Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)
Survey Track

Evaluation Methods for Representation Learning: A Survey

Kento Nozawa[1,2] and **Issei Sato**[1]
[1]The University of Tokyo
[2]RIKEN AIP
{nzw, sato}@g.ecc.u-tokyo.ac.jp

Abstract

Representation learning enables us to automatically extract generic feature representations from a dataset to solve another machine learning task. Recently, extracted feature representations by a representation learning algorithm and a simple predictor have exhibited state-of-the-art performance on several machine learning tasks. Despite its remarkable progress, there exist various ways to evaluate representation learning algorithms depending on the application because of the flexibility of representation learning. To understand the current applications of representation learning, we review evaluation methods of representation learning algorithms. On the basis of our evaluation survey, we also discuss the future direction of representation learning. The extended version, https://arxiv.org/abs/2204.08226, gives more detailed discussions and a survey on theoretical analyses.

1 Introduction

In deep neural networks [Goodfellow *et al.*, 2016], multiple nonlinear transformations from input space to output space are distinguished characteristics compared with other machine learning algorithms such as a kernel method. Nonlinear transformations enable deep neural networks to internally learn a feature vector, namely, feature "representation", that effectively captures informative features to optimize the objective function. For example, when we solve the MNIST classification task, the input image is transformed to a more abstract representation than the original input to predict its class label, which is a digit, after applying multiple nonlinear transformations by using convolutional neural networks. Thanks to this nonlinearity, deep learning algorithms often lower the priority of feature engineering. In other words, we require much less domain knowledge to carefully construct hand-crafted features when we solve the machine learning problem.

Motivated by the importance of learning feature representations, representation learning [Bengio *et al.*, 2013] is defined as a set of methods that automatically learn discriminative feature representations from a dataset to solve a machine learning task [LeCun *et al.*, 2015]. Empirically, the learned model is used as a feature extractor for other machine learning tasks, such as classification, regression, and visualization. In this sense, representation learning is also referred to as method to learn generic feature representations for *unseen* downstream tasks rather than end-to-end methods to solve a machine learning task directly. Unfortunately, we do not yet have a well-defined evaluation metric of representation learning for the latter case due to various applications of representation learning. Nevertheless, we believe that evaluation methods are critical in designing novel or analyzing existing algorithms.

We review the existing evaluation methods of representation learning algorithms to understand their applications and the current common practice. Specifically, we propose four evaluation perspectives of representation learning algorithms. In addition, we discuss the future direction on the basis of our review. Note that we *do not* aim to provide a comprehensive survey on the state-of-the-art algorithms compared with existing representation learning surveys such as [Jing and Tian, 2019].

2 Background: Representation Learning

We give a high-level overview and formulation of representation learning. We give two formulations in terms of the existence of supervised signals during representation learning: supervised representation learning (Section 2.1) and unsupervised representation learning (Section 2.2).

2.1 Supervised Representation Learning

Suppose supervised dataset $\mathcal{D}_{\text{sup}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}$ is an input sample, and y is a supervised signal such as a class label in classification or a real-valued target vector in regression. As a running example, we suppose classification, where $\mathbf{x}$ is an input sample, and $y \in \{1, \dots, Y\}$ is a categorical value in a pre-defined class set. A supervised representation learning algorithm trains parameterized feature extractor $\mathbf{h}$ by solving a supervised task on $\mathcal{D}_{\text{sup}}$. Feature extractor $\mathbf{h} : \mathbb{R}^I \to \mathbb{R}^d$ maps an input representation $\mathbf{x}$ to a feature representation $\mathbf{h}(\mathbf{x}) \in \mathbb{R}^d$, where d tends to be smaller than I, the dimensionality of $\mathbf{x}$. Depending on the supervised task, an additional function, $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^O$, yields the output representation to evaluate a supervised objective function given feature representation $\mathbf{h}(\mathbf{x})$. For multi-class classification case, $O = Y$. Formally, we minimize training loss function $\widehat{L}(\mathbf{h}, \mathbf{g})$ such as a cross-entropy loss to obtain pre-trained $\widehat{\mathbf{h}}$ and $\widehat{\mathbf{g}}$ as follows:

$$\widehat{\mathbf{h}},\ \widehat{\mathbf{g}} = \operatorname*{argmin}_{\mathbf{h},\mathbf{g}} \widehat{L}(\mathbf{h}, \mathbf{g}),$$
$$\text{where}\quad \widehat{L}(\mathbf{h}, \mathbf{g}) = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{\exp(\mathbf{g}_{y_i}(\mathbf{h}(\mathbf{x}_i)))}{\sum_{y\in\mathcal{Y}} \exp(\mathbf{g}_{y}(\mathbf{h}(\mathbf{x}_i)))}. \quad (1)$$

After minimizing supervised loss $\widehat{L}$ (1), we use $\widehat{\mathbf{h}}$ as a feature extractor for other machine learning tasks. The rest part, $\widehat{\mathbf{g}}$, tends to be removed after the optimization of $\widehat{L}$ because representation $\widehat{\mathbf{g}}(\widehat{\mathbf{h}}(\mathbf{x}))$ leads to poor downstream performance in practice [Donahue *et al.*, 2014]. [Yosinski *et al.*, 2014] explained that feature representations extracted by using near the final layer in neural networks are too specialized to solve the upstream supervised task without fine-tuning. Similar results have been reported in unsupervised representation learning [Bachman *et al.*, 2019]. By following the notations above, we define supervised representation learning as follows:

