Discriminability-Transferability Trade-Off: An Information-Theoretic Perspective (Appendix)

Quan Cui[1,2*†] Bingchen Zhao[1,3*†] Zhao-Min Chen[1,4†] Borui Zhao[1]
Renjie Song[1] Jiajun Liang[1] Boyan Zhou[5] Osamu Yoshie[2]
[1]MEGVII Teconology [2]Waseda University [3]Tongji University [4]Wenzhou University [5]ByteDance
cui-quan@toki.waseda.jp, zhaobc.gm@gmail.com

A. Appendix

The appendix is composed of five parts. In Sec. A.1, we discuss the correlation between mutual information maximization and the Principal Components Analysis (PCA). In Sec. A.2, we elaborate the method for estimating mutual information, *i.e.*, MINE [1]. In Sec. A.3, we detail implementation details of all experiments in the manuscript. In Sec. A.4, we provide ablation studies on key components of CTC. In Sec. A.5, we provide a pseudo code of CTC.

A.1. Mutual Information and PCA

PCA is a linear dimensionality reduction method which finds the linear projection(s) of the data that has the maximum variance. Specifically, given a dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$ where $x_n \in \mathcal{R}^D$ with an assumption that the data is zero mean, $\frac{1}{N}\sum_i x_i = 0$ and $P(x)$ is Gaussian. Solving for the linear projection $y = w^\top x$ with PCA is solving for the following equation:

$$w^* = \arg\max_{w} var(y),$$

where $var(\cdot)$ is the variance function. Increasing the norm of w increases the variance of y, so we limited the norm of w to be a unit, *i.e.* $\|w\| = 1$.

Consider we are interested in finding another linear projection $\hat{y} = \hat{w}^\top x$ that has the maximum mutual information $I(x;\hat{y})$. We can have:

$$I(x;\hat{y}) = H(\hat{y}) - H(\hat{y}|x) = H(\hat{y})$$

So the goal is to maximize the entropy of $H(\hat{y})$, and because x is a zero mean gaussian, the linear transformation of x is still gaussian. Then, we have:

$$H(\hat{y}) = -\int p(\hat{y})\log p(\hat{y})d\hat{y} = \frac{1}{2}\ln|\Sigma| + \frac{D}{2}(1+\ln 2\pi),$$

where Σ is the covariance matrix. Therefore, maximize the entropy of $H(\hat{y})$ is maximize the variance of $\hat{y} = \hat{w}^\top x$ which is solving for:

$$\hat{w}^* = \arg\max_{\hat{w}} var(\hat{y}) \text{ subject to } \|\hat{w}\| = 1$$

So solving the PCA and solving for a linear projection that have the maximum mutual information is the same.

A.2. Mutual Information Neural Estimation

Mutual Information Neural Estimation (MINE) [1] estimates mutual information $I(X;Y)$ by training a classifier to distinguish between samples from the joint, $\mathbb{J}$, and the product of marginals, $\mathbb{M}$, of random variables X and Y.

We implement our estimator based on the open-source code from GitHub [1]. In our implementation, we adopt a Multi-Layer Perceptron (MLP) composed of four fully-connected layers with hidden dimension 1024, and ReLU is used as the activation function. For calculating $I(X;T)$, the input dimension is set to 3072 $(32 \times 32 \times 3) + 512$, which is the summation of the resolution of tiny images and the dimension of representations. The batch size and learning rate is set to 5K and 1e-4, respectively. For each model, we train the MLP for 10K steps. For calculating $I(T;Y)$, the input dimension is $512 + C$, which represents the summation of the dimention of representations and the number of classes in the dataset. The batch size and learning rate is set to 5K and 1e-5, respectively. For each model, we train the MLP for 20K steps. The Adam [**?**] optimizer is used.

A.3. Implementation Details

In Sec. 3 of the manuscript, we develop the baseline used for temporal analyses .

Sec. 3.1: Training on CIFAR-100. We optimize a ResNet18 [6] on the source dataset CIFAR-100 [7], with the SGD optimizer and a batch size of 64. The initial learning rate is 5e-2, and the learning rate follows a cosine decay scheduler. The weight decay is set to 5e-4 and the total training epoch is set to 200.

* Equal contribution
† Work done during an internship at MEGVII

[1]https://github.com/sungyubkim/MINE-Mutual-Information-Neural-Estimation-

Sec. 3.1: Transferring to STL-10 and CINIC-10. For STL-10 [2] and CINIC-10 [4] datasets, we re-train a classifier on top of the backbone learned on the source dataset at each epoch. Specifically, for every evaluated model, we train the classifier for 15K steps. We set the batch size to 512 and the initial learning rate to 4e-1. The learning rate is decayed by 0.1 at the 5K-th and 10K-th steps, respectively.

In Sec. 5.1 of the manuscript, we prove our motivation on CIFAR-100, and then conduct image classification tasks on both CIFAR-100 and ImageNet [11].

