Noise Contrastive Method for Visual and Graph Representation Learning: A Note

Jiaxu Liu *
Department of Electrical Engineering and Electronics
University of Liverpool, UK
jiaxu.liu@liverpool.ac.uk

Abstract

We aim to give a gentle introduction on contrastive self-supervised learning, the first two sections are intuition and derivation of Noise Contrastive Estimation (NCE) and Contrastive Predictive Coding (CPC) followed by the InfoNCE loss. This note also included some case studies on visual and graph contrastive learning, with the association of these models with the CPC approach and their intuitive and mathematical working principles.

1 Noise-Contrastive Estimation

1.1 Preliminary

A sample $X = (\mathbf{x}_1, ..., \mathbf{x}_{N_d})$ of a random vector $\mathbf{x} \in \mathbb{R}^n$ is observed which follows an unknown probability density function (pdf) p_d (data pdf). We use a parameterized family of functions $\{p_m(\cdot; \boldsymbol{\theta})\}_{\boldsymbol{\theta}}$ where $\boldsymbol{\theta}$ is a vector of parameters. We assume that p_d belongs to this family, *i.e.*, there exist optimum $\boldsymbol{\theta}^\star$ such that $p_d = p_m(\cdot; \boldsymbol{\theta}^\star)$. Therefore, the parametric density estimation problem is about finding $\boldsymbol{\theta}^\star$ from the given (observed) data X. Any solution $\hat{\boldsymbol{\theta}}$ to this estimation problem must yield a properly normalized density $p_m(\cdot; \hat{\boldsymbol{\theta}})$, where *normalized* model is defined by for all $\boldsymbol{\theta}$

$$\int p_m(\mathbf{u}; \boldsymbol{\theta}) d\mathbf{u} = 1, \qquad p_m(\cdot; \boldsymbol{\theta}) \geq 0. \tag{1}$$

If a model is satisfied the latter positivity constraint rather than the normalization constraint, we say the model is *unnormalized*, denoted by $p^0_m(\cdot; \boldsymbol{\alpha})$. Since the model does not essentially integrates to one, we define the partition function $Z(\boldsymbol{\alpha})$ as

$$Z(\boldsymbol{\alpha}) = \int_\Omega p^0_m(\mathbf{u}'; \boldsymbol{\alpha}) d\mathbf{u}' \tag{2}$$

such that an unnormalized model can be converted to a normalized model by

$$p_m(\mathbf{u}; \boldsymbol{\theta}) = \frac{p^0_m(\mathbf{u}; \boldsymbol{\alpha})}{\int p^0_m(\mathbf{u}'; \boldsymbol{\alpha}) d\mathbf{u}'} = \frac{p^0_m(\mathbf{u}; \boldsymbol{\alpha})}{Z(\boldsymbol{\alpha})}. \tag{3}$$

We formalize the density estimation problem as maximum likelihood estimation, where the log-likelihood is defined as

$$\mathcal{L}_{\text{MLE}} = \mathbb{E}_{\mathbf{u} \sim p_d} \ln p_m(\mathbf{u}; \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{u} \sim p_d} \ln \frac{p^0_m(\mathbf{u}; \boldsymbol{\alpha})}{Z(\boldsymbol{\alpha})}, \tag{4}$$

*If there is any problem with the theoretical explanation of this note, or if there is any request to include some extra models, please contact the author directly via email.

Preprint. Under review.

we optimize the log-likelihood in accordance with the parameter $\boldsymbol{\theta}$ (or $\boldsymbol{\alpha}$ equally), since p_m^0 normally cannot be integrate analytically, $Z(\boldsymbol{\alpha})$ is therefore computational expensive. For low-dimensional $\mathbf{u}$, we can use techniques like Monte Carlo integration to approximate $Z(\boldsymbol{\alpha})$, however such numerical integration can have large variance for high-dimensional problems. To tackle with such infeasibility of estimating unnormalized models, Noise-Contrastive Estimation (NCE) [Gutmann and Hyvärinen, 2012] is proposed where the objective function aims to learn the difference between the observed data and manually crafted noise data, such that we find the latent patterns within observed data. NCE method leverage noise which the data is contrasted to, hence called "noise-contrastive".

1.2 NCE Objective

We first introduce the general settings of parametric density estimation used in Noise-Contrastive Estimation. Assume we have the observed data X and noise data Y are respectively sample $(\mathbf{x}_1, ..., \mathbf{x}_{N_d})$ of random variable $\mathbf{x} \in \mathbb{R}^n$ with pdf p_d and sample $(\mathbf{y}_1, ..., \mathbf{y}_{N_n})$ of random variable $\mathbf{y} \in \mathbb{R}^n$ with pdf p_n. We normally cannot directly infer p_d, while if we know the difference between X and Y, *i.e.*, the ratio p_d/p_n, and we know the reference (noise) distribution p_n, we can obtain p_d.

We use $U = (\mathbf{u}_1, ..., \mathbf{u}_{N_d+N_n})$ to denote the union of X and Y. Assign to each data point $\mathbf{u}_t$ a binary class C_t, if $C_t = 1$ then $\mathbf{u}_t \in X$, otherwise if $C_t = 0$ then $\mathbf{u}_t \in Y$. The class prior probabilities can be defined as

$$p(C = 1) = \frac{N_d}{N_d + N_n}, \qquad p(C = 0) = \frac{N_n}{N_d + N_n}.$$

The class-conditional probability of $C = 0$ is by definition the pdf of the noise data $\mathbf{y}$, *i.e.*

$$p(\mathbf{u}|C = 0) = p_n(\mathbf{u}),$$

since the pdf of data $\mathbf{x}$ (p_d) is unknown, we model the conditional probability $p(\cdot|C = 1)$ with the parameterized density function $p_m(\cdot; \boldsymbol{\theta})$, *i.e.*

$$p(\mathbf{u}|C = 1; \boldsymbol{\theta}) = p_m(\mathbf{u}; \boldsymbol{\theta}).$$

