between multiple Views of the same image (SwAV) [167] is proposed as an online algorithm by Inria and FAIR. SwAV introduces clustering to substitute the previous comparison between pairs, which gains more memory with the help of non-queue architecture. In this method, the clustering prototype joins the computation of the defined loss function. This prototype is encoded as the concatenation of vectors learned through the backpropagation in CNNs. Thus, there is no need for SwAV to compare the encoded representations between different views.

Based on the existing SwAV, a novel model called SElf-supERvised (SEER) [168] aims to learn a pre-trained encoder from any random image and unbounded dataset in the wild. The base network is RegNetY architectures [169] trained with the SwAV SSL method [167]. This method proves that the SSL is not specific to a curated dataset such as ImageNet, and the scalability of recent RegNet releases the limitation of traditional backbones such as ResNet. In addition, this method encourages the research community to explore more backbones suitable for universal SSL.

Attracting the attention in the recent SSL, FAIR conducts empirical experiments on the SSL by utilizing the structure of Simple Siamese (SimSiam) networks. This method [170] can avoid the design of negative sample pairs, large batches (or memory banks), and momentum encoders in traditional contrastive learning. The two encoders in Fig. 10 with identical parameters that process two different views t and t' of image x are substituted by the only siamese network. MLP predictor g is used for one of the view representations, and then the stop-gradient operation is applied to another view representation.

4.7 Learning by Clustering

DeepCluster [49] is the first model that adopts the clustering algorithm for large-scale dataset learning. This method groups the representations into different clusters and labels these clusters as supervised signals to

Table 2: Summary of the PFM in CV.

Year	Conference	Method	Pretext Task	Architecture	Downstream Task ¹	Code
2014	NeurIPS	Exemplar-CNN [134, 135]	discrimination	CNN	cla, rec	https://limb.informatik.uni-freiburg.de/...
2015	ICCV	Context [136]	context prediction	CNN	cla, det, clu	https://github.com/.../deepcontext
2016	CVPR	Inpainting [137]	inpainting	GAN, CNN	cla, det, seg, inp	https://github.com/.../context-encoder
2016	ECCV	Colorization [138]	colorization	CNN	cla, det, seg	https://github.com/.../colorization
2016	ECCV	Jigsaw [140]	Jigsaw puzzles	CNN	cla, det, seg, ret	https://github.com/.../JigsawPuzzleSolver
2017	CVPR	Split-Brain [139]	channel prediction	CNN	cla, det, seg	https://richzhang.github.io/splitbrainauto
2017	ICCV	Counting [142]	counting	CNN	cla, det, seg, ret	https://github.com/clvrai/...
2017	ICML	NAT [143]	noise	CNN	cla, det	-
2017	ICLR	BiGAN [48]	generation	GAN, CNN	cla, det, seg	https://github.com/.../bigan
2018	WACV	CDJP [141]	Jigsaw puzzles	CNN	cla, det, seg	-
2018	ICLR	RotNet [138]	rotation	NIN, CNN	cla, det, seg	https://github.com/gidariss/...
2018	arXiv	CPC [145]	patch overlapping	CNN, GRU	cla	-
2018	CVPR	NPID [153]	instance discrimination	CNN	cla	https://github.com/.../lemniscate.pytorch
2018	ECCV	DeepCluster [49]	clustering	CNN	cla, det, seg	https://github.com/.../deepcluster
2019	ICCV	LA [161]	local aggregation	CNN	rec, det	https://github.com/.../LocalAggregation
2019	NeurIPS	BigBiGAN [147]	generation	GAN, CNN	gen, cla	https://thub.dev/.../bigbigan
2019	CVPR	AET [172]	transformation	CNN	cla	https://github.com/.../AET
2019	NeurIPS	AMDIM [173]	discrimination	CNN	cla	https://github.com/.../amdim-public
2020	CVPR	ClusterFit [174]	clustering	CNN	cla, seg	-
2020	ICML	CPC v2 [146]	patch overlapping	CNN	cla, det	-
2020	CVPR	PIRL [162]	Jigsaw puzzles	CNN	cla, rec, dec	https://github.com/.../PIRL
2020	CVPR	MoCo [163]	discrimination	CNN	cla, rec, dec, pos, seg	https://github.com/.../moco
2021	ICLR	PCL [171]	clustering	CNN	cla, det	https://github.com/.../PCL
2020	arXiv	MoCo v2 [164]	discrimination	CNN	cla, dec	https://github.com/.../moco
2020	ICLR	SeLa [175]	self-labelling	CNN	cla, det, seg	https://github.com/.../self-label
2020	ICML	SimCLR [166]	discrimination	CNN	cla	https://github.com/.../simclr
2020	NeurIPS	SimCLR v2 [176]	self-distillation [160]	CNN	cla	https://github.com/.../simclr
2020	ECCV	CMC [177]	view matching [178]	CNN	cla, seg	https://hobbitlong.github.io/CMC
2020	NeurIPS	InfoMin [179]	discrimination	CNN	cla, det, loc, seg	https://hobbitlong.github.io/InfoMin
2020	NeurIPS	SwAV [167]	cropping	CNN, Transformer	cla, det	https://github.com/.../swav
2020	NeurIPS	BYOL [165]	discrimination	CNN	cla, det, seg	https://github.com/.../byol
2021	arXiv	MoCo v3 [180]	discrimination	CNN, Transformer	cla	-
2021	ICLR	ReLIC [181]	discrimination	CNN	cla, rel	-
2021	ICLR	PCL v2 [171]	clustering	CNN	cla, det	https://github.com/.../PCL
2021	CVPR	SimSiam [170]	discrimination	CNN	cla, det, seg	https://github.com/.../simsiam
2021	ICML	DirectPred [182]	discrimination	CNN	cla	https://github.com/.../ssl
2021	ICCV	DINO [151]	discrimination	CNN, Transformer	cla, seg	https://github.com/.../dino
2021	arXiv	MoBY [183]	discrimination	CNN, Transformer	cla, det, seg	https://github.com/.../Transformer-SSL
2021	NeurIPS	MST [184]	token prediction	CNN, Transformer	cla, det, seg	-
2022	ICLR	BEiT [185]	token prediction	Transformer	cla, seg	https://github.com/.../beit
2022	CVPR	MAE [154]	reconstruction	Transformer	cla, det, seg	https://github.com/facebookresearch/mae
2022	CVPR	SimMIM [155]	reconstruction	Transformer	cla, det, seg	https://github.com/microsoft/SimMIM
2022	ArXiv	UM-MAE [158]	reconstruction	Transformer	cla, det, seg	https://github.com/implus/UM-MAE
2022	ArXiv	LoMaR [159]	reconstruction	Transformer	cla, det, seg	https://github.com/junchen14/LoMaR
2022	ArXiv	CAE [186]	reconstruction	Transformer	cla, det, seg	https://github.com/lxtGH/CAE
2023	AAAI	PeCo [187]	reconstruction	Transformer	cla, det, seg	-
2023	ArXiv	SAM [156]	reconstruction	Transformer	det, gen, seg	https://github.com/facebookresearch/segment-anything

¹ Downstream task types: classification (cla), recognition (rec), detection (det), localization (loc), segmentation (seg), clustering (clu), inpainting (inp), retrieval (ret), generation (gen), pose estimation (pos), reinforcement learning (rel).

pretrain the parameters of the backbone network. It demonstrates SOTA performance on a wide range of standard transferred tasks used in unsupervised learning.

When it comes to the connection between contrastive learning and clustering, SwAV [167] has utilized prototypes that serve as a clustering center to help classify the sample pairs during pretraining, while Prototypical Contrastive Learning (PCL) [171] first targets bridging contrastive learning with clustering. Compared to instance discrimination as pretext tasks learning low-level representations, clustering can help to encode more semantic information. Then more semantic-based downstream tasks will benefit from it. As shown in Fig. 12, prototypical contrastive learning uses prototypes to substitute one of the views of generated samples in NCE loss (Eq. (15)), which is the proposed ProtoNCE loss in PCL. In addition, PCL is also a method based on soft parameter sharing, in which the momentum encoder is updated as Eq.(14).

4.8 Summary

This section extensively investigates recent progress in PFM on images for representation learning, from the early perspective of designing pretext tasks for self-labeling to present contrastive loss-based SSL. The pipelines of the main methods are clearly illustrated. We hope this section can prepare the incoming researchers to acquire a basic understanding of this novel area and some worthwhile research direction. We

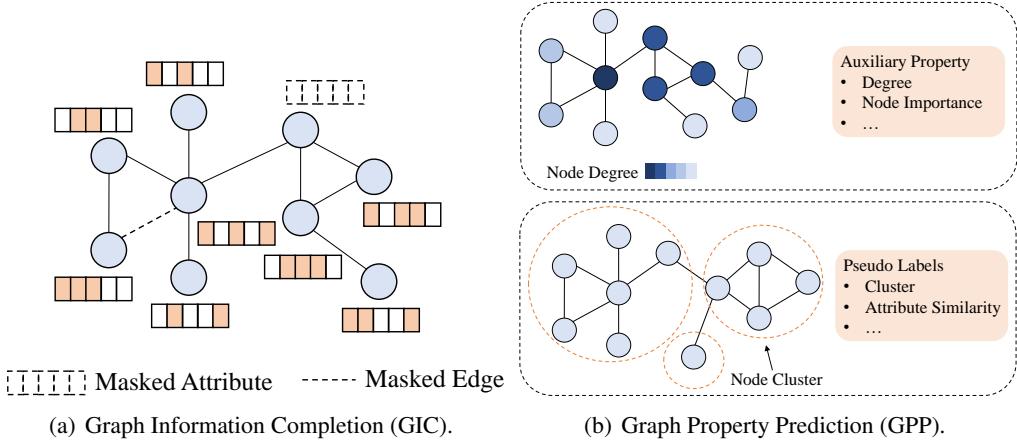


Figure 13: Graph Information Completion (GIC) and Graph Property Prediction (GPP).

believe the powerful generalization ability of PFM would extremely reduce training computation overhead by “pretraining once and transferring forever”. Recent transformer-based PFM have gradually outperformed traditional training from scratch on target datasets. This discovery will spur further exploration and research into this exciting field.

5 PFM for Graph Learning

With the development of deep learning in graphs, the parameters (i.e., graph embedding) of the model began to increase rapidly. Therefore, large-scale labeled data is needed for training the models to avoid under-fitting or over-fitting. However, the cost of constructing large-scale labeled datasets for graphs is too subjective, expensive, and time-consuming, especially in domains that require professional knowledge and timeliness. While some semi-supervised approaches have temporarily mitigated the reliance of graph embedding models on label scale, they have not fundamentally resolved this problem. In recent times, researchers have turned their attention towards the application of PFM in the field of graphs, inspired by their success in CV and NLP. However, for most graphs, obtaining large-scale pretraining data directly is challenging due to the unique nature of information such as nodes and edges. Therefore, recent studies have focused on utilizing the inherent information of a graph’s attributes, topology, and community to enhance the effectiveness of the node’s features. We have summarized the graph-related PFM in **Table 3**.

5.1 Learning by Graph Information Completion

The essential motivation of pretraining based on graph information completion (GIC) is to mask part of the information of the input graph data and recover the masked information based on the unmasked graph data, so as to pretrain the graph embedding, as shown in Fig. 13. Similar ideas appeared earlier in the field of image and text processing. For instance, in image processing, information such as image pixels and colors are recovered to pretrain the image encoder; in text processing, many methods implement pretraining of word embeddings and encoders by recovering part of the information in a sentence based on context words. These methods inspire the design of graph completion tasks on graph PFM.

Among them, You et al. [188] are inspired by image inpainting, and first propose to cover them by removing the features of the target nodes, and then recover/predict the features of the masked nodes. In order

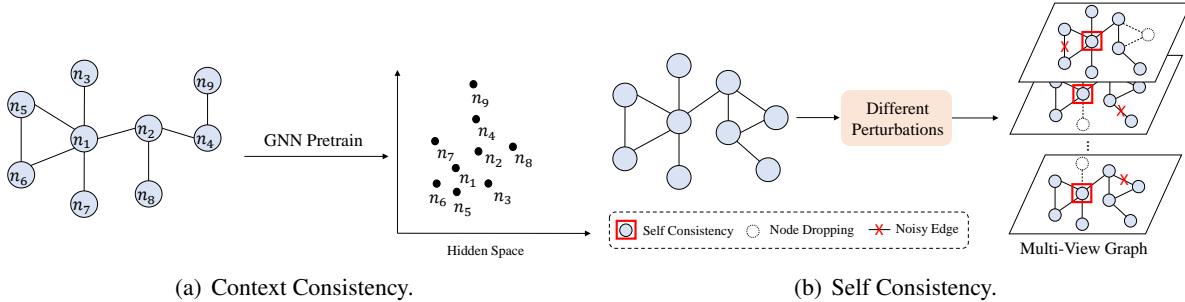


Figure 14: Graph Consistency Analysis (GCA).

to recover/predict the masked information, GraphCompetition [188] is achieved by providing GCNs with unmasked node features (limited to the 2-layer GCNs of the second-order neighbors of each target node). The purpose of GraphCompetition’s pretraining is to help the model better perform feature representation and teach the model to extract features from the context. You et al. [188] propose the attribute mask task (namely, AttributeMask), which masks node attributes randomly, and then requires the self-supervising module to reconstruct the masked attributes. Jin et al. [189] think deeply about SSL on graph data, and propose the edge mask task (namely, EdgeMask), seeking to develop self-supervision in pairs based not only on a single node itself but on the connection between two nodes in the graph. In particular, EdgeMask randomly masks some edges and then asks the model to reconstruct the masked edges. In short, EdgeMask is expected to help GNN learn local connectivity information. Hu et al. [190] propose a PFM that masks node and edge attributes and then predicts this masked information based on the adjacent structure.

5.2 Learning by Graph Consistency Analysis

Different from the aforementioned methods that focus on individual elements in the graph, graph consistency analysis (GCA) mainly explores the consistency of the distribution of two elements in the graph. Specifically, the consistency of two elements with similar semantics should be significantly stronger than two elements with unrelated semantics, and this characteristic can be used to pretrain the graph model. According to the judgment object of consistency, such methods can be roughly divided into the following three categories.

Context Consistency Based on the early homogeneity assumption, a mass of graph models tends to project contextual nodes to similar positions in semantic space. Such consistency of the context in the graph is also applied to the pretraining graph model, which attempts to adjust the node representation by capturing the distribution characteristics of the nodes in the context, as shown in Fig. 14 (a).

Random walk is an efficient method to acquire context. It can capture the distribution characteristics of different perspectives in the context by designing a variety of walk strategies. The DeepWalk [191] adopts a truncated random walk strategy to represent the node context as the form of a sequence of nodes. By introducing the idea of NLP into the network embedding model, DeepWalk regards the node sequence as a “sentence” and models it based on the skip-gram model, providing an unsupervised and scalable training method for node representation. Furthermore, on the basis of DeepWalk, node2vec [192] uses two different parameter-controlled random walk strategies to obtain deviated node sequences to fully capture the context information.

Different from randomly sampling nodes from the context, some recent methods directly consider the

relationship between the node’s k-order neighbor distribution (as positive examples) and non-adjacent nodes (as negative examples), and use this to train the graph model. LINE [193] respectively proposes first- and second-order proximity to describe the local similarity between pairs of nodes in the graph from different perspectives, and uses it to optimize node representation. Meanwhile, LINE uses negative sampling and edge sampling techniques to optimize the second-order traversal and excessive training storage overhead. VGAE [194] introduces a variational autoencoder to encode graph structure data, and model the node first-order neighbor through a GCN encoder and a simple inner product decoder.

Self Consistency In the field of NLP and CV, contrastive learning as an efficient self-supervised mechanism is widely used in the pretraining of models. In fact, the internal comparison mechanism of such methods is based on the mutual information estimation of the original graph data and the augmented graph data to maintain the consistency of the data itself, as shown in Fig. 14 (b). Inspired by contrastive learning, some studies have begun to generate augmented samples of original data samples in the graph model. Among them, two augmented samples from the same original sample are regarded as positive pairs, and two augmented samples from different original samples are regarded as negative pairs.

For node-level tasks, GCC [195] devises the pretext task as subgraph instance discrimination in and across networks. And GCC also enhances the ability of GNNs to learn the intrinsic and transferable structural representations by introducing contrastive learning. Specifically, GCC samples subgraphs from the whole graph as augmentations via random walk with restart and artificially designs positional node embedding as node initial features. As a novel graph representation learning model, GCA [196] incorporates various priors for topological and semantic aspects of the graph to achieve adaptive contrastive augmentation. Specifically, GCA devises an enhancement scheme based on node centrality measures to highlight important connection structures, while corrupting node features by adding noise to specific nodes to lead the pretraining model to recognize underlying semantic information.

For graph-level tasks, some studies have attempted to introduce more diverse contrastive learning strategies. Among them, You et al. [197] introduce four common graph augmentation tasks (i.e., node dropping, edge perturbation, attribute masking, and subgraph sampling) into the GL model based on underlying prior and propose a unified comparative learning framework: GraphCL. Meanwhile, GraphCL discusses in depth the role of data augmentation in comparative learning and gives experimental demonstration that joint multiple augmentation strategies can improve model performance.

Cross Scale Consistency Unlike the above two methods that consider the consistency of elements in the same scale, contrasting elements in graph data of different scales can also be used to train graph models, e.g., node-subgraphs. Most of such methods have the idea of maximizing mutual information [198, 199]. Specifically, the readout function is usually used to obtain the summary of the graph/subgraph, and the MI estimator can be calculated using the Jensen-Shannon divergence.

As a representative method, DGI [200] relies on maximizing the MI between the patch representation and the summary of the corresponding high-level graphs, which are all derived using the established graph convolutional network architecture, to learn the node representation. To generate negative samples on a single graph, DGI corrupts the original graph by randomly scrambling node features while keeping the structure unchanged. Similarly, Hassani and Khasahmadi propose CMVRL [201], which generates an additional structural view of a sample graph based on graph diffusion. The sample graph and a regular view are sub-sampled together, and the node representation and graph representation are learned based on two shared MLPs, and then contrast learning is achieved through the consistency loss provided by the discriminator.

SUBG-CON [202] samples a series of context subgraphs from the original graph and inputs them to

the encoder to obtain the pooled central node and subgraph representation. For the specified node, the context subgraph is expressed as a positive sample, and other randomly sampled subgraphs are expressed as a negative sample. The contrast loss of the latent space will force the encoder to identify positive samples and negative samples in order to distinguish different nodes based on regional structure information.

5.3 Learning by Graph Property Prediction

Considering the attribute and structural information of the graph as the target of information completion, pretraining based on graph property prediction (GPP) can also be used to build the graph model in different forms. One of the most common methods is to generate self-supervised signals by exploring the auxiliary property in the graph data and to take the graph property prediction task as the pretraining task of the graph model. According to the different settings of the pretext task, it can roughly classify two categories: property regression and property classification.

Property Regression (PR) In the graph model, different from the GIC mentioned above, property regression primarily focuses on mining the relationship between the broader numerical structure and property attributes within the graph. Specifically, this branch of methods extracts richer self-supervised signals in graph data for pretraining graph models.

For example, similar but different from masking node attributes, the goal of NodeProperty [189] is to predict each node’s auxiliary property in the graph, e.g., degree, local node importance, and local clustering coefficient. In other words, NodeProperty is used to encourage GNN to capture richer local structural information while optimizing the specific downstream tasks. Specifically, NodeProperty regards the node degree as a representative local node property, i.e., self-supervised signal, and takes other node properties as future work. Meanwhile, NodeProperty emphasizes that the intuition of devising self-supervised pretext tasks related to local node property is to ultimately guide the feature embedding of GNN (i.e., node representation) to save this information, which relies on the assumption that the node property information is relevant to the particular task.

Property Classification (PC) Different from the property regression task, the task of property classification is usually implemented by defining pseudo-labels based on a certain distribution in the graph data, which is a typical self-supervised method. Among them, the structure density, similarity of node attributes, and difference between local and global distributions are the most commonly used. We will briefly introduce the application of such methods in GL pretraining.

Among these methods, clustering is the most common and effective source of pseudo-labels. Among them, M3S [203] designs a multi-stage training strategy, using the idea of graph clustering to iteratively train the graph encoder, achieving enlarged labeled data with virtual labels in the case of very small samples. You et al. [188] further propose two pretraining strategies. Among them, Node Clustering assigns K (hyper-parameter) pseudo labels to nodes based on attribute clustering and pretrain node representation by node classification. In addition, You et al. also present Graph Partitioning based on the topology density assumption. In Graph Partitioning, the nodes of a graph are divided into approximately equal K (hyper-parameter) subsets to minimize the number of edges connecting nodes among subsets, and then pseudo labels are provided for nodes.

In addition to clustering methods, some researchers generate pseudo labels based on other statistical characteristics of graph data. For instance, in the molecular field, Rong et al. [204] use the molecular bonds of subgraphs and related statistical information to guide GNN to learn Context-Sensitive Properties (CSP)

and then apply them to prediction. Rong et al. [204] propose a Motif Prediction (MP) task, which can be expressed as a multi-label classification problem, in which each motif corresponds to a label. Specifically, let’s assume that K motifs in molecular data are considered. For a specific molecule (abstracted as graph G), they use RDKit to detect whether each motif appears in G , and then take it as the target of the motif prediction task.

5.4 Learning by Masked Autoencoder

The masked autoencoder (MAE) is first applied in MAGE [205], the masked autoencoders for self-supervised learning on graphs. Following MAE [154], MGAE operates on a partial network structure (without masked edges) that is based on convolutions. Besides, the decoder of MGAE is designed to model the cross-correlation between the head and tail nodes of an anchor edge. Empirical results demonstrate that MGAE performs better than traditional graph autoencoders and graph SSL approaches. Furthermore, GMAE [206] extends this approach by using a transformer instead of convolutions and reconstructing the features of masked nodes rather than masked edges. In addition to empirical improvements, MaskGAE [207] further provides theoretical justifications for the potential benefits of masked graph modeling. Designing algorithms to accommodate graphs of various complex properties is a promising direction. For instance, to tackle the heterogeneous graphs scenario, HGMAE [208] proposes meta-path masking and adaptive attribute masking with a dynamic mask to enable effective and stable learning on complex graph structure. Moreover, several training strategies are developed, including meta-path-based edge reconstruction to incorporate complex structural information, target attribute restoration to utilize various node attributes, and positional feature prediction to encode node positional information. Besides dealing with more complex graph structures, how to improve the learning efficiency of MAE on graph data remains an open question.

5.5 Other Learning Strategies on Graph Data

In addition to the above methods, there are lots of pretraining methods that use relatively novel or hybrid strategies. For example, CG³ [209] generates an improved node representation by designing a semi-supervised consistency loss to maximize the consistency between different views of the same data or data from the same category. Next, CG³ uses the graph generation loss related to the input feature to extract the potential deterministic relationship between the data feature and the input graph topology as a supplementary supervision signal for SSL.

Based on the attention mechanism, Graph-Bert [210] trains itself to reconstruct node attributes and topological structure with sampled linkless subgraphs within their local contexts. GMI [211] extends the traditional mutual information computing idea from the vector space to the graph domain and proposes to jointly maximize feature mutual information (between the node’s embedding and raw features of its neighbors) and edge mutual information (embedding of two adjacent nodes) for graph representation learning. GPT-GNN [212] proposes a self-supervised graph generation task to guide itself to capture the topological and semantic attributes of the graph. GPT-GNN roughly divides the possibility of graph generation into attribute generation and edge generation to untangle the intrinsic dependence between node attributes and graph topology.

5.6 Summary

In the graph model, as traditional feature learning methods are often accompanied by information loss in the process of feature learning, and the information taken into consideration is relatively one-sided, the

Table 3: Summary of PFM in GL.

