A. Theory and Proofs

A.1. Proof of Lemma 1: From RINCE to InfoNCE

We show that RINCE becomes asymptotically equivalent to InfoNCE when $q \to 0$. In particular, we prove the convergence of RINCE and its derivative in the limit of $q \to 0$.

Proof. We first prove the convergence in the function space with the L'Hôpital's rule:

$$
\begin{aligned}
&\lim_{q\to 0} \mathcal{L}^{\lambda,q}_{\text{RINCE}}(\mathbf{s}) \\
&= \lim_{q\to 0} \frac{-e^{q\cdot s^+}}{q} + \frac{\left(\lambda\cdot(e^{s^+} + \sum_{i=1}^K e^{s_i^-})\right)^q}{q} \\
&= \lim_{q\to 0} \frac{1-e^{q\cdot s^+}}{q} + \frac{-1+\left(\lambda\cdot(e^{s^+} + \sum_{i=1}^K e^{s_i^-})\right)^q}{q} \\
&= \lim_{q\to 0} \frac{1-e^{q\cdot s^+}}{q} + \lim_{q\to 0} \frac{-1+\left(\lambda\cdot(e^{s^+} + \sum_{i=1}^K e^{s_i^-})\right)^q}{q} \\
&= -\log(e^{s^+}) + \log\left(\lambda(e^{s^+} + \sum_{i=1}^K e^{s_i^-})\right) && \text{(L'Hôpital's rule)} \\
&= -\log\frac{e^{s^+}}{\lambda\left(e^{s^+} + \sum_{i=1}^K e^{s_i^-}\right)} \\
&= \mathcal{L}_{\text{InfoNCE}}(\mathbf{s}) + \log(\lambda).
\end{aligned}
$$

To prove the convergence in its derivative, we analyze the derivative with respect to the positive score s^+ and the negative score s_i^-. We begin with RINCE:

$$
\begin{aligned}
\text{(positive score)} \quad & \lim_{q\to 0} \frac{\partial}{\partial s^+} \mathcal{L}^{\lambda,q}_{\text{RINCE}}(\mathbf{s}) \\
&= \lim_{q\to 0} \frac{\partial}{\partial s^+} \frac{-e^{q\cdot s^+}}{q} + \frac{\partial}{\partial s^+} \frac{\left(\lambda\cdot(e^{s^+} + \sum_{i=1}^K e^{s_i^-})\right)^q}{q} \\
&= \lim_{q\to 0} -e^{q\cdot s^+} + (\lambda\cdot(e^{s^+} + \sum_{i=1}^K e^{s_i^-}))^{q-1}\cdot\lambda\cdot e^{s^+} \\
&= -1 + \frac{e^{s^+}}{e^{s^+} + \sum_{i=1}^K e^{s_i^-}}; \\
\text{(negative score)} \quad & \lim_{q\to 0} \frac{\partial}{\partial s_i^-} \mathcal{L}^{\lambda,q}_{\text{RINCE}}(\mathbf{s}) \\
&= \lim_{q\to 0} \frac{\partial}{\partial s_i^-} \frac{\left(\lambda\cdot(e^{s^+} + \sum_{i=1}^K e^{s_i^-})\right)^q}{q} \\
&= \lim_{q\to 0} (\lambda\cdot(e^{s^+} + \sum_{i=1}^K e^{s_i^-}))^{q-1}\cdot\lambda\cdot e^{s_i^-} \\
&= \frac{e^{s_i^-}}{e^{s^+} + \sum_{i=1}^K e^{s_i^-}}.
\end{aligned}
$$

We can see that the derivatives match the ones of InfoNCE

$$
\begin{aligned}
\text{(positive score)} \qquad & \frac{\partial}{\partial s^+}\mathcal{L}_{\text{InfoNCE}}(\mathbf{s}) \\
& = \frac{\partial}{\partial s^+} - \log \frac{e^{s^+}}{e^{s^+} + \sum_{i=1}^K e^{s_i^-}} \\
& = -\frac{e^{s^+}}{e^{s^+} + \sum_{i=1}^K e^{s_i^-}} \cdot \frac{(e^{s^+} + \sum_{i=1}^K e^{s_i^-}) \cdot e^{s^+} - e^{2s^+}}{(e^{s^+} + \sum_{i=1}^K e^{s_i^-})^2} \\
& = -1 + \frac{e^{s^+}}{e^{s^+} + \sum_{i=1}^K e^{s_i^-}}; \\
\text{(negative score)} \qquad & \frac{\partial}{\partial s_i^-}\mathcal{L}_{\text{InfoNCE}}(\mathbf{s}) \\
& = -\frac{e^{s^+} + \sum_{i=1}^K e^{s_i^-}}{e^{s_i^-}} \cdot \frac{-e^{s^+} \cdot e^{s_i^-}}{(e^{s^+} + \sum_{i=1}^K e^{s_i^-})^2} \\
& = \frac{e^{s_i^-}}{e^{s^+} + \sum_{i=1}^K e^{s_i^-}}.
\end{aligned}
$$