We use ν to denote the ratio of noise density against data $p(C = 0)/p(C = 1) = N_n/N_d$. We can therefore derive the posterior probability for the data class

$$\begin{aligned}
p(C = 1|\mathbf{u}; \boldsymbol{\theta}) &= \frac{p(\mathbf{u}|C = 1; \boldsymbol{\theta})p(C = 1)}{\sum_C p(\mathbf{u}|C)p(C)} \\
&= \frac{p(\mathbf{u}|C = 1; \boldsymbol{\theta})p(C = 1)}{p(\mathbf{u}|C = 1; \boldsymbol{\theta})p(C = 1) + p(\mathbf{u}|C = 0)p(C = 0)} \\
&= \frac{p_m(\mathbf{u}; \boldsymbol{\theta})\frac{N_d}{N_d+N_n}}{p_m(\mathbf{u}; \boldsymbol{\theta})\frac{N_d}{N_d+N_n} + p_n(\mathbf{u})\frac{N_n}{N_d+N_n}} \\
&= \frac{N_d p_m(\mathbf{u}; \boldsymbol{\theta})}{N_d p_m(\mathbf{u}; \boldsymbol{\theta}) + N_n p_n(\mathbf{u})} \\
&= \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})},
\end{aligned} \tag{5}$$

and posterior for the noise class

$$\begin{aligned}
p(C = 0|\mathbf{u}; \boldsymbol{\theta}) &= 1 - p(C = 1|\mathbf{u}; \boldsymbol{\theta}) \\
&= 1 - \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \\
&= \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})}.
\end{aligned} \tag{6}$$

Assume the class indicator C is a Bernoulli random variable, the conditional log-likelihood is given by

$$\begin{aligned}\mathcal{L}_{\text{NCE}}(\boldsymbol{\theta}) &= \ln \prod_{t=1}^{N_d+N_n} p(C_t=1|\mathbf{u}_t;\boldsymbol{\theta})^{C_t} p(C_t=0|\mathbf{u}_t;\boldsymbol{\theta})^{(1-C_t)} \\ &= \sum_{t=1}^{N_d+N_n} \{C_t \ln p(C_t=1|\mathbf{u}_t;\boldsymbol{\theta}) + (1-C_t)\ln p(C_t=0|\mathbf{u}_t;\boldsymbol{\theta})\} \\ &= \sum_{t=1}^{N_d} \ln p(C_t=1|\mathbf{u}_t;\boldsymbol{\theta}) + \sum_{t=1}^{N_n} \ln p(C_t=0|\mathbf{u}_t;\boldsymbol{\theta})\end{aligned} \tag{7}$$

Recall the binary cross-entropy loss defined in [Bishop, 2006]

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}, \tag{8}$$

where t_n denotes the binary indicator and y_n is the class probability, we observe that the negative NCE log-likelihood $-\mathcal{L}_{\text{NCE}}$ is simply the form of binary cross entropy loss. Thus in such way, a density estimation problem (unsupervised learning) can be performed by logistic regression (supervised learning).

1.3 NCE for Unnormalized Statistical Model

We have discussed about the Noise-Contrastive learning for density estimation, note that the above model only works when the parameterized function p_m is naturally normalized, however, most parameterized functions are not normalized, *e.g.*, arbitrary score function, neural networks. The NCE method makes it feasible to train the unnormalized model to predict the data density by leveraging the following assumptions.

Recall the conversion of unnormalized model to normalized model

$$p_m(\mathbf{u};\boldsymbol{\theta}) = \frac{p_m^0(\mathbf{u};\boldsymbol{\alpha})}{Z(\boldsymbol{\alpha})},$$

where $Z(\boldsymbol{\alpha}) = \int p_m^0(\mathbf{u};\boldsymbol{\alpha})d\mathbf{u}$ is the normalization term, we now consider Z not anymore a function of $\boldsymbol{\alpha}$ but a additional parameter $c = \ln 1/Z$ of normalized model, that is, we define the model as

$$p_m(\mathbf{u};\boldsymbol{\theta}) = \frac{p_m^0(\mathbf{u};\boldsymbol{\alpha})}{Z} \iff \ln p_m(\mathbf{u};\boldsymbol{\theta}) = \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) + c \tag{9}$$

where $\boldsymbol{\theta} = (\boldsymbol{\alpha}, c)$. The parameter c gurantee that the unnormalized model p_m^0 is scaled and satisfies the integrate-to-one property in Equation (1). Inline with such notation, we have the loss function $\mathcal{J}_{\text{NCE}}$ (continuing Equation (7)):

$$\begin{aligned}\mathcal{J}_{\text{NCE}}(\boldsymbol{\theta}) &= \mathcal{L}_{\text{NCE}}(\theta)/N_d \\ &= \frac{1}{N_d}\sum_{t=1}^{N_d} \ln p(C_t=1|\mathbf{u}_t;\boldsymbol{\theta}) + \frac{1}{N_d}\sum_{t=1}^{N_n} \ln p(C_t=0|\mathbf{u}_t;\boldsymbol{\theta}) \\ &= \frac{1}{N_d}\sum_{t=1}^{N_d} \ln \frac{p_m(\mathbf{u};\boldsymbol{\theta})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \frac{1}{N_d}\sum_{t=1}^{N_n} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \\ &= \frac{1}{N_d}\sum_{t=1}^{N_d} \ln \frac{p_m(\mathbf{u};\boldsymbol{\theta})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \frac{\nu}{N_n}\sum_{t=1}^{N_n} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})}.\end{aligned} \tag{10}$$