Definition 1. *Supervised representation learning aims to learn generic feature extractor* $\mathbf{h} : \mathbb{R}^I \to \mathbb{R}^d$ *by optimizing* $\widehat{L}$ *on labeled dataset* $\mathcal{D}_{\text{sup}}$ *automatically without feature engineering by domain experts.*

Some supervised learning algorithm can be viewed as supervised representation learning. Concretely, DeCAF [Donahue *et al.*, 2014] formulated ImageNet classification as a representation learning task and demonstrated the effectiveness of the learned feature extractor for downstream tasks.

One of the advantages of supervised representation learning is that we obtain feature extractor $\widehat{\mathbf{h}}$ as a by-product of supervised learning. For example, VGG [Simonyan and Zisserman, 2015] trained for ImageNet classification has been widely used for other vision tasks [Girshick *et al.*, 2014].

Empirically large sample size in supervised representation learning improves downstream performance [Sun *et al.*, 2017]. Unfortunately, enlarging the sample size is costly regarding time and money by hiring annotators and teaching them how to annotate data. In addition, we expect that the supervised task is not too easy to capture generic representations for downstream tasks. Intuitively, if we pre-train a model on a difficult task such as ImageNet classification, the model can generalize well to a simpler task, such as MNIST classification. However, the reverse probably does not hold; the pre-trained model on MNIST does not generalize well to ImageNet because the model trained on MNIST could not see complicated patterns during the training to solve ImageNet classification. Creating a dataset for a difficult task, which is ImageNet in the example above, does not only require skilled annotators but also easily contaminates the dataset that could hurt upstream performance [Beyer *et al.*, 2020]. As a result, the pre-trained model performs poorly as a feature extractor for downstream tasks. To overcome this disadvantage, unsupervised representation learning or weakly supervised representation learning [Mahajan *et al.*, 2018; Radford *et al.*, 2021] have been attracting much attention from the machine learning community.[1]

It has been reported that we can often predict the performance of downstream tasks by using the generalization performance in ImageNet when using ImageNet pretraining [Kornblith *et al.*, 2019; Abnar *et al.*, 2022]. However for even supervised representation learning, the best-performed model does not give the best performance on multiple downstream tasks [Abnar *et al.*, 2022]. Similar tendency has been reported in unsupervised representation learning [Ericsson *et al.*, 2021].

2.2 Unsupervised Representation Learning

Unsupervised representation learning[2] does not use label information at all to learn feature extractor $\mathbf{h}$. Suppose unlabeled dataset $\mathcal{D}_{\text{un}} = \{\mathbf{x}_i\}_{i=1}^{M}$, where M is the number of unlabeled samples. The difference from supervised representation learning is that unsupervised representation learning trains feature extractor $\mathbf{h}$ by solving an *unsupervised* task on $\mathcal{D}_{\text{un}}$. To do so, unsupervised representation learning studies reviewed in [Jing and Tian, 2019] proposed a novel unsupervised objective function that is called "pretext task" to train $\mathbf{h}$ without supervised signals. For example, autoencoders [Rumelhart *et al.*, 1986] minimize a reconstruction error as unsupervised loss $\widehat{L}_{\text{un}}$ defined by

$$\widehat{L}_{\text{un}}(\mathbf{h}, \mathbf{g}) = \frac{1}{M}\sum_{i=1}^{M} \left\|\mathbf{g}(\mathbf{h}(\mathbf{x}_i)) - \mathbf{x}_i\right\|_2 . \quad (2)$$

Intuitively, $\mathbf{h}$ compresses $\mathbf{x}$ such that $\mathbf{g}$ recovers $\mathbf{x}$ from $\mathbf{h}(\mathbf{x})$ by minimizing Eq. (2). We expect that such compressed feature representation $\mathbf{h}(\mathbf{x})$ captures useful features of $\mathbf{x}$ to solve other machine learning tasks. As the counterpart of Definition 1, we define unsupervised representation learning as follows:

Definition 2. *Unsupervised representation learning aims to learn generic feature extractor* $\mathbf{h} : \mathbb{R}^I \to \mathbb{R}^d$ *by optimizing* $\widehat{L}_{\text{un}}$ *on unlabeled dataset* $\mathcal{D}_{\text{un}}$ *automatically without feature engineering by domain experts.*

Thanks to the unsupervised nature, we can easily increase the sample size of D_{un} at almost no cost. For example, [Mikolov *et al.*, 2018] trained word representations on 630 billion words collected from `CommonCrawl` and [He *et al.*, 2020] trained self-supervised models on one billion images collected from `Instagram`. This property is desirable because the scale of a dataset is essential to improve the performance of downstream tasks in practice [He *et al.*, 2020; Kaplan *et al.*, 2020]. Surprisingly, even if we train a feature extractor on the same amount of data, unsupervised representation learning gives better transfer performance [Ericsson *et al.*, 2021] and better generalization on out-of-

[1]Due to the space limitation, we do not introduce the weakly supervised representation learning in this review. However, shortly they minimize the supervised loss with auxiliary information such as user-provided tags [Mahajan *et al.*, 2018] or an image description [Radford *et al.*, 2021].