□

A.2. Noisy Risk Bound for Exponential Loss

We justify the robustness of RINCE when $q = 1$ by extending Ghosh et al. [29]'s theorem to the exponential loss. The proof technique can be applied to other bounded symmetric classification losses.

Corollary 2. *Consider the setting of Ghosh et al. [29] and the exponential loss function $\mathcal{L}(s, y) = -ye^s$. Let $f_\eta^* = \arg\inf_{f\in\mathcal{F}} R_\mathcal{L}^\eta(f)$ be the minimizer of the noisy risk and $\epsilon = \inf_{f\in\mathcal{F}} R_\mathcal{L}(f)$ be the optimal risk. If $\eta_x \leq \eta_{\max} < 0.5$ for all $x \in \mathcal{X}$. If the prediction score is bounded by $s_{\max}$, we have $R(f_\eta^*) \leq (\epsilon + 2\eta_{\max}e^{s_{\max}})/(1 - 2\eta_{\max})$.*

Proof. Consider a binary classification loss with the following form:

$$\tilde{\mathcal{L}}_x(f(x), y) = B + \mathcal{L}_x(f(x), y) = B - y \cdot e^{f(x)} \geq 0,$$

where the prediction score $f(x)$ is bounded by $s_{\max} = \log(B)$. Note that the boundedness assumption holds for general representation learning on hypersphere, where the prediction score is the inner product between normalized feature vectors. Importantly, the loss satisfies

$$\tilde{\mathcal{L}}(f(x), 1) + \tilde{\mathcal{L}}(f(x), -1) = 2B.$$

By construction, the optimal risk takes the following value:

$$\inf_{f\in\mathcal{F}} R_{\tilde{\mathcal{L}}}(f) = \inf_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mu}[\tilde{\mathcal{L}}(f(x), y_x)] = \epsilon + B := \tilde{\epsilon},$$

and $f^* = \arg\inf_{f\in\mathcal{F}} R_{\tilde{\mathcal{L}}}(f)$. Note that f^* is also a minimizer w.r.t. the original loss $\mathcal{L}$($f^* = \arg\inf_{f\in\mathcal{F}} R_\mathcal{L}(f)$), as an additive constant will not change the optimum solutions. Expanding the noisy risk gives

$$
\begin{aligned}
R_{\tilde{\mathcal{L}}}^\eta(f) &= \mathbb{E}_{(x,y)\sim\mu}[(1 - \eta_x)\tilde{\mathcal{L}}(f(x), y_x) + \eta_x\tilde{\mathcal{L}}(f(x), -y_x)] \\
&= \mathbb{E}_{x\sim\mu}[(1 - \eta_x)\tilde{\mathcal{L}}(f(x), y_x) + \eta_x(2B - \tilde{\mathcal{L}}(f(x), y_x))] \qquad \text{(Symmetry)} \\
&= \mathbb{E}_{x\sim\mu}[(1 - 2\eta_x)\tilde{\mathcal{L}}(f(x), y_x)] + 2B\mathbb{E}_{x\sim\mu}[\eta_x].
\end{aligned}
$$

Let $f_\eta^* = \arg\inf R_\mathcal{L}^\eta(f_\eta^*) = \arg\inf R_{\tilde{\mathcal{L}}}^\eta(f_\eta^*)$, we have

$$R_{\tilde{\mathcal{L}}}^\eta(f^*) - R_{\tilde{\mathcal{L}}}^\eta(f_\eta^*) = \mathbb{E}_{x\sim\mu}[(1 - 2\eta_x)(\tilde{\mathcal{L}}(f^*(x), y_x) - \tilde{\mathcal{L}}(f_\eta^*(x), y_x))] \geq 0$$

since f_η^* is the minimizer of $R_{\tilde{\mathcal{L}}}^\eta$, which implies that

$$E_{x\sim\mu}[(1-2\eta_x)\tilde{\mathcal{L}}(f_\eta^*(x),y_x)] \leq E_{x\sim\mu}[(1-2\eta_x)\tilde{\mathcal{L}}(f^*(x),y_x)] \leq \tilde{\epsilon},$$

since $0 < 1-2\eta_x \leq 1$ by assumption. Let $\eta_{max} = \sup_{x\in\mathcal{X}} \eta_x$, we have

$$(1-2\eta_{\max})E_{x\sim\mu}[\tilde{\mathcal{L}}(f_\eta^*(x),y_x)] \leq \epsilon,$$

since the loss is non-negative, which implies

$$R_{\tilde{\mathcal{L}}}(f_\eta^*) \leq \frac{\tilde{\epsilon}}{1-2\eta_{\max}}.$$