Following the weak law of large numbers [Leove, 1977], that is, when the number of negative samples $N_d \to \infty$ we have

$$\mathcal{J}_{\text{NCE}}(\boldsymbol{\theta}) \xrightarrow{P} \mathcal{J}(\boldsymbol{\theta}) \implies \sup_{\boldsymbol{\theta}} |\mathcal{J}_{\text{NCE}}(\boldsymbol{\theta}) - \mathcal{J}(\boldsymbol{\theta})| \xrightarrow{P} 0 \tag{11}$$

where

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})} \ln \frac{p_m(\mathbf{u};\boldsymbol{\theta})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \nu \mathbb{E}_{\mathbf{u}\sim p_n(\mathbf{u})} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})}. \tag{12}$$

This NCE transformed optimization objective $\mathcal{J}$ is essentially an approximation to the maximum likelihood estimation, and the conclusion is that such approximation becomes more accurate as the ratio ν between the number of noise (negative) and data (positive) samples increases, which explains why the authors suggest that we set ν as large as possible (normally one positive against many negative). Lets first consider the data likelihood in Equation (4), as we want to maximize the likelihood and $\boldsymbol{\theta}$ is identical to $\boldsymbol{\alpha}$ in meaning, we calculate the derivative with respect to $\boldsymbol{\alpha}$ (which is identical to with respect to $\boldsymbol{\theta}$)

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\alpha}} \mathcal{L}_{\text{MLE}} &= \mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\alpha}} \ln \frac{p_m^0(\mathbf{u};\boldsymbol{\alpha})}{Z(\boldsymbol{\alpha})} \\
&= \mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\alpha}} \{\ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) - \ln Z(\boldsymbol{\alpha})\} \\
&= \mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) - \frac{\partial}{\partial \boldsymbol{\alpha}} \ln Z(\boldsymbol{\alpha}),
\end{aligned} \tag{13}$$

since the reference space Ω in $Z(\boldsymbol{\alpha}) = \int_\Omega p_m^0(\mathbf{u}';\boldsymbol{\alpha}) d\mathbf{u}'$ is the space of parameterized function $p_m(\cdot;\boldsymbol{\theta})$ which has not correlated with empirical distribution p_d, thus remove the expectation $\mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})}$. The latter term $\frac{\partial}{\partial \boldsymbol{\alpha}} \ln Z(\boldsymbol{\alpha})$ can be further derived as

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\alpha}} \ln Z(\boldsymbol{\alpha}) &= \frac{1}{Z(\boldsymbol{\alpha})} \frac{\partial}{\partial \boldsymbol{\alpha}} Z(\boldsymbol{\alpha}) \\
&= \frac{1}{Z(\boldsymbol{\alpha})} \frac{\partial}{\partial \boldsymbol{\alpha}} \int_{\Omega(\mathbf{u})} p_m^0(\mathbf{u};\boldsymbol{\alpha}) d\mathbf{u} \\
&= \int_{\Omega(\mathbf{u})} \frac{1}{Z(\boldsymbol{\alpha})} \frac{\partial}{\partial \boldsymbol{\alpha}} p_m^0(\mathbf{u};\boldsymbol{\alpha}) d\mathbf{u} \\
&= \int_{\Omega(\mathbf{u})} p_m(\mathbf{u};\boldsymbol{\theta}) \frac{1}{p_m^0(\mathbf{u};\boldsymbol{\alpha})} \frac{\partial}{\partial \boldsymbol{\alpha}} p_m^0(\mathbf{u};\boldsymbol{\alpha}) d\mathbf{u} \\
&= \mathbb{E}_{\mathbf{u}\sim p_m(\mathbf{u};\boldsymbol{\theta})} \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}),
\end{aligned} \tag{14}$$

replace the latter term in Equation (13) with the result in Equation (14) gives

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\alpha}} \mathcal{L}_{\text{MLE}} &= \mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) - \frac{\partial}{\partial \boldsymbol{\alpha}} \ln Z(\boldsymbol{\alpha}) \\
&= \mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) - \mathbb{E}_{\mathbf{u}\sim p_m(\mathbf{u};\boldsymbol{\theta})} \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) \\
&= \sum_{\mathbf{u}} p_d(\mathbf{u}) \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) - \sum_{\mathbf{u}} p_m(\mathbf{u};\boldsymbol{\theta}) \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) \\
&= \sum_{\mathbf{u}} \{(p_d(\mathbf{u}) - p_m(\mathbf{u};\boldsymbol{\theta})) \frac{\partial}{\partial \boldsymbol{\alpha}} \ln p_m^0(\mathbf{u};\boldsymbol{\alpha})\}.
\end{aligned} \tag{15}$$

As we discussed above and also demonstrated in above equation, the calculation of the "normalization term" $Z(\boldsymbol{\alpha})$ is still inevitable in maximum likelihood estimation. Now, we show that the objective of NCE is an universal approximation to data likelihood. Following Equation (12) we have

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{u}\sim p_d(\mathbf{u})} \ln \frac{p_m(\mathbf{u};\boldsymbol{\theta})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \nu \mathbb{E}_{\mathbf{u}\sim p_n(\mathbf{u})} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})}, \tag{16}$$

take derivative with respect to $\boldsymbol{\theta}$ gives

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) &= \frac{\partial}{\partial \boldsymbol{\theta}} \left(\mathbb{E}_{\mathbf{u} \sim p_d(\mathbf{u})} \ln \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \nu \mathbb{E}_{\mathbf{u} \sim p_n(\mathbf{u})} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \right) \\
&= \frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_{\mathbf{u} \sim p_d(\mathbf{u})} \ln \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \frac{\partial}{\partial \boldsymbol{\theta}} \nu \mathbb{E}_{\mathbf{u} \sim p_n(\mathbf{u})} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \\
&= \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{\mathbf{u}} p_d(\mathbf{u}) \ln \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \nu \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{\mathbf{u}} p_n(\mathbf{u}) \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \\
&= \sum_{\mathbf{u}} p_d(\mathbf{u}) \frac{\partial}{\partial \boldsymbol{\theta}} \ln \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \nu \sum_{\mathbf{u}} p_n(\mathbf{u}) \frac{\partial}{\partial \boldsymbol{\theta}} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})},
\end{aligned} \tag{17}$$

