

# From Parameter Estimation to Contrastive Learning

Haiyang Xu

PB20000326

University of Science and Technology of China

xuhaiyang@mail.ustc.edu.cn

## Abstract

*This report discusses two parameter estimation methods: Noise Contrastive Estimation(NCE) and its variant InfoNCE. Moreover, this report discusses several classic papers in Contrastive Learning which used InfoNCE loss. Meanwhile, this report makes detailed reviews and comparisons of these papers: what makes good research and how we can follow those excellent researchers.*

## 1. Main Structure Overview

### section 1: Main Structure Overview

#### section 2: Introduction

**2.1 :** Introduce Maximum Likelihood Estimation(MLE) and Maximum Posterior Probability(MAP), which are preparations for learning NCE and InfoNCE.

**2.2 :** Firstly, introduce self-supervised learning and its important composition——pretext task. Then, introduce two kinds of pretext tasks: generative method and contrastive learning method. Moreover, point out contrastive learning's task and its difficulty: construct positive and negative samples, construct loss function.

#### section 3: NCE and InfoNCE

**3.1.1 :** Define what problem NCE is trying to solve: computing softmax when the total number of categories are huge is a very expensive and time-

consuming task.

**3.1.2 :** Discover the core idea of NCE: converts the multi-category classification problem into a binary classification problem by learning the difference between data distribution samples and noisy distribution samples.

**3.1.3 :** Detailed derivation process of NCE.

**3.2 :** From NCE to InfoNCE

## section 4: Contrastive Learning

### 4.1 : InstDisc

This paper puts forward the contrastive learning task of Instance Discrimination: see every single figure as one single category. To solve the complexity of softmax, this paper introduce InfoNCE as the loss. This report discusses not only discusses its highlight but also its remained issues.

### 4.2 : InvaSpread

This paper follows the instance discrimination task, puts forward a different method of selecting positive and negative samples. This report discusses not only its highlight but also its remained issues.

### 4.3 : MoCo

This paper puts forward two innovations: a queue to solve the batchsize problem, and a momentum encoder to solve the incompatible problem in InstDisc. This report discusses what makes good research and what we can learn from the great researcher in computer vision: Kaiming He.

### 4.4 : SimCLR

This paper puts forward a projector to improve the performance of the method in InvaSpread. Meanwhile, rich computational resources prove to have great importance. This report discusses what makes good research.

## section 5: Conclusion

## 2. Introduction

### 2.1. Parameter Estimation

To introduce NCE and InfoNCE, we should first be familiar with two most frequently used parameter estimation method: **MLE** and **MAP**.

Figure 1 is Prof. Liu's slides [1] of Maximum Likelihood Estimation(MLE), which is clear and easy to understand.

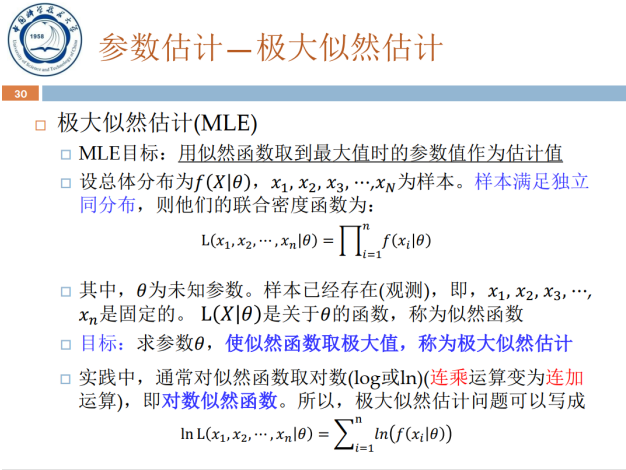


图 1. Prof. Liu's slides of MLE

Maximum likelihood estimation(MLE) is a very widely used method for parameter estimation. It is also based on a simple principle: using a known sample, find the parameter that is most likely to generate that sample.

Let  $x_1, x_2, \dots, x_n$  be observations from  $n$  independent and identically distributed random variables drawn from a Probability Distribution  $f_0$ , where  $f_0$  is known to be from a family of distributions  $f$  that depend on some parameters  $\theta$ . For example,  $f_0$  could be known to be from the family of normal distributions

$f$ , which depend on parameters  $\sigma$  (standard deviation) and  $\mu$  (mean), and  $x_1, x_2, \dots, x_n$  would be observations from  $f_0$ . The goal of MLE is to maximize the likelihood function:

$$L = f(x_1, x_2, \dots, x_n | \theta) \\ = f(x_1 | \theta) \times f(x_2 | \theta) \times \dots \times f(x_n | \theta)$$

Often, the average log-likelihood function is easier to work with:

$$\hat{\ell} = \frac{1}{n} \log L = \frac{1}{n} \sum_{i=1}^n \log f(x_i | \theta)$$

Figure 2 is Prof. Liu's slides [1] of Maximum posterior probability(MAP), which is also clear and easy to understand.

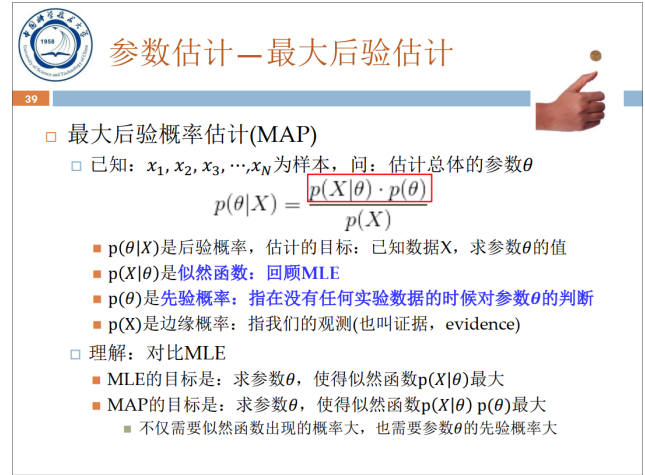


图 2. Prof. Liu's slides of MAP

The MAP estimate of the random variable  $X$ , given that we have observed  $Y = y$ , is given by the value of  $x$  that maximizes  $f_{X|Y}(x | y)$  if  $X$  is a continuous random variable,  $P_{X|Y}(x | y)$  if  $X$  is a discrete random variable.