Year	Conference	Method	Pretext Task	Encoder	Code
2014	KDD	DeepWalk [191]	GC-C	Shallow NN	https://github.com/phanein/deepwalk
2015	WWW	LINE [193]	GC-C	Shallow NN	https://github.com/tangjianpu/LINE
2016	NeurIPS	VGAE [194]	GC-C	GCN	-
2016	KDD	node2vec [192]	GC-C	Shallow NN	https://github.com/aditya-grover/node2vec
2017	NeurIPS	GraphSage [214]	GC-C	Shallow NN	https://github.com/williamleif/GraphSAGE
2018	ICLR	DGI [200]	GC-CS	GCN/SAGE	https://github.com/PetarV-/DGI
2020	ICML	GraphCompetition [188]	GIC	GCN	https://github.com/Shen-Lab/SS-GCNs
2020	ICLR	AttMasking [190]	GIC	GCN	http://snap.stanford.edu/gnn-pretrain
2020	ICML	AttributeMask [188]	GIC	GCN	https://github.com/Shen-Lab/SS-GCNs
2020	arXiv	EdgeMask [189]	GIC	GCN	https://github.com/ChandlerBang/SelfTask-GN
2020	arXiv	NodeProperty [189]	GPP-PR	GCN	https://github.com/ChandlerBang/SelfTask-GN
2020	AAAI	M3S [203]	GPP-PC	GCN	-
2020	ICML	Node Clustering [188]	GPP-PC	GCN	https://github.com/Shen-Lab/SS-GCNs
2020	ICML	Graph Partitioning [188]	GPP-PC	GCN	https://github.com/Shen-Lab/SS-GCNs
2020	NeurIPS	CSP [204]	GPP-PC	GCN	-
2020	NeurIPS	MP [204]	GPP-PC	GCN	-
2020	NeurIPS	SELAR [215]	GC-C	GNN	https://github.com/mlvlab/SELAR
2020	KDD	GCC [195]	GC-S	GIN	https://github.com/THUDM/GCC
2020	NeurIPS	GraphCL [197]	GC-S	GCN	https://github.com/CRIpac-DIG/GCA
2020	ICML	CMVRL [201]	GC-CS	GCN	-
2020	ICDM	SUBG-CON [202]	GC-CS	GCN	https://github.com/yzjiao/Subg-Con
2020	ICLR	InfoGraph [216]	GC-CS	GCN	https://github.com/fanyun-sun/InfoGraph
2020	AAAI	DMGI [217]	GC-CS	GCN	https://github.com/pcy1302/DMGI
2020	arXiv	Graph-Bert [210]	Hybrid	Transformer	https://github.com/jwzhanggy/Graph-Bert
2020	WWW	GMI [211]	Hybrid	GCN	-
2020	KDD	Gpt-GNN [212]	Hybrid	GNN	https://github.com/acbull/GPT-GNN
2021	ICML	JOAO [218]	GC-S	GCN	https://github.com/Shen-Lab/GraphCL_Automated
2021	AAAI	CSSL [219]	GC-S	GCN	https://github.com/UCSD-AI4H/GraphSSL
2021	PAKDD	GIC [198]	GC-CS	GCN	https://github.com/cmavro/Graph-InfoClust-GIC
2021	WWW	SUGAR [199]	GC-CS	GCN	https://github.com/RingBDStack/SUGAR
2021	ICML	GraphLoG [220]	GC-CS	GCN	https://github.com/DeepGraphLearning/GraphLoG
2021	WWW	SLICE [221]	GC-CS	GCN	https://github.com/pnml/SLICE
2021	WSDM	BiGI [222]	GC-CS	GCN	https://github.com/caojiangxia/BiGI
2021	WWW	GCA [196]	GC-S	GCN	https://github.com/CRIpac-DIG/GCA
2021	KDD	HeCo [223]	GC-CS	GCN	https://github.com/liun-online/HeCo
2021	AAAI	CG ³ [209]	Hybrid	GCN	-
2021	ICLR	SuperGAT [224]	GC-C	GAT	https://github.com/dongkwan-kim/SuperGAT
2021	KDD	MoCL [225]	Hybrid	GNN	https://github.com/illidanlab/MoCL-DK
2022	ArXiv	MGAE [205]	Maksed Edge Reconstruction	GCN	-
2022	KDD	GMAE [206]	Maksed Node Reconstruction	Transformer	https://github.com/THUDM/GraphMAE
2022	Arxiv	MaskGAE [207]	Partial Maksed Node Reconstruction	Transformer	https://github.com/EdisonLeeeee/MaskGAE
2022	Arxiv	HGMAE [208]	Metapath Masking Reconstruction	Transformer	-

obtained graph representation is relatively rough and loses mass information. People began to focus on the distribution of data and attributes in the graph data as self-supervised signals to pretrain the graph model so that it can capture more valuable information. By transforming the distribution of nodes, attributes, and edges in the graph into different pretext tasks, and using GNNs for modeling, the graph model can fully fit the original distribution of the input graph. In lots of unsupervised or semi-supervised scenarios, such pretrained graph models have been proven to benefit downstream tasks. Besides, federated training large graph models [213] can be a promising solution for building pretrained foundation models. Currently, with the in-depth study of contrastive learning strategies, some work has attempted to apply contrastive learning in different forms to the pretraining of graph models. Through the consistency analysis of context, self, and cross-scale, this kind of method greatly improves the performance of the pretrained graph model on different graphs.

6 PFMs for Other Data Modality

With the rapid development of the PFMs, except for text, image, and graph, the PFMs have also carried out a lot of research on speech, video, text-image, and cross-data. Besides, researchers have started investigating the unified PFMs that incorporate all three mentioned domains recently. Therefore, in this section, we introduce some other advanced and unified PFMs.

6.1 PFM_s for Speech

In the field of speech, wav2vec [226] obtains speech representation by capturing contextual information on large-scale unlabeled datasets, and fine-tuning on a few samples by noise comparison binary classification task, which greatly improves the performance of downstream tasks. Furthermore, vq-wav2vec [227] and wav2vec 2.0 [228] propose a discrete unsupervised pretraining method on the basis of wav2vec, discretizing the original continuous speech signal, so that the methods in the mature NLP community can be migrated and applied. Meanwhile, lots of research have tried to design different mechanisms to use the representation obtained by speech pretraining as the initial input, and apply it to different tasks, e.g., automatic speech recognition [229, 228], phoneme recognition [230], and speech synthesis [231]. In particular, the extensive application of spoken language understanding has promoted the research of joint pretraining of speech and text. For example, SpeechBERT [229] applies MLM to speech and text pairs to perform representation learning on discrete information. Unlike [232], which relies on a large amount of labeled data for joint pre-training, SPLAT [233] uses unlabeled speech data to pretrain the speech embedding module, and proposes a label-level alignment method suitable for label-level downstream tasks based on sequence alignment. MusicBERT [234] is a pretrained model designed for processing music data. It was developed by training on a vast symbolic music corpus consisting of over one million songs. To improve the pretraining process with symbolic music data, MusicBERT employs several mechanisms, such as OctupleMIDI encoding and a bar-level masking strategy. Huang et al. [235] suggest incorporating a metrical structure in the input data, which allows Transformers to better recognize the hierarchical structure of music at the beat-bar-phrase level. AudioTransformer [236] is a model that enhances the performance of Transformer architectures by implementing certain techniques, such as pooling, which were previously used in convolutional networks. Verma et al. [236] demonstrate how they leverage multi-rate signal processing ideas based on wavelets to improve the Transformer embeddings and obtain better results.

6.2 PFM_s for Video

Video is similar to the RGB features of image and sequence information of the text. Many meaningful explorations in self-supervised video representation learning can not only perform efficiently well in video datasets but also generalize to the learning in other areas. Odd-One-Out Networks (O3N) [237] is a technique that targets to predict the odd video subsequence among real subsequences sampled from a video in a training dataset. The experiments are conducted by using different video-clip encoders for O3N to prove consistent improvements of this pretraining design. Similarly, Shuffle and Learn [238] aims to learn the correct temporal order from a sequence of frames in a video. However, Kim et al. [239] designed a new self-supervised task called Space-Time Cubic Puzzles to train 3D CNNs. This task requires a pretrained backbone to arrange permuted 3D spatiotemporal crops. The performance of downstream tasks proves that effective video representations have been learned while solving such puzzles.

Inspired by the contrastive learning in images, many pretraining models in the video also utilize the contrastive loss to learn video presentations for downstream tasks. Inter-Intra Contrastive (IIC) framework [240] can learn video representations by using positive and negative pairs generated from different videos. Specifically, different modalities in the same video are treated as positive pairs, and video clips from different videos as negative ones. Temporal Contrastive Pretraining (TCP) [241] is another contrastive method based on CPC to learn video representations. Different from the existing GAN-based method that generates future frames for the video directly, TCP can predict the latent representation of future frames of the video, which is better for long-term predictions. Sequence Contrastive Learning (SeCo) [242] is a novel method considering both intra- and inter-frame instance discrimination in sequence order-based task.

6.3 PFM_s for Multimodal

The multimodal PFM among text and image can be divided into two categories: single-stream model and cross-stream model. The single-stream model refers to integrating text information and visual information at the beginning of the model. The Cross-stream model refers to text information and visual information encoded by two independent coding modules, respectively. Then different modal information is fused by mutual attention mechanism.

Single-Stream Model VisualBERT [243] inputs text and images into the model simultaneously, which are aligned and fused using Transformer’s self-attention mechanism. The input of the text is the same as BERT, and the input of the image is the image features extracted by Fasters-RCNN. VisualBERT also does pretraining and then fine-tuning the specific task. It adopts two pretraining tasks, namely MLM and sentence-image prediction, determining whether the input sentence describes the corresponding image. The structure of Unicoder-VL [244] is very similar to VisualBERT, except for the processing of the image. Unicoder-VL extracts the image feature through Faster-RCNN and concatenates the feature with image position-encoding mapping to the same space. It enhances the image label prediction task, which predicts the categories of images. The pretraining task of VL-BERT [245] is the same as Unicoder-VL. The image input of VL-BERT includes four parts: the image region features extracted by Fasters-RCNN, the location of the region in the original image, location coding, fragment encoding, and [IMG] encoding.

Cross-Stream Model In ViLBERT [246], the text and image modes are first encoded separately, and their outputs go through a standard attention module. This module is based on the Transformer structure. Still, in the self-attention mechanism, each module uses its query to calculate attention with the value and key of another module to integrate the information between different modules. The model is pretrained on two tasks. The first task is the mask task, which is the same as BERT. On the image side, the goal of the task is that when the region image is masked, the classification distribution of the output of the model can be as consistent as possible with the output distribution of the model used to extract the region features (such as Faster-RCNN). The second task is the language image matching task. DALL-E is a series of deep learning models developed by OpenAI to generate images from natural language prompts. The first version of DALL-E uses a transformer-based architecture, similar to the one used in the GPT LMs, to process the textual prompts and generate image-like representations. The model is trained on a dataset of images and their associated textual descriptions based on GPT-3. DALL-E 2 [247] is the improved version by employing contrastive language-image pretraining (CLIP) [248] for capturing semantic association between image-text pairs and GLIDE diffusion model [249] for text-conditional image synthesis. Furthermore, GPT-4 is proposed by OpenAI recently. It is a large-scale multimodal model which adopts RLHF and demonstrates human-level performance on various professional and academic benchmarks.

Based on the multi-modal data containing more available information than previous single-modality data, thus the performance of these models gets enhanced by combining with the SSL on the benchmark dataset. Cross and Learn [250] is the first method that reveals crossmodal information as an alternative source of supervision and obtains powerful feature representations from combining crossmodal loss and diversity loss in both RGB and optical flow modalities. Different from the existing methods that learn feature representations from only a single task in cross-domain datasets, Ren and Lee et al. [251] propose a novel deep multi-task network to learn more generalizable visual representations to overcome the domain difference and further utilize the cross-domain information in different tasks. In that paper, the cross-domain datasets are real and synthetic datasets generated by a GAN-based network, while the multiple tasks are the predictions of the surface normal, depth, and instance contour in RGB images. This model performs better

than any previous single-task-based SSL methods by learning general-purpose visual representations from cross-domain multi-task feature learning. Tian et al. [252] believe that a powerful representation is one that models cross-view factors from the perspective of humans view to understand the world. They propose Contrastive Multiview Coding (CMC) to learn a video representation by maximizing mutual information between different views of the same scene.

6.4 PFM for Code Generation

Code generation with LLMs involves using pretrained language models to automatically generate code based on natural language descriptions of a desired program. This approach has the potential to greatly improve the efficiency of software development by reducing the need for manual coding and allowing developers to focus on higher-level tasks.

The technique involves training large-scale language models on vast amounts of natural language text and then fine-tuning the models on specific programming tasks. By inputting natural language descriptions of code, the model can generate code snippets that are syntactically and semantically correct. Code generation with LLMs has been applied in various programming domains, including web development, NLP, and data analysis. The models used for code generation include GPT-4, T5, and Codex, among others. For example, Andrei et al. [253] have investigated and assessed the fine-tuning of transformer models for personalized code generation. Specifically, they have evaluated the effectiveness of various personalization techniques in the domain of generating unit tests for Java methods and learning to personalize for a specific software project. Shailja et al. [254] assess the capacity of LLMs to generate Verilog that is useful. To achieve this, pretrained LLMs are fine-tuned on Verilog datasets collected from GitHub and Verilog textbooks. An evaluation framework is then constructed, consisting of test benches for functional analysis and a flow for testing the syntax of Verilog code generated in response to problems of varying degrees of difficulty. An open-source CodeGen LLM that has undergone fine-tuning has been shown to outperform the current leading commercial Codex LLM. The CodeGen [255] is a group of LLMs that have up to 16.1B parameters and can handle both natural language and programming language data. Additionally, they have released the training library JAX FORMER as open-source. Their work demonstrates that the model can perform as well as the previous state-of-the-art zero-shot Python code generation on HumanEval, showcasing the practical applications of the trained model. Synchromesh, introduced in the study by Poesia et al. [256], adopts a novel approach called Target Similarity Tuning (TST) to retrieve a small set of examples from a training bank. Then, Synchromesh utilizes these examples to train a pretrained language model and generates programs by applying Constrained Semantic Decoding (CSD). CSD is a general framework that can restrict the output to valid programs in the target language. In this work, the authors show that the combined use of CSD and TST results in significant improvements in prediction accuracy, as well as preventing runtime errors.

However, there are still some limitations to code generation with LLMs, such as the models' tendency to generate overly verbose or inefficient code and their inability to handle complex programming tasks. Nevertheless, the technology has shown significant promise and has the potential to revolutionize the software development industry.

6.5 SOTA Unified PFMs

A big convergence of PFMs handling multiple modalities is emerging, such as backbone architecture, pre-training task, and model scaling up [29]. Therefore, many unified PFMs proposed by researchers arise. A unified PFM is a unified model pretrained on unimodal and multimodal data with single or multiple trans-

formers as the backbone, which has the ability to perform a large variety of downstream AI tasks, including unimodal tasks and multimodal tasks. There are currently three types of SOTA unified models based on model architectures. We defined them as the single-transformer model, multi-transformer model, and comb-transformer model. A single-transformer model refers to a PFM model which only has a large-scale transformer as its backbone, whereas a multi-transformer model refers to a PFM model having multiple transformers. A comb-transformer model is the PFM model with the combination of both single and multiple transformer structures.

Single-transformer Model UNITER [257] is a large-scale PFM for joint image-text embedding, which consists of an Image Embedder, a Text Embedder, and a multi-layer Transformer. It first encodes visual features and bounding box features for image regions using Image Embedder and tokens and positions using Text Embedder. Then, a Transformer module is applied to learn generalizable contextualized embeddings for images and text through four pretraining tasks. Instead of applying random joint masking to both modalities, conditional masking on pretraining tasks is used. Six vision-language tasks are selected as the downstream tasks.

Uni-Perceiver [258] is a single siamese model with shared parameters having the ability to process different modalities regarding vision and language tasks. Different task inputs and targets are encoded into unified token sequences with modality-specific tokenizers, which are then decoded by a modality-agnostic weight-sharing Transformer encoder into the shared representation space. Any perception task is modeled as finding the maximum likelihood target for each input through the similarity of their representations. Uni-Perceiver is pretrained on unimodal and multimodal tasks. The evaluation results on various downstream tasks show that the performance is close to SOTA methods by conducting prompt tuning on 1% of downstream task data.

Gato [259] builds a single large transformer sequence model that works as a multimodal, multi-task, multi-embodiment generalist policy. It can perform various tasks using a single neural network with the same set of weights. Gato is trained on 604 tasks, where different types of data, such as images, text, proprioception, joint torques, and other discrete and continuous observations and actions, are serialized into a flat sequence of tokens, batched, and processed by the transformer. During deployment, sampled tokens are assembled into different actions based on the context.

OFA [26] is a simple sequence-to-sequence learning framework with a unified instruction-based task representation that unifies various tasks. In the pretraining and finetuning stages, OFA requires no extra task-specific layers for downstream tasks to achieve Task-Agnostic. The Modality-Agnostic compute engine is a Transformer with the constraint that no learnable task- or modality-specific components are added to downstream tasks. OFA is pretrained on small-size image-text pairs to achieve crossmodal tasks while attaining highly competitive performances on unimodal tasks.

UNIFIED-IO [27] is a sequence-to-sequence model using a unified architecture that performs large and diverse tasks. UNIFIED-IO is a transformer model where both the encoder and decoder are composed of stacked transformer layers. The unified architecture does not need specific task or modality branches, which is accomplished by homogenizing the input and output of each task into a sequence of discrete vocabulary tokens. It trains a single transformer-based architecture on over 90 diverse datasets in the vision and language fields. UNIFIED-IO is the first model to perform various tasks and produce strong results across 16 diverse benchmarks without finetuning.

BEiT-3 [29] is a general-purpose multimodal pretrained model on language, vision, and vision-language tasks. The big convergence of BEiT-3 can be seen from three aspects, including backbone architecture, pretraining task, and model scaling up. It introduces a shared Multiway Transformer as backbone network

performing masked data modeling on both unimodal and multimodal data. To process different modalities, every Multiway Transformer block has a shared self-attention module, and a pool of feed-forward networks. It is a giant-size foundation model that contains 1.9B parameters. Experimental results show that BEIT-3 can outperform SOTA models on both vision and vision-language tasks.

Multi-transformer Model FLAVA [28] is an alignment model that targets all modalities at once and aims at solving vision and language tasks, and vision-language tasks. It utilizes a common transformer model architecture to learn strong representations from unimodal and multimodal data. An image encoder transformer is used to capture unimodal image representations. A text encoder transformer is adopted to process unimodal text information. A multimodal encoder transformer takes both encoded unimodal images and text as inputs and integrates their representations for multimodal reasoning. During pretraining, masked image modeling (MIM) and MLM losses are applied to the image and text encoders, respectively. On the other hand, masked multimodal modeling (MMM) and image-text matching (ITM) loss are used over paired image-text data. For downstream tasks, classification heads are applied to the outputs from the image, text, and multimodal encoders, respectively, for visual recognition, language understanding, and multimodal reasoning tasks. FLAVA shows good performance on 35 tasks across different domains. A noticeable advantage is that smaller datasets it used compared with other models.

Comb-transformer Model UNIMO [260] can learn both single modality and multimodalities with one model to achieve robust and generalizable representations. It employs multi-layer self-attention Transformers to learn general textual and visual representations simultaneously and unifies them into the same semantic space via cross-modal contrastive learning (CMCL). The main idea behind CMCL is to keep paired image and text representations close to the representation space while keeping non-paired representations far away. All of them are encoded by the same unified-modal Transformer in pairs or individually, and the representations of images and texts are extracted to compute the contrastive loss.

7 Other Advanced Topics on PFMs

As the number of parameters of the pretraining model increases, the pretraining model requires more memory and computing resources. It increases the training cost of PFMs and limits their deployment on resource-constrained devices. Therefore, to improve the efficiency of the pretraining model, PFM improves computational efficiency from the following two aspects: model efficiency and model compression. The model efficiency and compression of the PFM refer to simplifying the redundancy of model parameters and structure. Under the condition that the task completion degree is not affected, the model with fewer parameters and a more concise structure is obtained.

7.1 Model Efficiency

Model efficiency devotes to exploring more efficient pretraining methods to pretrain large-scale PFMs with a lower-cost solution. More efficient learning algorithms require more effective training methods and more efficient model architecture. Traditional pretraining tasks may be inefficient. For example, the commonly used masked token prediction task requires the model to predict masked tokens based on context. However, the masked tokens in the samples are usually a subset of the input tokens, and the model can only learn from this part of the tokens, so the training efficiency is low. To solve this problem, ELECTRA [30] proposes an RTD task that predicts whether each input marker is replaced by other tokens, which enables the ELECTRA

to train against all input tokens. In addition to effective training methods, more efficient architecture can also improve the efficiency of PFMS. For most PFMS based on the Transformer algorithm, a more efficient model architecture can be obtained by reducing the complexity of the Transformer algorithm.

7.2 Model Compression

Model compression requires less computing resources and memory. It is a potential approach to reduce the model size and enhance computation efficiency. The model compression strategy can be divided into two ways: parameter compression and structure compression.

The methods of parameter compression include parameter pruning, parameter quantization, low-rank decomposition, and parameter sharing. Parameter pruning refers to designing evaluation criteria for model parameters to delete redundant parameters based on a sizeable PFM. For example, Compressing BERT [35] prunes BERT before training while maintaining the performance equivalent to that of the original model. Parameter quantization is the quantization of model parameters from 32-bit full-precision floating-point numbers to lower-order numbers. For example, Q8BERT [84] uses 8-bit quantization to compress parameters fourfold with little impact on model performance. Low-rank decomposition is to reduce the dimension of a high-dimensional parameter vector into a sparse low-dimensional vector. Parameter sharing refers to the structured matrix or clustering methods to map model parameters and reduce the number of parameters. For example, the ALBERT [36] uses decomposition-embedded parameterization and cross-layer parameter sharing to reduce the parameters in the model.

Structure compression refers to compact networks and knowledge distillation. A compact network means reducing the number of parameters and calculations by designing a new compact network structure. Knowledge distillation refers to the transfer of knowledge from the larger teacher model to the smaller student model through the use of a soft label, etc. DistilBERT [261], for example, uses the knowledge distillation method to compress BERT, reducing the size of the BERT model by 40% while retaining 97% of its language comprehension.

7.3 Security and Privacy

The security risks, social bias, and data privacy in PFMs become an important research topic. Qiu et al. [5] recognize that deep neural networks can be attacked by adversarial samples, which mislead the model to produce false predictions. Due to the excellent portability of pretraining models, they have been widely used in NLP, CV, and GL. However, it has been found that the pretraining model is susceptible to the influence of adversarial samples. A tiny interference of the original input may mislead the pretraining model to produce specific false predictions. Meanwhile, it is possible to recover the data samples by querying the PFMs which can cause privacy leakage.

Generation Adversarial Samples The adversarial sample originates from the image. The adversarial samples of the image are hard to recognize with an invisible change. For example, only one pixel of the image is modified. Human beings do not easily detect such disturbance, but the neural network can identify the modified image, which is the original purpose of the adversarial sample. Some work has found that pre-trained LMs are vulnerable in some scenarios. Jin et al. [262] successfully attack the three target models of BERT, CNN, and RNN by generating natural adversarial samples, which indicates that the current language processing model still has a large room for improvement in terms of security. However, it is difficult to achieve due to the distinct discreteness of languages in NLP. In particular, the generation of adversarial sam-

ples in the text must take into account linguistic characteristics to ensure that the sample’s syntax and fluency are not harmed while affecting the model’s output. For example, [263] uses adversarial samples to attack the fine-tuning stage of the BERT model for text classification and entailment successfully. [264] combines the sememe-based word substitution method and search algorithm based on particle swarm optimization to generate adversarial samples.

Model Defects Some unrelated human factors can also mislead the PFM to make wrong predictions. For example, [33] discovers that the performance of BERT is limited in the reasoning task due to utilizing false statistical information in the dataset, which dramatically affects the performance by destroying this property. [265] defines universal adversarial triggers. When triggers are connected to any input, it can induce the model to generate specific predictions.

Backdoor Attacks There are still many methods to manipulate the predicted results of the pretraining model employing a backdoor attack. [266] demonstrates that it is possible to construct a weight poisoning attack in which pretrained weights are injected. After the fine-tuning stage, the backdoor is exposed. Attackers manipulate model predictions easily by injecting arbitrary keywords. [267] shows that PFMs in NLP can be manipulated by modifying the model corpus. The “meaning” of new words or existing words can be controlled by changing their weight parameters.

Defense Against Attacks The human-in-the-loop method [31, 32] has been proposed and applied to generate more natural, efficient, and diversified adversarial samples. Some defense approaches have been proposed to defend against such attacks. [268] designs an auxiliary anomaly detection classifier and uses a multi-task learning procedure to defend against adversarial samples. On the other hand, some defects in the PFM may be inherited by the custom models in transfer learning, such as the adversarial vulnerabilities and backdoors mentioned above. To mitigate this issue, [269] proposes a relevant model slicing technique to reduce defect inheritance during transfer learning while retaining useful knowledge from the PFM.

Data Privacy in PFMs LLMs and other PFMs have been trained on private datasets [270]. The researchers have discovered that by querying the massive LMs, it is feasible to recover specific training samples. An adversary may, for instance, obtain IRC discussions and personally identifiable information. Even worse, because large models have so many parameters, it is simple for PFM to memorize or learn private information, making larger models more prone to attack than smaller ones. Many PFMs such as the LLMs have been trained on private datasets. The researchers have found that it is possible to recover individual training examples by querying the LLMs. For instance, an adversary can extract examples including personally identifiable information, and Internet Relay Chat (IRC) conversations. Even worse, because of the billion parameters of large models, it is easy for PFM to learn private information, making the larger model more vulnerable than smaller models. We must take privacy-preserving measures into account during all PFM processes, including data processing, model training, model inference, and system deployment, in order to reduce the risks of privacy leakage.

8 Future Research Challenges and Open Problems

The PFM can avoid training models from the scratch, which is a breakthrough from weak AI to general AI. At present, due to the characteristics of PFM such as large-scale parameters, a large amount of training data,

and high computational complexity, there are still many technical challenges in PFM. We summarize the future research challenges of PFM from four perspectives: data, foundation, model design, and upstream and downstream tasks. Meanwhile, we point out some open problems in the future research direction.

8.1 Challenges on Data

Most pretrained datasets are for single modes and single languages. It is very important for the development of PFM to construct pretrained datasets for multimodal, multi-lingual, and graph data. For the characteristics of these data, the existing technical challenges are as follows:

Data Deficiencies Unlike NLP and CV, except for the reusable nodes in a few molecular and protein networks, most of the nodes and edges in the graph data do not have a large amount of unlabeled data for pretraining. Meanwhile, the pretraining research of the graph model is still in its initial state. Besides, data from the Internet of Things (IoT) will be enormous and contains rich physical world information. For example, inertial measurement unit sensor data can capture users' social activity information [271, 272]. The theoretical basis, various definitions of the pretext task, and the augmented design of contrastive learning are all imperfect, and new research urgently needs to be supplemented.

Multimodal PFM Some research work has been done on multimodal PFM, such as text and image, text and audio, etc. These are mostly PFM between two modalities. At present, the learning of multimodal PFM requires new multimodal datasets, which demand the establishment of the data between different modes. Thus, the construction of multimodal datasets is also an urgent problem to be solved.

Multi-lingual PFM The multi-lingual PFM solves the resource shortage problem in multiple languages, and it aids in the achievement of new improvements in QA, text summarization, low-resource neural machine translation, and so on. However, the current PFM is still a mask LM. To improve the performance of the multi-LM, some suitable new tasks need to be added. In addition, multi-lingual vocabularies are much larger than single-language vocabularies, resulting in a sharp increase in model parameters to be learned.

8.2 Challenges on Foundation

For a PFM, a theoretical foundation is essential to model performance, whether it is a “black box” or “white box” method. The foundation studied mainly includes theoretical foundation, semantic understanding, and explicable exploration.

Lack of Theoretical Foundation SSL in CV learns the experience from the NLP. There is no profound theory to support all kinds of tentative experiments, and further exploration has no handbook. Although there are several theoretical analysis that tries to understand the collapse of pretraining or the generalization ability of the learning representation, the lack of theoretical foundation is still a huge cloud upon the head of SSL.