where the left derivative of the RHS:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \ln \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} &= -\frac{\partial}{\partial \boldsymbol{\theta}} \ln \left(\frac{\nu p_n(\mathbf{u})}{p_m(\nu; \boldsymbol{\theta})} + 1 \right) \\
&= -\frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \left(\frac{\nu p_n(\mathbf{u})}{p_m(\nu; \boldsymbol{\theta})} \right) \\
&= -\frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \times -\frac{\nu p_n(\mathbf{u})}{(p_m(\mathbf{u}; \boldsymbol{\theta}))^2} \frac{\partial}{\partial \boldsymbol{\theta}} p_m(\mathbf{u}; \boldsymbol{\theta}) \\
&= \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{1}{p_m(\mathbf{u}; \boldsymbol{\theta})} \frac{\partial}{\partial \boldsymbol{\theta}} p_m(\mathbf{u}; \boldsymbol{\theta}) \\
&= \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p_m(\mathbf{u}; \boldsymbol{\theta}),
\end{aligned} \tag{18}$$

similarly the right derivative of the RHS:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} &= -\frac{\partial}{\partial \boldsymbol{\theta}} \ln \left(\frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{\nu p_n(\mathbf{u})} + 1 \right) \\
&= -\frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{1}{(\nu p_n(\mathbf{u}))^2} \frac{\partial}{\partial \boldsymbol{\theta}} \left(p_m(\mathbf{u}; \boldsymbol{\theta}) \nu p_n(\mathbf{u}) \right) \\
&= -\frac{1}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} p_m(\mathbf{u}; \boldsymbol{\theta}) \\
&= -\frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{1}{p_m(\mathbf{u}; \boldsymbol{\theta})} \frac{\partial}{\partial \boldsymbol{\theta}} p_m(\mathbf{u}; \boldsymbol{\theta}) \\
&= -\frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p_m(\mathbf{u}; \boldsymbol{\theta}).
\end{aligned} \tag{19}$$

Therefore, following Equation (17) we have

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) \\
&= \sum_{\mathbf{u}} p_d(\mathbf{u}) \frac{\partial}{\partial \boldsymbol{\theta}} \ln \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} + \nu \sum_{\mathbf{u}} p_n(\mathbf{u}) \frac{\partial}{\partial \boldsymbol{\theta}} \ln \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \\
&= \sum_{\mathbf{u}} p_d(\mathbf{u}) \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p_m(\mathbf{u}; \boldsymbol{\theta}) - \nu \sum_{\mathbf{u}} p_n(\mathbf{u}) \frac{p_m(\mathbf{u}; \boldsymbol{\theta})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p_m(\mathbf{u}; \boldsymbol{\theta}) \\
&= \sum_{\mathbf{u}} p_d(\mathbf{u}) \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p_m(\mathbf{u}; \boldsymbol{\theta}) - \sum_{\mathbf{u}} p_m(\mathbf{u}; \boldsymbol{\theta}) \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p_m(\mathbf{u}; \boldsymbol{\theta}) \\
&= \sum_{\mathbf{u}} \{ (p_d(\mathbf{u}) - p_m(\mathbf{u}; \boldsymbol{\theta})) \frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u}; \boldsymbol{\theta}) + \nu p_n(\mathbf{u})} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p_m(\mathbf{u}; \boldsymbol{\theta}) \}.
\end{aligned} \tag{20}$$

Leveraging the assumption in Equation (9), *i.e.*, $\ln p_m(\mathbf{u};\boldsymbol{\theta}) = \ln p_m^0(\mathbf{u};\boldsymbol{\alpha}) + c$, we therefore have

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) &= \sum_{\mathbf{u}}\{(p_d(\mathbf{u}) - p_m(\mathbf{u};\boldsymbol{\theta}))\frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})}\frac{\partial}{\partial\boldsymbol{\theta}}\ln p_m(\mathbf{u};\boldsymbol{\theta})\} \\ &= \sum_{\mathbf{u}}\{(p_d(\mathbf{u}) - p_m(\mathbf{u};\boldsymbol{\theta}))\frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})}\frac{\partial}{\partial\boldsymbol{\theta}}\ln(p_m^0(\mathbf{u};\boldsymbol{\alpha}) + c)\},\end{aligned} \tag{21}$$

It is discovered that a fixed $Z = 1$ instead of learning it does not affect the performance of the resulting models [Mnih and Teh, 2012]. This is possibly because the model has plenty of degrees of freedom so it is easy to meet the approximate normalization constraint encouraged by the objective function. Thus in a special case where $Z = 1 \iff c = \ln 1/Z = 0$,

$$\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) = \sum_{\mathbf{u}}\{(p_d(\mathbf{u}) - p_m(\mathbf{u};\boldsymbol{\theta}))\frac{\nu p_n(\mathbf{u})}{p_m(\mathbf{u};\boldsymbol{\theta}) + \nu p_n(\mathbf{u})}\frac{\partial}{\partial\boldsymbol{\theta}}\ln p_m^0(\mathbf{u};\boldsymbol{\alpha})\}, \tag{22}$$

remember ν is the ratio of number of negative samples toward positive, if let $\nu \to \infty$ we have

$$\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) = \sum_{\mathbf{u}}\{(p_d(\mathbf{u}) - p_m(\mathbf{u};\boldsymbol{\theta}))\frac{\partial}{\partial\boldsymbol{\theta}}\ln p_m^0(\mathbf{u};\boldsymbol{\alpha})\}, \tag{23}$$