To find the MAP estimate, we need to find the value of  $x$  that maximizes

$$f_{X|Y}(x | y) = \frac{f_{Y|X}(y | x)f_X(x)}{f_Y(y)}.$$

Note that  $f_Y(y)$  does not depend on the value of  $x$ . Therefore, we can equivalently find the value of  $x$  that maximizes

$$f_{Y|X}(y | x)f_X(x).$$

This can simplify finding the MAP estimate significantly, because finding  $f_Y(y)$  might be complicated.

## 2.2. Contrastive Learning

To talk about contrastive learning, we should first start with unsupervised learning and self-supervised learning.

Unsupervised learning, which is highly regarded by Lecun(see Figure 3), is a kind of machine learning method does not require manually labeled category information. And self-supervised learning is a type of unsupervised learning paradigm. Work such as BEiT [2] and MAE [3], which have been very hot last year, have demonstrated that self-supervised learning can outperform supervised learning methods on some specific tasks.

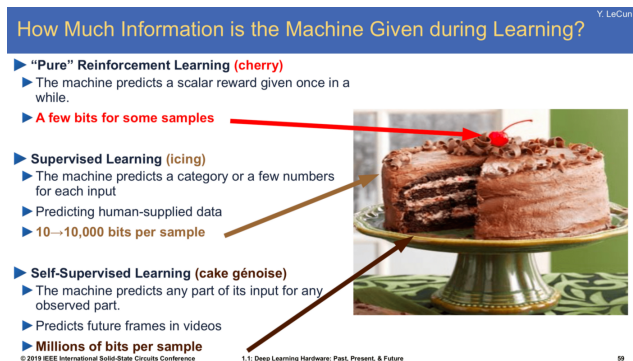


图 3. Facebook AI Chief Yann LeCun introduced his now-famous “cake analogy” at NIPS 2016: “If intelligence is a cake, the bulk of the cake is unsupervised learning, the icing on the cake is supervised learning, and the cherry on the cake is reinforcement learning (RL).”

Self-supervised learning directly uses the data itself as supervised information to learn feature representations of sample data and use them for downstream tasks. Most of the current mainstream approaches to machine learning are supervised learning methods, which rely on manually labeled labels, which can have a number of drawbacks.

The data itself provides much more information than sparse labels, so training models using supervised learning methods requires a large amount of la-

beled data, and the resulting models are sometimes “fragile”. Supervised learning tends to learn only task-specific knowledge, but not a general knowledge, so the feature representations learned by supervised learning are difficult to transfer to other tasks. However, Self-supervised learning avoids this problem because it directly uses the data itself to provide supervised information to guide learning.

Then, without using the labels, how do self-supervised learning use the data? The answer is to design a good **pretext task**, i.e. auxiliary tasks. For example, in the field of computer vision, reconstruct image pixels could be taken as an pretext task, that is,  $Loss$  is the  $MSE_{loss}$  between the input original image  $x$  and the reconstructed image  $x_{hat}$ . This kind of method is called **generative method**, because they generate features for self-supervised learning.

Besides generative method, there is another kind of pretext task, which is to select positive and negative samples (we will see the definition of positive and negative samples later in this report), and the  $Loss$  requires the features of positive and negative samples to be as far away as possible. This is called: **Contrastive Learning**.

Contrastive learning is to learn the feature representation of a sample by comparing the data with positive and negative samples in the feature space. Contrastive Methods only need to learn the distinguishability on the feature space. Thus Contrastive Methods do not focus too much on pixel details, but are able to focus on abstract semantic information, and the optimization becomes simpler compared to pixel-level reconstruction.

The main difficulties in contrastive learning are how to construct positive and negative samples and how to construct the loss function. This report will focus closely on these two issues, first introducing the commonly used loss function in contrast learning: InfoNCE loss and its predecessor NCE parameter estimation method in Section 3; then introducing several of the most classic papers in contrast learning in Section

4, discussing their methods for selecting positive and negative samples.

### 3. NCE and InfoNCE

I learned a lot about NCE and InfoNCE from these blogs: [4], [5]

#### 3.1. NCE

##### 3.1.1 Task background

First, let's see the task background, i.e., what problem NCE aims to solve: huge numbers of categories resulting in complex softmax calculation.

First, let's see a language model. Consider a sequence  $s = \{w_1, w_2, \dots, w_m\}$  of words  $w$ , and the language model is to get a  $p(w_1, w_2, \dots, w_m)$  to calculate the probability of occurrence of this sentence  $s$ . Intuitively we want to get this language model based on the chain rule of conditional probability:

$$\begin{aligned} p(w_1, \dots, w_m) &= p(w_1) * p(w_2 | w_1) * \dots * p(w_m | w_1, \dots, w_{m-1}) \\ &= \prod_{i=1}^m p(w_i | w_1, w_2, \dots, w_{i-1}) \\ &= \prod_{i=1}^m p(w_i | c_i) \end{aligned}$$

In the language model, the Markov hypothesis is introduced, i.e., "the probability of a word occurring is related to  $n$  words occurring before it", and the  $n$  words are called a gram, so the model can be simplified as:

$$p(w_1, w_2, w_3, \dots, w_m) = \prod_{i=1}^m p(w_i | w_{i-n+1}, \dots, w_{i-1})$$

we can write its log-likelihood function as follows using maximum likelihood estimation:

$$\mathcal{L}_{MLE} = \sum_{w_i \in s} \log p_{\theta}(w_i | c_i)$$

Then maximize the log-likelihood function  $\mathcal{L}_{MLE}$ , where that  $p(w | c)$  is viewed as a function of  $w$  and  $c$ , with  $\theta$  being the set of parameters to be determined.

$$p_{\theta}(w | c) = F(w, c; \theta)$$

Thus, once the optimal set of parameters  $\theta^*$  can be determined, the function  $F$  will also be determined,

and then for any probability  $p(w | c)$  can be computed with the function  $F(w, c; \theta^*)$ .