[2]We use unsupervised representation learning and self-supervised representation learning interchangeably.

distribution [Sariyildiz *et al.*, 2021] than supervised representation learning depending on downstream tasks.

One of the disadvantages of unsupervised representation learning is the difficulty of evaluation at the representation phase. We do not even know the existence of a universal unsupervised objective that indicates the minimizer can guarantee downstream performance. As an empirical observation, [Kolesnikov *et al.*, 2019] reported that lower validation loss of representation learning tasks did not imply better validation accuracy on ImageNet classification across different models. Hence, the generalization performance of a downstream task is often used as an evaluation metric in practice, as explained in Section 3.1.

3 Evaluation Methods of Representation Learning

We now organize the current evaluation methods of representation learning algorithms. We do not discuss general evaluation metrics for machine learning algorithms, such as computing efficiency. For future directions of representation learning, we also briefly discuss the theoretical work from our evaluation perspectives.

For all evaluation perspectives except for "Representation Learning as an Auxiliary Task" described in Section 3.4, suppose that we have R pre-trained representation learning models, $\{\widehat{\mathbf{h}}_r\}_{r=1}^{R}$. Given R pre-trained models, we would like to determine the best one.

3.1 Representation Learning for Pre-training

Since representations play an important role in solving machine learning problems, as explained in Section 1, we expect that extracted representations by a representation learning algorithm generalize to unseen machine learning tasks: downstream tasks, such as classification. Motivated by this expectation, the most common evaluation method is how learned representations help solve downstream tasks. In this sense, representation learning can be considered the pre-training [Hinton *et al.*, 2006] of the feature extractor of downstream tasks.

As a running example, we suppose a classification problem for a downstream task. Downstream dataset is denoted $\mathcal{D}_D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_D}$, where N_D is the number of samples and $y_i \in \mathcal{Y}$ is a class label. Suppose that pre-trained model $\widehat{\mathbf{h}}$ is used in the model of the downstream task denoted by $\mathbf{h}_D$. For example, both $\widehat{\mathbf{h}}$ and $\mathbf{h}_D$ are the same neural networks to extract feature representations from $\mathbf{x}$. We use the pre-trained parameters of $\widehat{\mathbf{h}}$ as initialization values of the parameters of $\mathbf{h}_D$. To solve the downstream task, we require an additional function $\mathbf{g}_D$ that maps feature space to label space: $\mathbb{R}^d \to \mathbb{R}^{|\mathcal{Y}|}$ since $\widehat{\mathbf{h}}$ is designed for the representation learning task, not for the downstream task. Note that $\mathbf{g}_D$ tends to be implemented as a simple function such as logistic regression, support vector machines, or shallow neural networks. This is because such a simple $\mathbf{g}_D$ is enough to solve the downstream task if extracted representations already capture discriminative features [Bengio *et al.*, 2013].

Experimental Procedure

Given R pre-trained feature extractors $\{\widehat{\mathbf{h}}_r\}_{r=1}^{R}$ and downstream dataset $\mathcal{D}_D$, we compare the extractors by using the evaluation metric of the downstream task. This is equivalent to treating the feature extractor as a hyperparameter in the model of the downstream task. The evaluation procedures are as follows: i) Train $\mathbf{h}_D, \mathbf{g}_D$ on downstream dataset $\mathcal{D}_D$ with pre-trained feature extractor $\widehat{\mathbf{h}}_r$ for each r. ii) Compare evaluation metric of the downstream task such as validation accuracy. At the first step, there are two common protocols to evaluate feature extractors: "frozen" and "fine-tuning".

Frozen protocol This evaluation protocol has been quite common in recent representation learning experiments. Since we expect that $\widehat{\mathbf{h}}$ can extract discriminative features for the downstream task, we do not train $\mathbf{h}_D$ initialized by $\widehat{\mathbf{h}}$ during the training of the downstream task. Training $\mathbf{g}_D$ requires less computing budget and converges faster than the training of $\mathbf{h}_D$ and $\mathbf{g}_D$ from scratch on the downstream dataset. Formally, we solve the following problem:

$$\min_{\mathbf{g}_D} \widehat{L}_D(\widehat{\mathbf{h}}, \mathbf{g}_D), \tag{3}$$

where $\widehat{L}_D$ is an empirical risk on $\mathcal{D}_D$ such as Eq. (1) for classification. The standard choice of $\mathbf{g}_D$ is a linear classifier [Donahue *et al.*, 2014; He *et al.*, 2020] or non-parametric method, such as k-nearest neighbors. When we use a linear classifier as $\mathbf{g}_D$, the evaluation protocol is also called "linear probing". To attain a further performance gain with additional computing cost, we implement $\mathbf{g}_D$ as a nonlinear model, for example, shallow neural networks with a nonlinear activation function [Bachman *et al.*, 2019].