Recall the derivative of MLE loss Equation (15) we observe that

$$\frac{\partial}{\partial\boldsymbol{\alpha}}\mathcal{L}_{\text{MLE}} = \sum_{\mathbf{u}}\{(p_d(\mathbf{u}) - p_m(\mathbf{u};\boldsymbol{\theta}))\frac{\partial}{\partial\boldsymbol{\alpha}}\ln p_m^0(\mathbf{u};\boldsymbol{\alpha})\} = \frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}),$$

thus when ν approx infinity, the gradient of the NCE objective function in Equation (23) and the gradient of the MLE log-likelihood function in Equation (15) are equivalent, which means that our NCE-transformed optimization objective is essentially an approximation to the maximum likelihood estimation method, and the approximation becomes more accurate as the ratio ν, which explains why the authors suggest setting ν as large as possible, and this is why in current applications (*e.g.*, representation learning pretext task [He et al., 2020]), we use one positive sample toward many negative samples.

2 Contrastive Predictive Coding

2.1 InfoNCE Objective

The main intuition behind Contrastive Predictive Coding [Oord et al., 2018] is to learn the representation of shared information between high dimensional signal and the encoded context, as well as discarding the low-level information. Mutual Information is a good way to measure the shared information between target x and context c, we maximize the mutual information between the encoded representations of x and c to extract the potential features shared by the inputs. The mutual information (MI) is defined as

$$I(x;c) = \sum_{x,c} p(x,c)\ln\frac{p(x|c)}{p(x)}. \tag{24}$$

It is hard to obtain the joint distribution $p(x,c)$, thus to maximize Equation 24, we maximize an lowerbound of mutual information. Given a set of $X = \{x_1, ..., x_N\}$ of N random samples, where there is one positive sample from $p(x|c)$, and other $N-1$ negative samples from $p(x)$, c is the latent representation of positive samples, the loss function is defined as

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}_X\left[\ln\frac{f_k(x_i,c)}{\sum_{x_j\in X} f_k(x_j,c)}\right] \tag{25}$$

where f_k is an unnormalized function that measures the score of if c matches x_i, and thus a better f_k results in $\frac{f_k(x_i,c)}{\sum_{x_j\in X} f_k(x_j,c)}$ (normalized function, indicating the probability of positive pair among all pair samples) closer to 1, as well as a smaller loss. If we regard x_i as the positive sample drawn from $p(x|c)$, and x_j as the negatives from $p(x)$, optimizing the loss also leads to f_k estimating a higher probability of ratio $\frac{p(x_i|c)}{p(x_i)}$. As this could be hard to understand, we give an illustration:

Lets assume when $\mathcal{L}_{\text{InfoNCE}}$ is optimized (minimized), then $f_k(x_i, c)$ is the largest among all $f_k(X, c)$. An optimal $f_k(x_i, c)$ semantically means x_i is the mostly likely to be correlated to c (the score), and thus means the probability that x_i is the positive sample (but not any $\{x_j\}$) is the largest, in the context of NCE, this is a binary classification problem, where the classifier tries to classify the x_i as positive while discriminating any other samples as negative, so the optimal classifier gives

$$\begin{aligned} p(\{x_i \text{ is positive}\}|X, c) &= \frac{p(x_i|c)\prod_{l\neq i} p(x_l)}{\sum_{j=1}^{N} p(x_j|c_t)\prod_{l\neq j} p(x_l)} \\ &= \frac{\frac{p(x_i|c)}{p(x_i)}}{\sum_{j=1}^{N}\frac{p(x_j|c)}{p(x_j)}}. \end{aligned} \tag{26}$$

The above equation means x_i has been correctly classified as positive rather than negative, its easy to observe that Equation 26 is quite similar to the form of Equation 25, in fact, their optimization target are the same, correctly classifying x_i as the positive sample semantically means $f_k(x_i, c)$ should have the highest score, and vice versa. This is also why the author claims that the optimal value

$$f_k(x_i, c) \propto \frac{p(x_i|c)}{p(x_i)}. \tag{27}$$

In practice, we could use any positive real score function as well as neural networks as f_k, while most existing works simply use a log-bilinear model:

$$f_k(x_i, c) = \exp\left(z_i^T W_k c\right), \tag{28}$$

where z_i stand for the encoded x_i, *i.e.*, $z_i = g_{\text{enc}}(x_i)$. Same to NCE, the loss Equation 25 is the categorical cross-entropy of classifying the positive sample correctly, the implementation-wise discussion will be in the next section.

2.2 Maximizing Lower Bound on MI

We prove that by optimizing Equation 25 we maximize the mutual information between the variables c_t and x_{t+k} by maximizing a lowerbound of mutual information:

$$I(x_i, c) \geq \ln(N) - \mathcal{L}_{\text{InfoNCE}}. \tag{29}$$

Recall Equation 26, the optimal $f_k(x_i, c)$ is given by $\frac{p(x_i|c)}{p(x_i)}$, inserting which back to Equation 25 we have

$$\begin{aligned} \mathcal{L}^{\text{opt}}_{\text{InfoNCE}} &= -\mathbb{E}_X \ln\left[\frac{\frac{p(x_i|c)}{p(x_i)}}{\frac{p(x_i|c)}{p(x_i)} + \sum_{x_j\in X_{\text{neg}}}\frac{p(x_j|c)}{p(x_j)}}\right] \\ &= \mathbb{E}_X \ln\left[1 + \frac{p(x_i)}{p(x_i|c)}\sum_{x_j\in X_{\text{neg}}}\frac{p(x_j|c)}{p(x_j)}\right]. \end{aligned} \tag{30}$$