Finally, we recover the original exponential loss without the additive term B. Plugging the form we have

$$B + R_{\mathcal{L}}(f_\eta^*) \leq \frac{\epsilon + B}{1-2\eta_{max}},$$

which implies

$$R_{\mathcal{L}}(f_\eta^*) \leq \frac{\epsilon + B}{1-2\eta_{max}} - B = \frac{\epsilon + 2B\eta_{max}}{1-2\eta_{max}}.$$

For exponential loss, setting B to $e^{s_{\max}}$ completes the proof. □

For instance, when the noise level is 40%, we have $R_{\mathcal{L}}(f_\eta^*) \leq 5\epsilon + 4B$. Note that the prediction score is bounded by $1/t$ in our case as the representations are projected onto the unit hypersphere.

A.3. Lower bound of Wasserstein Distance

We now establish RINCE as a lower bound of WDM [33]. WDM is based on the Wasserstein distance, a distance metric between probability distributions defined via an optimal transport cost. Letting μ and $\nu \in \text{Prob}(\mathbb{R}^d \times \mathbb{R}^d)$ be two probability measures, we define the Wasserstein-1 distance with a Euclidean cost function as

$$\mathcal{W}(\mu,\nu) = \inf_{\pi\in\Pi(\mu,\nu)} \mathbb{E}_{\substack{(X,V)\\(X',V')}\sim\pi} \left[\|X-X'\| + \|V-V'\|\right],$$

where $\Pi(\mu,\nu)$ denotes the set couplings whose marginals are μ and ν, respectively. We are now ready to state our theorem.

Theorem 3. *If $\lambda K > 1-\lambda$ and f projects the representation to a unit hypersphere, we have*

$$-\mathbb{E}\left[\mathcal{L}_{\text{RINCE}}^{\lambda,q=1}(\boldsymbol{s})\right] \leq \frac{\text{Lip}(f)\cdot(1-\lambda)\cdot e^{1/t}}{t}\mathcal{W}_1(P_{XV}^\phi, P_X^\phi P_V^\phi).$$

Proof. By the additivity of expectation, we can bound the negative symmetric loss as follows

$$\begin{aligned}
&-\mathbb{E}\left[\mathcal{L}_{\text{RINCE}}^{\lambda,q=1}(\mathbf{s})\right] \\
&= \mathbb{E}_{\substack{x\sim P_X\\ v\sim P_{V|X=x}\\ v_i\sim P_V}}[(1-\lambda)e^{f(\phi(x))^T f(\phi(v)/t} - \lambda\sum_{i=1}^K e^{f(\phi(x))^T f(\phi(v_i)/t}] \\
&= \mathbb{E}_{(x,v)\sim P_{XV}}\left[(1-\lambda)e^{f(\phi(x))^T f(\phi(v)/t}\right] - \mathbb{E}_{\substack{x\sim P_X\\ v_i\sim P_V}}\left[\lambda\sum_{i=1}^K e^{f(\phi(x))^T f(\phi(v_i)/t}\right] \\
&= \mathbb{E}_{(x,v)\sim P_{XV}}\left[(1-\lambda)e^{f(\phi(x))^T f(\phi(v)/t}\right] - \mathbb{E}_{\substack{x\sim P_X\\ v_i\sim P_V}}\left[\lambda\sum_{i=1}^K e^{f(\phi(x))^T f(\phi(v_i)/t}\right] \\
&= \mathbb{E}_{(x,v)\sim P_{XV}}\left[(1-\lambda)e^{f(\phi(x))^T f(\phi(v)/t}\right] - \lambda K\mathbb{E}_{\substack{x\sim P_X\\ v\sim P_V}}\left[e^{f(\phi(x))^T f(\phi(v)/t}\right] \\
&\leq (1-\lambda)\cdot(\mathbb{E}_{(x,v)\sim P_{XV}}\left[e^{f(\phi(x))^T f(\phi(v)/t}\right] - \mathbb{E}_{\substack{x\sim P_X\\ v\sim P_V}}\left[e^{f(\phi(x))^T f(\phi(v)/t}\right]),
\end{aligned}$$