Fine-tuning protocol To achieve further performance gain for the downstream task, we train both $\mathbf{h}_D$ initialized by $\widehat{\mathbf{h}}$ and $\mathbf{g}_D$ as a single model on the downstream task. This procedure is called "fine-tuning". Formally, we solve the following problem:

$$\min_{\mathbf{h}_D, \mathbf{g}_D} \widehat{L}_D(\mathbf{h}_D, \mathbf{g}_D), \text{ where } \mathbf{h}_D \text{ is initialized by } \widehat{\mathbf{h}}. \tag{4}$$

We might fine-tune $\mathbf{h}_D$ with a smaller learning rate in gradient descent-based optimization than the random initialization. This is because we expect that $\mathbf{h}_D$ with pre-trained weights has already been able to extract useful feature representations for the downstream task. If we set an optimizer's inappropriate hyperparameters, such as too large a learning rate or too many epochs, $\mathbf{h}_D$ is likely to forget the pre-trained weights. As a result, the model overfits $\mathcal{D}_D$. To avoid this explicitly, we can use $\widehat{\mathbf{h}}$ as a regularizer as reviewed in Section 3.2. Even though the fine-tuning protocol requires more computing budget than the frozen protocol, it empirically performs better than the frozen protocol [Zhai *et al.*, 2019].

Efficiency Complementary to the two evaluation protocols, varying the size of the downstream dataset and computing budget is concerned in representation learning experiments [Hénaff *et al.*, 2020]. Intuitively, if feature extractor $\widehat{\mathbf{h}}$ captures discriminative feature representations, training $\mathbf{h}_D$

and $\mathbf{g}_D$ requires fewer labeled data or less computing budget, i.e., fewer epochs in a gradient descent algorithm, than the same model with random initialization to achieve similar generalization performance [Erhan *et al.*, 2010].

Discussion

Fair comparison We need to pay attention to the size of representation learning data and feature extractor $\widehat{\mathbf{h}}$ to compare different representation learning algorithms. For deep neural network-based representation learning algorithms, there exists a positive correlation between the model size of $\widehat{\mathbf{h}}$ and downstream performance, for example, [Kolesnikov *et al.*, 2019] for vision and [Devlin *et al.*, 2019] for language. This phenomenon is known as the scaling-law [Kaplan *et al.*, 2020]. Enlarging the size of data makes this tendency stronger [Kolesnikov *et al.*, 2020]. Suppose we propose a novel representation learning algorithm to improve the state-of-the-art performance on downstream tasks. In this case, we should use the same architecture and dataset to disentangle the factors of performance gain. We also highly recommend following suitable suggestions to representation learning experiments by [Oliver *et al.*, 2018].

Best representations in the layers of neural networks For deep neural network-based models, we have multiple candidates of $\widehat{\mathbf{h}}$ depending on which layer we select as a feature extractor. The optimal feature extractor among the layers differs depending on the downstream task. Concretely, the representations extracted by using until the last layer tend to be specialized for the representation task [Yosinski *et al.*, 2014]. As a result, such $\widehat{\mathbf{h}}$ performs poorly as a feature extractor on the downstream task, especially without fine-tuning. Empirically, removing the last few layers from the network is a common technique to improve the downstream performance [Girshick *et al.*, 2014; Donahue *et al.*, 2014; Bachman *et al.*, 2019]. We recommend trying different intermediate representations as a hyperparameter of the downstream task, especially in the frozen protocol.

Relation to other machine learning problems The described experimental protocols are similar to transfer learning settings. The frozen and fine-tuning protocols are similar to "feature-representation-transfer" and "parameter transfer" in transfer learning [Pan and Yang, 2010], respectively. In transfer learning terminology, we train feature extractor $\mathbf{h}$ on a source domain, and then we transfer pre-trained $\widehat{\mathbf{h}}$ to a target domain, a downstream task. Another formulation is semi-supervised learning [Chapelle *et al.*, 2006], where we train a predictor from many unlabeled data and a few labeled data. Since unsupervised representation learning does not require a labeled dataset, we train $\mathbf{h}$ on the unlabeled data, then train $\mathbf{h}_\mathrm{D}$ and $\mathbf{g}_D$ with $\widehat{\mathbf{h}}$ on the labeled data [Zhai *et al.*, 2019]. We will discuss other representation learning-based approaches for the semi-supervised learning scenario in the other evaluation perspectives described in Sections 3.2 and 3.4.

Benefits for optimization As described above, pre-trained weights of representation learning model $\widehat{\mathbf{h}}$ behave as the initialization of downstream task's model $\mathbf{h}_D$. Since an initialization method is a key factor to improve performance in the gradient-based optimization of deep neural nets [Sutskever *et al.*, 2013], pre-trained models help the optimization of the downstream task. Concretely, we can compare representation learning algorithms in terms of stability [Erhan *et al.*, 2010]. If pre-trained feature extractor $\widehat{\mathbf{h}}$ is the good initialization of $\mathbf{h}_D$, the variance of optimum among multiple runs with different random seeds is smaller than random initialization, which means the pre-trained $\widehat{\mathbf{h}}$ is stable initialization to the randomness of the training for the downstream task.

Theoretical Analysis

A pre-training setting is also a common scenario regarding theoretical analysis in representation learning. Several theoretical analyses show the sample complexity bound of learning theory or the inequality of losses of representation learning and downstream tasks, usually classification by specifying the representation learning task. For example, contrastive learning [Arora *et al.*, 2019], language modeling [Saunshi *et al.*, 2021], unsupervised learning with data-augmentation [Nozawa and Sato, 2021], and references therein. These analyses mainly assume the datasets of representation learning and downstream tasks are the same; they do not cover transfer scenarios. We believe that combining these bounds and transfer learning analysis is worth exploring future directions for theoretical analysis to understand the transferability of representation learning algorithms.