When N is large enough, we have

$$\mathbb{E}_X \ln\left[1 + \frac{p(x_i)}{p(x_i|c)}\sum_{x_j\in X_{\text{neg}}}\frac{p(x_j|c)}{p(x_j)}\right] = \mathbb{E}_X \ln\left[1 + \frac{p(x_i)}{p(x_i|c)}(N-1)\mathbb{E}_{X_{\text{neg}}}\frac{p(x_j|c)}{p(x_j)}\right], \tag{31}$$

since negative samples does not correlated to context c, $p(x_j|c) = p(x_j)$ for all X_{neg}. Therefore

$$\begin{aligned} \mathcal{L}^{\text{opt}}_{\text{InfoNCE}} &= \mathbb{E}_X \ln\left[1 + \frac{p(x_i)}{p(x_i|c)}(N-1)\mathbb{E}_{X_{\text{neg}}}\frac{p(x_j|c)}{p(x_j)}\right] \\ &= \mathbb{E}_X \ln\left[1 + \frac{p(x_i)}{p(x_i|c)}(N-1)\right] \\ &= \mathbb{E}_X \ln\left[\frac{Np(x_i) + (p(x_i|c) - p(x_i))}{p(x_i|c)}\right] \\ &\geq \mathbb{E}_X \ln\left[N\frac{p(x_i)}{p(x_i|c)}\right], \end{aligned} \tag{32}$$

For the last two steps in Equation 32, it is because f_k is by definition a positive arbitrarily large score, and thus the esimated $p(x_i|c)/p(x_i)$ is likely to exceed one. Hence we have

$$\begin{aligned} &\mathbb{E}_X \ln \left[N \frac{p(x_i)}{p(x_i|c)} \right] = \ln(N) + \mathbb{E}_X \ln \left[\frac{p(x_i)}{p(x_i|c)} \right] = \ln(N) - I(x_i, c) \\ \iff &\mathcal{L}_{\text{InfoNCE}}^{\text{opt}} \geq \ln(N) - I(x_i, c), \end{aligned} \tag{33}$$

which is exactly the form in Equation 29, and this lower bound becomes tighter when N becomes larger.

3 Self-Supervised Representation Learning

3.1 Self-Supervised Paradigm

Supervised learning refers to the learning paradigm that leverages well-defined manual labels to train machine learning models. Conversely, unsupervised/semi-supervised learning refers to the learning paradigm with partial or without using any manual labels. Self-supervised learning indicates the learning paradigm where supervision signals are generated from data itself, which can also be viewed as a subset of unsupervised learning [Liu et al., 2021]. For visual tasks, the general definition of self-supervised learning is: given a dataset X, for each data x_i, along with pseudo label p_i, where they are automatically generated from a pre-defined pretext task without involving human annotation, the pseudo label can be generated by using attributes of image or using manually crafting methods, our task is to minimize the loss between x_i and p_i in order to extract useful information (embedding) in between the input data which can be used on downstream tasks. For graph level representation learning, similar to visual tasks, we are given a set of graphs $\mathbf{G} = \{G_i\}_{i=1}^N$ and the expected embedding size δ, our task is to learn a δ-dimensional graph embedding of every graph (or node, for node-level task), where it can be viewed as high level representation (summary) that can be used in many graph downstream tasks. Different from unsupervised learning, the self-supervised learning method generally include two parts: pretext tasks and loss function, models are trained with pretext tasks to obtain better performance and generalization on downstream tasks.

3.1.1 Pretext Tasks

The pretext tasks typically refer to the pre-designed tasks for model to solve, for instance in contrastive learning, we design manual labels and train binary classifier to indirectly learn more generalized representation from unlabeled data, we can utilized one of the encoders/sub-networks for downstream tasks (*i.e.*, true analytic tasks that evaluate the performance of the feature representation) and hopefully they can achieve better performance with these indirectly trained sub-modules.

3.1.2 Loss Function

Recall the loss function of InfoNCE in Equation 25, where c represents the context, x_i represents the positive sample (conditioning on context c) and $\{x_j\}$ represent the negative samples, the loss intuitively measures the similarities of sample pairs in a representation space, where it attract positive pairs (commonly one pair) as well as discriminating different negative pairs, since this is intuitively a binary classification problem so one can use cross-entropy as the loss function. The representations from contrastive self-supervised pre-training can outperform their supervised counterparts in certain tasks.

3.2 Visual Representation Learning

3.2.1 End-to-end

Following the notation in [He et al., 2020], we have an encoded query q which represents the encoded query sample (from data distribution) $\tilde{x}_q = f_{\text{q}}(x^q)$ where f_{q} is the encoder network. Similarly, we have the key $\tilde{x}_{k_+}$ as the positive encoded sample $\tilde{x}_{k_+} = f_{\text{k}}(x^k)$, and K negative encoded samples $\tilde{x}_{k_-}$ (manually designed). In practice, the encoder network f_{q} and f_{k} can be identical, shared weight, or different in structure. The SimCLR framework [Chen et al., 2020] introduces the MLP head $g_q(\cdot)$ and $g_k(\cdot)$ which are simply a 2-layer MLP projection head to project the representation to lower

dimensional latent space, we have the output representation $q = g_q(\tilde{x}_q)$ and $k = g_k(\tilde{x}_k)$. Let dot product denote the similarity between pairs, the loss function for positive pair of examples q and k_+ is defined as

$$\mathcal{L}_{q,k_+,\{k_-\}} = -\ln \frac{\exp(q \cdot k_+/\tau)}{\exp(q \cdot k_+/\tau) + \sum_{k_-} \exp(q \cdot k_-/\tau)} \tag{34}$$

where τ is the temperature hyper-parameter [Wu et al., 2018]. This loss is in exactly same form to the InfoNCE loss we mentioned above (Equation 25), where q is similar to the context c, k_+ and k_-s are positive and negative samples, for visual tasks, this loss is intuitively the log loss of a softmax-based classifier that distinguishes q as k_+.