where the last equality follows by $\lambda K > 1 - \lambda$. Note that for $\frac{-1}{t} \leq s \leq \frac{1}{t}$, which implies $|\nabla_s e^s| \leq e^{1/t}$. Therefore, by the mean value theorem, we have

$$
\begin{aligned}
&|e^{f(\phi(x))^T f(\phi(v))/t} - e^{f(\phi(x'))^T f(\phi(v'))/t}| \\
&\leq \frac{e^{1/t}}{t} |\langle f(\phi(x)), f(\phi(v))\rangle - \langle f(\phi(x')), f(\phi(v'))\rangle| && \text{(Mean Value Theorem)} \\
&= \frac{e^{1/t}}{t} |\langle f(\phi(x)) - f(\phi(x')), f(\phi(v))\rangle + \langle f(\phi(x')), f(\phi(v) - f(\phi(v'))\rangle| \\
&\leq \frac{e^{1/t}}{t} (|\langle f(\phi(x)) - f(\phi(x')), f(\phi(v))\rangle| + |\langle f(\phi(x')), f(\phi(v) - f(\phi(v'))\rangle|) \\
&\leq \frac{e^{1/t}}{t} (\|f(\phi(x)) - f(\phi(x'))\| \|f(\phi(v))\| + \|f(\phi(v) - f(\phi(v'))\| \|f(\phi(x'))\|) && \text{(Cauchy–Schwarz Ineq.)} \\
&= \frac{e^{1/t}}{t} (\|f(\phi(x)) - f(\phi(x'))\| + \|f(\phi(v) - f(\phi(v'))\|) && (f(\phi(x)) \text{ is unit norm}) \\
&\leq \frac{\text{Lip}(f) \cdot e^{1/t}}{t} (\|\phi(x) - \phi(x')\| + \|\phi(v) - \phi(v')\|) \\
&= \frac{\text{Lip}(f) \cdot e^{1/t}}{t} d((\phi(x), \phi(v)), (\phi(x'), \phi(v'))).
\end{aligned}
$$

We can see that the Lipschitz constant of $\exp(f(\cdot, \cdot))$ with respect to the metric d is bounded by $\frac{\text{Lip}(f) \cdot e^{1/t}}{t}$. Therefore, by Kantorovich-Rubinstein duality, we have

$$
\begin{aligned}
&- \mathbb{E}\left[\mathcal{L}^{\lambda, q=1}_{\text{RINCE}}(\mathbf{s})\right] \\
&\leq (1 - \lambda) \cdot (\mathbb{E}_{(x,v) \sim P_{XV}}[e^{f(\phi(x))^T f(\phi(v)/t}] \\
&\qquad\qquad - \mathbb{E}_{\substack{x \sim P_X \\ v \sim P_V}}[e^{f(\phi(x))^T f(\phi(v)/t}]), \\
&\leq \frac{\text{Lip}(f) \cdot (1 - \lambda) \cdot e^{1/t}}{t} \mathcal{W}_1(\phi_\# P_{XV}, \phi_\# P_X \cdot \phi_\# P_V)
\end{aligned}
$$

□

A.4. Noisy Wasserstein Dependency Measure

The result is a simple combination of Corollary 2 and Theorem 3. If $\lambda \geq \frac{\eta K - \eta + 1}{\eta K - \eta + 1 + K}$, by the assumption of additive noisy models and the symmetry of loss, we have