3.2 Representation Learning for Regularization

Even though we fine-tune the weights of $\mathbf{h}_D$ initialized by pre-trained $\widehat{\mathbf{h}}$ as described in Section 3.1, we obtain poor feature extractor $\mathbf{h}_D$ after fine-tuning such that they are far from $\widehat{\mathbf{h}}$ due to inappropriate hyper-parameters: too large learning rate or too many iterations for stochastic gradient-based optimization. As a result, the performance of the downstream task degrades because the model forgets the pre-trained weights to extract useful representations, and the downstream model overfits the downstream dataset. To avoid this, learned feature extractor $\widehat{\mathbf{h}}$ works as the explicit regularizer of $\mathbf{h}_D$ [Li *et al.*, 2018].

Suppose the same notations and classification introduced in Section 3.1. Given pre-trained feature extractor $\widehat{\mathbf{h}}$, we define the loss function for the downstream task with the regularizer of $\mathbf{h}_D(.)$ as follows:

$$\min_{\mathbf{h}_D, \mathbf{g}_D} \widehat{L}_D(\mathbf{h}_D, \mathbf{g}_D) + \frac{\lambda}{N_D} \sum_{i=1}^{N_D} \Omega\left(\mathbf{h}_D(\mathbf{x}_i), \widehat{\mathbf{h}}(\mathbf{x}_i)\right), \quad (5)$$

where $\lambda \in \mathbb{R}_{\geq 0}$ and regularization function $\Omega : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$.[3] A common choice of Ω is the L_2 distance.

We use the parameters of $\widehat{\mathbf{h}}$ for the regularizer of $\mathbf{h}_D$ rather than for only one representation (5). Suppose pre-trained feature extractor $\widehat{\mathbf{h}}$ is modeled by a neural network with J layers. Let feature extractor's parameters be $\widehat{\boldsymbol{\theta}} = \{\widehat{\mathbf{w}}^{(1)}, \widehat{b}^{(1)}, \dots, \widehat{\mathbf{w}}^{(J)}, \widehat{b}^{(J)}\}$, where weights $\mathbf{w}$ and bias b. Similarly, let $\boldsymbol{\theta}$ be parameters in $\mathbf{h}_D$: $\boldsymbol{\theta} =$

[3]Note that this formulation is called "feature-based knowledge distillation" in the distillation context [Gou *et al.*, 2021].

$\{\mathbf{w}^{(1)}, b^{(1)}, \dots, \mathbf{w}^{(J)}, b^{(J)}\}$. For $\boldsymbol{\theta}$, regularization term $\Omega(.,.)$ with L_2 distance is defined as

$$\Omega\left(\boldsymbol{\theta}, \widehat{\boldsymbol{\theta}}\right) = \sum_{j=1}^{J} \left(\|\mathbf{w}^{(j)} - \widehat{\mathbf{w}}^{(j)}\|_2 + \|b^{(j)} - \widehat{b}^{(j)}\|_2 \right). \quad (6)$$

As a special case of Eq. (6), we obtain L_2 regularization when we set $\widehat{\boldsymbol{\theta}} = \mathbf{0}$ instead of pre-trained weights $\widehat{\boldsymbol{\theta}}$. [Li *et al.*, 2018] reported that Eq. (6) performed better on transfer learning tasks than L_2 regularization.

Experimental procedure

Since we use these regularizations to solve a downstream task, the evaluation procedure is the same as in Section 3.1.

Discussion

Compared with Section 3.1, these regularizations consume more memory space, particularly Eq. (6) because the number of parameters doubles for $\mathbf{h}_D$, which might make training infeasible, especially for large parametric models, such as deep neural networks. Therefore fine-tuning protocol with hyper-parameter tuning is a more practical evaluation method than this explicit regularization.

Theoretical Analysis

A similar regularization term to Eq. (6) can be obtained from PAC-Bayesian analysis [McNamara and Balcan, 2017]. Through the lens of the PAC-Bayes analysis, $\widehat{\boldsymbol{\theta}}$ can be considered the prior of $\boldsymbol{\theta}$. If $\widehat{\boldsymbol{\theta}}$ is the good prior of $\boldsymbol{\theta}$, Eq. (6) helps solve the downstream task. In contrast, if we pick poor $\widehat{\boldsymbol{\theta}}$, the regularization hurts the optimization of the downstream task, making optimization unstable or leading to a poor feature extractor.

3.3 Representation Learning for Dimensionality Reduction

Dimensionality reduction [Espadoto *et al.*, 2021] maps a raw data sample into a lower-dimensional space such that the mapped representation preserves important information of the original data sample. Representation learning works as dimensionality reduction when the dimensionality of extracted feature representation d is smaller than the dimensionality of the original input I. Indeed, matrix factorization-based dimensionality reduction algorithms are compared with unsupervised representation learning [Perozzi *et al.*, 2014; Baroni *et al.*, 2014] to extract feature vectors.