Here, we use a neural network to fit  $F$ . Given the current context  $c$  and a parameter set of  $\theta$ , we can directly compute the probability of the next word  $w$ . Assuming that the result before input to softmax is denoted by  $s_{\theta}(w, c)$ , then the conditional probability can be expressed in the following form:

$$\begin{aligned} p_{\theta}(w | c) &= \frac{\exp(s_{\theta}(w, c))}{\sum_{w' \in V} \exp(s_{\theta}(w', c))} \\ &= \frac{u_{\theta}(w, c)}{Z(c)} \end{aligned}$$

However, the number of word bases  $|V|$  is often very large, so **computing  $Z(c)$  is a very expensive and time-consuming task**, which is the problem NCE has to solve.

##### 3.1.2 Core Idea

The core idea of NCE is to discover some characteristics in the data by **learning the difference between data distribution samples and noisy distribution samples**. Because this method relies on comparison with noisy data, it is called "Noise Contrastive Estimation" (NCE). **In this way, NCE converts the multi-category classification problem into a binary classification problem, where the classifier is able to differentiate the data samples and the noise samples, but not to learn the complex distribution of huge numbers of data sample categories**. Meanwhile, the parameter  $\theta$  of this classifier is equivalent to the distribution of the data samples we want to obtain. (It can be proved in [6])

##### 3.1.3 Derivation Process

OK, let's start the long but interesting derivation of NCE.

Now suppose that the data distribution for a particular context  $c$  is  $\tilde{p}(w | c)$  and we call the sample taken from it a positive sample, such that the category

$D = 1$ ; and another noisy distribution unrelated to  $c$  is  $q(w)$  and we call the sample taken from it a negative sample, such that the category  $D = 1$ . Now,  $k_d$  positive samples and  $k_n$  negative samples are now taken out, and these samples are mixed to form a mixed distribution  $p(w | c)$ .

We obtain the following probabilities:

$$\begin{aligned} p(D = 1) &= \frac{k_d}{k_d + k_n} \\ p(D = 0) &= \frac{k_n}{k_d + k_n} \\ p(w | D = 1, c) &= \tilde{p}(w | c) \\ p(w | D = 0, c) &= q(w) \end{aligned}$$

The MAP can be calculated:

$$\begin{aligned} p(D = 0 | w, c) &= p(D = 0)p(w | D = 0, c) \\ &= \frac{p(D = 0)p(w | D = 0, c) + p(D = 1)p(w | D = 1, c)}{p(D = 0)p(w | D = 0, c) + p(D = 1)p(w | D = 1, c)} \\ &= \frac{\frac{k_n}{k_d + k_n} \times q(w)}{\frac{k_d}{k_d + k_n} \times \tilde{p}(w | c) + \frac{k_n}{k_d + k_n} \times q(w)} \\ &= \frac{\frac{k_n}{k_d} \times q(w)}{\tilde{p}(w | c) + \frac{k_n}{k_d} \times q(w)} \\ p(D = 1 | w, c) &= p(D = 1)p(w | D = 1, c) \\ &= \frac{p(D = 0)p(w | D = 0, c) + p(D = 1)p(w | D = 1, c)}{p(D = 0)p(w | D = 0, c) + p(D = 1)p(w | D = 1, c)} \\ &= \frac{\frac{k_d}{k_d + k_n} \times \tilde{p}(w | c)}{\frac{k_d}{k_d + k_n} \times \tilde{p}(w | c) + \frac{k_n}{k_d + k_n} \times q(w)} \\ &= \frac{\tilde{p}(w | c)}{\tilde{p}(w | c) + \frac{k_n}{k_d} \times q(w)} \end{aligned}$$

Let the ratio of negative and positive samples be:

$$k = \frac{k_n}{k_d}$$

, then we have:

$$\begin{aligned} p(D = 0 | w, c) &= \frac{k \times q(w)}{\tilde{p}(w | c) + k \times q(w)} \\ p(D = 1 | w, c) &= \frac{\tilde{p}(w | c)}{\tilde{p}(w | c) + k \times q(w)} \end{aligned}$$

NCE is to replace  $\tilde{p}(w | c)$  with  $p_\theta(w | c)$  so that the posterior probability becomes a function about  $\theta$ . However, the problem is that this form now still requires the computation of  $Z(c)$ . To solve this problem, NCE makes two assumptions:

1.  $Z(c)$  is regarded as a parameter  $z_c$ , which is equivalent to other parameters in neural network.