Semantic Understanding Does the pretrained LM learn the meaning of the language, or just rely on corpus learning? Many models perform well on various datasets with helpful information that can be extracted, where some approaches even exceed human levels. However, the performance is poor on domain datasets or

relatively small datasets. The models cannot reach a better level of stability and match different downstream tasks. This means that the model cannot serve the real purpose of human language use.

8.3 Challenges on Model Design

Most existing structures of PFM are tried for text, image, and graph. The primary method is to increase data, improve computation power, and design training procedures to achieve better results. How to make a trade-off between data, computing resources, and predictive performance is worth studying.

Model Variety There are many attempts at model design, such as generation-based models in the CV area. However, GAN-based approaches are not popular for the following two reasons: 1) the discriminator has learned meaningful feature representations, but they are forgotten during training [273]; 2) the mode collapse causes the generator to output samples in singular mode to cheat the discriminator. As a result, although researchers attempt to apply GAN-based approaches on SSL for pretraining, the difficulties in the convergence of discriminator and divergence of generator hinder development and progress in this area.

Model Compression With the wide application of the Transformer and the pretraining model showing a general trend of growth, the computational complexity of the pretraining model has become the focus of attention. Due to the huge hardware requirements of model training and other reasons, the high threshold makes it difficult for researchers to train from scratch. BERT-base and GPT-3 contain about 108 million parameters and 175 billion parameters, respectively. It is not conducive to the development of relevant research work. There are some works for pretraining model compression, such as ALBERT having fewer parameters and better effect than BERT-base. The improvement models still require powerful computing equipment, making them difficult to apply universally. Reducing the high computing cost is one of the main challenges in future research.

Model Robustness Although many researchers have designed different pretext tasks for the pretraining, the main problem remains on how to design robust pretext tasks and judge the performance before large-scale computations. In addition, how to compare these proposed methods fairly is also a big issue. As for NLP, deep neural networks are vulnerable to adversarial inputs because of their linear characteristics. Although pretraining models perform well on different NLP tasks, most are based on deep neural networks, which generally have poor robustness. Operations such as cutting and rotating do not change the nature of the image in CV. In contrast, operations such as adding, deleting, and substituting a word in the text are likely to affect the semantics of the text. Therefore, how to improve the robustness of the PFM in NLP is a technical challenge.

Model Anti-attack The PFMs are vulnerable to attack by adversarial examples, which can easily mislead the model to produce specific false predictions. It is difficult to process due to the unique discreteness of language in the NLP field. Thus, the current PFMs have huge room for improvement in model anti-attack.

8.4 Challenges on Finetuning and Prompt

The pretrained model in NLP, CV, and GL fields can achieve good performance in most upstream tasks, but not all good in downstream tasks for fine-tuning and prompt. How to achieve consistent results both on upstream and downstream tasks is still a challenge for the PFMs.

Saturation Phenomena Google Research [274] observed the nonlinear relationship between the performance of upstream and downstream tasks. The higher training accuracy with more data on the upstream tasks does not always lead to better performance on the target downstream tasks. This observation challenges the most intuitive understanding of the pretraining process. Even in the most extreme case, the performance of upstream and downstream is at odds.

Pretext Task There are too many self-supervised tasks, also known as pretext tasks. The pretext task can be used for any downstream tasks, such as detection and classification. It is difficult to match the relationship between pretext tasks and downstream tasks.

Task-based Graph Much of the pretraining on the graph is based on the task graph. Different tasks construct different graphs, where nodes need to be reused. This makes it impossible to pretrain on the graph by introducing as much data as NLP and CV.

8.5 Open Problems for Future PFM

First of all, a big convergence of text, image, graph, and multimodal pretraining is expected. Till the survey is written, no work has considered the graph in their unified PFM. All of the SOTA unified models mainly focus on the language, vision, and language-vision tasks, while neglecting the importance of the graph in the data domain. Second, a unified backbone architecture for unified PFM in future research will become more popular. It can be seen that a unified PFM model which only has a large-scale transformer as its backbone, i.e., a single-transformer model, is more focused by researchers than other types of unified PFM. Third, a unified PFM is expected to achieve SOTA transfer performance for all different tasks in all data domains, including text, image, graph, and multimodalities. Most unified PFM are only outstanding in a single data domain, whereas the performance in other domains is not competitive. BEiT-3 [29] shows a great example in both vision and vision-language tasks towards this research direction. Besides, in terms of RL usage in PFM, even though ChatGPT build the milestone in NLP, CV and GL do not have significant research published yet. More work in this direction is expected in the future.

9 Conclusion

Existing PFM in text, image, and graph domains are principally summarized in this survey. Firstly, we introduce the basic components of NLP, CV, and GL. Then, we provide a summary of existing models designed for pretraining in the three domains and summarize the necessary information in terms of model structures. Furthermore, we study some other research for PFM, including other advanced and unified PFM, model efficiency and compression, and security and privacy. Finally, we present the main challenges and open problems in PFM research.

A Basic Components

A.1 Basic Components on NLP

Table 4: Commonly used notations on NLP and graph.

NLP		Graph	
Notations	Descriptions	Notations	Descriptions
N	The length of input text.	$ \cdot $	The length of a set.
w_i	The i -th word in input text.	\mathcal{G}	The set of graphs.
$ V $	The word corpus size.	G	A graph.
H_x	The representation of the input sequence.	V	The set of nodes in the graph.
$\overrightarrow{\theta}_f$	The parameters for forward modeling.	v	A node.
$\overleftarrow{\theta}_b$	The parameters for backward modeling.	E	The set of edges in the graph.
θ	The shared parameter in all permutations.	e_{ij}	An edge between v_i and v_j .
Z_N	The set of all possible permutations of T .	A	The adjacency matrix of a graph.
$z_{T=t}$	The t -th element of z .	\mathcal{T}	The set of node types in a graph.
$z_{T < t}$	The $[1, 2, \dots, t-1]$ elements of z .	X	The feature matrix of a graph.
z	A permutation of T .	\mathcal{Y}	The set of ground truth labels in a graph.
m	The dimension of the feature vector.	D	The given graph data.
b_1, b_2	The bias values of the hidden layer and the output layer.	M_{GL}	The GL model.

A.1.1 Language Model

With the rapid development of deep learning, LMs are more and more applicable to the pretraining of NLP models. The LM can estimate the probability of rationality of a paragraph of the text. There are two main types of LMs: statistical LM and neural network LM.

Statistical LM The statistical LM is a mathematical model to solve the context-related characteristics of natural language from the perspective of probability and statistics. The core of statistical LMs is to determine the probability of a sentence appearing in a text. As the theoretical basis of the probabilistic LM, the N-gram model profoundly influences the subsequent LM. It plays a pivotal role in the field of the LM. The N-gram LM introduces the Markov hypothesis, which assumes that the probability of the occurrence of the current word only depends on the nearest $n - 1$ words. The maximum likelihood probability of a word w_i can be calculated by

$$p(w_i | w_1, w_2, \dots, w_N) = p(w_i | w_{i-n+1}, w_{i-n+2}, \dots, w_{i-1}) = \frac{C(w_{i-n+1}, w_{i-n+2}, \dots, w_i)}{\sum_N C(w_{i-n+1}, w_{i-n+2}, \dots, w_{i-1})}, \quad (16)$$

where $T = [w_1, w_2, \dots, w_N]$ is the text sequence and $C(w_{i-n+1}, w_{i-n+2}, \dots, w_i)$ is the co-occurrence frequency of $(w_{i-n+1}, w_{i-n+2}, \dots, w_i)$. The $p(w_i | w_1, w_2, \dots, w_N)$ is calculated according to the chain rule

$$p(w_1, w_2, \dots, w_N) = \prod_{i=1}^N p(w_i | w_1, w_2, \dots, w_{i-1}). \quad (17)$$

N-gram uses the probabilities of each word in the sequence to represent the co-occurrence probability of the whole text sequence. When N is large, it indicates a more vital constraint on the occurrence of the next word in the sequence and leads to more sparse frequency information. When N is small, the statistical results have higher reliability and better generalization ability, but the constraint will be weaker.

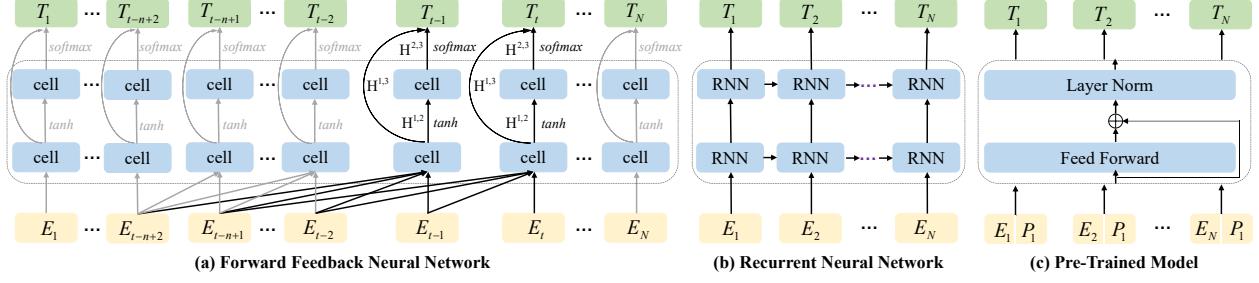


Figure 15: The model architectures of forward feedback neural network, recurrent neural network and Pretrained LMs. $H^{1,2}$, $H^{2,3}$ and $H^{1,3}$ are the weight matrices used to connect each layer.

Neural LM The statistical LM adopts maximum likelihood estimation, which is intuitive and easy to understand. However, there are still problems such as a lack of long-term dependence, the rapid growth of parameter space and sparse data. Therefore, the neural network is introduced to map the LM to a continuous space. Neural LMs use distributed representations of words to model natural language sequences. Unlike class-based N-gram models, neurolinguistic models are able to recognize two similar words without losing the ability to encode each word as different from the other. It can be directly used for NLP tasks. It mainly introduces Forward Feedback Neural Networks (FFNN), Recurrent Neural Networks (RNN), and pretrained LMs.

As shown in Fig. 15 (a), FFNN according to the all former words of $x = [w_1, \dots, w_{i-1}]$ calculates the probability of w_i . In order to predict the conditional probability of w_i , x is sharing the projection matrix $M \in R^{|V| \times m}$ to a continuous feature vector space according to the projection index, $|V|$ is word library size, m is the dimension of the feature vector. The output is represented as

$$y = b_2 + H_x^{1,3} + H_x^{2,3} \tanh(b_1 + H_x^{1,2}), \quad (18)$$

where $H^{1,2}$, $H^{2,3}$ and $H^{1,3}$ are the weight matrices used to connect each layer, and b_1 and b_2 are the bias values of the hidden layer and the output layer respectively.

The structure of the FFNN contains only limited information about the foregoing and has some limitations on the length of the input sequence. Therefore, the RNN LM comes into being. As shown in Fig. 15 (b), RNN can accept input of any variable length. When the input window is moved, its internal state mechanism can avoid repeated calculation, and parameter sharing further reduces the number of model parameters. Therefore, compared with FFNN, RNN has a great advantage.

The pretraining LM is to get a set of model parameters by pretraining some tasks. It initializes the model with these parameters and then trains to improve the model performance effectively. The commonly used pretraining models are fixed embedding (Word2vec [12], Glove [69], etc), variable embedding (Embeddings from LMs (ELMO) [275], Generative Pretrained Transformer (GPT) [50] and Bidirectional Encoder Representations from Transformers (BERT) [13], etc). Here, we give an example of the GPT model, as shown in Fig. 15 (c). It adopts a two-stage process. In the first stage, the Transformer decoder is used as the basic unit of the model to perform text prediction. In the second stage, the GPT is initialized differently for different downstream tasks, training the model and fine-tuning the parameters.

A.2 Basic Components on GL

Due to the extensive use of graph data in many fields, some communities (e.g., chemistry, protein, and social network) have recently focused on the study of graph pretraining. These pretraining models encode

graph attributes, structures, and other information into node representations from multiple perspectives by designing different pretext tasks, which are used to optimize downstream tasks. In this section, we introduce the definition of the basic concepts of graphs, and then provide a formal definition of the PFM on the graph.

A.2.1 Notations and Definitions of Graphs

Unless particularly specified, the notations used in this article are illustrated in Table 4. We use $\mathcal{G} = \{G_i\}_i^N$ to represent a set of graphs, where N represents the number of graphs. Depending to the graph's definition of the edges and nodes, graph data can be classified into the following types.

Definition 1. An **unattributed graph** is $G = (V, E)$, where $v \in V$ is a node, $e \in E$ is an edge, and naturally $E \subseteq V \times V$. Adjacency matrix $A \in \mathbb{R}^{n \times n}$ represents the topology of graph G , where $n = |V|$. $A_{i,j} = 1$ denotes there is an edge between node v_i and v_j , otherwise $A_{i,j} = 0$.

Definition 2. An **attributed graph** is $G = (V, E, X_v, X_e)$, where $X_v \in \mathbb{R}^{n \times d_v}$ and $X_e \in \mathbb{R}^{m \times d_e}$ are the feature matrices of nodes and edges, $|V| = n$, $|E| = m$, d_v and d_e denotes the feature dimensions of node and edge. In fact, in most application scenarios, only nodes have attributes, and edges have no attributes or only weights.

Definition 3. An **undirected graph** is $G = (V, E)$, where $e_{i,j} \in E$ means an unordered node pair (v_i, v_j) . In particular, the adjacency matrix A of the undirected graph is a symmetric matrix (i.e., $A_{i,j} = A_{j,i}$).

Definition 4. A **directed graph** is $G = (V, E)$, where $e_{i,j} \in E$ means an ordered node pair (v_i, v_j) .

Definition 5. G has a node-type mapping function $f_v : V \rightarrow \mathcal{T}^v$ and an edge-type mapping function $f_e : E \rightarrow \mathcal{T}^e$. When $|\mathcal{T}^v| = |\mathcal{T}^e| = 1$, the graph $G = (V, E)$ is a **homogeneous graph**. In other words, all nodes in G belong to a type, and all edges also belong to one type.

Definition 6. When $|\mathcal{T}^v| > 1$ and/or $|\mathcal{T}^e| > 1$, the graph $G = (V, E)$ is a **heterogeneous graph**. In particular, a heterogeneous graph must be an attributed graph.

A.2.2 Learning Settings on Graphs

GL methods are usually used to solve machine learning tasks on graph data, and we introduce different settings (supervision mode and learning mode) for GL.

Before that, we first provide the notations of the corresponding mathematical formulation of GL. $C = \{c_1, c_2, \dots, c_K\}$ is a set of target components defined in a graph set \mathcal{G} ($G^{c_i} \in \mathcal{G}$), and c_i is associated with a corresponding ground truth label $y_i \in \mathcal{Y} = \{1, 2, \dots, N_y\}$, where K denotes the total number of target components, and N_y is the number of classes being predicted. Then the graph data can be represented as $D = \{c_i, G^{c_i}, y_i\}_i^K$, and a complete GL model M_{GL} can also be determined by $y_i = M_{GL}(c_i, G^{c_i})$. For instance, in a node classification task, c_i is the node to be classified, y_i denotes c_i 's label in graph G^{c_i} . Similarly, in a node clustering task, c_i is the node to be clustered, y_i denotes the corresponding cluster label in graph G^{c_i} .

Supervision Mode Depending on the source and scale of the training data, the supervision settings of GL can be divided into four types as shown in Figure 16. **Supervised** GL is the most common mode in the real scenario. Given the target component c_i and the corresponding ground truth label y_i , the goal is to minimize the loss function between the predicted label of the GL model (i.e., $y_i^{pred} = M_{GL}(c_i, G^{c_i})$) and the expected label y_i of all c_i . Compared with supervised learning, **unsupervised** GL refers to situations in which no

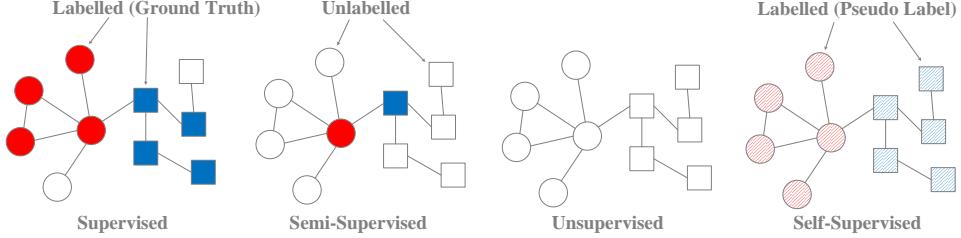


Figure 16: Schematic of different supervision modes.

labeled data is provided, only the attributes and structure distribution of graph data (i.e., (c_i, G^{c_i})) can be used. **Self-supervised** GL is a special case of both supervised and unsupervised learning. Specifically, self-supervised learning mainly uses pretext tasks (e.g., clustering, completion, and partition) to mine its own supervised information (i.e., pseudo-labels) from large-scale unsupervised graph data, and trains the GL model M_{GL} through the self-supervised information, so that it can learn to the valuable features of downstream tasks. In other words, the supervised information of self-supervised learning is not manually labeled, but the pretext tasks automatically construct supervised information from large-scale unsupervised data for supervised learning or training. **Semi-supervised** learning is a combination of unsupervised and supervised learning, who aims at learning data distribution to predict unlabeled data to solve the problem of difficulty in obtaining labeled data in real scenarios. In GL, semi-supervised learning refers to the realization of pattern recognition given a few labeled data and mass unlabeled data.

Learning Mode The GL model M_{GL} is optimized by the given training samples, and adjusted on the validation samples to participate in the test. According to the visibility of the graph data at different stages, the learning settings of GL model M_{GL} can be classified into two categories: inductive learning and transductive learning.

Definition 7. Inductive Learning, which is the most common setting in machine learning tasks, trains the model on labeled data and then tests on samples that have never appeared in the training stage. Formally, given a training sample $\{(c_i, G^{c_i}, y_i)\}_{i=1}^{N_l}, \{(c_j, G^{c_j})\}_{j=1}^{N_u}$, where N_l and N_u are the numbers of labeled/unlabeled samples. Inductive learning learns a function $f^{ind} : \mathcal{G} \mapsto \mathcal{Y}$ so that f^{ind} is expected to be a good classifier on the future graph data $\{(c_k, G^{c_k})\}$, beyond $\{(c_j, G^{c_j})\}_{j=1}^{N_u}$.

Definition 8. Transductive Learning is different from inductive learning in that all samples are visible during both the training and testing stages. Formally, given a training sample $\{(c_i, G^{c_i}, y_i)\}_{i=1}^{N_l}, \{(c_j, G^{c_j})\}_{j=1}^{N_u}$, transductive learning learns a function $f^{trans} : \mathcal{G}^{l+u} \mapsto \mathcal{Y}^{l+u}$ so that f^{trans} is expected to be a good classifier on the unlabeled data $\{(c_j, G^{c_j})\}_{j=1}^{N_u}$.

Under the supervised setting (including semi-/self-supervised), the unified classifier optimization methods of inductive learning and transductive learning can be written as:

$$\mathcal{L} = \frac{1}{K} \sum_{i=1}^K \mathcal{L}(f_\theta^{(\cdot)}(c_i, G^{c_i}), y_i), \quad (19)$$

where \mathcal{L} is the cross-entropy loss, c_i can be node, edge or subgraph of its associated graph G^{c_i} , and $f_\theta^{(\cdot)}$ denotes inductive/transductive function with parameter θ .

Compared with using only one pretext task, some methods have designed some integration mechanisms to incorporate the advantages of multiple pretext tasks into a unified framework.

B Traditional Learning Methods

B.1 Traditional Text Learning

NLP is a research field that integrates linguistics and computer science. Its main research tasks include part-of-speech tagging, named entity recognition, semantic role labeling, machine translation, question answering, sentiment analysis, text summarization, text classification, relationship extraction, event extraction, etc. The LM can be considered the cornerstone of the downstream NLP tasks. It experiences four processes: grammar rule LM, probabilistic LM, neural network LM, and pretraining LM. A PFM trains on a large benchmark dataset to obtain a model which can solve new similar tasks, which has become a new hotspot in current LM research.

Word representations play a significant role in downstream tasks, which is the basis of NLP. The N-gram model preprocesses text features and encodes adjacent N words as a group, which makes it overly dependent on the richness of the training corpus. Otherwise, data-sparse is likely to occur, and the computational complexity will increase exponentially with the increase of N . Neural Network LM (NNLM) [11] adopts the idea of word vector for the first time, and the low-dimensional word vector of distributed representation can solve the discrete problem caused by word embedding well. However, it is still challenging to solve the problem of high computational complexity. The computational complexity of the word2vec model is independent of the selected window size but is determined by the dictionary size and the word vector dimension. Many downstream tasks can be significantly improved by training on a large corpus using word vector embedding after initial training. However, the problem of polysemy for the static word vector is still unsolved, and it still belongs to the shallow LM [276] [277]. Therefore, more effective models are urgently needed to deal with the dataset more flexibly. To capture high-level concepts of context, such as polysemy elimination, syntactic structure, etc. Neelakantan et al. [278] propose to learn multiple embeddings per word type. Zhou et al. [279] integrate the features on both dimensions of the matrix to enrich semantics by using subword information. Based on the Continuous Bag Of Words (CBOW) [12] in word2vec, Hui et al. [280] fine-tune the generated word vectors for emotion and obtain the word vectors containing both semantic meaning and emotional tendency, which significantly improved the performance in the Weibo sentiment classification task. Liu et al. [281] propose a model of hierarchical translation for machine translation. It uses the neural LM based on RNN as the word vector generation model. Liang et al. [282] propose an approach based on the double-layer self-attention mechanism for machine reading comprehension, and the model is divided into three parts: single document encoder, multi-document encoder, and answer prediction. In the single document encoder, the problem of the context information is represented by the Gated Recurrent Unit (GRU) model. Zhang et al. [283] propose an INDependent RNN (INDRNN) and attention mechanism for user intention classification, using word vectors generated by word2vec as input. The model introduces a word-level attention mechanism to effectively quantify the contribution of domain vocabulary to the intention category.

B.2 Traditional Image Learning

There are several types of neural networks in the deep learning era, from the beginning of most famous convolutional neural networks (CNNs) to the subsequent Attention- and Transformer-based networks. A deep neural network refers to an artificial neural network with more hidden layers, and more parameters are used to represent the target model, which leads to the SOTA performance on the benchmark dataset from image to video. Here, we introduce the milestone networks in CV chronologically.

B.2.1 Convolution-Based Networks.

ImageNet [284], as one of the most important databases in computer vision, has aroused many milestone network architectures in image classification, including AlexNet [285], NIN [286], VGG [287], GoogLeNet [288], ResNet [289], DenseNet [290], etc. When it comes to object detection and semantic segmentation, researchers explore R-CNNs [291, 292, 293, 294], FCN [295], SSD [296], YOLOs [297, 298, 299, 300, 301], SegNet [302], PSPNet [303], Deeplabs [304, 305, 306, 307], RefineNet [308], etc. on common benchmark datasets, such as PASCAL VOC [309, 310], MS COCO [311], etc.

There are several shared features among these popular convolution-based networks: 1) *data augmentation*. Deep models require much more data to fit a complicated model, thus the data augmentation technique such as flipping, rotation, cropping, scaling, translation, and even adding noises enlarges the training dataset; 2) *convolution*. The convolutional kernel is used to extract the features of original image data, which maintains the spatial structure for the adjacent pixels; 3) *deep architecture*. The deep architecture contains more parameters, which enhance the capability of the model. These common features contribute to the SOTA performance of convolutional neural networks (CNNs) in computer vision for nearly recent 10 years.

B.2.2 Recurrent neural networks

Different from CNNs targeting 2D-dimensional image applications, recurrent neural networks (RNNs) [312, 313, 314] try to use recursive cells to process pictures in sequence, i.e., video data. However, the weaknesses of gradient explosion and long-term dependencies restrict further development of this model. To handle these problems embedded inside the RNN-based models, long short-term memory (LSTM) [315] was proposed by Hochreiter and Schmidhuber in 1997. In addition, the improved capability of LSTMs produces popularity and attracts attention both in NLP and CV [316, 317, 318, 319, 320].

B.2.3 Generation-Based Networks

Generative Adversarial Networks (GANs) [321] have provided a paradigm to learn representations for unlabelled data, and spawn many GAN-based approaches on downstream tasks. In image translation, pix2pix software [322] first proposes the conditional adversarial networks as a solution to the image-to-image translation problems, and achieves reasonable results on real-world datasets. Markovian Generative Adversarial Networks (MGANs) [323] is a method to generate texture synthesis, which can be applied to style transfer and video stylization. CycleGAN [324] provides a learning algorithm to translate an original image from the source domain to a target domain without containing pairs of images in datasets for supervised learning. StyleGAN [325] is a style-based generator to serve as an alternative architecture for traditional GANs. Pixel Recurrent Neural Networks (PixelRNN) [326] aims to complete images by modeling full dependencies between the color channels. DiscoGAN [327] is designed to learn relations between different domains.

GANs have also provided a novel direction to study data synthesis because it perfectly simulates the distribution of the original data. Laplacian Pyramid of Adversarial Networks (LAPGAN) [328] uses a cascade of convolutional networks to generate images in a coarse-to-fine fashion. Similarly, Stacked Generative Adversarial Networks (SGAN) [329] decompose variations into multiple levels and gradually resolve uncertainties by stacking several GANs in a top-down way.

B.2.4 Attention-Based Networks

Based on the success of CNNs in the area of CV, the attention module is designed to equip with the popular CNNs. For example, SENet [330] proposes a channel attention module, which won first place in the competition of ILSVRC2017. In addition, CBAM [331] sequentially infers attention maps along both channel and spatial dimensions. Many innovative works, such as GCNet [332] and CCNet [333], are inspired by this idea of soft-attention mechanism, which outperforms the traditional CNNs on major benchmarks for both recognition and segmentation tasks. In particular, the self-attention mechanism [334], calculating the response at a position among all entities in a sequence by attending to all positions within the same sequence, is proposed to estimate the relevance of one position to other positions in feature maps. To control the expected entities and model more complex relations among different elements in the sequence, masked self-attention and multi-head attention [38] are the key components proposed to substitute the function of convolutions in the era of transformers.