$$
\begin{aligned}
&- \mathbb{E}\left[\mathcal{L}^{\lambda, q=1}_{\text{RINCE}}(\mathbf{s})\right] \\
&= \mathbb{E}_{\substack{(x,v) \sim P^\eta_{XV} \\ v_i \sim P_V}}[(1 - \lambda) e^{f(\phi(x))^T f(\phi(v)/t} - \lambda \sum_{i=1}^{K} e^{f(\phi(x))^T f(\phi(v_i)/t}] \\
&= \mathbb{E}_{(x,v) \sim P^\eta_{XV}}\left[(1 - \lambda) e^{f(\phi(x))^T f(\phi(v)/t}\right] - \mathbb{E}_{\substack{x \sim P_X \\ v_i \sim P_V}}\left[\lambda \sum_{i=1}^{K} e^{f(\phi(x))^T f(\phi(v_i)/t}\right] \\
&= (1 - \lambda)(1 - \eta) \mathbb{E}_{(x,v) \sim P_{XV}}\left[e^{f(\phi(x))^T f(\phi(v)/t}\right] - K \cdot (\lambda - \eta + \eta\lambda) \mathbb{E}_{\substack{x \sim P_X \\ v \sim P_V}}\left[e^{f(\phi(x))^T f(\phi(v)/t}\right] && \text{(symmetry)} \\
&\leq (1 - \lambda)(1 - \eta) \cdot (\mathbb{E}_{(x,v) \sim P_{XV}}\left[e^{f(\phi(x))^T f(\phi(v)/t}\right] - \mathbb{E}_{\substack{x \sim P_X \\ v_i \sim P_V}}\left[e^{f(\phi(x))^T f(\phi(v_i)/t}\right]) && (\lambda \geq \tfrac{\eta K - \eta + 1}{\eta K - \eta + 1 + K}) \\
&\leq (1 - \eta) \cdot \frac{\text{Lip}(f) \cdot (1 - \lambda) \cdot e^{1/t}}{t} \mathcal{W}_1(\phi_\# P_{XV}, \phi_\# P_X \cdot \phi_\# P_V) \\
&= (1 - \eta) \cdot L \cdot I_{\mathcal{W}}(\phi(X), \phi(V)).
\end{aligned}
$$

A.5. InfoNCE is not symmetric

Note that by taking the derivative with respect to the prediction score s, the definition is equivalent to $\frac{\partial \mathcal{L}(s,1)}{\partial s} + \frac{\partial \mathcal{L}(s,-1)}{\partial s} = 0 \; \forall s \in \mathbb{R}$.

$$\frac{\partial \mathcal{L}_{\text{InfoNCE}}(\mathbf{s})}{\partial s^+} = \frac{-1}{\mathcal{L}_{\text{InfoNCE}}(\mathbf{s})} \cdot \frac{e^{s^+} \cdot \sum_{i=1}^{K} e^{s_i^-}}{(e^{s^+} + \sum_{i=1}^{K} e^{s_i^-})^2}$$
$$\frac{\partial \mathcal{L}_{\text{InfoNCE}}(\mathbf{s})}{\partial s_i^-} = \frac{-1}{\mathcal{L}_{\text{InfoNCE}}(\mathbf{s})} \cdot \frac{e^{s^+}(1 - e^{s_i^-}) + \sum_{i=1}^{K} e^{s_i^-}}{(e^{s^+} + \sum_{i=1}^{K} e^{s_i^-})^2}.$$

Within a batch of data, the gradients with respect to s^+ and s^- are entangled and do not sum to a constant, which fail to meet the symmetry condition.

B. Experiment Details

B.1. CIFAR-10

We follow the experiment setup in [11], where the SimCLR [6] models are trained with Adam optimizer for 500 epochs with learning rate 0.001 and weight decay 1e−6. The encoder is ResNet-50 and the dimension of the latent vector is 128. The temperature is set to $t = 0.5$. The models are then evaluated by training a linear classifier for 100 epochs with learning rate 0.001 and weight decay 1e−6. We use the PyTorch code in Figure 9 to generate the data augmentation noise.

```
def get_train_transform(noise_rate):
    train_transform = transforms.Compose([
        transforms.RandomResizedCrop(32),
        transforms.RandomApply([transforms.RandomResizedCrop(32, scale=(0.2, 0.2))], p=noise_rate
    ),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomApply([transforms.ColorJitter(0.4, 0.4, 0.4, 0.1)], p=0.8),
        transforms.RandomGrayscale(p=0.2),
        transforms.GaussianBlur(kernel_size=int(0.1*32)),
        transforms.ToTensor(),
        transforms.Normalize([0.4914, 0.4822, 0.4465], [0.2023, 0.1994, 0.2010])])
    return train_transform
```

Figure 9. PyTorch code for CIFAR-10 data augmentation noise.

B.2. ImageNet

SimCLR We adopt the SimCLR implementation[2] from PyTorch Lightning [84]. In addition, we spot a bug and fix the implementation of negative masking of PyTorch Lightning according to Figure 10 and achieve 68.9 top-1 accuracy on ImageNet (the one reported in the PyTorch Lightning's website is 68.4). To implement RINCE, we only modify the lines that calculates loss according to Figure 2.

Mocov3 We adopt the official code[3] from Mocov3 [35]. To implement RINCE, we only modify the lines that calculates loss in `moco/builder.py` according to Figure 2.