2. It turns out ([7]) that for neural networks with many parameters, we fix  $z_c$  to 1 for each  $c$  is still valid.

Thus the posterior probability can be written in the following form:

$$\begin{aligned} p_\theta(D = 0 | w, c) &= \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)} \\ p_\theta(D = 1 | w, c) &= \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} \end{aligned}$$

Now we have the binary classification problem with parameter  $\theta$ , and assuming that the label  $D_t$  is Bernoulli distributed. Using MLE, we can write conditional log likelihood  $\mathcal{L}_{NCE}^c$  as follows:

$$\begin{aligned} \mathcal{L}_{NCE}^c &= \sum_{t=1}^{k_d+k_n} D_t \log P(D = 1 | w_t, c) \\ &\quad + \sum_{t=1}^{k_d+k_n} (1 - D_t) \log P(D = 0 | w_t, c) \\ &= \sum_{t=1}^{k_d} \log P(D = 1 | w_t, c) + \sum_{t=1}^{k_n} \log P(D = 0 | w_t, c) \\ &= \sum_{t=1}^{k_d} \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} + \sum_{t=1}^{k_n} \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)} \end{aligned}$$

This function needs to be divided by the number of positive samples  $k_d$

$$\begin{aligned} J_{NCE}^c &= \frac{1}{k_d} \left[ \sum_{t=1}^{k_d} \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} + \sum_{t=1}^{k_n} \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)} \right] \\ &= \frac{1}{k_d} \sum_{t=1}^{k_d} \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} \\ &\quad + \frac{1}{k_d} \sum_{t=1}^{k_n} \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)} \\ &= \frac{1}{k_d} \sum_{t=1}^{k_d} \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} \\ &\quad + \frac{k}{k_n} \sum_{t=1}^{k_n} \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)} \end{aligned}$$

When the number of data is large, according to the law of large numbers, the above equation can also

be written as

$$\begin{aligned}
J_{NCE}^c &= \frac{1}{k_d} \sum_{t=1}^{k_d} \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} \\
&+ \frac{k}{k_n} \sum_{t=1}^{k_n} \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)} \\
&= \mathbb{E}_{w \sim \tilde{p}(w|c)} \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} \\
&+ k \mathbb{E}_{w \sim q(w)} \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)}
\end{aligned}$$

Finally, maximize the above log-likelihood function!

$$\begin{aligned}
J_{NCE}^c &= \mathbb{E}_{w \sim \tilde{p}(w|c)} \log \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)} \\
&+ k \mathbb{E}_{w \sim q(w)} \log \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)}
\end{aligned}$$

This is the final form of NCE target function!

### 3.1.4 Concise Conclusion of NCE

The number of classes in the softmax calculation might be too large, so NCE simply reduced the classification to a binary classification problem—data and noise. We are happy with the binary classification—the probability could be easily (the idea and method are simple, though the process is not very easy) estimated directly by MLE.

## 3.2. InfoNCE

In fact, InfoNCE is the introduction of NCE to contrastive learning, which is a specific variant of NCE.

Let's explain InfoNCE more specifically, based on the ideas provided in NCE:

1. the data taken from the condition  $p(x_{t+k} | c_t)$  is called a "positive sample", which is the predicted data made from the context  $c_t$ , and it is formed into a "positive sample pair" together with this context, with the category label set to 1.

2. The sample taken from  $p(x_{t+k})$  is called "negative sample", which is the random data not necessarily related to the current context  $c_t$ , and it forms a "negative sample pair" together with this context  $c_t$ , with the category label set to 0.

So all we have to do is to train a logistics classification model to distinguish between these two pairs of positive and negative samples.

According to the setting in NCE, we give  $X = \{x_1, \dots, x_N\}$  of size  $N$ , which contains 1 positive sample sampled from  $p(x_{t+k} | c_t)$  and  $N - 1$  negative samples sampled from a noisy distribution  $p(x_{t+k})$  sampled from a negative sample. Assuming that the first  $x_i$  is a positive sample and the following  $i = t + k$  samples is the context  $c_t$ , then the possibility when the positive sample  $x_{t+k}$  and the  $N - 1$  negative sample can be found correctly at the same time can be written in the following form

$$\begin{aligned}
p(d = i | X, c_t) &= p(x_{t+k} | c_t) \\
&= \frac{p(x_{t+k} | c_t) \prod_{l \neq t+k} p(x_l)}{\sum_{j=1}^N p(x_j | c_t) \prod_{l \neq j} p(x_l)} \\
&= \frac{\frac{p(x_{t+k} | c_t)}{p(x_{t+k})}}{\sum_{j=1}^N \frac{p(x_j | c_t)}{p(x_j)}}
\end{aligned}$$

Maximize the above equation, therefore maximize the model's ability to successfully discriminate between each positive and negative sample.

Then we can write a prediction of the form  $x_{t+k}$  based on  $c_t$

$$p(x_{t+k} | c_t) = \frac{\exp(s_\theta(x_{t+k}, c_t))}{\sum_{x_j \in X} \exp(s_\theta(x_j, c_t))}$$

In the above equation, we know that  $s_\theta(x, c)$  is a scoring function, and the output score is used to quantify the match of  $x$  in the context  $c$ ;  $s_\theta(x_{t+k}, c_t)$  is also to quantify the similarity between the predicted and true results for  $x_{t+k}$ . Quantify the cosine similarity and defines  $\exp(s_\theta(x_{t+k}, c_t))$  as  $f_k(x_{t+k}, c_t)$ , that is

$$p(x_{t+k} | c_t) = \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}$$

$f_k(x_{t+k}, c_t)$  can actually be used as a representation of the density ratio  $\frac{p(x_{t+k}|c_t)}{p(x_{t+k})}$ , which is not directly equivalent but positively related in meaning, i.e.

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k} | c_t)}{p(x_{t+k})}$$

Now our optimization goal is to maximize the result of the above equation, so we can write the corresponding form of CE(cross entropy) loss as follows

$$\begin{aligned}\mathcal{L}_N &= -\sum_X \left[ p(x, c) \log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)} \right] \\ &= -\mathbb{E}_X \left[ \log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)} \right]\end{aligned}$$

This is InfoNCE loss!

## 4. Contrastive Learning

After a hard, long bunch of mathematical formula derivation, let's relax and enjoy a few papers that use InfoNCE loss.

The sequence of included papers is based on the development path of contrastive learning:

1. InstDisc [8]
2. InvaSpread [9]
3. MoCo v1 [10]
4. SimCLR v1 [11]

Let's go!