B.2.5 Transformer-Based Networks

Recently, inspired by the self-attention mechanism and subsequent success of the transformer in NLP, researchers in CV also try to use the transformer as an alternative to the convolution. Self-attention-based transformer models always operate in a two-stage training mechanism: 1) pretraining on a primitive dataset (always big but not well labeled) by defining pretext tasks; 2) transferring the pretrained weights to the downstream tasks and adjusting the parameters on the target domain dataset by finetuning. Vision Transformer (ViT) [40] is applied on CV and achieves the SOTA performance on major benchmark datasets. Data-efficient image Transformers (DeiT) [335] was proposed by Facebook AI to train image transformers more efficiently and maintain the SOTA performance simultaneously. DETection TRansformer (DETR) [336] significantly outperforms competitive baselines in both object detection and semantic segmentation. LeViT [337] outperforms existing benchmarks with respect to balancing the accuracy and training speed. Image GPT [149] is inspired by a sequence transformer in NLP, which can compete with several self-supervised benchmarks on ImageNet. On the basis of this research, DeepViT [338] explores a deeper architecture to improve performance consistently by making the transformer go deeper. Moreover, many researchers try to apply the transformer to more specific tasks. Pyramid Vision Transformer (PVT) [339] introduces the pyramid structure to overcome the difficulties of porting the transformer to various dense prediction tasks, and achieves the SOTA performance on major benchmark datasets. M3DeTR [340] is a novel research on multi-representation, multi-scale, and mutual-relation 3D object detection with transformers. Medical Transformer (MedT) [341] has focused on medical image segmentation and outperforms previous CNN-based and transformer-based architecture. In conclusion, the transformer has become a novel and popular research area in CV and its performance is proved by many existing works.

B.3 Traditional Graph Learning

GL aims to embed the graph as a low-dimensional representation while preserving the desired properties of the original graph data. Classical GL methods are usually implemented using statistical methods or artificially designed components.

Dimension Reduction As a commonly used method in feature engineering, dimension reduction aims to reduce the dimension of high-dimensional attribute graph data into a lower-dimensional representation. In GL, it highlights the remaining information at the cost of losing part of the attributes. According to different dimensionality reduction strategies, such methods can be classified into two types. The first type is subspace

learning under the linear assumption. Based on the assumption that the principal components [342] related to the larger variance represent important structural information, and those smaller variances represent noise, principal component analysis calculates a low-dimensional representation that maximizes the variance of the data. Linear Discriminant Analysis (LDA) [343] achieves dimension reduction by maximizing the ratio of inter-class scattering and intra-class scattering to obtain a linear projection matrix. Multi-Dimensional Scaling (MDS) [344] is a distance-maintaining manifold learning method. It produces a mapping in a lower dimension to preserve dissimilarities between nodes as much as possible. The second type is nonlinear dimension reduction, which aims to automatically learn nonlinear topology to achieve manifold learning. Isomap [345] first constructs a neighborhood graph on the manifold and calculates the shortest path between pairs of nodes, and then uses MDS to construct a low-dimensional embedding. Locally Linear Embedding (LLE) [346] first allocates neighbors for each node. Then, it calculates the weighted $W_{i,j}$, the best linear reconstruction feature X_i from its neighbors. Finally, calculate the low-dimensional embedding for the optimal reconstruction of $W_{i,j}$.

Matrix Factorization Greatly influenced by the idea of dimension reduction, the models based on matrix factorization emerged in the early research of GL. Such models aim to reconstruct the adjacency matrix of the graph to achieve dimension reduction while maintaining structural information. Although these models have significant limitations, in fact, their ideas still inspire many current studies. Depending on how the matrix is constructed, such methods often append specific constraints. Graph Laplacian eigenmaps [347] minimizes a loss function to ensure that nodes close to each other on the manifold are mapped into the low-dimensional space and still maintain the local distances. Node proximity matrix factorization [348] minimizes the objective function $|W - YY^{cT}|$ through matrix factorization to approximate the proximity of nodes in the low-dimensional space, where Y and Y^c are the embeddings for nodes and context nodes, and W is the default node proximity matrix. GraRep [349] aims to preserve the high-order proximity of graphs in the embedding space, thus it derives a k -th order transition matrix, A^k , by multiplying the adjacency matrix to itself k times. The transition probability from node v_i to node v_j is the entry in the i -th row and j -th column of the k -th order transition matrix, i.e., $p_k(v_i|v_j) = A_{i,j}^k$. Then GraRep defines the loss function using the skip-gram model and negative sampling. To capture the high-order proximity between node pairs, HOPE [350] preserves asymmetric transitivity in approximating the high-order proximity. Specifically, the goal of HOPE is to minimize the objective function $\|S - WC^T\|_F^2$, where the elements $s_{i,j} \in S$ represent a certain edge feature (e.g., Katz index, the Rooted Page-Rank, the Common Neighbors, and the Adamic-Adar) between the corresponding node pairs (v_i, v_j) , W is the node representation matrix, and C is the embedding of the node as the context. To reconstruct the matrix S more simply and elegantly, HOPE proposes to obtain W and C directly based on the low-rank singular value decomposition (SVD).

Graph Kernel The kernel method is an important algorithm in pattern recognition and machine learning. Its basic idea is to give the graph embedding $x \in X$ in the original low-dimensional space X , and maps the embeddings to a high-dimensional feature space H through a nonlinear function f^{ker} . Then the nonlinear problem in X can be solved by constructing a linear algorithm in H . There are two main types of kernel methods on graph data. The first type uses the embedding method to convert the graph data into vectorial representation, and then directly implements the application based on the kernel function. However, due to the loss of mass graph structure information when transforming graphs into vectorial representation, such methods do not perform well in real scenarios. The second type of method introduces the graph kernel function to solve this problem. Based on retaining the advantages of the original kernel function, it directly represents the structural information of the graph data in the high-dimensional Hilbert space. The definition of the traditional method of graph kernel comes from R-convolution. According to the difference between the contrast substructure and the decomposition method of the graph structure, a large number of methods based on graph kernel have been proposed. For example, the work of [351, 352] proposed a random-walk

kernel based on calculating the number of common synchronization between two graph structures, To reduce the computational complexity and optimize the random walk strategy, a graph kernel based on comparing the shortest path information between two graph structures is proposed. To capture more complex topological information, the Weisfeiler-Lehman subtree graph kernel is proposed, which is based on a one-dimensional Weisfeiler-Lehman isomorphism test algorithm to find isomorphic subtree structures in a bunch of graph structures [353].

C PFM_s Theory

Since pretraining has received great attention from the research community, the investigation in the theory-backed explanation is similarly eye-catching. During the unsupervised pretraining era before SSL, Erhan et al. [354, 355] shed some light on the theoretical explanation for the confirmation and clarity of learning difficulties. [354] researches the influence of pretraining with respect to architecture depth, model capacity, and the number of training samples, and demonstrates the robustness of pretraining from the perspective of both the optimization and the regularization. [355] further prove the regularizer role of the unsupervised pretraining in the downstream supervised tasks.

C.1 Different Perspectives

Pretext Tasks [356] posits a mechanism based on approximate conditional independence (CI) to connect pretext and downstream task data distributions, which suggests that pretext tasks can self-supervisedly learn the representations from unlabelled data that reduce the sample complexity of downstream supervised tasks. The experiments both on CV and NLP task supports this theory. Representation Learning via Invariant Causal Mechanisms (RELIC) [181] also provides a theoretical understanding from the perspective that the explicit invariance constraints across augmentations can yield improved generalization guarantees.

Multi-View Redundancy From the perspective of a multi-view setting, [357] understands contrastive learning as exploiting multiple views of data for representation learning. This theory provides a theoretical analysis that the linear functions of these representations from pretraining are still competitive compared with the non-linear optimal predictor of the label. In other words, the linear functions of the learned representations are nearly optimal on downstream prediction tasks whenever the different views provide redundant information about the label.

C.2 Different Categories

Contrastive Learning Although experimental results show us that previous designs such as contrastive loss or momentum updating can produce impressive performance in SSL. However, one of the most important questions that remain in SSL is why these methods can maintain representation consistency during the pretraining process. A naive view is the minimization between positive pairs can boost invariance learning, while the maximization between negative pairs contributes to avoiding representational collapse. [358] shows that contrastive learning can achieve competitive bound via intra-class concentration, thus leading to the reduction of sample complexity on downstream tasks from the benefit of transferred representations. This research also provides a framework that can be utilized both on the guarantees of the quality of learning representations during the pretraining phase and the future assumptions added to the framework that allow tighter guarantees.

Non-Contrastive Learning While contrastive learning shows an effect by capturing the similarity and dissimilarity among the unlabelled examples, and further converging to an average local optimum which represents the general representations, recent non-contrastive SSL methods such as BYOL and SimSiam also shows the SOTA performance without the design of comparison between negative pairs. Based on the analysis of the eigenspaces, Tian et al. [182] study the behavior of non-contrastive SSL training and prove that the effects are from both the predictor and stop-gradient signal. Based on this theory, a novel and simple **DirectPred** method is proposed as a by-product of this theoretical exploration.

D Pretext Task Taxonomy on CV

Pretext tasks are always designed to use pseudo labels generated from the data itself to pretrain the proxy model. There are five categories of pretext tasks for self-supervised: 1) generation-based methods; 2) transformation-based methods; 3) context-based methods; 4) semantic-based methods; 5) view-based methods.

Generation-Based Methods This type of method is GAN-based in the deep learning era. For image generation, there are several applications including image colorization [138, 359], image super-resolution [360], image editing [361], context encoders [137], image-to-image translation [324], etc. On the other hand, video generation tasks contains future prediction [145], video action recognition [241], video generation [362, 363], and video representaion [364].

Transformation-Based Methods Transformation is a typical technology that serves as a data augmentation method to enlarge the training dataset in traditional DL. However, if transformations of the same image are labeled as positive samples and others as negative samples, this pretext task can be used for self-supervised pretraining [166]. Popular transformation in self-supervised learning (SSL) contains color transformation (such as Jitter, Gaussian blur, and adjusting brightness) and geometric transformation (such as flipping, cropping, scaling, and rotation).

Context-Based Methods Basically, the design and construction of many artificial tasks, such as solving Jigsaw puzzles [140], comparing context similarity, and discriminating sequence order. Solving Jigsaw puzzles is defined as identifying the correct position of patches from an image. This task can help the model to learn an encoder for transfer learning [365, 141], and the feature representations are effective after the pretrained dataset is big enough. In addition, the design of video Jigsaw is also proposed for unsupervised learning [366]. Differently, context similarity tries to label the patches from the same images as positive samples and others as negative samples, then use a predefined similarity function to scale the distance between different pairs [49].

Semantic-Based Methods Semantic-based methods contain object detection, semantic segmentation, and depth prediction. These tasks also involve pretext tasks because their pixel-based labels can learn a more robust feature representation than simpler tasks. These pre-text tasks always establish on video dataset [367, 368].

View-Based Methods This type of method contains both single-modal data and multi-modal data. For the single-modal data, the original data is treated as the anchor and different viewpoints generate its positive pair samples. Sometimes the time slices in sequence-based data are treated as negative pairs because the scene is changed as time goes [369]. In addition, multi-modal data is usual in view-based methods, which are also called cross-modal-based methods here. Such as audio-video cooperative learning [370], RGB and optical flow cross-modal distance training [250].

E PFM_s for Reinforcement Learning

The success of pretraining learning methods in the supervised learning domain has spurred interest in the reinforcement learning (RL) domain to study whether the same paradigms can be adapted to RL algorithms. General pretraining RL can include broad directions, such as Reward-Free RL [371, 372, 373, 374], Goal-condition RL [375, 376, 377], and Representation Learning in RL [378, 379, 380, 381]. Here we focus the Representation Learning in RL. Specifically, this direction seeks to *improve the performance by pretraining the visual perception competent of RL agent, i.e., the state encoder, with some large-scale datasets using unsupervised/self-supervised data augmentation techniques*. The pretraining process empowers the state encoder to capture the essential structure information from the raw inputs (pixel-level input for CV). An RL policy network is built based on the pretrained state encoder to learn the specific downstream control tasks in the fine-tuning stage. Recent studies have demonstrated that can greatly benefit both in sample efficiency and learning effectiveness from unsupervised [382, 383, 384], semi-supervised [385], and self-supervised [386, 387] learning techniques. Specifically, this direction could be roughly classified into the following two categories: Model-based Pretraining RL and Contrastive-like Pretraining RL.

Model-based Pretraining RL Model-based Pretraining RL aims to first pretrain a generative world model to capture the underlying structure of the environment and then leverage the world model as a state encoder or simulator during fine-tuning. World Models [382] is the first work that proposes to learn a compressed spatial and temporal representation of the environment in an unsupervised manner using a simple Variational Autoencoder, which greatly improves the sample efficiency compared to training from scratch. However, learning the world model without being aware of the environment’s dynamic might lead to ignorance of some key information in the environment. Dreamer [388, 389] proposed to learn latent dynamics by approximating the representation, transition, and reward model. They then train RL agents purely by imagination in a latent space, which is more efficient since it brings a low memory footprint and enables fast predictions of thousands of imagined trajectories in parallel. Furthermore, DreamerPro [390] proposes a reconstruction-free approach based on prototypical representations to migrate the task-irrelevant visual distractions problem in the latent dynamics modeling. DreamerPro significantly outperforms previous SOTA methods when there are complex background distractions. To verify whether learning accurate world models for the real world is promising, Daydreamer [391] applies Dreamer to the real-world physical robots problem and empirically demonstrates significant learning efficiency gains.

Contrastive-like Pretraining RL Contrastive-like Pretraining RL techniques seek to improve the representation ability of state encoders by pretraining the state encoder with a large amount of out-of-domain data or adding some auxiliary loss using unsupervised learning or data augmentation techniques. CURL [392] combines instance contrastive learning and by using MoCo [163] mechanism, which significantly improves the data efficiency of RL agents. Furthermore, RAD [393] proposes an implicit approach that directly trains the RL objective on multiple augmented observations views, which outperforms CURL on some of the environments in the DeepMind Control Suite. Concurrent to RAD, DrQ [394] introduces a simple regularization term, which applies image augmentation to compute current and target Q values. They demonstrate that data efficiency can be significantly improved after applying it to DQN. DrQ-v2 [395] further extends this approach to solve complex humanoid locomotion tasks by inserting similar techniques into the DDPG algorithm. Orthogonal to this direction, [379, 378, 396, 397] demonstrate that pretraining the vision part of RL agent using supervised or unsupervised methods on out-of-domain data can improve the learning efficiency of downstream RL control tasks. Besides ensuring consistency across different views of observation, SPR [381] additionally trains a dynamics model which enforces the representations to be temporally predic-

tive. Based on SPR, SGI [380] proposes to pretrain representations using a combination of latent dynamics modeling, unsupervised goal-conditioned, and inverse dynamics modeling. Compared to previous methods, SGI can better capture the environment’s dynamics and facilitate downstream RL control task training.

F Evaluation Metrics

Classification Task The classification task, according to a labeled training document, determines the relationship between document features and document categories. The learned relationship model is then used to determine the category of new documents.

Accuracy and Error Rate The key metrics for a text classification model are Accuracy and Error Rate. The terms Accuracy and Error Rate are defined as follows:

$$\text{Accuracy} = \frac{(TP + TN)}{N}, \quad (20)$$

$$\text{ErrorRate} = 1 - \text{Accuracy} = \frac{(FP + FN)}{N}, \quad (21)$$

where TP and FP denote true positive and false positive, TN and FN stand for true negative and false negative.

Precision, Recall and F1 Regardless of the standard type and error rate, there are very important metrics used for unbalanced testing sets. These metrics are similar to the concept of the class label in the testing samples. F1 is defined as the harmonic average of Precision and Recall. Thus, Accuracy, Recall, and F1 can be represented as:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad (22)$$

$$F1 = \frac{2\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (23)$$

When the accuracy, F1, and recall values hit 1, the desired results are obtained. On the other hand, when the values turn 0, we get the worst consequence. For the multi-class classification task, the precision and recall values of each class can be determined independently, and then the individual and overall performance can be analyzed.

Micro – F1 The *Micro – F1* [398] is a metric that measures all labels’ overall accuracy and recall. We denote *Micro – F1* as:

$$\text{Micro} - F1 = \frac{2P_t \times R_t}{P + R}, \quad (24)$$

$$P = \frac{\sum_{t \in S} TP_t}{\sum_{t \in S} TP_t + FP_t}, \quad R = \frac{\sum_{t \in S} TP_t}{\sum_{t \in S} TP_t + FN_t}. \quad (25)$$

where TP_t and FP_t mean true and false positive of the t th label on a text.

Macro – F1 The *Macro – F1* calculates the average $F1$ of all labels by giving equal weight to them. *Macro – F1* is denoted as:

$$\text{Macro} - F1 = \frac{1}{S} \sum_{t \in S} \frac{2P_t \times R_t}{P_t + R_t}, \quad (26)$$

$$P_t = \frac{TP_t}{TP_t + FP_t}, \quad R_t = \frac{TP_t}{TP_t + FN_t}. \quad (27)$$

where TN_t and FN_t represent true and false negative of the t th label. S stands for the label set of all samples.

Mean Reciprocal Rank (MRR) The MRR is commonly used to evaluate the performance of ranking algorithms on Question Answering (QA) and Information Retrieval (IR) tasks. MRR is represented as

$$\text{MRR} = \frac{1}{Q} \sum_{i=1}^Q \frac{1}{rank_i}, \quad (28)$$

where $rank_i$ is the ranking of the i th ground-truth answer. The number of predicted labels on each text is denoted by Q . Moreover, there are some metrics, such as EM, Hamming-loss [399], P@K and NDCG@K.

Generation Task Generation task uses LMs to predict the next most likely word or sentence based on input data.

Bilingual EvaLuation Understudy (BLEU) BLEU compares the generated sentences to the reference sentence and makes predictions using automatic machine translation algorithms. The language creation problem is also supported by deep learning technologies such as speech recognition, image caption generation, and text summarization. They can't discover anything better, but it has a few advantages: it's simple to comprehend, correlates well with human judgment, and is language-independent. As a bilingual evaluation aid, BLEU is mainly used to evaluate the quality of machine translation [400]. BLEU compares the degree of overlap between the N-gram in the candidate text and the N-gram in the reference text. The higher overlap indicates better translation quality. The formula for the computation is:

$$BLEU = BP \times \exp \left(\sum_{n=1}^N W_n \log P_n \right), \quad (29)$$

where N represents N-gram, BP is penalty factor, P_N is multivariate precision, and $W_N = 1/N$ is the corresponding weight of multivariate precision. r represents the length of the shortest reference translation, and c represents the length of the candidate translation, then the specific calculation method of penalty factor BP is as follows:

$$BP = \begin{cases} 1, & l_t > l_a \\ e^{1-l_a/l_t}, & l_t \leq l_a \end{cases}, \quad (30)$$

where l_t is the number of words in machine translation and l_a is the number of words in reference answer. The penalty factor is mostly used to penalize large gaps between machine and reference translations.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) ROUGE stands for N-gram co-occurrence statistics, which are used in automatic evaluation methods. It is expanded on the similarity of N-grams, which means that an N-gram is a subsequence of the main document text in terms of N words. There are four types of ROUGE, including ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. The first two are commonly used, and the N in rouge-N refers to N-gram, which is calculated similarly to BLEU, except BLEU is based on accuracy, while ROUGE is based on recall. L in ROUGE-L refers to the Longest Common Subsequence, which is calculated as the Longest Common Subsequence between the candidate abstract and the reference abstract. Thus, the longer the length, the higher the score, based on the F value. The calculation

formula of ROUGE-N and ROUGE-L is mainly introduced. The calculation formula of ROUGE-N is as follows:

$$ROUGE - N = \frac{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \{\text{ReferenceSummaries}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)}, \quad (31)$$

where N stands for N-gram, $\text{Count}(\text{gram}_n)$ represents the frequency of occurrence of an N-gram, and $\text{Count}_{\text{match}}(\text{gram}_n)$ represents the frequency of co-occurrence of an N-gram. The calculation formula of ROUGE-L is as follows:

$$ROUGE - L = F_{\text{lcs}} = \frac{(1 + \beta^2) R_{\text{lcs}} P_{\text{lcs}}}{R_{\text{lcs}} + \beta^2 P_{\text{lcs}}}, \quad (32)$$

$$R_{\text{lcs}} = \frac{LCS(X, Y)}{M}, \quad (33)$$

$$P_{\text{lcs}} = \frac{LCS(X, Y)}{N}, \quad (34)$$

where X is the candidate abstract, Y represents the reference abstract, $LCS(X, Y)$ table indicates the length of the Longest Common Subsequence (LCS) of the candidate abstract and references abstract, M stands for the length of reference abstract, and N denotes the length of the candidate abstract. The ROUGE method is characterized by N-gram co-occurrence statistics, based on recall rate (ROUGE-N) and F-value (ROUGE-L). They are often used in text summaries. It is worth noting that ROUGE is word-based correspondence rather than semantic-based correspondence, but this can be mitigated by increasing the number of reference summaries.

METEOR METEOR, also known as an explicitly sorted translation evaluation metric [401], is an improved version of the BLEU standard that aims to address some flaws in the BLEU standard. Using WordNet to calculate matching relationships between specific sequences, synonyms, roots, affixes, and definitions improves BLEU performance and makes it more relevant to manual discrimination. The calculation formula is as follows:

$$METEOR = (1 - Pen) \times F_m, \quad (35)$$

$$F_m = \frac{PR}{\alpha P + (1 - \alpha)R}, \quad (36)$$

$$P = \frac{m}{\sum_k h_k(c_i)}, \quad (37)$$

$$R = \frac{m}{\sum_k h_k(s_{ij})}, \quad (38)$$

where $Pen = \gamma(\frac{ch}{m})^\theta$ is a penalty factor, which punishes the word order in candidate translation that is different from that in reference translation. ch refers to the number of chunks, which are clustered units of matched units adjacent to each other in both the candidate translation and the candidate reference translation. α, β, θ is the adjustable parameter, m is the number of unary groups that can be matched in the candidate translation, c is the length of the candidate translation, $h_k(c_i)$ is the number of occurrences in candidate translations c_i , and $h_k(s_{ij})$ is the number of occurrences in reference translations s_{ij} .

Perplexity Perplexity is also called the degree of confusion [402]. Its core idea is: first, according to the testing sentence, learn a LM P . Then, according to the LM P , the score of the optional sentence is calculated. Finally, the above scores are standardized according to sentence length. The calculation formula is as follows:

$$PPL(W) = P(w_1, w_2, \dots, w_M)^{-\frac{1}{M}}, \quad (39)$$

where W is the candidate translation, M is the length of the candidate translation, P is the LM obtained according to the reference translation, and $P(w_1, w_2, \dots, w_M)$ is the score calculated by the LM for the candidate translation. The Perplexity assessment indicator is based on a LM. The lower the degree of confusion, the better the translation quality, which is often used in machine translation and LMs. Its disadvantages are as follows: the larger the dataset is, the faster the degree of confusion decreases; the punctuation in the data will impact the PPL of the model; and the interference of common words.

G Datasets

G.1 Downstream Tasks and Datasets on NLP

There are many available datasets in the NLP domain, divided according to different tasks. We summarize them in Table 5. It mainly comprises two categories: the task of classification of texts and the task of generating texts. The text classification tasks mainly include Sentiment Analysis (SA), News Classification (NC), Topic Labelling (TL), Natural Language Inference (NLI), Named Entity Recognition (NER), Question Answering (QA), Dialogue Act Classification (DAC), etc. The generation tasks mainly include text summaries and machine translation.

Sentiment Analysis (SA) It consists of judging the emotional polarity and dividing it into several classes. Depending on the granularity of sentiments, the SA is divided into three categories: dichotomy (positive and negative), trichotomy (positive, negative, and neutral), and multiple categories. Here we introduce several datasets in detail.

Stanford sentiment treebank (SST) [473] The dataset is an extension of MR [474]. SST-1 is a version of SST. It is divided into five categories and the number of training texts and testing texts is 8,544 and 2,210, respectively. It also consists of 20 average tokens. The SST-2 [475] contains 9,613 movie reviews including 6,920 training texts, 872 development texts, and 1,821 testing texts.

Semantic textual similarity benchmark (STS-B) [476] It is used in semantic textual similarity tasks organized in the SemEval context between 2012 and 2017 [477]. It consists of text from image titles, news titles and forums. On a scale of 1 to 5, STS-B displays the semantic similarity of two sentences. It includes 5,749 training sets, 1,379 development sets, and 1,377 testing sets.

Multi-Perspective Question Answering (MPQA) [478, 479] This is an opinion dataset which has two categories. It contains 10,606 sentences from various news sources that have been manually annotated for opinions and other private states. It is worth noting that there are 3,311 positive articles and 7,293 negative articles, having no labels for each article.

IMDB reviews [480] The dataset is the world’s most authoritative source for binary sentiment classification of film reviews. The number of content in each class is the same and it can be divided into training and testing sets whose number of comments is 25,000 on average.

News Classification (NC) As one of the most vital information sources, news content exerts a critical effect on people. The NC facilitates users to acquire essential knowledge in real time. Its applications mainly include news topic identification and recommendation of relevant news based on user interests. Here we introduce several datasets in detail.

Table 5: The statistics of the datasets on NLP. For the QA task, the class represents the sum number of candidate answers and the correct answer. For dialogue, class is the number of slots. Length means the average tokens in turn.