Sec. 5.1: Benchmarking on CIFAR-100. We mainly follow the baseline settings in Sec. A.3. Specifically, we set the initial learning rate to 5e-2 and the batch size to 64. The SGD optimizer and cosine learning rate scheduler are used. Note that we optimizer the model for 300 epochs as the baseline for fair comparisons with our CTC. As settings about CTC, we optimize the first stage (information aggregation stage) for 200 epochs and the second stage (information revitalization stage) for 100 epochs. The initial learning rate of the second stage is set to 5e-3 and also follows a cosine scheduler. The α and β are set to 0.01 and 1.0, respectively.

Sec. 5.1: Benchmarking on ImageNet. For experiments on the ImageNet dataset, we set the initial learning rate to 0.01 with a batch size of 256. We use the SGD optimizer and the cosine learning rate scheduler. The model is trained for a total of 200 epochs, where 120 epochs are for the first stage and the last 80 epochs are for the second stage. The parameters α and β are set to 0.2 and 1.0 respectively.

In Sec. 5.2 of the manuscript, we introduce the AutoAugment [3] and plug in our CTC for better transferability.

Sec. 5.2: Benchmarking on CIFAR-100. We set the initial learning rate to 5e-2 and the batch size to 64. The SGD optimizer and cosine learning rate scheduler are used. Note that we optimizer the model for 300 epochs as the baseline for fair comparisons with our CTC. As settings about CTC, we optimize the first stage (information aggregation stage) for 200 epochs and the second stage (information revitalization stage) for 100 epochs. The α and β are set to 0.01 and 1.0, respectively.

Sec. 5.2: Benchmarking on ImageNet. The experiments on ImageNet with AutoAugment are the same with Sec. 5.1.

In Sec. 5.3 of the manuscript, we transfer representations to various tasks, *i.e.*, object detection on COCO [9] and fine-grained visual categorization (FGVC) on CUB200 [13], Aircraft [10], and iNaturalist-18 [12].

Sec. 5.3: Object detection on COCO. For the experiments transferring the learned representation to object detection on COCO, we use the `train2017` split for training the model and the `val2017` split to test the finetuned model. We adopted the Mask-RCNN [5] with FPN [8] as the architecture for detection, and the model is trained with the $1\times$ schedule with a maximum of 180K iterations of training, the learning rate is set to 0.02 and the batchsize is 16, step decay schedule is used, the learning rate will be multiplied by 0.1 at 120K and 160K iterations.

Sec. 5.3: FGVC on CUB200, Aircraft, and iNaturalist-18. For transfer learning experiments on fine-grained classification datasets, we finetune the pretrained model with 100 epochs, and the learning rate is set to 5e-3 with cosine decay, and the batchsize is 64 for CUB-200 and Aircraft, and is 256 for iNaturalist-18.

A.4. Ablation Studies

We conduct ablation studies on the source dataset CIFAR-100 and target dataset STL-10.

Ablation: effects of $\mathcal{L}_{\text{IAS}}$. To prove the effectiveness of $\mathcal{L}_{\text{IAS}}$, we provide an ablation study on the loss function of the information aggregation stage. Specifically, we optimize the first stage with only the $\mathcal{L}_{\text{CE}}$ and then fine-tune the second stage with both $\mathcal{L}_{\text{IRS}}$ and $\mathcal{L}_{\text{CE}}$. Classification results are as followed:

method	$\mathcal{L}_{\text{IAS}}$	CIFAR-100 top-1 (%)
ResNet18+CTC(Ours)	–	**80.1**
ResNet18+CTC(Ours)	✓	**80.1**

It can be observed that the final classification accuracy is not influenced. After the training, we transfer learned representations to STL-10 [2]. Results are shown in Figure 1. Without the $\mathcal{L}_{\text{IAS}}$, the information bank (at the of the first stage) cannot provide a good transferability, which can result from the over-compression, and thus the second stage achieves worse transferability. It proves the effectiveness of $\mathcal{L}_{\text{IAS}}$ on selecting a satisfactory information bank.

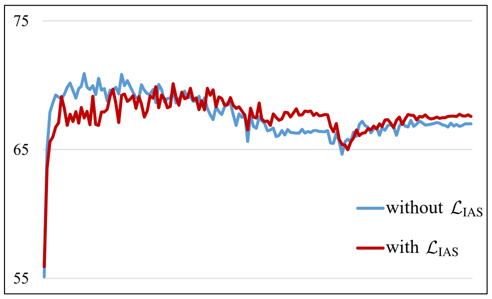

Figure 1. Temporal transferring results on the STL-10 dataset. X and Y axes represent the training process and classification accuracy, respectively. Red and blue lines represent training with and without $\mathcal{L}_{\text{IAS}}$. It demonstrates that training without $\mathcal{L}_{\text{IAS}}$ will lead to worse transferability.

Ablation: effects of α. It is worth mentioning that our proposed CTC is a flexible method. In the manuscript, we use $\alpha = 0.01$ for achieving both better transferability and discriminability. Besides, a small α ensures the final classification accuracies on CIFAR-100 and ImageNet are better than the baseline in Table 1–4 of the manuscript.