### 4.1. InstDisc [8]

InstDisc is from the paper: Unsupervised Feature Learning via Non-Parametric **I**nstance **D**iscrimination.

InstDisc is an early paper to define a brand-new contrastive learning task—**Instance Discrimination**. And it uses the InfoNCE loss, which we have introduced before.

Its main inspiration is shown in Figure 4. When the input image is a leopard, labels like leopard and snow leopard are ranked higher not because these images have similar labels, but because these images just look more alike —cheetahs and snow leopards just look similar, and bookshelves just don't. So this can lead to easy misclassification (e.g. splitting cheetah into snow leopard). How to solve this problem? InstDisc proposes a simple and violent way: regard each picture as one specific category —instance discrimination—and then hope that the each picture's representation in the representation space are as far away

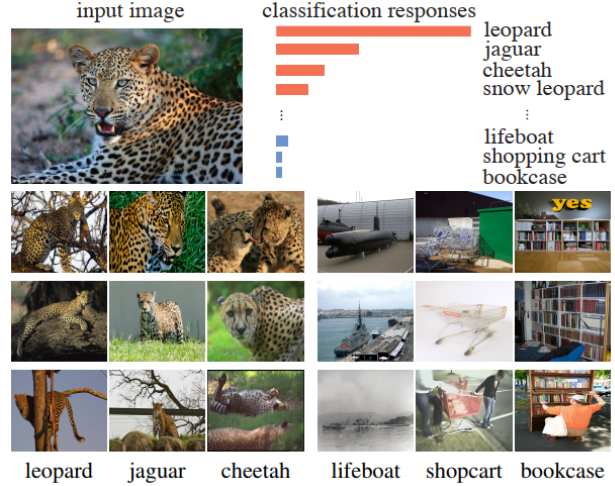


图 4. InstDisc finds out that, if you feed a leopard image to a classifier that has been trained with supervised learning, you will find that all the top categories are related to leopards (e.g. cheetah and snow leopard), while the bottom categories are often not related to leopards at all.

as possible (see the representation space ball in Figure 5, i.e., we want the feature vectors to be uniformly distributed on this ball). In this way, ideally, the higher ranked image will be only "individual class" to which the image belongs, and the other classes will be more distant from it, so it will not be easy to misclassify. Contrastive Learning is reflected in: these "individual classes" are their own **positive samples**, while all the other images serve as **negative samples**.

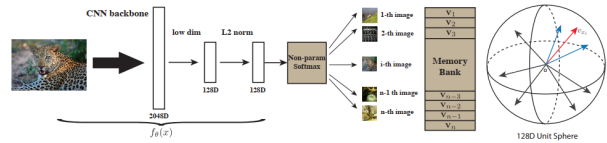


图 5. Framework of InstDisc

The method is also relatively simple, the figure gets 128D(dimension) vector after the encoder, and this 128D vector is the positive sample in contrastive learning. Here there is another concept **Memory Bank**, which is actually the positive sample and all other images in the dataset (128w-1 images in the case of Im-



ageNet). Then each time we do a Loss, we randomly sample  $K$  images from the Memory Bank as negative samples. After each epoch of loss, all the figures sampled in this epoch are updated with the updated encoder. Simple Method, right?

The loss here is InfoNCE Loss, which we have discussed before:

$$L_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)} \quad (1)$$

Now, I want to explain this loss from the perspective of contrastive learning but not probability distribution (though it's inferred from probability distribution, which we have discussed before). Here  $q$  is the input query picture and  $k_+$  is it in the memory bank (i.e., the positive sample). The denominator of  $k_i$  is all the samples in the memory bank (one positive sample +  $n-1$  negative samples). The  $K$  is the number of negative samples taken. This is very similar to softmax and is well understood. We want  $q \cdot k_+$  to be as large as possible (the distance between positive samples and query is as small as possible) and  $q \cdot k_i$  to be as small as possible (the distance between positive samples and query is as small as possible), so that the loss becomes smaller, which means the distance between different embeddings in the feature space becomes larger, this is what we want. Here  $\tau$  is the temperature coefficient, the larger the value the smoother the distribution, the smaller the distribution the sharper it is, a hyperparameter used to control the convergence of the model.

In my opinion, this paper explicitly uses contrastive learning in a computer vision task and designs a simple but effective selection method of positive and negative samples —instance discrimination— and then trying to separate positive and negative samples. It is also innovative in that it is based on the similarity of visual semantic information and uses a simple but effective Loss, InfoNCE Loss.

However, this also raises some questions for me:

This memory bank, which includes 128w numbers, seems to be too large. And updating it would require a lot of computational resources and computational time.

And the big sample number cannot take full advantage of InfoNCE. Can we find a better way?

## 4.2. InvaSpread [9]

InvaSpread is from the paper: Unsupervised Embedding Learning via **I**nvariant and **S**preading Instance Feature.

Follow the instance discrimination task, InvaSpread puts forward a different method of selecting positive and negative samples—minibatch pairs data augmentation.

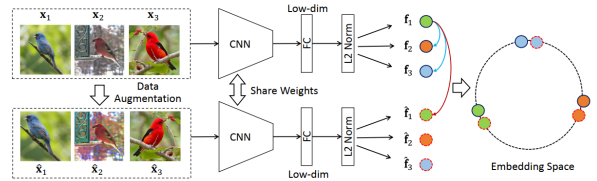


图 6. Framework of InvaSpread

The main inspiration point of InvaSpread is the same as InstDisc. It is stated that the embedding (Same meaning of "representation", but this paper calls it as embedding, so we follow) should be the same for the same image (Instance) by encoder, and different for different images. For similar images their embedding should remain as invariant (similar) as possible, and the embedding of different classes of images should be spreading (dissimilar). Well, this paper and InstDisc are two descriptions of the same thing.