Type	Task	Datasets	Class	Length	Number	Related Papers
Classification	Sentiment Analysis	MR	2	20	10662	[403, 404, 405, 406, 407]
		SST-1	5	18	11,855	[408, 403, 409, 410, 411]
		SST-2	2	19	9,613	[408, 403, 412, 413, 13]
		MPQA	2	3	10,606	[414, 403, 415]
	IMDB	2	294		50,000	[416, 417, 412, 413, 418, 14]
News Classification	20NG	20	221		18,846	[419, 420, 421, 406, 422, 279]
	AG News	4	45/7		127,600	[423, 424, 425, 405, 14]
	R8	8	66		7,674	[406, 422, 426]
	R52	52	70		9,100	[406, 422, 426]
Topic Labeling	DBPedia	14	55		630,000	[423, 424, 418, 427]
	Ohsmed	23	136		7,400	[406, 422, 426]
	YahooA	10	112		1,460,000	[423, 428]
Natural Language Inference	SNLI	3	-		570,152	[429, 430, 55, 431, 13, 275]
	MNLI	3	-		433,000	[432, 13, 14, 55, 36]
	QNLI	2	-		115,667	[13, 14, 36]
	WNLI	2	-		852	[431, 36]
	RTE	2	-		5,768	[36]
	SICK	3	-		10,000	[433]
	MSRP	2	-		5,801	[434]
Named Entity Recognition	CoNLL 2003	4	-		2,302	[275, 13, 435, 436, 437, 438]
	OntoNotes 4.0	18	-		-	[439, 440]
	OntoNotes 5.0	18	-		2,945,000	[13, 435, 436, 438]
	MSRA	3	-		-	[439, 13, 440, 438]
	ACE 2004	7	-		443	[441, 442, 443, 444, 438]
	ACE 2005	7	-		437	[441, 442, 443, 445, 438]
	KBP2017	-	-		-	[445, 438]
Question Answering	QQP	2			799,266	[13, 36]
	MRPC	2	-		-	[36]
	SQuAD	-	5,000		5,570	[275, 55, 36]
	RACE	5	-		100,000	[446, 14, 431, 36]
	TREC	6	10		6,400	[404, 412, 425, 279, 405, 427]
	WikiQA	-	873		243	[447, 448]
Dialog Act Classification	DSTC 4	89	-		30,000	[449, 450]
	MRDA	5	-		62,000	[451, 449]
	SwDA	43	-		1,022,000	[449, 452, 453]
Generation	Text Summarization	NYT	-	-	109,910	[454, 455]
		CNN	-	760	92,579	[456, 457, 458, 459, 460]
		Dailymail	-	653	219,506	[461, 457, 454, 462, 459]
		Gigaword	-	-	3,991,000	[463, 457]
	Machine Translation	WMT14	-	-	-	[464, 465]
		WMT16	-	-	-	[466, 465]
		WMT17	-	-	-	[467, 468, 466, 464, 469]
		WMT18	-	-	-	[467, 466, 468]
Dialogue	DSTC2	-	-		3,000	[470]
	MWOZ	35	15.03		10,438	[470, 471, 472]
	GSIM	-	-		3,008	[470]
	OOS	151	-		23,700	[470]

20 Newsgroups (20NG) [481] 20NG is a text dataset derived from newsgroups. There are 20 classes with the same number of articles per class, including 18846 articles in total. The average number of tokens is 221.

AG News [423, 482] This is an academic news search engine, which is divided into four categories. It contains news headlines and introductions. It includes 120,000 training texts and 7,600 testing texts. The number of average tokens is 45/7.

R8 and R52 [483] They come from Reuters [484]. R8 contains 8 classes consisting of 66 average tokens and includes 2,189 and 5,485 testing and training courses. There are 52 classes in R52, which consists of 70 average tokens. It is divided into 6,532 and 2,568 training and testing texts.

Topic Labeling (TL) The task mainly obtains the meaning of the file by defining complex file themes. It is a critical component of topic analysis technology, which aims at simplifying topic analysis by assigning each article to one or more topics. Here, we introduce a few in detail.

DBpedia [485] It is a large-scale multilingual knowledge base generated by Wikipedia's most commonly used information boxes. It releases DBpedia every month, adding or removing classes and attributes in each version. The most popular version of DBpedia has 14 categories, separated into 560,000 training data and 70,000 testing data. The number of average tokens is 55.

Ohsuemed [486] This is a biomedical literature database. The number of texts is 7,400. It has 23 cardiovascular disease categories and consists of 136 average tokens. All texts are medical abstracts that are categorized into one or more classes.

Yahoo answers (YahooA) [423] The dataset is a topic labeling task having 10 categories. The number of average tokens is 136. There are 140,000 training data and 5,000 testing data. Each text in YahooA has question titles, question contexts, and best answers.

Natural Language Inference (NLI) This task is used to forecast whether the meaning of a text can be inferred from another. Interpretation is a broad form of NLI. By comparing the semantic similarity of sentence pairings, it determines whether a sentence is the interpretation of another one. Here we introduce several primary datasets in detail.

The Stanford Natural Language Inference (SNLI) [429] It is commonly used in NLI tasks. It contains 570,152 human-annotated sentence pairs, which are annotated with three sorts of relationships: neutral, derived, and conflicting. Multi-genre Natural Language Inference (MNLI) [487] has 3 categories and consists of 430,000 sentence pairs annotated with textual information, which is usually used in textual inference tasks. Question Natural Language Inference (QNLI) [488], whose task with 2 classes is to determine whether a given text pair is a question-answer. Winograd Natural Language Inference (WNLI) [489] which consists of 2 categories is a dataset that captures the standard reference information between two paragraphs.

Microsoft Research Paraphrase (MSRP) [434] The dataset contains sentence pairs for the text-similarity task, including 1,725 training and 4,076 testing sets. A binary label annotates each pair, discriminating whether they are paraphrases.

Sentences Involving Compositional Knowledge (SICK) [433] It includes nearly 10,000 English sentence pairs, marked with similarity, and the scale range is 1-5. It has neutral, entailment, and contradictory three categories.

Named Entity Recognition (NER) This is a fundamental task of NLP to identify people, places, organizations, and other entities in text. It is a crucial primary tool for many NLP tasks, including information extraction, question answering, semantic parsing, machine translation, etc.

CoNLL 2003 [275] It consists of newswire text from the Reuters RCV1 corpus. It contains four different entity types (Location, Organization, Person, and Miscellaneous) and includes 1,393 English news articles, and 909 German news articles.

OntoNotes 5.0 [13] The dataset consists of 174,5K English, 900K Chinese, and 300K Arabic text data. It comes from telephone conversations, news agencies, radio news, radio conversations, and blogs. It has 18 entity classes containing 11 types, seven values, and 2,945,000 text data.

MSRA [439] This is a Chinese dataset that is obtained from the news domain. It has three types of entities and is used as a shared task on SIGNAN back in 2006.

Question Answering (QA) There are two types of QA systems: the extraction guidance system and the generation guidance system. The extractive QA can be regarded as a particular case of text classification. Here we detail several datasets.

Microsoft Research Paraphrase Corpus (MRPC) [490] It contains 5,800 sentence pairs extracted from Internet news, and the task type is similar to the QQP dataset. Sentence pairs are derived from comments on the same news item and determine whether the two sentences are semantically the same. The assessment criteria were classification accuracy and F1 score.

Stanford Question Answering Dataset (SQuAD) [275] This is a large-scale machine-reading comprehension dataset that contains two tasks. SQuAD 1.1 [488] provides questions and corresponding answers, and the dataset contains 100,000 samples in total, while SQuAD 2.0 [491] adds unanswered questions and expands the scale to 150,000.

RACE [492] The dataset has 5 categories, containing nearly 100,000 questions extracted from middle and high school English tests, with corresponding answers given by experts. The average length of RACE text is more significant than 300, which is longer than other reading comprehension datasets (such as SQuAD) sequences.

Dialog Act Classification (DAC) The dialogue act is a specific verbal component, which marks the dialogue according to the meaning category of the dialogue. DAC categorizes tags according to the meaning of the dialogue to help understand the speaker's intentions.

Dialog State Tracking Challenge 4 (DSTC 4) [450] It belongs to the dialog act classification task and mainly focuses on dialog state tracking on human-human dialogs. It is divided into 89 training classes and contains 24,000 training texts and 6,000 test texts.

ICSI Meeting Recorder Dialog Act (MRDA) [451] It includes about 75 hours of speech from 75 naturally occurring meetings among 53 speakers. The number of categories is 5, and it contains 51,000 training texts, 11,000 test texts, and 11,000 validation texts.

Switchboard Dialog Act (SwDA) [493] The dataset extends the dialogue behavior label with rounds/discourses. The label summarizes the sentence structure, and relevant and pragmatic information of the relevant turn. The SwDA is split into 43 training classes and includes 1,003,000 training texts, 19,000 test texts, and 112,000 validation texts.

Text Summarization Text summarization is a summary of given single or multiple documents. It is kept as concise as possible while ensuring that it reflects the critical content of the original document. It can be divided into extractive summarization and generative summarization. Extractive summarization is generated by extracting and splicing the critical sentences in documents. Generative summarization is generated by a model, which summarizes documents according to the required content expressed in documents.

NYT [454] The dataset comes from the corpus annotated by the New York Times. The named entities are annotated using the Stanford NER tool in conjunction with the Freebase knowledge base. It contains 9,076 articles, with the remaining 100,834 divided into a training set (96,834 examples) and a validation set (4,000 samples).

CNN/Daily Mail [456] It is used for the passage-based question-answering task, and it is popular in assessing ATS systems. The dataset consists of CNN/Daily Mail news stories paired with multi-sentence human-generated summaries. There are 287,226 training instances, 13,368 validation instances, and 11,490 testing instances in total.

Gigaword [463] This is a dataset of English news chapters consisting of nearly 950 pieces. Headlines – stories from multiple sources, including the New York Times – include some articles with a one-sentence, short news feed.

Machine Translation (MT) It refers to the task of translation from one language to another with its semantic equivalence by a computer. There are three categories, rule-based machine translation, statistics-based machine translation, and neural network-based machine translation.

WMT14 [464] It is a grouping of datasets used in the Ninth Workshop on Statistical Machine Translation shared tasks, including a news translation task, a quality estimation task, a metrics task, and a medical text translation task.

WMT16 [465] This dataset is a grouping of datasets used in the First Conference on Machine Translation shared tasks. It has ten shared tasks, including a news translation task, an IT domain translation task, a biomedical translation task, an automatic post-editing task, a metrics task, a quality estimation task, a tuning task, a pronoun translation task, a bilingual document alignment task, and a multimodal translation task.

WMT17 [464] The dataset includes three MT tasks (news, biomedical, and multimodal), an automatic post-editing task, a quality estimation task, a task dedicated to the training of neural MT systems, a task on bandit learning for MT, an automatic post-editing task, and a metrics task.

WMT18 [467] It mainly features six shared tasks: a news translation task, a biomedical translation task, an automatic post-editing task, a metrics task, a quality estimation task, and a multimodal translation task. Participants must evaluate their approaches to the machine translation topic using the standard datasets created for the shared tasks.

Dialogue As an essential way of man-machine interaction, the dialogue system offers a wide range of applications. The existing dialogue systems can be grouped into task-oriented dialogue systems and non-task-oriented dialogue systems from application scenarios. Among them, the non-task type of conversation system can also be called a chatbot.

DSTC2 [470] This is a multi-round dialogue dataset of restaurant reservation fields, including 1,612 training data, 506 verification data, and 1,117 test data. It allows the user's goals to change compared to

DSTC1. DSTC2 is also richer in terms of the conversation state representation, including the slot value pairs of the user’s targets and the ways to find them.

MWOZ [470] It contains 8,420/1,000/1,000 conversations for training, validation, and test sets, respectively. It contains 30 pairs in seven domains being a multi-domain fully-labeled corpus. Every sample includes a goal, multiple user and agent utterances, and annotations regarding slot values.

Out-Of-Scope (OOS) [470] The dataset includes 15,100 training, 3,100 validation, and 5,500 test sets, respectively. It contains 151 intent classes, containing 150 in-scope and one out-of-scope intent. The out-of-scope intent indicates that a user utterance failed to classify to given predefined objectives.

G.2 Downstream Tasks and Datasets on CV

Table 6: The statistics of the datasets used on downstream tasks.

Type	Name	Usage	Domain	Class	Size	Related Papers
Classification	ImageNet	Pretrain & Downstream	-	1000+	1,200,000+	[136, 137, 140, 141, 139, 142, 143, 138, 145, 494, 174, 179, 173, 151, 183, 182, 146, 153] [161, 162, 163, 164, 165, 48, 495, 49, 172, 166, 167, 170, 175, 177, 176, 180, 181, 496]
	CIFAR-10	Downstream	-	10	60,000	[134, 135, 138, 165, 172, 175, 166, 173, 182]
	CIFAR-100	Downstream	-	100	60,000	[165, 175, 166, 173]
	STL-10	Downstream	-	10	6,000	[134, 135, 177, 179, 173, 182]
	Caltech-101	Downstream	object	101	9,146	[134, 135, 165, 166]
	MNIST-10	Downstream	digit	10	60,000	[48, 179]
	SVHN	Downstream	digit	10	73,257	[175]
	Places205	Downstream	scene	205	2,448,873	[138, 139, 142, 161, 162, 49, 172, 494, 175, 167, 173, 174, 496]
	SUN397	Downstream	scene	899	130,519	[166]
	HMDB51	Downstream	action	51	7000	[177]
Detection	UCF101	Downstream	action	101	-	[177]
	Food-101	Downstream	food	101	101,000	[165, 166]
	Birdsnap	Downstream	bird	500	49,829	[166]
	Cars	Downstream	car	196	16,185	[166, 165]
	Aircraft	Downstream	aircraft	102	10,200	[165, 166]
	Pets	Downstream	pet	37	7,400	[165, 166]
	Flowers	Downstream	flower	102	8,189	[165, 166]
	DTD	Downstream	texture	47	5,640	[165, 166]
	iNaturalist2018	Downstream	species	8,000+	450,000+	[162, 167, 174, 496]
	JFT-300M	Pretrain	-	3,000+	300,000,000+	[40, 496]
Segmentation	COCO	Downstream	object	80	200,000	[142, 163, 164, 179, 167, 170, 183, 496]
	VOC07	Downstream	object	20	9,963	[137, 138, 140, 139, 142, 143, 138, 146, 161, 162, 163, 164, 165, 49, 494, 175, 167, 170, 141]
	VOC12	Downstream	object	20	2,913	[137, 138, 140, 139, 142, 49, 141]
	NYU-Depth V2	Downstream	scene	894	1,449	[139, 165, 177]
	VOC11	Downstream	object	20	3,334	[136]
	ADE20K	Downstream	scene	3,688	27,574	[183, 496]
	Cityscapes	Downstream	scene	25	25,000+	[163]
	LVIS	Downstream	vocabulary	1,200+	160,000+	[163]
	DAVIS	Downstream	scene	150	-	[151]
	Paris StreetView	Downstream	scene	-	15,000	[136, 137]
Inpainting	Moving-MNIST	Downstream	digit	-	10,000	[179]
-	YFCC100M	Pretrain	multimedia	-	100,000,000+	[49]

The datasets in CV mainly contain three types from the perspective of tasks: classification, detection, and segmentation. The popular datasets are concluded in Table 6, and some infrequently mentioned datasets in long tails are discussed in the text.

Classification In this part, we first cover the popular large-scale datasets used frequently in both the pretext and downstream tasks. Then the domain datasets only used for the downstream tasks are unfolded.

MNIST [497] It’s a collection of handwritten digits that includes 60,000 samples in training and 10,000 in testing. The images are fixed-size with 28×28 pixels. The pixel values are from 0 to 255.0 in which pixel values smaller than 255.0 can be understood as background (white) and 255 means foreground (black). The labels are from 0 to 9 and only one of these digits exists in an image. Both traditional and deep learning methods are based on this most popular dataset despite advanced methods showing perfect results. Thus, Geoffrey Hinton has described it as “the drosophila of machine learning”.

Street View House Numbers (SVHN) [498] In the domain of digit numbers, it collects real-world digit numbers from house numbers in Google Street View images. It includes 73,257 digits for training, 26,032 digits for testing, and 531,131 additional. All of them are 32×32 color images with both class labels and character-level bounding boxes.

CIFAR [499] As more advanced methods show perfect results on the simple datasets, more sophisticated datasets such as CIFAR-10 and CIFAR-100 are conducted. These two datasets are closer to the real-world object. The CIFAR-10 contains 50,000 training images and 10,000 testing images, with 6,000 images per class and 32×32 pixels in each RGB color image. The CIFAR-100 is similar to the CIFAR-10 but with more detailed label information. There are 100 classes containing 500 training images and 100 testing images in each class. In addition, these 100 "fine" classes are grouped equally into 20 "coarse" classes. Researchers can adapt it to suitable learning methods.

STL-10 [500] Inspired by the CIFAR-10 dataset, STL-10 is another 96×96 color image dataset containing similar 10 real-world classes. Each class has 500 training images and 800 testing images. The biggest difference is that STL-10 has 100,000 unlabeled images for unsupervised learning. More construction information can be seen in [501].

Caltech-101 [502] It collects roughly 300×200 color images of objects belonging to 101 categories, with 40 to 800 images per category and 50 on average. The outlines of the objects in the pictures are annotated for the convenience of different learning methods.

ImageNet [284] This is one of the most popular and large-scale datasets on computer vision. It is built according to the hierarchical structure of WordNet [503]. The full ImageNet dataset contains 14,197,122 images and 21,841 synsets indexed, attaching on average 1,000 images to demonstrate each synset. The most frequently-used subset of ImageNet is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset from 2010 to 2017, containing tasks of classification, localization, and detection. The number of samples in training and testing datasets and the labels of images are determined by the specific task, more details are seen in [504].

HMDB51 [505, 506] In addition to the popular MNIST, there still exist many domain datasets used for the downstream tasks in the classification problem. HMDB51 is an action video database for a total of 7,000 clips in 51 action classes. It contains five types of facial actions and body movements.

UCF101 [507] It is another action video dataset designed for more realistic action recognition. It is an extension of the UCF50 [508] dataset containing only 50 action categories with 101 action categories, collected from YouTube. What makes it a famous recognition dataset is the workshop in ICCV13 with UCF101 as its main competition benchmark.

Food-101 [509] This is a real-world food dataset of 101 food categories, with 750 and 250 images per class in training and testing dataset respectively.

Birdsnap [510] It is a fine-grained visual categorization of birds on a broad scale, with bounding boxes and the locations/annotations of 17 parts in the object. It contains 49,829 images of the 500 most common species in North America, with each species containing 69 to 100 images and most species having 100. In addition, some images are also labeled as male or female, immature or adult, and breeding or non-breeding plumage.

SUN397 To target the scene categorization, the extensive Scene UNderstanding (SUN) database [511, 512] fills the gap of the existing dataset with the limited scope of categories. This database has 899 categories and 130,519 images, and only images with more than 200×200 pixels were kept. SUN397 is a more well-

sampled subset that maintains 397 categories with at least 100 images per category, in which other categories containing relatively few unique photographs are discarded.

Places205 [513] Places205 dataset is another large scale scene dataset consists of 2,448,873 images from 205 scene categories.

Cars [514] The dataset in the domain of cars contains 16,185 color images of 196 classes (at the level of Make, Model, Year) of cars. For convenience, this dataset is split into training and testing sets in roughly equal quantities.

Aircraft [515] It is another fine-grained visual classification designed for aircraft (also known as FGVC-Aircraft). A popular form of this dataset is the fine-grained recognition challenge 2013 (FGComp2013) [516] ran in parallel with the ILSVRC2013. There exist four-level hierarchies: Model, Variant, Family, Manufacturer, from finer to coarser to organize this database. The more detailed information is shown in [517].

Pets [518] It represents The Oxford-IIIT Pet Dataset that collects 37 pet categories with roughly 200 images per category. All images have an associated ground truth annotation of breed for classification, head ROI for detection, and pixel-level trimap for segmentation.

Flowers [519] Similarly, Flowers is another domain dataset in flowers also collected by Oxford; it contains Oxford-17 Flowers of 17 categories and Oxford-102 Flowers of 102 categories.

Describable Textures Dataset (DTD) [520] This is an evolving collection of textural images in the wild, which consists of 5,640 images of 47 categories, with 120 images per category.

iNaturalist2018 [521] It is a large-scale species classification competition conducted on the FGVC5 workshop at CVPR2018. This dataset contains over 8,000 species categories, with more than 450,000 images in the training and validation dataset collected from iNaturalist [522].

JFT-300M [523] JFT-300M is an internal Google dataset introduced by Sun et al [523] and well-known from ViT Model [40]. It is labeled by algorithms that utilize human-computer communications and target classification tasks. This dataset finally contains 300M images with over 1000M labels, thus leading to the multiple labels attached to this large-scale dataset.

Detection The detection is a popular task in the CV, and almost all the research is conducted on COCO and PASCAL VOC datasets.

COCO [311] This is a large-scale dataset for object detection, segmentation, and caption; it contains 330,000 RGB images, with more than 200,000 labeled. There are 1.5 million object instances of 80 object categories involved. Thus, it is one of the most popular benchmark dataset in detection and segmentation in parallel with the following PASCAL VOC.

PASCAL VOC [524] From 2005 through 2012, the dataset has run challenges assessing performance on object class recognition and has provided standardized image datasets for object class recognition. The main datasets used in self-supervised learning are VOC07, VOC11, and VOC12. Main competitions in VOC07 [525] contain classification and detection tasks; both of them consist of 20 objects and contain at least one object in each image. Thus, it is common to use VOC07 to serve as the downstream task for the detection.

Segmentation The segmentation is a semantics-based pixel-level classification. These datasets are difficult to obtain and annotate, thus they are always used as a downstream task.

VOC11 [526] & *VOC12* [527] Both VOC11 and VOC12 contains classification, detection, and segmentation tasks in the main competition, thus leading to the common use of downstream task for the segmentation.

ADE20K [528, 529] It collects 27,574 images from both the SUN and Places205 databases, in which 25,574 for training and 2,000 for testing. All 707,868 objects from 3,688 categories existing in images are annotated. Especially, this dataset contains 193,238 annotated object parts and parts of parts, and additional attributes, annotation time, depth ordering for the benefit of the research community.

NYU-Depth V2 [530] This is a dataset consisting of images and video sequences from 464 indoor scenes that are recorded by both the RGB and Depth cameras from 3 cities. It contains 1,449 images with the ground truth of depth, and the original RGB values are also provided. In addition, there are 407,024 new unlabeled frames and additional class labels for the objects in images.

Cityscapes [531, 532] It is a dataset of urban street scenes from 50 cities with the ground truth of semantic segmentation. The main instances are vehicles, people, and construction. The high-quality dense pixel annotations contain a volume of 5,000 images. In addition to the fine annotations, coarser polygonal annotations are provided for a set of 20,000 images. Moreover, the videos consist of not consistent images with high-quality annotations, and these annotated images with consistently changing views are provided for researchers.

LVIS [533] It is a dataset for large vocabulary instance segmentation. It features that 1) a category or word in one image is related to the only segmentation object; 2) more than 1,200 categories are extracted from roughly 160,000 images; 3) long tails phenomenon exist in these categories; and 4) more than 2,000,000 high-quality instance segmentation masks.

Densely Annotated VIdeo Segmentation (DAVIS) [534] It is a video dataset designed for the in-depth analysis of the SOTA in video object segmentation, in which DAVIS 2017 [535] contains both semi-supervised (human-guided at the testing time) and unsupervised (human non-guided at test time) video sequences with multiple annotated instances.

Others There are many datasets designed for special visual tasks such as inpainting. In addition, this part covers the data collection in the wild.

Paris StreetView [536] The dataset is designed for image inpainting task, which contains 14,900 training images and 100 testing images collected from street views of Paris. This dataset is collected from Google Street View and mainly focuses on the buildings in the city.

Moving-MNIST [537] Based on MNIST, it is a video dataset designed for evaluating sequence prediction or reconstruction, which contains 10,000 sequences. Each video is long of 20 frames and consisted of two digits (possibly overlapped) moving inside a 64×64 patch. The first benchmark is reported on [538] by the method of LSTMs.

Yahoo Flickr Creative Commons 100 Million (YFCC100M) [539, 540] The dataset is the largest public multimedia collection that is allowed to search by users for their own targets; this dataset can browse both images and videos. It is free and for researchers to explore and investigate subsets of the YFCC100M in real time. Subsets of the complete dataset can be retrieved by any keyword search and reviewed directly. In addition, the text information attached to any image or video is abundant, such as containing location information and user tags. Briefly, it is more a multimedia library than a domain dataset.

Data in the Wild More generalized dataset concept in the self-supervised learning era is composed of

multimedia websites, APP, or search engines such as Instagram, Flickr, Google Images, etc. I think pictures in the wild will play a major role in the future study of CV because of the quantity of data, the computation source, and the learning power of PFM.

G.3 Downstream Tasks and Datasets on Graph

The purpose of the pretraining graph model is to improve the performance of downstream tasks. According to the different analysis objects of the downstream tasks, they can be divided into nodes, edges, and graphs. Meanwhile, the PFMs of GL have been widely used in a mass of fields. In this section, we combine the downstream tasks to conduct statistics on the pretraining datasets and the downstream task datasets.

Node-Level Tasks Nodes are the most basic element of the graph, so lots of downstream tasks mainly focus on the analysis of nodes.

Node Classification Node ClassiFication (NCF) is one of the most prevalent graph-based tasks, which has important analytical value in most of the different types of graph data. Different from the pseudo-labels assigned to nodes in the graph in self-supervised methods, the labels in NCF often come from external information such as manual annotation. Based on Definition 7 and 8, NCF can be divided into two types: transductive and inductive according to the visibility during training, verification, and testing. In addition, the result of NCF can be single-label or multi-label according to the mutual exclusion of labels. The statistical results of common NFC datasets are shown in Table 7.

Node Clustering The goal of Node ClusterIng (NCI) is to divide a graph into different classes or clusters according to a certain standard so that the correlation of nodes in the same cluster is as large as possible, and the irrelevance of nodes that are not in the same cluster is also minimized. Although in the above-mentioned pretraining tasks, NCI is used as a pretext task has appeared, NCI can still test pretraining graph models based on other pretext tasks.

Top-K Search The goal of task Top-K Search (TKS) is to search the K nodes with the highest predefined associations for a given node in the graph. Usually, TKS is used for search tasks such as recommendation and alignment. The detailed statistical results of the datasets are shown in Table 7.