Data visualization can be viewed as a special case of dimensionality reduction when extracted feature representations are in $\mathbb{R}^2$ or $\mathbb{R}^3$, where a human can recognize features visually. Related to this, it is also common to apply a visualization algorithm to extracted feature representations $\left\{\widehat{\mathbf{h}}(\mathbf{x}_i)\right\}_{i=1}^{N_D}$ rather than directly learn feature representations in $\mathbb{R}^2$ or $\mathbb{R}^3$ to see samples with the same label are close to others [Donahue *et al.*, 2014].

Experimental Procedure

Suppose R pre-trained feature extractors $\{\widehat{\mathbf{h}}_r\}_{r=1}^{R}$ and downstream dataset $\mathcal{D}_D$. Note that we might not require the labels of $\mathcal{D}_D$ depending on the evaluation metric.

Dimensionality reduction i) Extract feature representations with each feature extractor $\widehat{\mathbf{h}}_r$ from $\mathcal{D}_D$. ii) Compare sets of extracted representations using evaluation metric for dimensionality reduction [Espadoto *et al.*, 2021].

Visualization i) Extract feature representations with each feature extractor $\widehat{\mathbf{h}}_r$ from $\mathcal{D}_D$. ii) If $d > 3$, apply a dimensionality reduction algorithm to each set of extracted feature representations for visualization. iii) Compare visualized features, usually with scatter plots.

Discussion

To our best knowledge, numerical evaluation of representation learning as dimensionality reduction is not common. However, investigating what information is implicitly embedded in learned representations is an interesting analysis and actively performed to understand extracted feature representations. For example, [Adi *et al.*, 2017] the existence of the word, or order of two words, in order to understand sentence feature extractors. Intuitively, if the properties are embedded into extracted sentence representations, the model can predict the properties accurately.

Numerical evaluation of visualization is challenging because it tends to require human evaluation. This is why representation learning papers [Donahue *et al.*, 2014; Kornblith *et al.*, 2019] showed only generated figures without numerical evaluation on visualization. Another problem is that visualization algorithms generate different plots depending on their hyper-parameters. For example, t-SNE algorithm gives largely different visualization results depending on its hyper-parameters [Wattenberg *et al.*, 2016]. Hence we do not encourage evaluating representation learning algorithms using visualizations without careful the hyper-parameters tuning of visualization algorithms. Another approach is to use label information if we access a label of each sample by following t-SNE's theoretical analysis [Arora *et al.*, 2018]. In this case, we can use the similar evaluation protocol discussed in Section 3.1 with a simple classifier such as a linear classifier.

Theoretical Analysis

Since representation learning can be seen as dimensionality reduction, researchers try to explain what information is embedded using a representation learning algorithm by specifying its loss function. Matrix factorization-based analysis [Levy and Goldberg, 2014] is a common approach, especially for word representation learning. [Kong *et al.*, 2020] unified widely used algorithms in natural language processing from a mutual information perspective, including more recent models such as BERT [Devlin *et al.*, 2019]. Similarly, a specific unsupervised loss function can be interpreted as a lower bound of mutual information [Oord *et al.*, 2018].

3.4 Representation Learning as Auxiliary Task

This evaluation perspective differs from the others. We focus on a representation learning algorithm itself rather than pre-trained feature extractor $\widehat{\mathbf{h}}$. Since representation learning attempts to learn generic feature representations from a dataset, the representation learning algorithm is expected to improve another machine learning algorithm's performance by optimizing its loss and the loss of the downstream task simultaneously or cyclically. For example, supervised learning [Islam *et al.*, 2021], semi-supervised learning [Weston *et al.*, 2008; Zhai *et al.*, 2019], and reinforcement learning [Oord *et al.*, 2018]

Suppose the same notations and classification introduced in Section 3.1. Recall that we aim to learn a classifier that consists of feature extractor $\mathbf{h}_D$ and classification head $\mathbf{g}_D$ by minimizing supervised loss L_D, e.g., cross-entropy loss (1). Let L_{aux} be a representation learning's loss, e.g., mean squared loss (2) for an auto-encoder-based representation learning algorithm and $\mathbf{g}_{\mathrm{aux}}$ be a representation learning specific projection head, e.g., the decoder of the auto-encoder model. Suppose that the supervised and representation learning models share $\mathbf{h}_D$. Formally, we optimize the following loss function to solve supervised loss L_{D} with representation learning loss L_{aux}:

$$\min_{\mathbf{h}_D, \mathbf{g}_D, \mathbf{g}_{\mathrm{aux}}} L_{\mathrm{D}}(\mathbf{h}_D, \mathbf{g}_D) + \beta L_{\mathrm{aux}}(\mathbf{h}_D, \mathbf{g}_{\mathrm{aux}}), \tag{7}$$

where predefined coefficient $\beta \in \mathbb{R}_{\geq 0}$.

Experimental Procedure

Suppose R representation learning algorithms. For each representation learning algorithm, we optimize Eq. (7). We select the best one using the evaluation metric, such as validation accuracy.