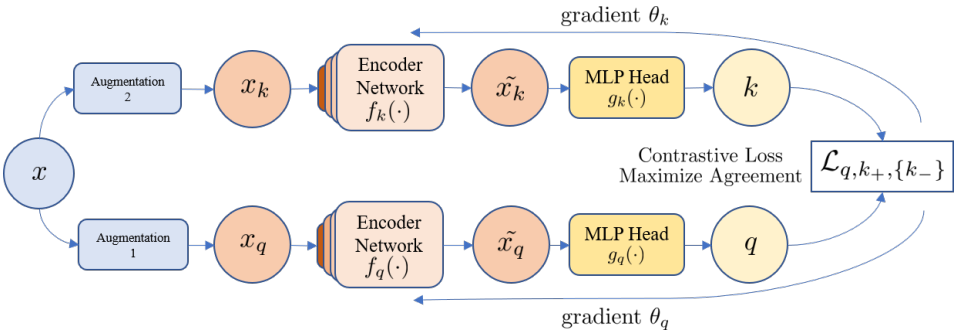


Figure 1: end-to-end framework

We give an illustration of general end-to-end framework in Figure 1, to highlight, during the back-propagation of training, the framework will compute the gradient though not only query encoder f_q but the key encoder f_k, recall Equation 34, computing the gradient of key encoder in the denominator (normalization) term requires large computation resource (since the negative data size can be arbitrarily large), especially when performing batch updating. We cannot simply copy the weight from f_q to f_k without mathematical arguments, and many empirical results also proved doing so will result in poor performance.

3.2.2 Momentum Contrast

The concept of momentum contrast is proposed by [He et al., 2020] (MoCo v1) and improved by [Chen et al., 2021] (MoCo v2 & v3), here we focus on the details of the classic MoCo v1 implementation. Recall end-to-end framework in previous section, the main drawback is that f_k cannot be easily updated, and we cannot naïvely copy f_q to f_k. The concept of momuntum update is proposed to adress this problem, where the parameter θ_k of the key encoder f_k can be updated based on the θ_q of f_q by

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q, \tag{35}$$

where $m \in [0, 1)$ is the momentum coefficient. In this case, we only need to compute the gradient through θ_q during back-propagation, and θ_k is smoothly updated with out any extra computation on large mini-batches. In experiment, a larger momentum coefficient works better than smaller values.

Figure 2 illustrates the pipeline of MoCo framework, where it also introduced dictionary queue for alternating negative samples. To specify, we maintain a queue that contains several previous encoded keys from immediate preceding mini-batches. By every end of forward propagation, the part of samples in the dictionary queue are replaced, where the current mini-batch of keys is enqueued and the oldest is dequeued. The queue of data always represents a sampled subset of data and all of the queue members are used as negative samples k_- for current contrast.

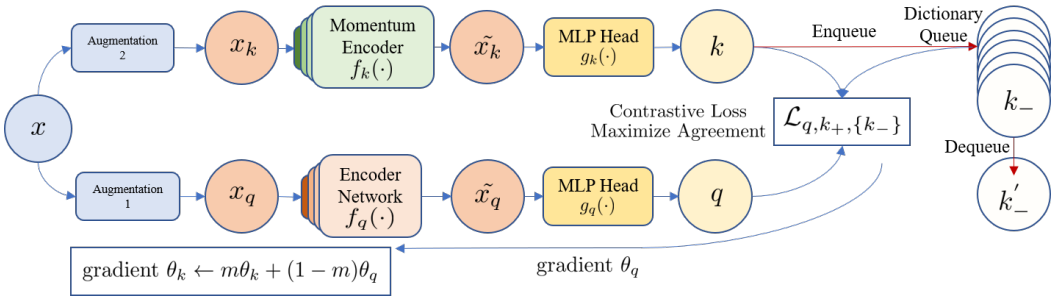


Figure 2: MoCo framework

In summary, the process of the algorithm is: we first get x_q and x_k through random augmentation of original data sample x, we encode to get $q = g_q(f_q(x_q))$ and $k = g_k(f_k(x_k))$ during forward pass, where the parameter of encoder and MLP head are together represented by θ_q and θ_k, then we perform matrix multiplication for q to k to get positive logits, and q to k_- in the queue to get negative logits, we train a binary classifier to distinguish between positive logits and negative logits using cross-entropy loss, finally we update the parameter θ_q and momentumly update θ_k with θ_q (Equation 35).

3.2.3 Deep InfoMax

The motivation of visual contrastive learning such as MoCo, SimCLR etc. are basically formulating global-wise contrast method as well as image augmentation and network structure augmentation. The concept of Deep InfoMax (DIM) is proposed by [Hjelm et al., 2018], which is not a typical contrastive learning framework but focusing on learning representation by maximizing mutual information from both global and local image feature. The author first introduces a base encoder, which is a combination of feature encoder (*e.g.*convnet) and summary operation (*e.g.*, average)