Link-Level Tasks The edge is also an important part of the graph structure, which associates independent nodes and is the key to distinguishing graph data from non-relational data. Especially in some specific fields (e.g., molecules, proteins), edges contain real information, so there are various tasks related to edges.

Link Classification Similar to the NCF, the Link Classification (LC) also assigns one or more labels to a given edge. In fact, in LC, the nodes at both ends of the edge are still taken into consideration.

Link Prediction Link Prediction (LP) is a common graph task (e.g., knowledge graph). The goal of LP is to predict edges that are removed or may exist in the graph. Similar to NCI, LP is also one of the pretext tasks in self-supervised learning, and its statistic results as shown in Table 8.

Top-K Recommendation Top-K Recommendation (TKR) is exactly the same as the definition of TKS, the difference lies in the sorting goal.

Graph-Level Tasks The graph-level task generally focuses on the distribution of nodes, edges, and attributes in a given graph, in order to infer the possible properties of the entire graph.

Table 7: The statistics of the datasets for node-level tasks. Homogeneous:Hom, Heterogeneous:Het.

Task	Name	Usage	Source	Type	Nodes	Edges	Class	Features	Related Paper
NCF	Academia	pretrain	Citation	Hom	138K	739K	-	-	[195]
	DBLP (SNAP)	pretrain	Citation	Hom	317K	2M	-	-	[195]
	DBLP (NetRep)	pretrain	Citation	Hom	540K	30M	-	-	[195]
	IMDB	pretrain	Movie	Hom	896K	8M	-	-	[195]
	Facebook	pretrain	Social	Hom	3M	47M	-	-	[195]
	LiveJournal	pretrain	Social	Hom	4M	86M	-	-	[195]
	Cora	Downstream	Citation	Hom	2,708	5,429	7	1,433	[203, 188, 214, 194, 224] [200, 201, 202, 198, 209, 211]
	CiteSeer	Downstream	Citation	Hom	3,327	4,732	6	3,703	[203, 188, 194, 224] [201, 202, 198, 209, 211]
	PubMed	Downstream	Citation	Hom	19K	44K	3	500	[203, 201, 202, 198, 188] [209, 211, 194, 224, 200]
	ACM	Downstream	Citation	Hom	8,994	26K	4	1,902	[215]
NTKS	Cora-Full	Downstream	Citation	Hom	20K	63K	70	500	[224, 541]
	Cora-ML	Downstream	Citation	Hom	2,995	8,158	7	2879	[224]
	Reddit-233K	Downstream	Social	Hom	233K	57M	210	5,414	[189, 214, 201, 202]
	BlogCatalog	Downstream	Social	Hom	10K	334K	39	-	[191, 192]
	YouTube	Downstream	Social	Hom	1M	3M	47	-	[191]
	Reddit-231K	Downstream	Social	Hom	231K	11M	41	602	[542, 543, 200, 211]
	Amazon	Downstream	Social	Het	130M	-	-	-	[212]
	PPI-30K	Downstream	Protein	Het	3,890	77K	50	-	[192, 200]
	PPI-57K	Downstream	Protein	Het	57K	819K	121	50	[542, 224, 543, 202, 211]
	IMDB	Downstream	Movie	Hom	12K	37K	4	1,256	[215]
	Four-Univ	Downstream	Movie	Hom	4,518	3,426	6	2,000	[224]
	Chameleon	Downstream	Web	Hom	2,277	36K	6	500	[224]
	Crocodile	Downstream	Web	Hom	12K	180K	6	500	[224]
	Flickr-89K	Downstream	Web	Hom	89K	450K	7	500	[224, 202]
	ogbn-arxiv	Downstream	Web	Hom	169K	117K	40	128	[224]
	Wiki-CS	Downstream	Web	Hom	12K	277K	10	300	[224, 541]
	DBLP	Downstream	Web	Hom	17K	53K	4	1639	[224, 543]
	Computers	Downstream	Co-purchase	Hom	14K	246K	10	767	[224, 198, 209, 541]
	Photo	Downstream	Co-purchase	Hom	7,650	119K	8	745	[224, 198, 209, 541, 544]
	CS	Downstream	Co-author	Hom	18K	82K	15	500	[224, 198, 209, 541, 544]
	Physics	Downstream	Co-author	Hom	35K	248K	5	500	[224, 198, 541]
	H-index	Downstream	Co-author	Hom	5,000	44K	-	-	[195]
	Flickr-81K	Downstream	Photo	Hom	81K	6M	195	-	[191]
	Wikipedia	Downstream	Word	Hom	4,777	185K	40	-	[192]
	US-Airport	Downstream	Airline	Hom	1,190	13K	-	-	[195]
	OAG	Downstream	Academic	Het	178M	2B	-	-	[212]
NTKS	KDD-ICDM	Downstream	Co-author	Hom	2,867/2,607	7,637/4,774	697	-	[195]
	SIGIR-CIKM	Downstream	Co-author	Hom	2,851/3,548	6,354/7,076	874	-	[195]
	SIGMOD-ICDE	Downstream	Co-author	Hom	2,626/2,559	8,304/6,668	898	-	[195]

Table 8: The statistics of the datasets for LC. Homogeneous:Hom, Heterogeneous:Het.

Name	Usage	Source	Type	Nodes	Edges	Class	Features	Related Paper
Cora	Downstream	Citation	Hom	2,708	5,429	7	1,433	[203, 188, 189, 545, 546, 214, 194, 224, 543, 542] [200, 201, 202, 198, 209, 210, 211, 544]
CiteSeer	Downstream	Citation	Hom	3,327	4,732	6	3,703	[203, 188, 189, 546, 542, 194, 224, 200, 543] [201, 202, 198, 209, 210, 211, 541, 544]
PubMed	Downstream	Citation	Hom	19K	44K	3	500	[203, 188, 189, 545, 546, 542, 194, 224, 543, 200] [201, 202, 198, 209, 210, 211, 544]
ML-100K	Downstream	Movie	Hom	2,625	100K	5	-	[545]
ML-1M	Downstream	Movie	Hom	9,940	1M	5	-	[545]
BlogCatalog-5K	Downstream	Social	Hom	5,196	172K	6	8,189	[542, 211]
Amazon	Downstream	Social	Het	130M	-	-	-	[212]
PPI-57K	Downstream	Protein	Het	57K	819K	121	50	[542, 224, 543, 202, 211]
Flickr-7K	Downstream	Photo	Hom	7,575	240M	9	12,047	[542, 211]
Last-FM	Downstream	Music	Hom	15K	73K	122	-	[215]
Book-Crossing	Downstream	Book	Hom	111K	443K	52	-	[215]
OAG	Downstream	Academic	Het	178M	2B	-	-	[212]

Graph Classification Graph Classification (GC) is commonly used in social, molecular, and protein graph data, which aims to predict the property of the given community, chemical compound, and protein. The statistic results as shown in Table 9.

Table 9: The statistics of the datasets for GC. Homogeneous:Hom, Heterogeneous:Het.

Name	Usage	Source	Type	Graphs	Nodes	Edges	Class	Related Paper
ZINC15	Pretraining	Molecule	Hom	2M	-	-	-	[190, 204]
ChEMBL	Pretraining	Molecule	Hom	456K	-	-	-	[190, 204]
PPI-pre	Pretraining	Protein	Het	395K	-	-	-	[190]
MUTAG	Downstream	Molecule	Hom	188	-	-	2	[190, 547, 201, 216, 199, 218, 225, 548]
PTC	Downstream	Molecule	Hom	344	-	-	2	[190, 547, 201, 216, 199, 548]
BBBP	Downstream	Molecule	Hom	2,039	-	-	2	[190, 204, 549, 218, 220, 225]
Tox21	Downstream	Molecule	Hom	7,831	-	-	24	[190, 204, 549, 218, 220, 225]
ToxCast	Downstream	Molecule	Hom	8,575	-	-	1,234	[190, 204, 549, 218, 220, 225]
SIDER	Downstream	Molecule	Hom	1,427	-	-	54	[190, 204, 549, 218, 220, 225]
ClinTox	Downstream	Molecule	Hom	1,478	-	-	4	[190, 204, 549, 218, 220, 225]
MUV	Downstream	Molecule	Hom	93K	-	-	34	[190, 218, 220]
HIV	Downstream	Molecule	Hom	41K	-	-	2	[190, 549, 218, 220]
BACE	Downstream	Molecule	Hom	1,513	-	-	2	[190, 549, 218, 220, 225]
PPI-88K	Downstream	Protein	Het	88K	-	-	80	[190]
IMDB-M	Downstream	Movie	Hom	1,500	19K	99K	3	[545, 195, 547, 201, 216]
IMDB-B	Downstream	Movie	Hom	1,000	19K	97K	2	[545, 195, 547, 201, 216, 218]
FreeSolv	Downstream	Molecule	Hom	642	-	-	-	[204]
ESOL	Downstream	Molecule	Hom	1,128	-	-	-	[204]
Lipophilicity	Downstream	Molecule	Hom	4,200	-	-	-	[204]
QM7	Downstream	Molecule	Hom	6,830	-	-	-	[204]
QM8	Downstream	Molecule	Hom	22K	-	-	-	[204]
COLLAB	Downstream	Co-author	Hom	5,000	373K	-	3	[195, 547, 218, 548]
RDT-B	Downstream	Co-author	Hom	2,000	859K	-	2	[195, 216, 218, 548]
RDT-M	Downstream	Co-author	Hom	5,000	3M	-	5	[195, 216, 218, 548]
NCII	Downstream	Molecule	Hom	4,110	123K	132K	2	[197, 219, 547, 199, 218, 548]
NCI109	Downstream	Molecule	Hom	4,127	123K	133K	2	[199]
PROTEINS	Downstream	Molecule	Hom	1,113	44K	81K	2	[197, 219, 199, 218, 548]
D&D	Downstream	Molecule	Hom	1,178	335K	843K	2	[199, 218]
Mutagenicity	Downstream	Molecule	Hom	4,337	131K	134K	2	[219]
METR-LA	Downstream	Traffic	Hom	1	207	-	-	[550]

Data Source The PFM of GL have been widely used in a mass of fields. We will descript the details of the pretraining datasets and the downstream task datasets.

Citation and Co-author network A citation is a basic local representation, whose structure reflects the citation relationships of papers in a research direction or field. Specifically, a citation network is a kind of relational data composed of research papers as nodes and citation relations as edges. Among them, the citation network used in the GL model usually comes from local samples of common citation databases, e.g., Cora, Citeseer, and PubMed, and serves as downstream tasks. Similarly, the co-author network is a dataset of scientific collaboration that corresponds to a researcher’s ego network, in which the researcher and their collaborators are nodes and an edge indicates collaboration between two researchers. According to different requirements of downstream tasks, such co-author networks can be used for various tasks, e.g., node classification and graph classification.

Molecular and protein network A molecular network usually refers to a compound composed of atoms and atomic bonds, and predicting the properties of the compound is usually regarded as a graph classification task. For example, MUTAG is a collection of nitroaromatic compounds whose goal is to predict their mutagenicity to *Salmonella typhimurium*. PTC uses a graph to show the structure of multiple compounds and aims to predict the carcinogenicity of different compounds in rats. The protein network is a collection of proteins classified as either enzymes or non-enzymes. The amino acids are represented by nodes, and two nodes are connected by an edge if they are less than 6 Angstroms apart.

Social and Movie network The social network is the social-relational data in the real network environment, which usually represents the relationship between users or posts. For instance, Reddit is a graph dataset comprised of Reddit posts made in September 2014. BlogCatalog is a graph dataset that represents a network of social relationships between bloggers who are listed on the BlogCatalog website. The movie network is usually composed of actors and their co-occurrence participation in the movie. For example, IMDB-B is a movie collaboration dataset that contains a large number of self-networks of actors who play movie roles in IMDB. Nodes in each graph represent actors/actresses, and if they appear in the same film, an edge connects them. These graphs are based on action and romance genres. The difference between IMDB-M and IMDB-B is that a node in the graph represents one or more actors.

Others Some of the rarer graph data are used to test the universality of the PFM, such as word networks (Wikipedia), book networks (Book-crossing), and airline networks (US-Airport). In addition, there are also some special graph structures adapted to specific models, such as spatiotemporal graphs (METR-LA).

References

- [1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, “On the opportunities and risks of foundation models,” *arXiv preprint arXiv:2108.07258*, 2021.
- [2] G. G. Chowdhury, “Natural language processing,” *Annual review of information science and technology*, 2003.
- [3] D. Forsyth and J. Ponce, *Computer vision: A modern approach*. 2011.
- [4] J. A. Bondy, U. S. R. Murty, *et al.*, *Graph theory with applications*. 1976.
- [5] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, “Pre-trained models for natural language processing: A survey,” *Science China Technological Sciences*, 2020.
- [6] J. Li, T. Tang, W. X. Zhao, and J.-R. Wen, “Pretrained language models for text generation: A survey,” *arXiv*, 2021.
- [7] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, *et al.*, “A survey on visual transformer,” *arXiv*, 2020.
- [8] S. Sanchez, H. Romero, and A. Morales, “A review: Comparison of performance metrics of pretrained models for object detection using the tensorflow framework,” in *IOP Conference Series: Materials Science and Engineering*.
- [9] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, “Pre-training graph neural networks,” *arXiv*, 2019.
- [10] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” *Proceedings of the IEEE*, 2020.
- [11] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *J. Mach. Learn. Res.*, 2003.
- [12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” in *Proc. ICLR*, 2013, 2013.

- [13] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *NAACL-HLT*, 2019.
- [14] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” in *NeurIPS*.
- [15] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, *et al.*, “Evaluating large language models trained on code,” *arXiv preprint arXiv:2107.03374*, 2021.
- [16] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy, *et al.*, “Text and code embeddings by contrastive pre-training,” *arXiv preprint arXiv:2201.10005*, 2022.
- [17] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” *Advances in neural information processing systems*, vol. 30, 2017.
- [18] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, “Learning to summarize with human feedback,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.
- [19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, “Training language models to follow instructions with human feedback,” *arXiv preprint arXiv:2203.02155*, 2022.
- [20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, “Language models are few-shot learners,” *arXiv*, 2020.
- [21] B. Lester, R. Al-Rfou, and N. Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021.
- [22] T. Schick and H. Schütze, “Exploiting cloze-questions for few-shot text classification and natural language inference,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, 2021.
- [23] Z. Zhang, A. Zhang, M. Li, and A. Smola, “Automatic chain of thought prompting in large language models,” in *International Conference on Learning Representations*, 2023.
- [24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” in *Advances in Neural Information Processing Systems*, 2022.
- [25] OpenAI, “Gpt-4 technical report,” 2023.
- [26] P. Wang, A. Yang, R. Men, J. Lin, S. Bai, Z. Li, J. Ma, C. Zhou, J. Zhou, and H. Yang, “Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework,” *arXiv preprint arXiv:2202.03052*, 2022.
- [27] J. Lu, C. Clark, R. Zellers, R. Mottaghi, and A. Kembhavi, “Unified-io: A unified model for vision, language, and multi-modal tasks,” *arXiv preprint arXiv:2206.08916*, 2022.

- [28] A. Singh, R. Hu, V. Goswami, G. Couairon, W. Galuba, M. Rohrbach, and D. Kiela, “Flava: A foundational language and vision alignment model,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15638–15650, 2022.
- [29] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som, *et al.*, “Image as a foreign language: Beit pretraining for all vision and vision-language tasks,” *arXiv preprint arXiv:2208.10442*, 2022.
- [30] K. Clark, M. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: pre-training text encoders as discriminators rather than generators,” in *ICLR*, 2020.
- [31] E. Wallace, P. Rodriguez, S. Feng, I. Yamada, and J. Boyd-Graber, “Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering,” 2019.
- [32] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela, “Adversarial NLI: A new benchmark for natural language understanding,” in *ACL*.
- [33] T. Niven and H. Kao, “Probing neural network comprehension of natural language arguments,” in *ACL*.
- [34] G. Wang, N. Ivanov, B. Chen, Q. Wang, and Q. Yan, “Graph learning for interactive threat detection in heterogeneous smart home rule data,” in *2023 ACM SIGMOD International Conference on Management of Data*, ACM, 2023.
- [35] M. A. Gordon, K. Duh, and N. Andrews, “Compressing BERT: studying the effects of weight pruning on transfer learning,” in *RepL4NLP@ACL*.
- [36] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” in *ICLR*, 2020.
- [37] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, L. Zhang, W. Han, M. Huang, *et al.*, “Pre-trained models: Past, present and future,” *AI Open*, 2021.
- [38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv*, 2017.
- [39] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, “Attention mechanisms in computer vision: A survey,” *Computational Visual Media*, vol. 8, no. 3, pp. 331–368, 2022.
- [40] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.
- [41] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, “Graph transformer networks,” *Advances in neural information processing systems*, vol. 32, 2019.
- [42] B. M. P. P. J. H. J. G. A. S. M. C. R. G. I. A. R. J. L. B. M. T. A. A. X. W. C. R. M. M. J. P. U. E. M. K. S. v. S. G. F. E. A. M. F. Y. A. O. F. H. J. B. M. P. C. A. A. G. V. B. C. V. Y. T. T. M. A. K. F. P. D. T. T. K. M. L. X. Z. D. K. J. H. N. H. Mostafa Dehghani, Josip Djolonga, “Scaling vision transformers to 22 billion parameters,” *arXiv preprint arXiv:2302.05442*, 2023.

- [43] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, “Palm: Scaling language modeling with pathways,” *arXiv preprint arXiv:2204.02311*, 2022.
- [44] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, “Spanbert: Improving pre-training by representing and predicting spans,” *Trans. Assoc. Comput. Linguistics*, 2020.
- [45] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in *ACL*.
- [46] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “Electra: Pre-training text encoders as discriminators rather than generators,” *arXiv*, 2020.
- [47] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” *arXiv*, 2019.
- [48] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” *arXiv*, 2016.
- [49] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, “Deep clustering for unsupervised learning of visual features,” in *ECCV*.
- [50] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.
- [51] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI blog*, 2019.
- [52] R. Caruana, “Multitask learning,” *Machine learning*, 1997.
- [53] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” *arXiv*, 2018.
- [54] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv*, 2016.
- [55] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized BERT pretraining approach,” *CoRR*, 2019.
- [56] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” *arXiv*, 2015.
- [57] K. Song, X. Tan, T. Qin, J. Lu, and T. Liu, “Mpnet: Masked and permuted pre-training for language understanding,” in *NeurIPS*, 2020.
- [58] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, “A survey on text classification: From traditional to deep learning,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 2, pp. 1–41, 2022.
- [59] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, “Mass: Masked sequence to sequence pre-training for language generation,” *arXiv*, 2019.

- [60] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, “Unified language model pre-training for natural language understanding and generation,” *arXiv*, 2019.
- [61] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, “Ernie: Enhanced representation through knowledge integration,” *arXiv*, 2019.
- [62] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, “Ernie 2.0: A continual pre-training framework for language understanding,” in *AAAI*.
- [63] Y. Cui, W. Che, T. Liu, B. Qin, and Z. Yang, “Pre-training with whole word masking for chinese BERT,” *T-ASL*, 2021.
- [64] S. Diao, J. Bai, Y. Song, T. Zhang, and Y. Wang, “ZEN: pre-training chinese text encoder enhanced by n-gram representations,” in *EMNLP*.
- [65] H. Tsai, J. Riesa, M. Johnson, N. Arivazhagan, X. Li, and A. Archer, “Small and practical bert models for sequence labeling,” *arXiv*, 2019.
- [66] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [67] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, *et al.*, “Lamda: Language models for dialog applications,” *arXiv preprint arXiv:2201.08239*, 2022.
- [68] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *arXiv*, 2013.
- [69] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *EMNLP*.
- [70] A. M. Dai and Q. V. Le, “Semi-supervised sequence learning,” *arXiv*, 2015.
- [71] P. Liu, X. Qiu, and X. Huang, “Recurrent neural network for text classification with multi-task learning,” *arXiv*, 2016.
- [72] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *TACL*, 2017.
- [73] B. McCann, J. Bradbury, C. Xiong, and R. Socher, “Learned in translation: Contextualized word vectors,” *arXiv*, 2017.
- [74] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” *arXiv*, 2019.
- [75] L. Kong, C. d. M. d’Autume, W. Ling, L. Yu, Z. Dai, and D. Yogatama, “A mutual information maximization perspective of language representation learning,” *arXiv*, 2019.
- [76] W. Wang, B. Bi, M. Yan, C. Wu, Z. Bao, J. Xia, L. Peng, and L. Si, “Structbert: Incorporating language structures into pre-training for deep language understanding,” *arXiv*, 2019.
- [77] W. Xiong, J. Du, W. Y. Wang, and V. Stoyanov, “Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model,” *arXiv*, 2019.

- [78] M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, “Knowledge enhanced contextual word representations,” *arXiv*, 2019.
- [79] H. Huang, Y. Liang, N. Duan, M. Gong, L. Shou, D. Jiang, and M. Zhou, “Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks,” *arXiv*, 2019.
- [80] J. M. Eisenschlos, S. Ruder, P. Czaplak, M. Kardas, S. Gugger, and J. Howard, “Multifit: Efficient multi-lingual language model fine-tuning,” *arXiv*, 2019.
- [81] I. Beltagy, K. Lo, and A. Cohan, “Scibert: A pretrained language model for scientific text,” *arXiv*, 2019.
- [82] S. Sun, Y. Cheng, Z. Gan, and J. Liu, “Patient knowledge distillation for bert model compression,” *arXiv*, 2019.
- [83] G. Lample and A. Conneau, “Cross-lingual language model pretraining,” *arXiv*, 2019.
- [84] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, “Q8BERT: quantized 8bit BERT,” in *EMC2@NeurIPS*.
- [85] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter,” *arXiv*, 2019.
- [86] W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju, “Fastbert: a self-distilling bert with adaptive inference time,” *arXiv*, 2020.
- [87] L. Martin, B. Müller, P. J. O. Suárez, Y. Dupont, L. Romary, É. de la Clergerie, D. Seddah, and B. Sagot, “Camembert: a tasty french language model,” in *ACL*.
- [88] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, “Unsupervised cross-lingual representation learning at scale,” in *ACL*.
- [89] N. Kitaev, L. Kaiser, and A. Levskaya, “Reformer: The efficient transformer,” in *ICLR*, 2020.
- [90] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, “Q-bert: Hessian based ultra low precision quantization of bert,” in *AAAI*.
- [91] Z. Chi, L. Dong, F. Wei, W. Wang, X.-L. Mao, and H. Huang, “Cross-lingual natural language generation via pre-training,” in *AAAI*.
- [92] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang, “K-bert: Enabling language representation with knowledge graph,” in *AAAI*.
- [93] Z. Jiang, W. Yu, D. Zhou, Y. Chen, J. Feng, and S. Yan, “Convbert: Improving BERT with span-based dynamic convolution,” in *NeurIPS*, 2020.
- [94] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, “Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers,” in *NeurIPS*, 2020.
- [95] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, “Multilingual denoising pre-training for neural machine translation,” *Trans. Assoc. Comput. Linguistics*, 2020.

- [96] T. Sun, Y. Shao, X. Qiu, Q. Guo, Y. Hu, X. Huang, and Z. Zhang, “Colake: Contextualized language and knowledge embedding,” in *COLING*.
- [97] H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbé, L. Besacier, and D. Schwab, “Flaubert: Unsupervised language model pre-training for french,” in *LREC*.
- [98] T. Shen, Y. Mao, P. He, G. Long, A. Trischler, and W. Chen, “Exploiting structured knowledge in text via graph-guided representation learning,” in *EMNLP*.
- [99] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, “Tinybert: Distilling BERT for natural language understanding,” in *EMNLP*.
- [100] P. Delobelle, T. Winters, and B. Berendt, “Robbert: a dutch roberta-based language model,” in *EMNLP*.
- [101] B. He, D. Zhou, J. Xiao, X. Jiang, Q. Liu, N. J. Yuan, and T. Xu, “Integrating graph contextualized knowledge into pre-trained language models,” in *EMNLP*.
- [102] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *J. Mach. Learn. Res.*, 2020.
- [103] T. Schick and H. Schütze, “Exploiting cloze-questions for few-shot text classification and natural language inference,” in *EACL*, 2021.
- [104] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang, “KEPLER: A unified model for knowledge embedding and pre-trained language representation,” *Trans. Assoc. Comput. Linguistics*, 2021.
- [105] T. Gao, X. Yao, and D. Chen, “Simcse: Simple contrastive learning of sentence embeddings,” *CoRR*, 2021.
- [106] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, *et al.*, “Glam: Efficient scaling of language models with mixture-of-experts,” in *International Conference on Machine Learning*, pp. 5547–5569, PMLR, 2022.
- [107] Z. Chi, S. Huang, L. Dong, S. Ma, S. Singhal, P. Bajaj, X. Song, and F. Wei, “Xlm-e: Cross-lingual language model pre-training via electra,” *arXiv preprint arXiv:2106.16138*, 2021.
- [108] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, *et al.*, “Multitask prompted training enables zero-shot task generalization,” *arXiv preprint arXiv:2110.08207*, 2021.
- [109] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, *et al.*, “Scaling language models: Methods, analysis & insights from training gopher,” *arXiv preprint arXiv:2112.11446*, 2021.
- [110] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, *et al.*, “Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model,” *arXiv preprint arXiv:2201.11990*, 2022.
- [111] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, “Training compute-optimal large language models,” *arXiv preprint arXiv:2203.15556*, 2022.