The framework of InvaSpread is shown in Figure 6, and its choice of positive and negative examples is quite different from InstDisc. In comparison with InstDisc, InvaSpread treats each minibatch as a small "memory bank", say 256 images. Then, we do data augmentation (cropping) on each image, so that we get 512 images. The loss used is also InfoNCE loss.

After reading InstDisc, the key idea of InvaSpread is very well understood. It features a random minibatch and uses the minibatch as the sampling range for positive and negative samples, which avoids the very large memory bank and the pressure on GPU memory



caused by the update of 128w sample features after each round of back propagation. At the same time, it introduces data enhancement operation, which greatly enhances the robustness of contrastive learning, that is, the same image should be projected to the same feature space even in different angles and different forms.

However, there still raise some questions to me, in which I'd like to point out one:

It's right that 256 avoids the computation and restoration complexity, but at the same time it is a small number. It's highly possible that it will cause unbalance between different minibatches. That is to say, every time it updates the negative samples, it only updates 256 samples. This will cause somewhat shortcut—the query tends to match the negative sample created with it in the same time, but not the semantically similar one. It's a terrible problem!

Following I will introduce the most famous two models in contrastive learning: **MoCo** and **SimCLR**. MoCo is a great improvement following InstDisc, and SimCLR is a great improvement following InvaSpread. The two models will show us what makes good work:

1. creative idea for applying seemingly natural method to new problem(Personally, I think this approach is more worthy of following, which I think is elegant and high-class scientific research)
2. rich computational resources(That's you, the big models engaged by the big AI companies!)

### 4.3. MoCo v1 [10]

MoCo v1 is from the paper: **Momentum Contrast** for Unsupervised Visual Representation Learning.

This paper put forward two innovations: a queue to solve the batchsize problem, and a momentum encoder to solve the incompatible problem in InstDisc.

It is led by Kaiming He. A joke says that when this man touches a specific region, then the region will definitely show great potential and value.

MoCo v1 completely follows InstDisc, but Kaiming summarizes the problem of contrastive learning as a dictionary query problem and creatively proposes to

use Queue and Momentum Encoder to do contrastive learning, as shown in Figure 7.

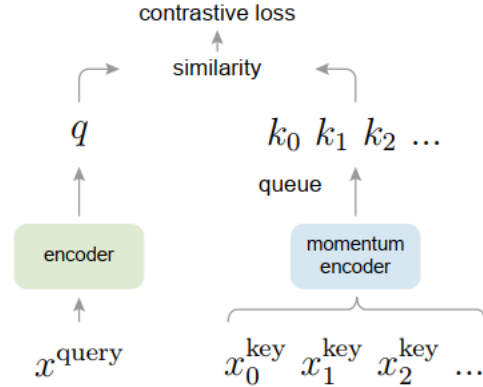


图 7. Framework of MoCo v1

It is still structured as the image is passed through encoder to get 128D embedding for InfoNCE loss calculation. It is required that, the cosine distance between query and positive samples(Kaiming regard the samples as key) is as small as possible, while that between query and negative keys is as large as possible. As if to answer my doubts: InstDisc's memory bank is too big for the hardware, and the sample features may not be aligned between different minibatches of InvaSpread? Kaiming says, I will create a large and consistent dictionary at the same time!

Kaiming believes that by making the dictionary larger, we can sample more of the high-dimensional feature space, and the more keys in the dictionary, the richer the visual information will be, and it will be easier to find the essential features that are distinguishable in the loss process; by making the dictionary consistent, we mean that all the keys in the dictionary should be generated by the same or similar encoder. If the keys in the dictionary are obtained by different encoders, then query is likely to find the key generated by the same or similar encoder as query, instead of the key with similar visual information, which leads to shortcut solution.

Based on this, in brief, MoCo's innovation point is mainly:

1. Create a Queue to update the dictionary. The queue is stored in Hard Disc, and when doing the forward propagation, a minibatch is firstly sampled from the queue, and then put on the GPU for calculation. In this way, the size of the dictionary is not limited by the GPU memory, and our dictionary (i.e.,  $K$  in InstDisc) can be made larger, which means the sample region will be greater, thus solving the problem of Large; at the same time, the first in first out feature of the queue allows the earliest minibatch entering the queue to also getting out of the queue the earliest. Then the encoder after the update will encode the new embeddings of the minibatch, which will enter the queue afterwards. It is an effective dynamic update, which partially solves the problem of consistency.

2. Create two encoders, one is the normal encoder of query side, which is used to extract the embedding of query, that is, positive samples; the other is the momentum encoder of momentum update, which is used to extract the embedding of negative samples. Here the momentum encoder solves the problem of consistency from another perspective: the backpropagation after loss calculation is only applied to the normal encoder; while the momentum encoder is the one that does the momentum update  $\theta_k = m \cdot \theta_{k-1} + (1 - m) \cdot \theta_q$ , that is,  $\theta_k$  is initially initialized by the encoder  $\theta_q$  on the query side, but if a relatively large momentum is added, then  $\theta_k$  is updated more slowly, which also ensures relative consistence between different minibatches and will not lead to serious inconsistency between minibatches.

Considerately, Kaiming also gives a diagram of the different ways of encoder backpropagation, see Figure 8. (a) end to end is actually what InvaSpread does, ran-

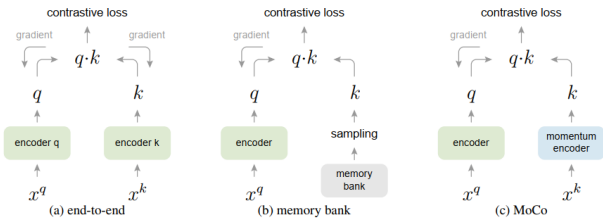


图 8. Different methods of encoder updates

domly taking minibatch as a dictionary, using it as a positive and negative sample for contrastive learning, and then updating it, but this leads to inconsistency problems; (b) memory bank is actually what InstDisc does, but requires huge memory (c) MoCo is the approach of this paper, which uses Queue and momentum encoder to achieve both Large and consistency.