B.3. Kinetics-400

We adopt the official implementation[4] from [19]. Similarly, we only modify the loss function in the `criterions` directory. In particular, we use the SimCLR style implementation for both InfoNCE and RINCE loss. We also adopt the same hyperparameters described in the git repository for training. We set the learning rate to $1e-3$ to finetune the models on downstream classification tasks such as UCF101 and HMDB51 with the provided evaluation code.

[2] https://github.com/PyTorchLightning/lightning-bolts/tree/master/pl_bolts/models/self_supervised/simclr

[3] https://github.com/facebookresearch/moco-v3

[4] https://github.com/facebookresearch/AVID-CMA

```
def compute_neg_mask(self):
    total_images = self.num_nodes * self.gpus * self.batch_size * self.num_pos
    world_size = self.num_nodes * self.gpus
    batch_size = self.batch_size * self.num_pos
    orig_images = self.batch_size
    rank = int(os.environ["LOCAL_RANK"])

    neg_mask = torch.zeros(batch_size, total_images)
    all_indices = np.arange(total_images)
    pos_members = orig_images * world_size * np.arange(self.num_pos)
    for anchor in np.arange(self.num_pos):
        for img_idx in range(orig_images):
            delete_inds = orig_images * rank + img_idx + pos_members
            neg_inds = torch.tensor(np.delete(all_indices, delete_inds)).long()
            neg_mask[anchor * orig_images + img_idx, neg_inds] = 1
    neg_mask = neg_mask.cuda(non_blocking=True)

    return neg_mask

def nt_xent_loss(self, out_1, out_2, temperature):
    if torch.distributed.is_available() and torch.distributed.is_initialized():
        out_1_dist = SyncFunction.apply(out_1)
        out_2_dist = SyncFunction.apply(out_2)
    else:
        out_1_dist = out_1
        out_2_dist = out_2

    out = torch.cat([out_1, out_2], dim=0)
    out_dist = torch.cat([out_1_dist, out_2_dist], dim=0)

    similarity = torch.exp(torch.mm(out, out_dist.t()) / temperature)

    ################################### original code ###################################
    # # from each row, subtract e^(1/temp) to remove similarity measure for x1.x1       #
    # neg = similarity.sum(dim=-1)                                                      #
    # row_sub = Tensor(neg.shape).fill_(math.e ** (1 / temperature)).to(neg.device)     #
    # neg = torch.clamp(neg - row_sub, min=eps)  # clamp for numerical stability      #
    #####################################################################################

    neg_mask = self.compute_neg_mask()
    neg = torch.sum(similarity * neg_mask, 1)

    pos = torch.exp(torch.sum(out_1 * out_2, dim=-1) / temperature)
    pos = torch.cat([pos, pos], dim=0)

    loss = -(torch.mean(torch.log(pos / (pos + neg))))

    return loss
```

Figure 10. PyTorch Lightening implementation of SimCLR. The original implementation of negative masking (commented out) is problematic because it subtracts $e^{1/t}$ to remove similarity measure for pairs that consist of the same images. However, subtracting a constant does not alter the gradient with respect to the model parameters. In particular, there are still gradients backpropagating through the false positive pairs. We fix it by directly filtering out those false pairs with a negative mask.

B.4. ACAV100M

We again modify the official implementation of [19] for the ACAV100M experiments, where we modify the data loader to adopt it to ACAV100M. Different from Kinetics-400 experiments, the input size is set to 8×224^2 during the finetuning process for computational efficiency. We again use the exact same set of hyperparameters from [19] for both training and testing.

B.5. TU-Dataset

We adopt the official implementation[5] from [40]. To implement RINCE, we only modify the loss in `gsimclr.py` file.

[5] https://github.com/Shen-Lab/GraphCL/tree/master/unsupervised_TU

C. Additional Results

C.1. Loss Visualization

We extend the analysis in section 4.1 by visualizing the loss and the scale of the gradients with respect to both positive scores s^+ and s^- in Figure 11. Interestingly, distinct from the analysis for positive pairs, two losses treat the negative pairs similarly. The gradient scale w.r.t. negative score increases when the negative score is large for both InfoNCE and RINCE as Figure 11 (c) shows, implying that both of them have the "hard negative sampling" scheme. The hard negative sampling strategy has been shown to improve the performance in downstream tasks [12]. In conclusion, InfoNCE ($q \to 0$) places more weights on hard positive and hard negative pairs, while fully symmetric RINCE ($q = 1$) put more emphasis on easy positive pairs and hard negative pairs.