- [112] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, *et al.*, “Opt: Open pre-trained transformer language models,” *arXiv preprint arXiv:2205.01068*, 2022.
- [113] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, “Finetuned language models are zero-shot learners,” in *International Conference on Learning Representations*, 2022.
- [114] O. Honovich, T. Scialom, O. Levy, and T. Schick, “Unnatural instructions: Tuning language models with (almost) no human labor,” *arXiv preprint arXiv:2212.09689*, 2022.
- [115] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Naik, A. Ashok, A. S. Dhanasekaran, A. Arunkumar, D. Stap, *et al.*, “Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks,” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, 2022.
- [116] S. Mishra, D. Khashabi, C. Baral, and H. Hajishirzi, “Cross-task generalization via natural language crowdsourcing instructions,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, 2022.
- [117] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, “Self-instruct: Aligning language model with self generated instructions,” *arXiv preprint arXiv:2212.10560*, 2022.
- [118] L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh, *et al.*, “Ethical and social risks of harm from language models,” *arXiv preprint arXiv:2112.04359*, 2021.
- [119] S. Kiegeland and J. Kreutzer, “Revisiting the weaknesses of reinforcement learning for neural machine translation,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1673–1681, 2021.
- [120] N. Jaques, J. H. Shen, A. Ghandeharioun, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard, “Human-centric dialog training via offline reinforcement learning,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3985–4003, 2020.
- [121] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, “Self-critical sequence training for image captioning,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7008–7024, 2017.
- [122] R. Y. Pang and H. He, “Text generation by learning from demonstrations,” in *Proceedings of the international conference on learning representations*, 2021.
- [123] M. Hausknecht, P. Ammanabrolu, M.-A. Côté, and X. Yuan, “Interactive fiction games: A colossal adventure,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 7903–7910, 2020.
- [124] C. Snell, I. Kostrikov, Y. Su, M. Yang, and S. Levine, “Offline rl for natural language generation with implicit language q learning,” *arXiv preprint arXiv:2206.11871*, 2022.
- [125] X. Lu, S. Welleck, L. Jiang, J. Hessel, L. Qin, P. West, P. Ammanabrolu, and Y. Choi, “Quark: Controllable text generation with reinforced unlearning,” *arXiv preprint arXiv:2205.13636*, 2022.
- [126] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, “Survey on reinforcement learning for language processing,” *Artificial Intelligence Review*, pp. 1–33, 2022.

- [127] R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi, and Y. Choi, “Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization,” *arXiv preprint arXiv:2210.01241*, 2022.
- [128] J. Wu, L. Ouyang, D. M. Ziegler, N. Stiennon, R. Lowe, J. Leike, and P. Christiano, “Recursively summarizing books with human feedback,” *arXiv preprint arXiv:2109.10862*, 2021.
- [129] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, *et al.*, “Webgpt: Browser-assisted question-answering with human feedback,” *arXiv preprint arXiv:2112.09332*, 2021.
- [130] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, *et al.*, “Improving alignment of dialogue agents via targeted human judgements,” *arXiv preprint arXiv:2209.14375*, 2022.
- [131] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhosseini, C. McKinnon, *et al.*, “Constitutional ai: Harmlessness from ai feedback,” *arXiv preprint arXiv:2212.08073*, 2022.
- [132] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, *et al.*, “Scaling instruction-finetuned language models,” *arXiv preprint arXiv:2210.11416*, 2022.
- [133] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large language models are zero-shot reasoners,” in *Advances in Neural Information Processing Systems* (A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, eds.), 2022.
- [134] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, “Discriminative unsupervised feature learning with convolutional neural networks,” *Advances in neural information processing systems*, 2014.
- [135] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, “Discriminative unsupervised feature learning with exemplar convolutional neural networks,” *TPAMI*, 2016.
- [136] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *ICCV*.
- [137] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context encoders: Feature learning by inpainting,” in *CVPR*.
- [138] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in *ECCV*, 2016.
- [139] R. Zhang, P. Isola, and A. A. Efros, “Split-brain autoencoders: Unsupervised learning by cross-channel prediction,” in *CVPR*.
- [140] M. Noroozi and P. Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *ECCV*.
- [141] D. Kim, D. Cho, D. Yoo, and I. S. Kweon, “Learning image representations by completing damaged jigsaw puzzles,” in *WACV*.
- [142] M. Noroozi, H. Pirsiavash, and P. Favaro, “Representation learning by learning to count,” in *ICCV*.

- [143] P. Bojanowski and A. Joulin, “Unsupervised learning by predicting noise,” in *ICML*, 2017.
- [144] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” *arXiv*, 2018.
- [145] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv*, 2018.
- [146] O. Henaff, “Data-efficient image recognition with contrastive predictive coding,” in *ICML*, 2020.
- [147] J. Donahue and K. Simonyan, “Large scale adversarial representation learning,” in *NeurIPS*.
- [148] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, “Adversarially learned inference,” *arXiv*, 2016.
- [149] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, “Generative pretraining from pixels,” in *ICML*, 2020.
- [150] H. Bao, L. Dong, S. Piao, and F. Wei, “Beit: Bert pre-training of image transformers,” in *International Conference on Learning Representations*, 2021.
- [151] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.
- [152] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” in *International Conference on Machine Learning*, pp. 8821–8831, PMLR, 2021.
- [153] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, “Unsupervised feature learning via non-parametric instance discrimination,” in *CVPR*.
- [154] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.
- [155] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, “Simmim: A simple framework for masked image modeling,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9653–9663, 2022.
- [156] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, “Segment anything,” *arXiv preprint arXiv:2304.02643*, 2023.
- [157] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.
- [158] X. Li, W. Wang, L. Yang, and J. Yang, “Uniform masking: Enabling mae pre-training for pyramid-based vision transformers with locality,” *arXiv preprint arXiv:2205.10063*, 2022.
- [159] J. Chen, M. Hu, B. Li, and M. Elhoseiny, “Efficient self-supervised vision pretraining with local masked reconstruction,” *arXiv preprint arXiv:2206.00790*, 2022.
- [160] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv*, 2015.

- [161] C. Zhuang, A. L. Zhai, and D. Yamins, “Local aggregation for unsupervised learning of visual embeddings,” in *ICCV*.
- [162] I. Misra and L. v. d. Maaten, “Self-supervised learning of pretext-invariant representations,” in *CVPR*.
- [163] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *CVPR*.
- [164] X. Chen, H. Fan, R. Girshick, and K. He, “Improved baselines with momentum contrastive learning,” *arXiv*, 2020.
- [165] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, *et al.*, “Bootstrap your own latent: A new approach to self-supervised learning,” *arXiv*, 2020.
- [166] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *ICML*.
- [167] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” *arXiv*, 2020.
- [168] P. Goyal, M. Caron, B. Lefauveux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, *et al.*, “Self-supervised pretraining of visual features in the wild,” *arXiv*, 2021.
- [169] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, “Designing network design spaces,” in *CVPR*.
- [170] X. Chen and K. He, “Exploring simple siamese representation learning,” in *CVPR*.
- [171] J. Li, P. Zhou, C. Xiong, and S. C. H. Hoi, “Prototypical contrastive learning of unsupervised representations,” in *ICLR*, OpenReview.net, 2021.
- [172] L. Zhang, G.-J. Qi, L. Wang, and J. Luo, “Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data,” in *CVPR*.
- [173] P. Bachman, R. D. Hjelm, and W. Buchwalter, “Learning representations by maximizing mutual information across views,” *arXiv*, 2019.
- [174] X. Yan, I. Misra, A. Gupta, D. Ghadiyaram, and D. Mahajan, “Clusterfit: Improving generalization of visual representations,” in *CVPR*, 2020.
- [175] Y. M. Asano, C. Rupprecht, and A. Vedaldi, “Self-labelling via simultaneous clustering and representation learning,” *arXiv preprint arXiv:1911.05371*, 2019.
- [176] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, “Big self-supervised models are strong semi-supervised learners,” *arXiv*, 2020.
- [177] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” *arXiv*, 2019.
- [178] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical data augmentation with no separate search,” *arXiv*, 2019.
- [179] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, “What makes for good views for contrastive learning,” *arXiv*, 2020.

- [180] X. Chen, S. Xie, and K. He, “An empirical study of training self-supervised vision transformers,” *arXiv*, 2021.
- [181] J. Mitrovic, B. McWilliams, J. C. Walker, L. H. Buesing, and C. Blundell, “Representation learning via invariant causal mechanisms,” in *ICLR*.
- [182] Y. Tian, X. Chen, and S. Ganguli, “Understanding self-supervised learning dynamics without contrastive pairs,” in *ICML*.
- [183] Z. Xie, Y. Lin, Z. Yao, Z. Zhang, Q. Dai, Y. Cao, and H. Hu, “Self-supervised learning with swin transformers,” *arXiv*, 2021.
- [184] Z. Li, Z. Chen, F. Yang, W. Li, Y. Zhu, C. Zhao, R. Deng, L. Wu, R. Zhao, M. Tang, *et al.*, “Mst: Masked self-supervised transformer for visual representation,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.
- [185] H. Bao, L. Dong, S. Piao, and F. Wei, “BEit: BERT pre-training of image transformers,” in *International Conference on Learning Representations*, 2022.
- [186] X. Chen, M. Ding, X. Wang, Y. Xin, S. Mo, Y. Wang, S. Han, P. Luo, G. Zeng, and J. Wang, “Context autoencoder for self-supervised representation learning,” *arXiv preprint arXiv:2202.03026*, 2022.
- [187] X. Dong, J. Bao, T. Zhang, D. Chen, W. Zhang, L. Yuan, D. Chen, F. Wen, and N. Yu, “Peco: Perceptual codebook for bert pre-training of vision transformers,” *arXiv preprint arXiv:2111.12710*, 2021.
- [188] Y. You, T. Chen, Z. Wang, and Y. Shen, “When does self-supervision help graph convolutional networks?,” in *ICML*.
- [189] W. Jin, T. Derr, H. Liu, Y. Wang, S. Wang, Z. Liu, and J. Tang, “Self-supervised learning on graphs: Deep insights and new direction,” *CoRR*, 2020.
- [190] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. S. Pande, and J. Leskovec, “Strategies for pre-training graph neural networks,” in *ICLR*, 2020.
- [191] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: online learning of social representations,” in *ACM SIGKDD*.
- [192] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *ACM SIGKDD*.
- [193] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: large-scale information network embedding,” in *WWW*.
- [194] T. N. Kipf and M. Welling, “Variational graph auto-encoders,” *CoRR*, 2016.
- [195] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, “GCC: graph contrastive coding for graph neural network pre-training,” in *KDD*.
- [196] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, “Graph contrastive learning with adaptive augmentation,” in *WWW*, 2021.
- [197] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, “Graph contrastive learning with augmentations,” in *NeurIPS*, 2020.

- [198] C. Mavromatis and G. Karypis, “Graph infoclust: Maximizing coarse-grain mutual information in graphs,” in *PAKDD*, 2021.
- [199] Q. Sun, J. Li, H. Peng, J. Wu, Y. Ning, P. S. Yu, and L. He, “SUGAR: subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism,” in *WWW*, 2021.
- [200] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep graph infomax,” in *ICLR*, 2019.
- [201] K. Hassani and A. H. K. Ahmadi, “Contrastive multi-view representation learning on graphs,” in *ICML*.
- [202] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu, “Sub-graph contrast for scalable self-supervised graph representation learning,” in *ICDM*.
- [203] K. Sun, Z. Lin, and Z. Zhu, “Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes,” in *AAAI*.
- [204] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, “Self-supervised graph transformer on large-scale molecular data,” in *NeurIPS*, 2020.
- [205] Q. Tan, N. Liu, X. Huang, R. Chen, S.-H. Choi, and X. Hu, “Mgae: Masked autoencoders for self-supervised learning on graphs,” *arXiv preprint arXiv:2201.02534*, 2022.
- [206] Z. Hou, X. Liu, Y. Dong, C. Wang, J. Tang, *et al.*, “Graphmae: Self-supervised masked graph autoencoders,” *arXiv preprint arXiv:2205.10803*, 2022.
- [207] J. Li, R. Wu, W. Sun, L. Chen, S. Tian, L. Zhu, C. Meng, Z. Zheng, and W. Wang, “Maskgae: Masked graph modeling meets graph autoencoders,” *arXiv preprint arXiv:2205.10053*, 2022.
- [208] Y. Tian, K. Dong, C. Zhang, C. Zhang, and N. V. Chawla, “Heterogeneous graph masked autoencoders,” *arXiv preprint arXiv:2208.09957*, 2022.
- [209] S. Wan, S. Pan, J. Yang, and C. Gong, “Contrastive and generative graph convolutional networks for graph-based semi-supervised learning,” in *AAAI*.
- [210] J. Zhang, H. Zhang, C. Xia, and L. Sun, “Graph-bert: Only attention is needed for learning graph representations,” *arXiv*, 2020.
- [211] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, “Graph representation learning via graphical mutual information maximization,” in *WWW*, 2020.
- [212] Z. Hu, Y. Dong, K. Wang, K. Chang, and Y. Sun, “GPT-GNN: generative pre-training of graph neural networks,” in *KDD*.
- [213] G. Wang, H. Guo, A. Li, X. Liu, and Q. Yan, “Federated iot interaction vulnerability analysis,” in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, IEEE, 2023.
- [214] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *NIPS*, 2017.
- [215] D. Hwang, J. Park, S. Kwon, K. Kim, J. Ha, and H. J. Kim, “Self-supervised auxiliary learning with meta-paths for heterogeneous graphs,” in *NeurIPS*, 2020.

- [216] F. Sun, J. Hoffmann, V. Verma, and J. Tang, “Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization,” in *ICLR*, 2020.
- [217] C. Park, D. Kim, J. Han, and H. Yu, “Unsupervised attributed multiplex network embedding,” in *AAAI*.
- [218] Y. You, T. Chen, Y. Shen, and Z. Wang, “Graph contrastive learning automated,” *CoRR*, 2021.
- [219] J. Zeng and P. Xie, “Contrastive self-supervised learning for graph classification,” in *AAAI*.
- [220] M. Xu, H. Wang, B. Ni, H. Guo, and J. Tang, “Self-supervised graph-level representation learning with local and global structure,” *CoRR*, 2021.
- [221] P. Wang, K. Agarwal, C. Ham, S. Choudhury, and C. K. Reddy, “Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks,” in *WWW*.
- [222] J. Cao, X. Lin, S. Guo, L. Liu, T. Liu, and B. Wang, “Bipartite graph embedding via mutual information maximization,” in *WSDM*, 2021.
- [223] X. Wang, N. Liu, H. Han, and C. Shi, “Self-supervised heterogeneous graph neural network with co-contrastive learning,” *KDD*, 2021.
- [224] D. Kim and A. Oh, “How to find your friendly neighborhood: Graph attention design with self-supervision,” in *ICLR*, 2021.
- [225] M. Sun, J. Xing, H. Wang, B. Chen, and J. Zhou, “Mocl: Contrastive learning on molecular graphs with multi-level domain knowledge,” *CoRR*, 2021.
- [226] S. Schneider, A. Baevski, R. Collobert, and M. Auli, “wav2vec: Unsupervised pre-training for speech recognition,” in *INTERSPEECH*.
- [227] A. Baevski, S. Schneider, and M. Auli, “vq-wav2vec: Self-supervised learning of discrete speech representations,” in *ICLR*.
- [228] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” in *NeurIPS*, 2020.
- [229] Y. Chung and J. R. Glass, “Generative pre-training for speech with autoregressive predictive coding,” in *ICASSP*.
- [230] X. Song, G. Wang, Y. Huang, Z. Wu, D. Su, and H. Meng, “Speech-xlnet: Unsupervised acoustic model pretraining for self-attention networks,” in *INTERSPEECH*.
- [231] Y. Chung, Y. Wang, W. Hsu, Y. Zhang, and R. J. Skerry-Ryan, “Semi-supervised training for improving data efficiency in end-to-end speech synthesis,” in *ICASSP*, 2019.
- [232] P. Denisov and N. T. Vu, “Pretrained semantic speech embeddings for end-to-end spoken language understanding via cross-modal teacher-student learning,” in *Interspeech*, 2020.
- [233] Y.-A. Chung, C. Zhu, and M. Zeng, “SPLAT: Speech-language joint pre-training for spoken language understanding,” in *ACL*, 2021.
- [234] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, “Musicbert: Symbolic music understanding with large-scale pre-training,” *arXiv preprint arXiv:2106.05630*, 2021.

- [235] Y.-S. Huang and Y.-H. Yang, “Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions,” in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 1180–1188, 2020.
- [236] P. Verma and J. Berger, “Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions,” *arXiv preprint arXiv:2105.00335*, 2021.
- [237] B. Fernando, H. Bilen, E. Gavves, and S. Gould, “Self-supervised video representation learning with odd-one-out networks,” in *CVPR*.
- [238] I. Misra, C. L. Zitnick, and M. Hebert, “Shuffle and learn: unsupervised learning using temporal order verification,” in *ECCV*, 2016.
- [239] D. Kim, D. Cho, and I. S. Kweon, “Self-supervised video representation learning with space-time cubic puzzles,” in *AAAI*.
- [240] L. Tao, X. Wang, and T. Yamasaki, “Self-supervised video representation learning using inter-intra contrastive framework,” in *ACM Multimedia*.
- [241] G. Lorre, J. Rabarisoa, A. Orcesi, S. Ainouz, and S. Canu, “Temporal contrastive pretraining for video action recognition,” in *WACV*.
- [242] T. Yao, Y. Zhang, Z. Qiu, Y. Pan, and T. Mei, “Seco: Exploring sequence supervision for unsupervised representation learning,” *arXiv*, 2020.
- [243] L. H. Li, M. Yatskar, D. Yin, C. Hsieh, and K. Chang, “Visualbert: A simple and performant baseline for vision and language,” *CoRR*, 2019.
- [244] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, “Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training,” in *AAAI*.
- [245] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, “VL-BERT: pre-training of generic visual-linguistic representations,” in *ICLR*, 2020.
- [246] J. Lu, D. Batra, D. Parikh, and S. Lee, “Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks,” in *NeurIPS*.
- [247] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.
- [248] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *International Conference on Machine Learning*, pp. 8748–8763, PMLR, 2021.
- [249] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, “Glide: Towards photorealistic image generation and editing with text-guided diffusion models,” *arXiv preprint arXiv:2112.10741*, 2021.
- [250] N. Sayed, B. Brattoli, and B. Ommer, “Cross and learn: Cross-modal self-supervision,” in *GCPR*, 2018.
- [251] Z. Ren and Y. J. Lee, “Cross-domain self-supervised multi-task feature learning using synthetic imagery,” in *CVPR*.

- [252] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” in *ECCV*.
- [253] A. Zlotchevski, D. Drain, A. Svyatkovskiy, C. B. Clement, N. Sundaresan, and M. Tufano, “Exploring and evaluating personalized models for code generation,” in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1500–1508, 2022.
- [254] S. Thakur, B. Ahmad, Z. Fan, H. Pearce, B. Tan, R. Karri, B. Dolan-Gavitt, and S. Garg, “Benchmarking large language models for automated verilog rtl code generation,” *arXiv preprint arXiv:2212.11140*, 2022.
- [255] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, “Codegen: An open large language model for code with multi-turn program synthesis,” *arXiv preprint arXiv:2203.13474*, 2022.
- [256] G. Poesia, O. Polozov, V. Le, A. Tiwari, G. Soares, C. Meek, and S. Gulwani, “Synchromesh: Reliable code generation from pre-trained language models,” *arXiv preprint arXiv:2201.11227*, 2022.
- [257] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, “Uniter: Universal image-text representation learning,” in *European conference on computer vision*, pp. 104–120, Springer, 2020.
- [258] X. Zhu, J. Zhu, H. Li, X. Wu, H. Li, X. Wang, and J. Dai, “Uni-perceiver: Pre-training unified architecture for generic perception for zero-shot and few-shot tasks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16804–16815, 2022.
- [259] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, “A generalist agent,” *arXiv preprint arXiv:2205.06175*, 2022.
- [260] W. Li, C. Gao, G. Niu, X. Xiao, H. Liu, J. Liu, H. Wu, and H. Wang, “Unimo: Towards unified-modal understanding and generation via cross-modal contrastive learning,” *arXiv preprint arXiv:2012.15409*, 2020.
- [261] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter,” *CoRR*, 2019.
- [262] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is bert really robust? a strong baseline for natural language attack on text classification and entailment,” in *AAAI*.
- [263] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is BERT really robust? A strong baseline for natural language attack on text classification and entailment,” in *AAAI*.
- [264] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun, “Word-level textual adversarial attacking as combinatorial optimization,” in *ACL*.
- [265] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, “Universal adversarial triggers for attacking and analyzing NLP,” in *EMNLP-IJCNLP*.
- [266] K. Kurita, P. Michel, and G. Neubig, “Weight poisoning attacks on pretrained models,” in *ACL*.
- [267] R. Schuster, T. Schuster, Y. Meri, and V. Shmatikov, “Humpty dumpty: Controlling word meanings via corpus poisoning,” in *IEEE Symposium on Security and Privacy*, 2020.

- [268] R. Bao, J. Wang, and H. Zhao, “Defending pre-trained language models from adversarial word substitution without performance sacrifice,” in *ACL/IJCNLP*.
- [269] Z. Zhang, Y. Li, J. Wang, B. Liu, D. Li, Y. Guo, X. Chen, and Y. Liu, “Remos: reducing defect inheritance in transfer learning via relevant model slicing,” in *Proceedings of the 44th International Conference on Software Engineering*, pp. 1856–1868, 2022.
- [270] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, U. Erlingsson, *et al.*, “Extracting training data from large language models.,” in *USENIX Security Symposium*, vol. 6, 2021.
- [271] G. Wang, L. Zhang, Z. Yang, and X.-Y. Li, “Socialite: Social activity mining and friend auto-labeling,” in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, IEEE, 2018.
- [272] F. Han, L. Zhang, X. You, G. Wang, and X.-Y. Li, “Shad: Privacy-friendly shared activity detection and data sharing,” in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 109–117, IEEE, 2019.
- [273] T. Chen, X. Zhai, M. Ritter, M. Lucic, and N. Houlsby, “Self-supervised gans via auxiliary rotation loss,” in *CVPR*, 2019.
- [274] S. Abnar, M. Dehghani, B. Neyshabur, and H. Sedghi, “Exploring the limits of large scale pre-training,” *arXiv*, 2021.
- [275] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *NAACL-HLT*, 2018.
- [276] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *NIPS*.
- [277] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa, “Natural language processing (almost) from scratch,” *J. Mach. Learn. Res.*, 2011.
- [278] A. Neelakantan, J. Shankar, A. Passos, and A. McCallum, “Efficient non-parametric estimation of multiple embeddings per word in vector space,” in *EMNLP*.
- [279] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, “Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling,” in *COLING*.
- [280] D. U. Hui, X. U. Xueke, W. U. Dayong, Y. Liu, Y. U. Zhihua, and X. Cheng, “A sentiment classification method based on sentiment-specific word embedding,” *Journal of Chinese Information Processing*, 2017.
- [281] Y. Liu, C. Ma, and Y. Zhang, “Hierarchical machine translation model based on deep recursive neural network,” *Chin. J. Comput.*, 2017.
- [282] X. Liang, F. Ren, Y. Liu, L. Pan, Y. Hou, Y. Zhang, and L. I. Yan, “N-reader: Machine reading comprehension model based on double layers of self-attention,” *Journal of Chinese Information Processing*, 2018.
- [283] Z. Zhichang, Z. Zhenwen, and Z. Zhiman, “User intent classification based on indrnn-attention,” *Journal of Computer Research and Development*, 2019.

- [284] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, 2009.
- [285] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, 2012.
- [286] M. Lin, Q. Chen, and S. Yan, “Network in network,” *arXiv*, 2013.
- [287] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv*, 2014.
- [288] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *CVPR*.
- [289] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*.
- [290] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *CVPR*.
- [291] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*.
- [292] R. Girshick, “Fast r-cnn,” in *ICCV*.
- [293] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *arXiv*, 2015.
- [294] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *ICCV*.
- [295] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*.
- [296] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, 2016.
- [297] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *CVPR*.
- [298] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in *CVPR*.
- [299] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv*, 2018.
- [300] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv*, 2020.
- [301] Q. Chen, Y. Wang, T. Yang, X. Zhang, J. Cheng, and J. Sun, “You only look one-level feature,” *arXiv*, 2021.
- [302] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, 2017.
- [303] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *CVPR*, 2017.

- [304] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” *arXiv*, 2014.
- [305] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE transactions on pattern analysis and machine intelligence*, 2017.
- [306] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv*, 2017.
- [307] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *ECCV*.
- [308] G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation,” in *CVPR*.
- [309] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International journal of computer vision*, 2015.
- [310] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, 2010.
- [311] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, 2014.
- [312] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, 1986.
- [313] M. I. Jordan, “Serial order: A parallel distributed processing approach,” in *Advances in psychology*, 1997.
- [314] J. L. Elman, “Finding structure in time,” *Cognitive science*, 1990.
- [315] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, 1997.
- [316] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE transactions on pattern analysis and machine intelligence*, 2008.
- [317] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *CVPR*.
- [318] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *arXiv*, 2014.
- [319] A. Graves, “Generating sequences with recurrent neural networks,” *arXiv*, 2013.
- [320] M. Sundermeyer, R. Schlüter, and H. Ney, “Lstm neural networks for language modeling,” in *INTERSPEECH*, 2012.
- [321] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *arXiv*, 2014.
- [322] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *CVPR*, 2017.

- [323] C. Li and M. Wand, “Precomputed real-time texture synthesis with markovian generative adversarial networks,” in *European conference on computer vision*, 2016.
- [324] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *ICCV*.
- [325] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *CVPR*, 2019.
- [326] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *International Conference on Machine Learning*, 2016.
- [327] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, “Learning to discover cross-domain relations with generative adversarial networks,” in *International Conference on Machine Learning*, 2017.
- [328] E. Denton, S. Chintala, A. Szlam, and R. Fergus, “Deep generative image models using a laplacian pyramid of adversarial networks,” *arXiv*, 2015.
- [329] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, “Stacked generative adversarial networks,” in *CVPR*.
- [330] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *CVPR*.
- [331] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, “Cbam: Convolutional block attention module,” in *ECCV*.
- [332] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, “Gcnet: Non-local networks meet squeeze-excitation networks and beyond,” in *ICCV Workshops*.
- [333] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, “Ccnet: Criss-cross attention for semantic segmentation,” in *ICCV*.
- [334] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in *International conference on machine learning*, 2019.
- [335] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention,” *arXiv*, 2020.
- [336] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European Conference on Computer Vision*, 2020.
- [337] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, “Levit: a vision transformer in convnet’s clothing for faster inference,” *arXiv*, 2021.
- [338] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, and J. Feng, “Deepvit: Towards deeper vision transformer,” *arXiv*, 2021.
- [339] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” *arXiv*, 2021.
- [340] T. Guan, J. Wang, S. Lan, R. Chandra, Z. Wu, L. Davis, and D. Manocha, “M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers,” *arXiv*, 2021.
- [341] J. M. J. Valanarasu, P. Oza, I. Hacihaliloglu, and V. M. Patel, “Medical transformer: Gated axial-attention for medical image segmentation,” *arXiv*, 2021.