Variety of the dataset for representation learning There exist several ways to calculate representation learning loss L_{aux}. The simplest way is to use the same labeled dataset, D_D [Islam *et al.*, 2021]. Other ways are to use an additional unlabeled dataset [Luong *et al.*, 2016] or the union of labeled and unlabeled datasets [Weston *et al.*, 2008; Zhai *et al.*, 2019]. When we use an unlabeled dataset, the unlabeled dataset can come from the same data distribution [Zhai *et al.*, 2019] or different data distribution [Luong *et al.*, 2016].

Discussion

This formulation can be seen as multi-task learning whose tasks are the combination of a supervised task and a representation learning task. Compared with the other evaluations, this evaluation is easy to tune hyperparameters of the representation learning algorithm because we can search the hyper-parameters in the single training stage. Recall that pre-training in Section 3.1 and regularization in Section 3.2 require two-stage training: training the representation learning model and training the model of the downstream task. Moreover, the final performance tends to be better than pre-training in practice [Zhai *et al.*, 2019].

Theoretical Analysis

[Le *et al.*, 2018] showed the stability bound of a supervised loss with a linear auto-encoder used for the auxiliary task. Similarly, [Garg and Liang, 2020] proposed sample complexity bounds for unsupervised representation learning and supervised learning losses. In their analysis, intuitively, representation learning can work as a learnable regularizer to reduce the hypothesis size of the supervised model. [Maurer *et al.*, 2016] gave sample complexity bounds for multi-task learning and learning-to-learn via shared feature extractor $\mathbf{h}_D$ for all tasks.

4 Conclusion and Future Directions

Representation learning trains a feature extractor that automatically extracts generic feature representations from a dataset. Unlike existing representation learning's survey papers, we reviewed four evaluation methods of representation learning algorithms to understand the current representation learning applications. We conclude this review by discussing the future directions based on Vapnik's principle.

A famous principle [Vapnik, 2000] to solve a problem says

When solving a given problem, try to avoid solving
a more general problem as an intermediate step.

Regarding the common evaluations in representation learning, representation learning seems to oppose Vapnik's principle. For example, suppose a binary image classification: dog versus cat, as a downstream task. We should not need a feature extractor that can distinguish the difference between Birman and Ragdoll, which are quite similar cat species, to solve the downstream task by following the principle. However, we impose such ability on representation learning because it learns the generic feature extractor from a massive dataset for unseen downstream tasks. In this sense, Vapnik's principle is inapplicable to representation learning. Therefore we believe that we need a different metric to evaluate representation learning algorithms rather than the performance of a single downstream task, such as validation accuracy on ImageNet-1K. One possible solution is to measure the averaged performance among diverse downstream tasks, such as VTAB [Zhai *et al.*, 2020] for vision or SuperGLUE [Wang *et al.*, 2019] for language. This idea can be generalized to modal-agnostic evaluation as discussed in [Tamkin *et al.*, 2021].

More pessimistically, solving the downstream task via representation learning, especially two-stage training, is less effective than solving the problem directly with comprehensive hyper-parameter tuning. We expect the learned representations to capture redundant features to solve the downstream task, i.e., distinguishing between Birman and Ragdoll. Such unnecessary expressiveness could hurt downstream tasks' performance. In transfer learning terminology, a negative transfer might cause this ineffectiveness. However, unsupervised representation learning has advantages compared with supervised learning, such as robustness to class imbalance [Liu *et al.*, 2022] or generalization to unseen classes [Sariyildiz *et al.*, 2021]. Hence we reach the same future direction as in the previous paragraph: *can we develop suitable evaluation metrics rather than only a single metric of a downstream task for representation learning?*

Acknowledgements

This work is partially supported by Next Generation AI Research Center, The University of Tokyo. We thank Han Bao and Yoshihiro Nagano for constructive discussion and for suggesting relevant work. We thank anonymous reviewers and editor of IJCAI-ECAI 2022 Survey track for giving constructive suggestions to improve our manuscript.

References

[Abnar *et al.*, 2022] Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the Limits of Large Scale Pre-training. In *ICLR*, 2022.

[Adi *et al.*, 2017] Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In *ICLR*, 2017.

[Arora *et al.*, 2018] Sanjeev Arora, Wei Hu, and Pravesh K. Kothari. An Analysis of the t-SNE Algorithm for Data Visualization. In *COLT*, 2018.

[Arora *et al.*, 2019] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A Theoretical Analysis of Contrastive Unsupervised Representation Learning. In *ICML*, 2019.

[Bachman *et al.*, 2019] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning Representations by Maximizing Mutual Information Across Views. In *NeurIPS*, 2019.

[Baroni *et al.*, 2014] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't Count, Predict! A Systematic Comparison of Context-counting vs. Context-predicting Semantic Vectors. In *ACL*, 2014.

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *TPAMI*, 2013.

[Beyer *et al.*, 2020] Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are We Done with ImageNet? *arXiv:2006.07159v1 [cs.CV]*, 2020.

[Chapelle *et al.*, 2006] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-supervised Learning*. MIT Press, 2006.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.

[Donahue *et al.*, 2014] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *ICML*, 2014.

[Erhan *et al.*, 2010] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *JMLR*, 2010.

[Ericsson *et al.*, 2021] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How Well Do Self-Supervised Models Transfer? In *CVPR*, 2021.

[Espadoto *et al.*, 2021] Mateus Espadoto, Rafael M. Martins, Andreas Kerren, Nina S.T. Hirata, and Alexandru C. Telea. Toward a Quantitative Survey of Dimension Reduction Techniques. *TVCG*, 2021.