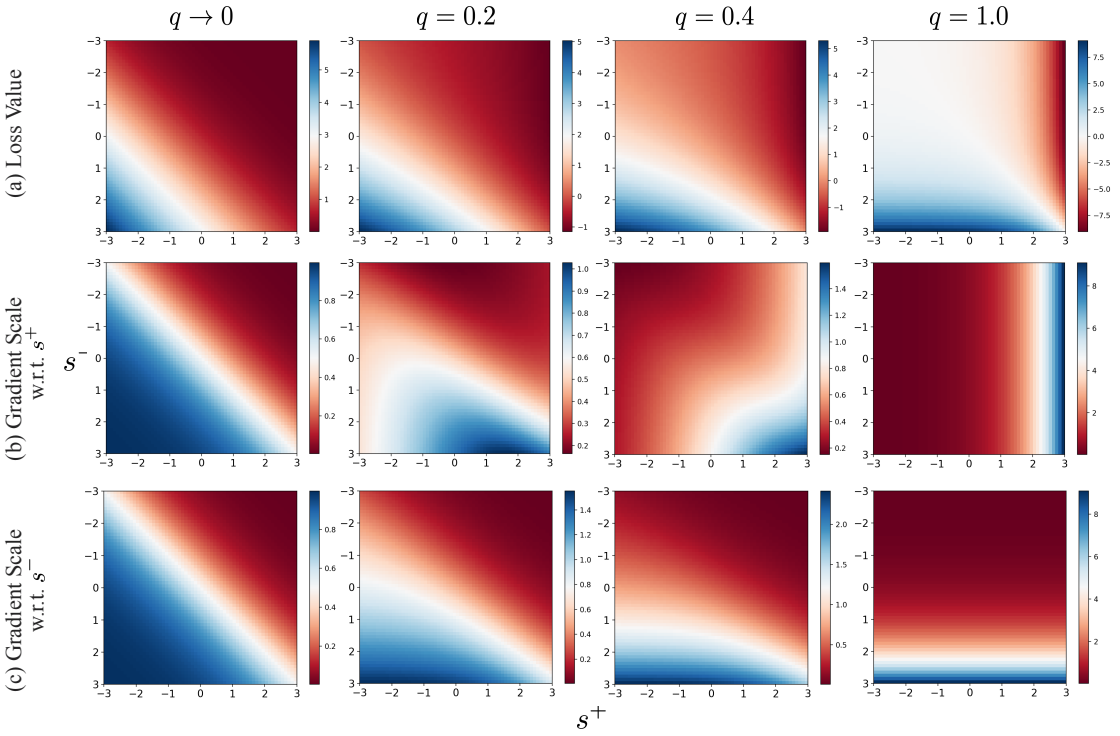


Figure 11. **Loss Visualization.** We visualize the (a) loss value and the (b) gradient scale with respect to the positive score s^+ and (c) gradient scale with respect to the negative score s^- for different q while setting $\lambda = 0.5$.

C.2. Exact Number of CIFAR-10 and ACAV100M Experiments

We first provide the exact numbers for CIFAR-10 and ACAV100M experiments.

η	InfoNCE	$q = 0.01$	$q = 0.1$	$q = 0.5$	$q = 1.0$
0.0	93.4±0.2	93.4±0.2	93.2±0.1	93.3±0.1	93.0±0.2
0.2	93.1±0.1	93.3±0.3	93.0±0.1	93.2±0.2	92.9±0.3
0.4	90.7±0.2	93.0±0.2	92.0±0.9	93.1±0.1	92.8±0.1
0.6	88.2±0.4	90.8±0.2	90.6±0.3	92.9±0.2	92.4±0.2
0.8	87.1±0.5	89.1±0.2	89.3±0.1	89.9±0.3	91.6±0.3
1.0	87.1±1.0	88.7±0.1	89.3±0.4	89.3±0.6	88.2±0.3

Table 4. CIFAR-10 Label Noise

η	InfoNCE	$q = 0.01$	$q = 0.1$	$q = 0.5$	$q = 1.0$
0.0	91.1±0.1	91.6±0.1	91.5±0.1	91.8±0.2	90.7±0.1
0.2	89.3±0.1	89.8±0.2	89.7±0.1	90.4±0.1	90.9±0.1
0.4	87.3±0.4	87.7±0.5	87.5±0.2	88.8±0.1	89.0±0.1
0.6	84.5±0.2	85.4±0.2	85.3±0.2	86.6±0.1	86.3±0.2
0.8	80.6±0.1	81.2±0.2	80.3±0.2	82.5±0.2	82.8±0.3
1.0	71.0±0.5	71.2±0.6	71.8±0.4	71.5±0.3	72.7±0.2