- [342] R. C. T. Lee, Y. H. Chin, and S. C. Chang, “Application of principal component analysis to multikey searching,” *IEEE Trans. Software Eng.*, no. 3, pp. 185–193, 1976.
- [343] J. Ye, R. Janardan, and Q. Li, “Two-dimensional linear discriminant analysis,” in *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pp. 1569–1576, 2004.
- [344] S. Robinson and R. Bennett, “A typology of deviant workplace behaviors: A multidimensional scaling study,” *Academy of Management Journal*, vol. 38, pp. 555–572, 1995.
- [345] O. Samko, A. D. Marshall, and P. L. Rosin, “Selection of the optimal parameter value for the isomap algorithm,” *Pattern Recognit. Lett.*, no. 9, pp. 968–979, 2006.
- [346] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [347] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural Comput.*, no. 6, pp. 1373–1396, 2003.
- [348] A. P. Singh and G. J. Gordon, “Relational learning via collective matrix factorization,” in *ACM SIGKDD*.
- [349] S. Cao, W. Lu, and Q. Xu, “Grarep: Learning graph representations with global structural information,” in *CIKM*.
- [350] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in *ACM SIGKDD*.
- [351] M. Sugiyama and K. M. Borgwardt, “Halting in random walk kernels,” in *NIPS*, 2015.
- [352] U. Kang, H. Tong, and J. Sun, “Fast random walk graph kernel,” in *SIAM*.
- [353] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-lehman graph kernels,” *J. Mach. Learn. Res.*, pp. 2539–2561, 2011.
- [354] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, “The difficulty of training deep architectures and the effect of unsupervised pre-training,” in *Artificial Intelligence and Statistics*, 2009.
- [355] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, “Why does unsupervised pre-training help deep learning?,” in *AISTATS*.
- [356] J. D. Lee, Q. Lei, N. Saunshi, and J. Zhuo, “Predicting what you already know helps: Provable self-supervised learning,” *arXiv*, 2020.
- [357] C. Tosh, A. Krishnamurthy, and D. Hsu, “Contrastive learning, multi-view redundancy, and linear models,” in *Algorithmic Learning Theory*, 2021.
- [358] S. Arora, H. Khandeparkar, M. Khodak, O. Plevrakis, and N. Saunshi, “A theoretical analysis of contrastive unsupervised representation learning,” *arXiv*, 2019.
- [359] S. Anwar, M. Tahir, C. Li, A. Mian, F. S. Khan, and A. W. Muzaffar, “Image colorization: A survey and dataset,” *arXiv*, 2020.

- [360] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *CVPR*.
- [361] G. Perarnau, J. Van De Weijer, B. Raducanu, and J. M. Álvarez, “Invertible conditional gans for image editing,” *arXiv*, 2016.
- [362] C. Vondrick, H. Pirsiavash, and A. Torralba, “Generating videos with scene dynamics,” *arXiv*, 2016.
- [363] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, “Mocogan: Decomposing motion and content for video generation,” in *CVPR*.
- [364] X. Wang and A. Gupta, “Unsupervised learning of visual representations using videos,” in *ICCV*.
- [365] C. Wei, L. Xie, X. Ren, Y. Xia, C. Su, J. Liu, Q. Tian, and A. L. Yuille, “Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning,” in *CVPR*.
- [366] U. Ahsan, R. Madhok, and I. Essa, “Video jigsaw: Unsupervised learning of spatiotemporal context for video action recognition,” in *WACV*.
- [367] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, “Learning features by watching objects move,” in *CVPR*.
- [368] I. Croitoru, S.-V. Bogolin, and M. Leordeanu, “Unsupervised learning from video to detect foreground objects in single images,” in *ICCV*.
- [369] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, “Time-contrastive networks: Self-supervised learning from video,” in *ICRA*.
- [370] B. Korbar, D. Tran, and L. Torresani, “Cooperative learning of audio and video models from self-supervised synchronization,” *arXiv*, 2018.
- [371] B. C. Stadie, S. Levine, and P. Abbeel, “Incentivizing exploration in reinforcement learning with deep predictive models,” *arXiv preprint arXiv:1507.00814*, 2015.
- [372] J. Achiam and S. Sastry, “Surprise-based intrinsic motivation for deep reinforcement learning,” *arXiv preprint arXiv:1703.01732*, 2017.
- [373] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, “Curiosity-driven exploration by self-supervised prediction,” in *International conference on machine learning*, pp. 2778–2787, PMLR, 2017.
- [374] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, “# exploration: A study of count-based exploration for deep reinforcement learning,” *Advances in neural information processing systems*, vol. 30, 2017.
- [375] P. Dey and S. Medya, “Manipulating node similarity measures in network,” *arXiv*, 2019.
- [376] B. Han, C. Zheng, H. Chan, K. Paster, M. Zhang, and J. Ba, “Learning domain invariant representations in goal-conditioned block mdps,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 764–776, 2021.
- [377] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, “Goal-conditioned imitation learning,” *Advances in neural information processing systems*, vol. 32, 2019.

- [378] R. Shah and V. Kumar, “Rrl: Resnet as representation for reinforcement learning,” *arXiv preprint arXiv:2107.03380*, 2021.
- [379] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik, “Masked visual pre-training for motor control,” *arXiv preprint arXiv:2203.06173*, 2022.
- [380] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, R. D. Hjelm, P. Bachman, and A. C. Courville, “Pretraining representations for data-efficient reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 12686–12699, 2021.
- [381] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, “Data-efficient reinforcement learning with self-predictive representations,” *arXiv preprint arXiv:2007.05929*, 2020.
- [382] D. Ha and J. Schmidhuber, “World Models,” Mar. 2018.
- [383] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, “Reinforcement Learning with Unsupervised Auxiliary Tasks,” Nov. 2016.
- [384] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, “DARLA: Improving Zero-Shot Transfer in Reinforcement Learning,” June 2018.
- [385] C. Finn, T. Yu, J. Fu, P. Abbeel, and S. Levine, “Generalizing skills with semi-supervised reinforcement learning,” *arXiv preprint arXiv:1612.00429*, 2016.
- [386] R. Shah and V. Kumar, “RRL: Resnet as representation for Reinforcement Learning,” Nov. 2021.
- [387] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, “Data-Efficient Reinforcement Learning with Self-Predictive Representations,” May 2021.
- [388] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, “Dream to control: Learning behaviors by latent imagination,” *arXiv preprint arXiv:1912.01603*, 2019.
- [389] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, “Mastering atari with discrete world models,” *arXiv preprint arXiv:2010.02193*, 2020.
- [390] F. Deng, I. Jang, and S. Ahn, “Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations,” in *International Conference on Machine Learning*, pp. 4956–4975, PMLR, 2022.
- [391] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, “Daydreamer: World models for physical robot learning,” *arXiv preprint arXiv:2206.14176*, 2022.
- [392] M. Laskin, A. Srinivas, and P. Abbeel, “Curl: Contrastive unsupervised representations for reinforcement learning,” in *International Conference on Machine Learning*, pp. 5639–5650, PMLR, 2020.
- [393] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, “Reinforcement learning with augmented data,” *Advances in neural information processing systems*, vol. 33, pp. 19884–19895, 2020.
- [394] I. Kostrikov, D. Yarats, and R. Fergus, “Image augmentation is all you need: Regularizing deep reinforcement learning from pixels,” *arXiv preprint arXiv:2004.13649*, 2020.
- [395] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, “Mastering visual continuous control: Improved data-augmented reinforcement learning,” *arXiv preprint arXiv:2107.09645*, 2021.

- [396] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, “R3m: A universal visual representation for robot manipulation,” *arXiv preprint arXiv:2203.12601*, 2022.
- [397] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta, “The unsurprising effectiveness of pre-trained vision models for control,” *arXiv preprint arXiv:2203.03580*, 2022.
- [398] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. 2008.
- [399] R. E. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” *Mach. Learn.*, no. 3, 1999.
- [400] E. Reiter, “A structured review of the validity of bleu,” *Computational Linguistics*, 2018.
- [401] M. Denkowski and A. Lavie, “Meteor universal: Language specific translation evaluation for any target language,” in *Proceedings of the ninth workshop on statistical machine translation*.
- [402] C.-Y. Lin and E. Hovy, “Automatic evaluation of summaries using n-gram co-occurrence statistics,” in *HLT-NAACL*, 2003.
- [403] Y. Kim, “Convolutional neural networks for sentence classification,” in *EMNLP*.
- [404] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences,” in *ACL*.
- [405] M. Yang, W. Zhao, J. Ye, Z. Lei, Z. Zhao, and S. Zhang, “Investigating capsule networks with dynamic routing for text classification,” in *EMNLP*.
- [406] L. Yao, C. Mao, and Y. Luo, “Graph convolutional networks for text classification,” in *AAAI*.
- [407] Y. Wang, A. Sun, J. Han, Y. Liu, and X. Zhu, “Sentiment analysis by capsules,” in *WWW*.
- [408] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *EMNLP*.
- [409] K. S. Tai, R. Socher, and C. D. Manning, “Improved semantic representations from tree-structured long short-term memory networks,” in *ACL*.
- [410] X. Zhu, P. Sobhani, and H. Guo, “Long short-term memory over recursive structures,” in *ICML*.
- [411] J. Cheng, L. Dong, and M. Lapata, “Long short-term memory-networks for machine reading,” in *EMNLP*.
- [412] P. Liu, X. Qiu, X. Chen, S. Wu, and X. Huang, “Multi-timescale long short-term memory neural network for modelling sentences and documents,” in *EMNLP*.
- [413] P. Liu, X. Qiu, and X. Huang, “Recurrent neural network for text classification with multi-task learning,” in *IJCAI*.
- [414] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, “Semi-supervised recursive autoencoders for predicting sentiment distributions,” in *EMNLP*.
- [415] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang, “Bi-directional block self-attention for fast and memory-efficient sequence modeling,” in *ICLR*, 2018.
- [416] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *ICML*.

- [417] M. Iyyer, V. Manjunatha, J. L. Boyd-Graber, and H. D. III, “Deep unordered composition rivals syntactic methods for text classification,” in *ACL*.
- [418] T. Miyato, A. M. Dai, and I. J. Goodfellow, “Adversarial training methods for semi-supervised text classification,” in *ICLR*, 2017.
- [419] S. Lai, L. Xu, K. Liu, and J. Zhao, “Recurrent convolutional neural networks for text classification,” in *AAAI*.
- [420] R. Johnson and T. Zhang, “Supervised and semi-supervised text categorization using LSTM for region embeddings,” in *ICML*.
- [421] Y. Bao, M. Wu, S. Chang, and R. Barzilay, “Few-shot text classification with distributional signatures,” in *ICLR*, 2020.
- [422] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, “Simplifying graph convolutional networks,” in *ICML*.
- [423] X. Zhang, J. J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *NIPS*, 2015.
- [424] R. Johnson and T. Zhang, “Deep pyramid convolutional neural networks for text categorization,” in *ACL*.
- [425] J. Wang, Z. Wang, D. Zhang, and J. Yan, “Combining knowledge with deep convolutional neural networks for short text classification,” in *IJCAI*.
- [426] L. Huang, D. Ma, S. Li, X. Zhang, and H. Wang, “Text level graph neural network for text classification,” in *EMNLP-IJCNLP*.
- [427] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune BERT for text classification?,” in *CCL*.
- [428] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, “Hierarchical attention networks for document classification,” in *NAACL-HLT*, 2016.
- [429] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” in *EMNLP*.
- [430] Z. Wang, W. Hamza, and R. Florian, “Bilateral multi-perspective matching for natural language sentences,” in *IJCAI*.
- [431] X. Liu, P. He, W. Chen, and J. Gao, “Multi-task deep neural networks for natural language understanding,” in *ACL*.
- [432] A. Williams, N. Nangia, and S. R. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” in *NAACL-HLT*, 2018.
- [433] M. Marelli, L. Bentivogli, M. Baroni, R. Bernardi, S. Menini, and R. Zamparelli, “Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment,” in *SemEval@COLING*.
- [434] B. Dolan, C. Quirk, and C. Brockett, “Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources,” in *COLING*, 2004.

- [435] J. Fu, P. Liu, and G. Neubig, “Interpretable multi-dataset evaluation for named entity recognition,” in *EMNLP*.
- [436] B. Lester, D. Pressel, A. Hemmeter, S. R. Choudhury, and S. Bangalore, “Constrained decoding for computationally efficient named entity recognition taggers,” in *EMNLP*.
- [437] Y. Luo, H. Zhao, and J. Zhan, “Named entity recognition only from word embeddings,” in *EMNLP*.
- [438] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li, “A unified MRC framework for named entity recognition,” in *ACL*.
- [439] Y. Zhang and J. Yang, “Chinese NER using lattice LSTM,” in *ACL*.
- [440] Y. Meng, W. Wu, F. Wang, X. Li, P. Nie, F. Yin, M. Li, Q. Han, X. Sun, and J. Li, “Glyce: Glyph-vectors for chinese character representations,” in *NeurIPS*.
- [441] A. Katiyar and C. Cardie, “Nested named entity recognition revisited,” in *NAACL-HLT*, 2018.
- [442] B. Wang and W. Lu, “Neural segmental hypergraphs for overlapping mention recognition,” in *EMNLP*.
- [443] Y. Luan, D. Wadden, L. He, A. Shah, M. Ostendorf, and H. Hajishirzi, “A general framework for information extraction using dynamic span graphs,” in *NAACL-HLT*, 2019.
- [444] T. Shibuya and E. H. Hovy, “Nested named entity recognition via second-best sequence learning and decoding,” *Trans. Assoc. Comput. Linguistics*, 2020.
- [445] H. Lin, Y. Lu, X. Han, and L. Sun, “Sequence-to-nuggets: Nested entity mention detection via anchor-region networks,” in *ACL*.
- [446] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. H. Hovy, “RACE: large-scale reading comprehension dataset from examinations,” in *EMNLP*.
- [447] Y. Yang, W. Yih, and C. Meek, “Wikiqa: A challenge dataset for open-domain question answering,” in *EMNLP*.
- [448] C. N. dos Santos, M. Tan, B. Xiang, and B. Zhou, “Attentive pooling networks,” *CoRR*, 2016.
- [449] J. Y. Lee and F. Dernoncourt, “Sequential short-text classification with recurrent and convolutional neural networks,” in *NAACL-HLT*, 2016.
- [450] S. Kim, L. F. D’Haro, R. E. Banchs, J. D. Williams, and M. Henderson, “The fourth dialog state tracking challenge,” in *Dialogues with Social Robots - Enablements, Analyses, and Evaluation, Seventh International Workshop on Spoken Dialogue Systems, IWSDS 2016, Saariselkä, Finland, January 13-16, 2016*, 2016.
- [451] J. Ang, Y. Liu, and E. Shriberg, “Automatic dialog act segmentation and classification in multiparty meetings,” in *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP ’05, Philadelphia, Pennsylvania, USA, March 18-23, 2005*, 2005.
- [452] Y. Wan, W. Yan, J. Gao, Z. Zhao, J. Wu, and P. S. Yu, “Improved dynamic memory network for dialogue act classification with adversarial training,” in *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, 2018.

- [453] V. Raheja and J. R. Tetreault, “Dialogue act classification with context-aware self-attention,” in *Proc. NAACL, 2019*, 2019.
- [454] J. Xu, Z. Gan, Y. Cheng, and J. Liu, “Discourse-aware neural extractive text summarization,” in *ACL*.
- [455] Y. Zou, X. Zhang, W. Lu, F. Wei, and M. Zhou, “Pre-training for abstractive document summarization by reinstating source text,” in *EMNLP*.
- [456] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li, “Generative adversarial network for abstractive text summarization,” in *AAAI*.
- [457] M. Yang, Q. Qu, W. Tu, Y. Shen, Z. Zhao, and X. Chen, “Exploring human-like reading strategy for abstractive text summarization,” in *AAAI*.
- [458] M. Bhandari, P. N. Gour, A. Ashfaq, P. Liu, and G. Neubig, “Re-evaluating evaluation in text summarization,” in *EMNLP*.
- [459] Y. Dong, S. Wang, Z. Gan, Y. Cheng, J. C. K. Cheung, and J. Liu, “Multi-fact correction in abstractive text summarization,” in *EMNLP*.
- [460] D. Huang, L. Cui, S. Yang, G. Bao, K. Wang, J. Xie, and Y. Zhang, “What have we achieved on text summarization?”, in *EMNLP*.
- [461] W. Kryscinski, R. Paulus, C. Xiong, and R. Socher, “Improving abstraction in text summarization,” in *EMNLP*.
- [462] W. Kryscinski, B. McCann, C. Xiong, and R. Socher, “Evaluating the factual consistency of abstractive text summarization,” in *EMNLP*.
- [463] P. Kouris, G. Alexandridis, and A. Stafylopatis, “Abstractive text summarization based on deep learning and semantic content generalization,” in *ACL*.
- [464] K. Chen, R. Wang, M. Utiyama, and E. Sumita, “Content word aware neural machine translation,” in *ACL*, 2020.
- [465] Z. Lin, X. Pan, M. Wang, X. Qiu, J. Feng, H. Zhou, and L. Li, “Pre-training multilingual neural machine translation by leveraging alignment information,” in *EMNLP*.
- [466] E. Bugliarello and N. Okazaki, “Enhancing machine translation with dependency-aware self-attention,” in *ACL*, 2020.
- [467] A. F. Aji, N. Bogoychev, K. Heafield, and R. Sennrich, “In neural machine translation, what does transfer learning transfer?”, in *ACL*.
- [468] C. Baizotis, B. Haddow, and A. Birch, “Language model prior for low-resource neural machine translation,” in *EMNLP*.
- [469] Q. Cui, S. Huang, J. Li, X. Geng, Z. Zheng, G. Huang, and J. Chen, “Directqe: Direct pretraining for machine translation quality estimation,” in *AAAI*.
- [470] C. Wu, S. C. H. Hoi, R. Socher, and C. Xiong, “TOD-BERT: pre-trained natural language understanding for task-oriented dialogue,” in *EMNLP*.
- [471] G. Campagna, A. Foryciarz, M. Moradshahi, and M. S. Lam, “Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking,” in *ACL*, 2020.

- [472] Q. Liu, L. Yu, L. Rimell, and P. Blunsom, “Pretraining the noisy channel model for task-oriented dialogue,” *CoRR*, 2021.
- [473] “SST Corpus.” <http://nlp.stanford.edu/sentiment>, 2013.
- [474] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” in *ACL*.
- [475] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *EMNLP*.
- [476] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, “Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation,” *arXiv*, 2017.
- [477] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, “Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals,” in *Proc. NAACL, 2009*, 2009.
- [478] J. Wiebe, T. Wilson, and C. Cardie, “Annotating expressions of opinions and emotions in language,” *Language Resources and Evaluation*, no. 2-3, 2005.
- [479] “MPQA Corpus.” <http://www.cs.pitt.edu/mpqa/>, 2005.
- [480] Q. Diao, M. Qiu, C. Wu, A. J. Smola, J. Jiang, and C. Wang, “Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS),” in *ACM SIGKDD*.
- [481] “20NG Corpus.” <http://ana.cachopo.org/datasets-for-single-label-text-categorization>, 2007.
- [482] “AG Corpus.” http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, 2004.
- [483] “Reuters Corpus.” <https://www.cs.umb.edu/~smimarog/textmining/datasets/>, 2007.
- [484] “Reuters Corpus.” <https://martin-thoma.com/nlp-reuters>, 2017.
- [485] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, “Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia,” *Semantic Web*, no. 2, 2015.
- [486] “Ohsumed Corpus.” <http://davis.wpi.edu/xmdv/datasets/ohsumed.html>, 2015.
- [487] A. Williams, N. Nangia, and S. R. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” *arXiv*, 2017.
- [488] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” *arXiv*, 2016.
- [489] H. Levesque, E. Davis, and L. Morgenstern, “The winograd schema challenge,” in *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2012.
- [490] W. B. Dolan and C. Brockett, “Automatically constructing a corpus of sentential paraphrases,” in *IWP*, 2005.

- [491] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” *arXiv*, 2018.
- [492] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, “Race: Large-scale reading comprehension dataset from examinations,” *arXiv*, 2017.
- [493] D. Jurafsky and E. Shriberg, “Switchboard swbd-damsl shallow-discourse-function annotation coders manual,” 1997.
- [494] J. Li, P. Zhou, C. Xiong, R. Socher, and S. C. Hoi, “Prototypical contrastive learning of unsupervised representations,” *arXiv preprint arXiv:2005.04966*, 2020.
- [495] J. Donahue and K. Simonyan, “Large scale adversarial representation learning,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [496] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” *arXiv preprint arXiv:2111.06377*, 2021.
- [497] <http://yann.lecun.com/exdb/mnist/>.
- [498] <http://ufldl.stanford.edu/housenumbers/>.
- [499] <https://www.cs.toronto.edu/~kriz/index.html>.
- [500] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.
- [501] <https://cs.stanford.edu/~acoates/stl10/>.
- [502] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.
- [503] G. A. Miller, *WordNet: An electronic lexical database*. 1998.
- [504] <https://image-net.org/>.
- [505] <https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/>.
- [506] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “HMDB: a large video database for human motion recognition,” in *ICCV*, 2011.
- [507] <https://www.crcv.ucf.edu/data/UCF101.php>.
- [508] <https://www.crcv.ucf.edu/data/UCF50.php>.
- [509] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101–mining discriminative components with random forests,” in *European conference on computer vision*, 2014.
- [510] T. Berg, J. Liu, S. Woo Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur, “Birdsnap: Large-scale fine-grained visual categorization of birds,” in *CVPR*, 2014.
- [511] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *2010 IEEE computer society conference on computer vision and pattern recognition*, 2010.

- [512] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, “Sun database: Exploring a large collection of scene categories,” *International Journal of Computer Vision*, 2016.
- [513] <http://places.csail.mit.edu/downloadData.html>.
- [514] http://ai.stanford.edu/~jkrause/cars/car_dataset.html.
- [515] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, “Fine-grained visual classification of aircraft,” tech. rep., 2013.
- [516] <https://sites.google.com/site/fgcomp2013/>.
- [517] <https://www.robots.ox.ac.uk/~vgg/data/fgvc-aircraft/>.
- [518] <https://www.robots.ox.ac.uk/~vgg/data/pets/>.
- [519] <https://www.robots.ox.ac.uk/~vgg/data/flowers/>.
- [520] <https://www.robots.ox.ac.uk/~vgg/data/dtd/>.
- [521] <https://sites.google.com/view/fgvc5/competitions/inaturalist>.
- [522] <https://www.inaturalist.org/>.
- [523] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” in *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.
- [524] <http://host.robots.ox.ac.uk/pascal/VOC/>.
- [525] <http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html>.
- [526] <http://host.robots.ox.ac.uk/pascal/VOC/voc2011/index.html>.
- [527] <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html>.
- [528] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *CVPR*.
- [529] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, “Semantic understanding of scenes through the ade20k dataset,” *International Journal of Computer Vision*, 2019.
- [530] https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.
- [531] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset,” in *CVPR Workshop on The Future of Datasets in Vision*, 2015.
- [532] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [533] A. Gupta, P. Dollar, and R. Girshick, “LVIS: A dataset for large vocabulary instance segmentation,” in *CVPR*, 2019.
- [534] <https://davischallenge.org/>.

- [535] <https://davischallenge.org/davis2017/code.html>.
- [536] C. Doersch, “Data analysis project: What makes paris look like paris?,”
- [537] http://www.cs.toronto.edu/~nitish/unsupervised_video/.
- [538] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using lstms,” in *International conference on machine learning*, 2015.
- [539] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, “Yfcc100m: The new data in multimedia research,” *Communications of the ACM*, 2016.
- [540] <http://projects.dfki.uni-kl.de/yfcc100m/>.
- [541] W. Jin, X. Liu, X. Zhao, Y. Ma, N. Shah, and J. Tang, “Automated self-supervised learning for graphs,” *CoRR*, 2021.
- [542] Z. Peng, Y. Dong, M. Luo, X. Wu, and Q. Zheng, “Self-supervised graph representation learning via global context prediction,” *CoRR*, 2020.
- [543] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, “Deep graph contrastive representation learning,” *CoRR*, 2020.
- [544] M. Jin, Y. Zheng, Y. Li, C. Gong, C. Zhou, and S. Pan, “Multi-scale contrastive siamese networks for self-supervised graph representation learning,” *CoRR*, 2021.
- [545] Z. Hu, C. Fan, T. Chen, K. Chang, and Y. Sun, “Pre-training graph neural networks for generic structural feature extraction,” *CoRR*, 2019.
- [546] Y. Zhu, Y. Xu, F. Yu, S. Wu, and L. Wang, “CAGNN: cluster-aware graph neural networks for unsupervised graph representation learning,” *CoRR*, 2020.
- [547] H. Zhang, S. Lin, W. Liu, P. Zhou, J. Tang, X. Liang, and E. P. Xing, “Iterative graph self-distillation,” *CoRR*, 2020.
- [548] S. Lin, P. Zhou, Z.-Y. Hu, S. Wang, R. Zhao, Y. Zheng, L. Lin, E. Xing, and X. Liang, “Prototypical graph contrastive learning,” 2021.
- [549] S. Zhang, Z. Hu, A. Subramonian, and Y. Sun, “Motif-driven contrastive learning of graph representations,” *CoRR*, 2020.
- [550] F. L. Opolka, A. Solomon, C. Cangea, P. Velickovic, P. Liò, and R. D. Hjelm, “Spatio-temporal deep graph infomax,” *CoRR*, 2019.