[Garg and Liang, 2020] Siddhant Garg and Yingyu Liang. Functional Regularization for Representation Learning: A Unified Theoretical Perspective. In *NeurIPS*, 2020.

[Girshick *et al.*, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *CVPR*, 2014.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[Gou *et al.*, 2021] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge Distillation: A Survey. *IJCV*, 2021.

[He *et al.*, 2020] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*, 2020.

[Hénaff *et al.*, 2020] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-Efficient Image Recognition with Contrastive Predictive Coding. In *ICML*, 2020.

[Hinton *et al.*, 2006] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 2006.

[Islam *et al.*, 2021] Ashraful Islam, Chun-Fu Richard Chen, Rameswar Panda, Leonid Karlinsky, Richard Radke, and Rogerio Feris. A Broad Study on the Transferability of Visual Representations with Contrastive Learning. In *ICCV*, 2021.

[Jing and Tian, 2019] Longlong Jing and Yingli Tian. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. *TPAMI*, 2019.

[Kaplan *et al.*, 2020] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. *arXiv:2001.08361v1 [cs.LG]*, 2020.

[Kolesnikov *et al.*, 2019] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting Self-Supervised Visual Representation Learning. In *CVPR*, 2019.

[Kolesnikov *et al.*, 2020] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big Transfer (BiT): General Visual Representation Learning. In *ECCV*, 2020.

[Kong *et al.*, 2020] Lingpeng Kong, Cyprien de Masson D'Autume, Wang Ling, Lei Yu, Zihang Dai, and Dani Yogatama. A Mutual Information Maximization Perspective of Language Representation Learning. In *ICLR*, 2020.

[Kornblith *et al.*, 2019] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do Better ImageNet Models Transfer Better? In *CVPR*, 2019.

[Le *et al.*, 2018] Lei Le, Andrew Patterson, and Martha White. Supervised Autoencoders: Improving Generalization Performance with Unsupervised Regularizers. In *NeurIPS*, 2018.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In *NeurIPS*, 2014.

[Li *et al.*, 2018] Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit Inductive Bias for Transfer Learning with Convolutional Networks. In *ICML*, 2018.

[Liu *et al.*, 2022] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised Learning is More Robust to Dataset Imbalance. In *ICLR*, 2022.

[Luong *et al.*, 2016] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task Sequence to Sequence Learning. In *ICLR*, 2016.

[Mahajan *et al.*, 2018] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the Limits of Weakly Supervised Pretraining. In *ECCV*, 2018.

[Maurer *et al.*, 2016] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The Benefit of Multitask Representation Learning. *JMLR*, 2016.

[McNamara and Balcan, 2017] Daniel McNamara and Maria-Florina Balcan. Risk Bounds for Transferring Representations With and Without Fine-Tuning. In *ICML*, 2017.

[Mikolov *et al.*, 2018] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in Pre-Training Distributed Word Representations. In *LREC*, 2018.

[Nozawa and Sato, 2021] Kento Nozawa and Issei Sato. Understanding Negative Samples in Instance Discriminative Self-supervised Representation Learning. In *NeurIPS*, 2021.

[Oliver *et al.*, 2018] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. In *NeurIPS*, 2018.

[Oord *et al.*, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748v2 [cs.LG]*, 2018.

[Pan and Yang, 2010] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *TKDE*, 2010.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *KDD*, 2014.

[Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021.

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Internal Representations by Error Propagation*. MIT Press, 1986.

[Sariyildiz *et al.*, 2021] Mert Bulent Sariyildiz, Yannis Kalantidis, Diane Larlus, and Karteek Alahari. Concept Generalization in Visual Representation Learning. In *ICCV*, 2021.

[Saunshi *et al.*, 2021] Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A Mathematical Exploration of Why Language Models Help Solve Downstream Tasks. In *ICLR*, 2021.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

[Sun *et al.*, 2017] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *ICCV*, 2017.

[Sutskever *et al.*, 2013] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the Importance of Initialization and Momentum in Deep Learning. In *ICML*, 2013.

[Tamkin *et al.*, 2021] Alex Tamkin, Vincent Liu, Rongfei Lu, Daniel Fein, Colin Schultz, and Noah Goodman. DABS: a Domain-Agnostic Benchmark for Self-Supervised Learning. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[Vapnik, 2000] Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory, Second Edition*. Statistics for Engineering and Information Science. Springer, 2000.

[Wang *et al.*, 2019] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *NeurIPS*, 2019.

[Wattenberg *et al.*, 2016] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to Use t-SNE Effectively. *Distill*, 2016.

[Weston *et al.*, 2008] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep Learning via Semi-Supervised Embedding. In *ICML*, 2008.

[Yosinski *et al.*, 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How Transferable Are Features in Deep Neural Networks? In *NeurIPS*, 2014.

[Zhai *et al.*, 2019] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S^{4}L: Self-Supervised Semi-Supervised Learning. In *ICCV*, 2019.

[Zhai *et al.*, 2020] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A Large-scale Study of Representation Learning with the Visual Task Adaptation Benchmark. *arXiv:1910.04867v2 [cs.CV]*, 2020.