After reading this article I was very impressed that Kaiming solved the problem by addressing the pain points of contrastive learning. Is his idea very fancy? Queue and momentum are not at all the kind of work that is similar to Transformer’s breakthrough type. So why is it still so appreciated and evocative? **I think this article is a good guide for our research work, for the pain points of the research content, more to try some seemingly CLASSIC methods, and apply them to the problem, sometimes SIMPLE but efficient!**

This is actually true of much of Kaiming’s work, and I really enjoy following Kaiming’s work. For example, his classic Resnet [12]. What was the pain point of Deep Learning at that time? Gradient dispersion and vanishing problem. Kaiming said, ”That’s easy, let’s add a constant mapping to the network structure, so that it will definitely add a 1 to the gradient, which means that in the worst case, the network will not learn anything, but it will not learn worse as the number of layers of the network increases, which is the perfect solution to the problem of gradient vanishing problem!” Another example is the MAE [3], Kaiming first introduced the method of ”mask” from the field of NLP(natural language processing) to the field of CV. As early as two years ago BERT proposed a pre-training method called MIM to mask the elements in a sentence and then predict them. So what are the pain points in the CV field that cannot take advantage of the method? Kaiming points out that it is because the inductive bias of CNN and the wrong setting of mask ratio. Inductive bias of CNN is an advantage, but it is a disadvantage in mask, because it is not global, and a lot of information dropped in mask often affects not

only the partial space, but also the information of the whole picture, which leads to poor performance of CNN in mask context. So what should we do? Kaiming utilizes ViT as the encoder, which proves to be success. Then Kaiming move his sight to mask ratio. BERT’s mask ratio is only 15%, while Kaiming’s mask ratio in MAE [3] reaches 75%! Why? ”The semantic information density contained in pictures is much lower than that of natural language”. In other words, the small mask ratio is difficult enough for NLP characters, such as ”I go \_”, do I go ”homing”, or go ”swimming”? But for CV tasks it’s too easy because the embedded semantic information is so low. For example, if you mask off the arm of a person swinging a tennis ball, it’s very easy to guess by the person and the ball that it’s the arm that should fill in the place. So, Kaiming says, let’s increase the difficulty and crank up the mask ratio! It’s like masking off the person and the ball, leaving only the arm, and let the model come back to predict. This requires the model to have a strong learning ability.

I think the important thing when doing this Research Report is not only the collection of information or the evaluation of the information, **but more importantly, our own thinking about these papers.** For example, after reading a lot of Kaiming’s work, when I saw MoCo, I saw the ”Kaiming style” of thinking about and solving problems: **Hold tightly to the pain point of the problem, then look for existing and classical methods to see if they can be used for that problem. In a sense, this will have a much greater impact on my future research work than the two innovative ideas proposed in the MoCo article itself.**

So, what else can make good work besides creative idea? The SimCLR introduced below answers this question: RICH COMPUTATIONAL RESOURCES.

#### 4.4. SimCLR v1 [11]

SimCLR is from the paper: A **S**imple Framework for **C**ontrastive **L**earning of Visual **R**epresentations.

This paper puts forward a projector to improve

the performance of the method in InvaSpread. Meanwhile, rich computational resources prove to have great importance.

SimCLR is an work of art created by Google Brain Team. Its **art** has a sense of beauty of (as in the title) and violence——incredibly rich computational resources. Meanwhile, of course it contains innovation, not just showing Google’s richness of TPU.

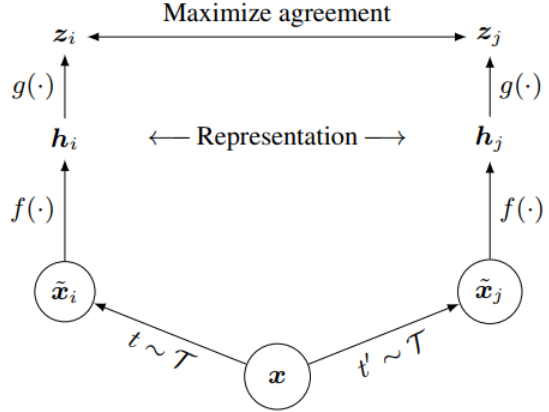


图 9. Framework of SimCLR

First, let’s look at the SimCLR framework, see Figure 9. Isn’t this the InvaSpread framework? Input image  $x$ , sample  $\tau$  positively, sample  $\tau'$  negatively, then pass an encoder to get the representation  $h$ . Here’s the Innovation: the representation  $h$  is mapped to a low-dimensional representation  $z$  by a function  $g(\cdot)$ , and then the  $z$  is applied to InfoNCE loss.

It turns out that, compared to InvaSpread, the innovations of SimCLR, in addition to the added projection function  $g(\cdot)$ , are:

1. SimCLR does more data augmentation. SimCLR tries very many ways of data augmentation, including crop and resize, color distort(drop), color distort(jitter), rotate, cutout(mask), Gaussian noise, Gaussian blur... As in Figure 10; Then, Google trenchantly did ablation experiments, and the ablation experiments for data augmentation alone are shown in Figure 11; which proves the necessity of random cropping and random color transformation, and other data