Table 5. CIFAR-10 Augmentation Noise

model	20K	50K	100K	200K	500K
InfoNCE (100 epoch)	72.482	75.205	77.161	79.937	82.717
InfoNCE (150 epoch)	72.429	76.13	78.8	80.095	83.082
InfoNCE (200 epoch)	72.429	76.183	78.641	79.94	83.388
RINCE (100 epoch)	73.635	76.685	78.694	81.153	83.505
RINCE (150 epoch)	74.632	77.505	79.064	82.263	83.399
RINCE (200 epoch)	74.253	78.086	79.355	82.368	83.769

Table 6. Top1 accuracy on UCF101 of models trained on ACAV100M.

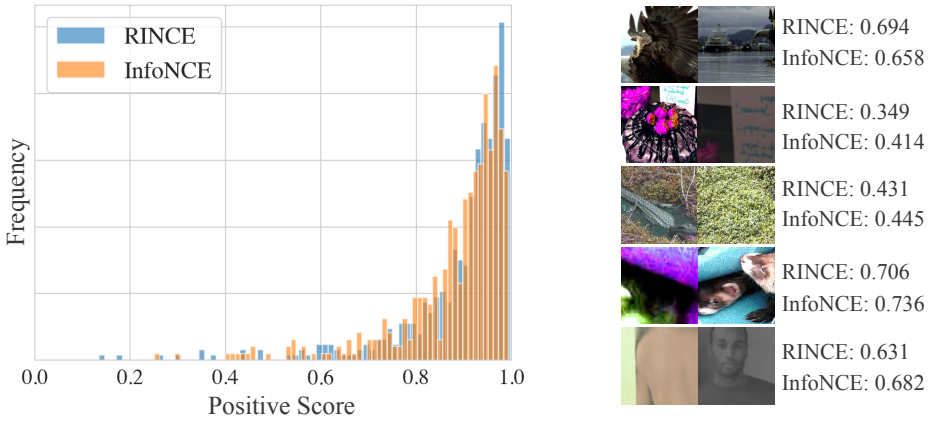
C.3. Positive Scores and Views, Continue

We extend our analysis of Figure 6 to InfoNCE baseline and discuss the impact of implicit weighting. We can see that the positive scores in both InfoNCE and RINCE models are correlated to the noisiness of positive pairs.

Figure 12. **Positive pairs and their scores.** The corresponding positive scores are shown below the image pairs. The positive scores $s^+ \in [-1, 1]$ are output by the trained InfoNCE and RINCE model (temperature $= 1$). Pairs that have lower scores are visually noisy, while informative pairs often have higher scores.

We then study the distribution of positive scores and compare the positive scores output by InfoNCE and RINCE on noisy views. As Figure 13 (a) shows, the positive scores of clean pairs output by RINCE is slightly higher, making the density of RINCE around score 1.0 larger than InfoNCE. Figure 13 (b) gives a closer look on scores versus noisy views. We can see that InfoNCE tends to output higher scores for noisy views than RINCE, corroborating our analysis: InfoNCE tends to

maximize the positive score of hard (noisy) pairs. This inherently makes the positive scores of clean pairs lower for InfoNCE, explaining the discrepancy between InfoNCE and RINCE in (a).



(a) Distribution of Positive Scores (b) Noisy Views and Scores

Figure 13. **Comparison between RINCE and InfoNCE.** (a) Distribution of Positive Scores for RINCE and InfoNCE; (b) InfoNCE outputs higher scores for noisy pairs.

C.4. Ablation Study on λ

Finally, we provide an ablation study on how λ affect the performance of RINCE with CIFAR-10 augmentation noise experiments. We can see that in both clean and noise setting, RINCE is not sensitive to the choice of λ as long as it is not too large. Therefore, we simply set $\lambda = 0.01$ for all vision experiments and $\lambda = 0.025$ for graph experiments.

Noise Rate	0.0	0.4
RINCE ($\lambda = 0.01$)	91.54	89.65
RINCE ($\lambda = 0.05$)	**91.81**	**89.81**
RINCE ($\lambda = 0.1$)	91.32	89.9
RINCE ($\lambda = 0.2$)	90.55	89.69
RINCE ($\lambda = 0.4$)	90.89	89.39

Table 7. CIFAR-10 Augmentation Noise