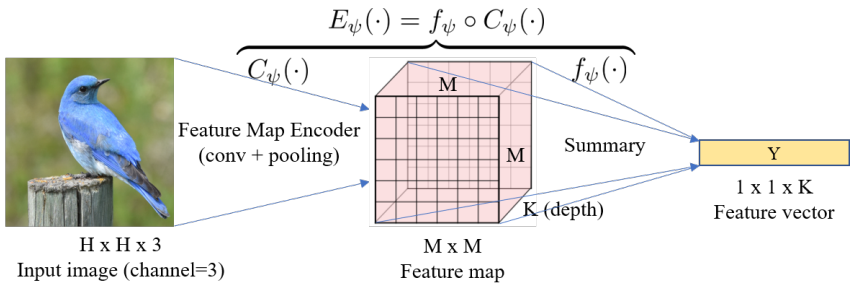
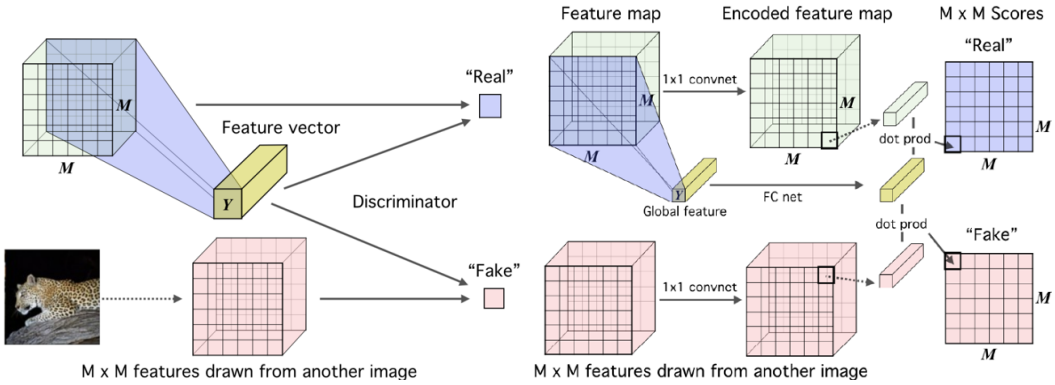


Figure 3: The base encoder model E_ψ

In Figure 3, an image X is encoded using a convnet until reaching a feature map of $M \times M$ feature vectors corresponding to $M \times M$ input patches. These vectors are summarized into a single feature vector Y . We want to train a good encoder network E_ψ such that it extract useful high-level representation that could be used in downstream task.



(a) Deep InfoMax global MI object, where the high-level feature vector Y and low-level $M \times M$ feature map of real input X_+ are passed through discriminator to get the score, same for the fake samples feature map from another uncorrelated image X_-.

(b) Deep InfoMax local-global MI object, where the mutual information between local and global features are maximized. In this case, every location of the feature map (local feature) are paired with the global summarized feature vector Y and M^2 of scores are produced with all local-global pairs for each image input.

Figure 4: Global and local MI maximization, figures from [Hjelm et al., 2018]

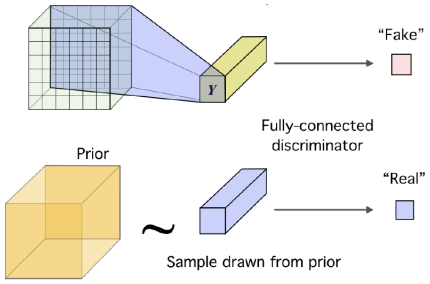


Figure 5: Matching the output of the encoder to a prior.

Figure 4(a) and 4(b) introduces two ways of MI maximization, to optimize the MI objective, we leverage the knowledge in InfoNCE and formulate the objective:

$$I^{(\text{InfoNCE})}_{\omega,\psi}(X; E_\psi(X)) = \mathbb{E}_{p_{\text{data}}}\left[T_{\psi,\omega}(x, E_\psi(x)) - \mathbb{E}_{p'_{\text{data}}}\left[\ln \sum_{x' \sim p'_{\text{data}}} e^{T_{\psi,\omega}(x', E_\psi(x))}\right]\right], \tag{36}$$

where T_ψ is the discriminator with parameter ω, we perform global and local MI maximization in Figure 4 by

$$\arg\max_{\omega_1,\omega_2,\psi}(\alpha I_{\omega_1,\psi}(X; E_\psi(X)) + \frac{\beta}{M^2}\sum_{i=1}^{M^2} I_{\omega_2,\psi}(X^{(i)}; E_\psi(X))), \tag{37}$$

where the first term is the global objective and second is local objective, ω_1 and ω_2 are respectively the discriminator parameter of global and local objectives, and α, β are hyperparameters. Till here, we almost completed our InfoNCE objective, additionally, we need a constraint to match the learnt representation to a prior distribution, where the prior is defined according to different applications. the objective can be defined as

$$\arg\min_{\psi} \arg\max_{\phi} \mathcal{D}_\phi(V \| U_{\psi,p}) = \mathbb{E}_V[\ln D_\phi(y)] + \mathbb{E}_P[\ln(1 - D_\phi(E_\psi(x)))] \tag{38}$$

where V denotes the prior and $U_{\psi,P}$ is the push-forward distribution from the local-global encoder, $\mathcal{D}$ denotes the divergence (non-negative difference between two distribution), $D_\phi : \mathcal{Y} \to \mathbb{R}$ is the discriminator which reconstruct the global feature to the pattern of empirical data. We want

the push-forward representation U to match the prior V, Figure 5 describes the process of prior matching, where the real samples are drawn from a prior and fakes are from the encoder output and sent to an extra discriminator, we adversarially train the discriminator to distinguish between sets of real-fake samples and train the encoder to fool the discriminator, so that the output via encoder in prior matching framework can generate representation that match the pattern of predefined prior and thus can be used in various specific tasks. Thus the overall objective of DIM can be defined as

$$\arg\max_{\omega_1,\omega_2,\psi}(\alpha I_{\omega_1,\psi}(X;E_\psi(X)) + \frac{\beta}{M^2}\sum_{i=1}^{M^2} I_{\omega_2,\psi}(X^{(i)};E_\psi(X))) + \arg\min_{\psi}\arg\max_{\phi}\mathcal{D}_\phi(V\|U_{\psi,p}) \tag{39}$$

A Appendix

References

Christopher M Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

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.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

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.

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.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

M. Leove. *Probability Theory: 4th Ed.* Graduate texts in mathematics, 45. Springer-Verlag, 1977.

Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S Yu. Graph self-supervised learning: A survey. *arXiv preprint arXiv:2103.00111*, 2021.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.