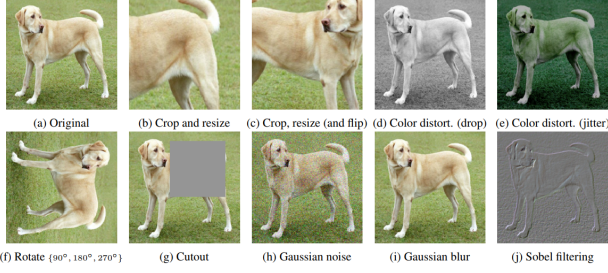


图 10. Different Data Augmentation

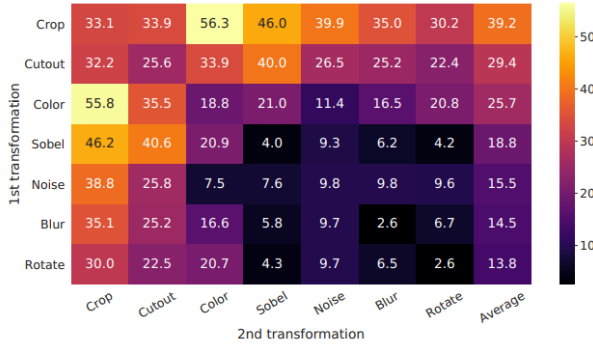


图 11. Ablation experiment of data augmentation

augmentation is just icing on the cake or optional.

2. SimCLR uses a larger batchsize (8192, while InvaSpread is 256), and the training epoch is much longer (100, while InvaSpread is 40). The degree of treacherousness is shown in Figure 12 below: Then Google

To keep it simple, we do not train the model with a memory bank (Wu et al., 2018; He et al., 2019). Instead, we vary the training batch size  $N$  from 256 to 8192. A batch size of 8192 gives us 16382 negative examples per positive pair from both augmentation views. Training with large batch size may be unstable when using standard SGD/Momentum with linear learning rate scaling (Goyal et al., 2017). To stabilize the training, we use the LARS optimizer (You et al., 2017) for all batch sizes. We train our model with Cloud TPUs, using 32 to 128 cores depending on the batch size.<sup>2</sup>

图 12. experimental configuration

Brain said, it works, and we trained so quickly that we could run the whole experiment in an hour and a half, as follows in Figure 13;

The first time I read this article, I felt that Google Brain was showing off its rich resources. Later, after

<sup>2</sup>With 128 TPU v3 cores, it takes  $\sim 1.5$  hours to train our ResNet-50 with a batch size of 4096 for 100 epochs.

图 13. TPU use

reading the article twice more, plus the recent research and attempts on models such as BEiT [2], MAE [3], I have a deeper understanding: the projection function  $g(\cdot)$ , which seems to be important.

We note that BEiT [2] and MAE [3] take the same approach as SimCLR (unlike all other papers in this report) when performing self-supervised pre-training: when training, encoder and decoder (I think the projection function  $g(\cdot)$  in SimCLR can be seen as a simple decoder, which is important) are utilized when training, and only encoder, not decoder, is utilized when doing inference. Is it a coincident? I don't think so. The design of a simple decoder compared to a complex encoder has been shown to work well in self-supervised pretraining, but we don't know why yet. In fact, self-supervised learning itself has some mysteries for the design of pretext tasks, and it is impossible to theoretically justify the best choice of pretext tasks, but different pretext tasks have a great impact on the performance of the downstream task inference —for example, InfoNCE loss in contrastive learning is a classic pretext task's loss —bringing similar samples closer together in the sample space and pushing unrelated samples farther away. From my perspective, this is what self-supervised learning can explore, and is also where the appeal of self-supervised learning lies.

## 5. Conclusion

Knowledge of mathematics, especially Probability and Statistics, is very important for machine learning work. To solve the problem of complexity in calculating softmax with huge numbers of categories, NCE is put forward. It aims to discover some characteristics in the data by learning the difference between data distribution samples and noisy distribution samples. In this way, NCE converts the multi-category classification problem (learn the complex distribution of all the

categories to calculate the softmax) into a binary classification problem(whether the sample is data sample or noise sample). Afterwards, its variant, InfoNCE is frequently used in Contrastive Learning.

From 2019, Contrastive Learning is keeping on developing very fast. Besides the classic papers this report introduces, more and more good papers exist, like SWaV [13], which introduce the concept of "Prototype" to Contrastive Learning; SimSiam [14], which even doesn't use negative samples directly; MoCo v3 [15], which applies the backbone of Transformer; HCSC [16], which combines Instance loss and Prototype loss together... I've finished reading this papers and learned a lot. I believe that after reading this report everyone will have a clear comprehension of what contrastive learning is, what's the difficulty of contrastive learning, how did former researchers solve the problems, what we can learn from them, etc.

**Keep on studying Mathematics, then we might rethink seemingly difficult problems and apply innovative methods based on math knowledge.**

**Keep on learning Machine Learning and following great researchers. Not only their professional knowledge is worthy learning from, but also their thinking patterns.**

## 参考文献

- [1] QiLiu. Prof. qiliu's slides on ad2022. <http://staff.ustc.edu.cn/~qiliuql/files/AD2022/3.2.pdf>. 2022-03-31. 2
- [2] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3, 12
- [3] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 3, 10, 11, 12
- [4] 山上的小酒馆. Moco 中的 infonce. [https://blog.csdn.net/weixin\\_45104951/article/details/123551076](https://blog.csdn.net/weixin_45104951/article/details/123551076). 2022-03-17. 4
- [5] Lethe. Noise contrastive estimation 前世今生——从 nce 到 infonce. <https://zhuanlan.zhihu.com/p/334772391>. 2020-12-08. 4
- [6] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2), 2012. 4
- [7] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012. 5
- [8] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742, 2018. 7
- [9] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019. 7, 8
- [10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 7, 9
- [11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 7, 11
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 10
- [13] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020. 13
- [14] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 13

- [15] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021. [13](#)
- [16] Yuanfan Guo, Minghao Xu, Jiawen Li, Bingbing Ni, Xuanyu Zhu, Zhenbang Sun, and Yi Xu. Hcsc: Hierarchical contrastive selective coding. *arXiv preprint arXiv:2202.00455*, 2022. [13](#)