In this part, by further finding a trade-off between discriminability and transferability, we demonstrate that our CTC can significantly improve transferability without damaging discriminability. Specifically, in the first stage, we assign a large weight α to the $\mathcal{L}_{\text{IAS}}$, and, unavoidably, the gradient of $\mathcal{L}_{\text{CE}}$ will be influenced, and the discriminability can be damaged. However, larger α could contribute to higher transferability and build a foundation for counteracting over-compression in the second stage. In the following, we first train the model with α 0.5 and then perform transferring tasks to show that our method can achieve outstanding transferability by sacrificing marginal discriminability and the final classification accuracy. The below table shows classification accuracies on CIFAR-100:

method	α	CIFAR-100 top-1 (%)
ResNet18+CosLr	–	79.3
ResNet18+CTC(Ours)	0.01	**80.1**
ResNet18+CTC(Ours)	0.5	79.4

It can be observed that increasing α to 0.5 would result in a worse final classification accuracy than setting α to 0.01. However, the result is still better than the baseline method. Then, we visualize the temporal transferring results on STL-10 in Figure 2. As illustrated, the transferability is greatly improved by setting α to 0.5. It supports the effectiveness of $\mathcal{L}_{\text{IAS}}$ and further demonstrates the existence of a trade-off between discriminability and transferability. More importantly, the potential of our method for achieving better transferability is proven, and our method can flexibly adjust the trade-off to achieve better discriminability or transferability.

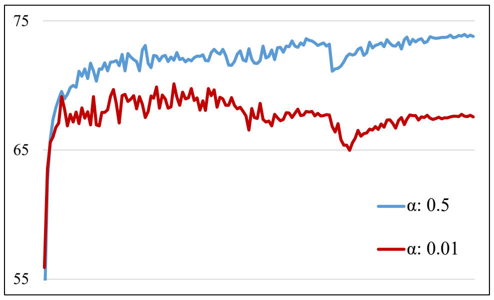

Figure 2. Temporal transferring results on the STL-10 dataset. X and Y axes represent the training process and classification accuracy, respectively. Red and blue lines represent training with $\alpha = 0.01$ and $\alpha = 0.5$.

Ablation: effects of β. With $\alpha = 0.01$, we adjust the value of β and conduct experiments on CIFAR-100. Results are shown in the following table:

method	β	CIFAR-100 top-1 (%)
ResNet18+CTC(Ours)	0.5	79.7
ResNet18+CTC(Ours)	1.0	**80.1**
ResNet18+CTC(Ours)	1.5	79.9

It can be observed that the model performs well with β ranging from 0.5 to 1.5. Then, we visualize temporal transferring results in Figure 3. As we can see, setting $\beta = 1.0$ yields the best transferring results on STL-10. For the best results of both image classification and transfer learning, we use $\beta = 1.0$ in all experiments of the manuscript.

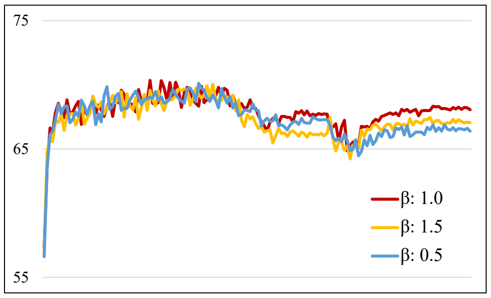

Figure 3. Temporal transferring results on the STL-10 dataset. X and Y axes represent the training process and classification accuracy, respectively. Red, yellow, and blue lines represent training with $\beta = 1.0$, $\beta = 1.5$ and $\beta = 0.5$, respectively.

A.5. Pseudo Code of CTC

A PyTorch-Style pseudo code of the CTC method is given in Alg 1.

References

[1] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. MINE: mutual information neural estimation. *arXiv:1801.04062*, 2018. 1

[2] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011. 2

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019. 2

[4] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv:1810.03505*, 2018. 2

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

Algorithm 1 Pseudo code of CTC in a PyTorch-like style.

```
# net: the network
# memory: the memory bank for holding representations
# e1, e2: numbers of epochs for two stages
# extract: a function to extract representations
# sample: a function to sample representations from the memory bank
# alpha, beta: hyper-parameters

for _ in e1: # Information aggregation stage
  for x in loader:
    logits = net.forward(x)
    t_1 = net.extract(x)
    v_1 = memory.sample(x) # Sampling contrastive samples from the memory
    loss_ias = ContrastiveLoss(t_1, v_1)# Calculating the IAS loss
    loss_ce = CrossEntropyLoss(logits, labels)
    loss = alpha * loss_ias + loss_ce
    loss.backward()
    update(net.param)
    update(memory, t_1) # Updating memory bank

information_bank = net

for _ in e2: # Information revitalization stage
  for x in loader:
    logits = net.forward(x)
    t_2 = net.extract(x)
    t_1_hat = information_bank.extract(x).detach()
    loss_irs = ContrastiveLoss(t_2, t_1_hat) # Calculating the IRS loss
    loss_ce = CrossEntropyLoss(logits, labels)
    loss = beta * loss_irs + loss_ce
    loss.backward()
    update(net.param)
```

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2

[10] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv:1306.5151*, 2013. 2

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 2

[12] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 2

[13] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2