A Appendix

The appendix is composed of 7 parts. In Sec. A.1, we discuss the correlation between mutual information maximization and the Principal Components Analysis (PCA). In Sec.A.2, we provide extensive experiments to reveal the drawback of over-compression. In Sec. A.3, we provide more experiments to validate our CTC alleviates the over-compression. In Sec. A.4, we provide training and test loss curves to prove that over-compression is not caused by over-fitting. In Sec. A.5, we elaborate the method for estimating mutual information, *i.e.*, MINE [1]. In Sec. A.6, we detail implementation details of all experiments in the manuscript. In Sec. A.7, we provide the pseudo code of CTC.

A.1 Connection between 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 Observing Over-Compression on More Datasets and Networks

In this part, we provide more transferring results on various datasets and networks. The source dataset is CIFAR-100 [8], and the target datasets are composed of

6 widely used datasets, *i.e.*, CIFAR-10 [8], STL-10 [2], CINIC-10 [4], TinyImageNet [1], SVHN [13], and Fashion-MNIST [17]. Backbone choices are composed of 4 effective models, *i.e.*, ResNet18 [6], WRN-28-4 [19], ShuffleNetV2 [11], and ResNext32-16x4d [18]. It could be observed in Figure 1 that, with different networks and datasets, the transferability consistently goes to a peak and begins to decrease.

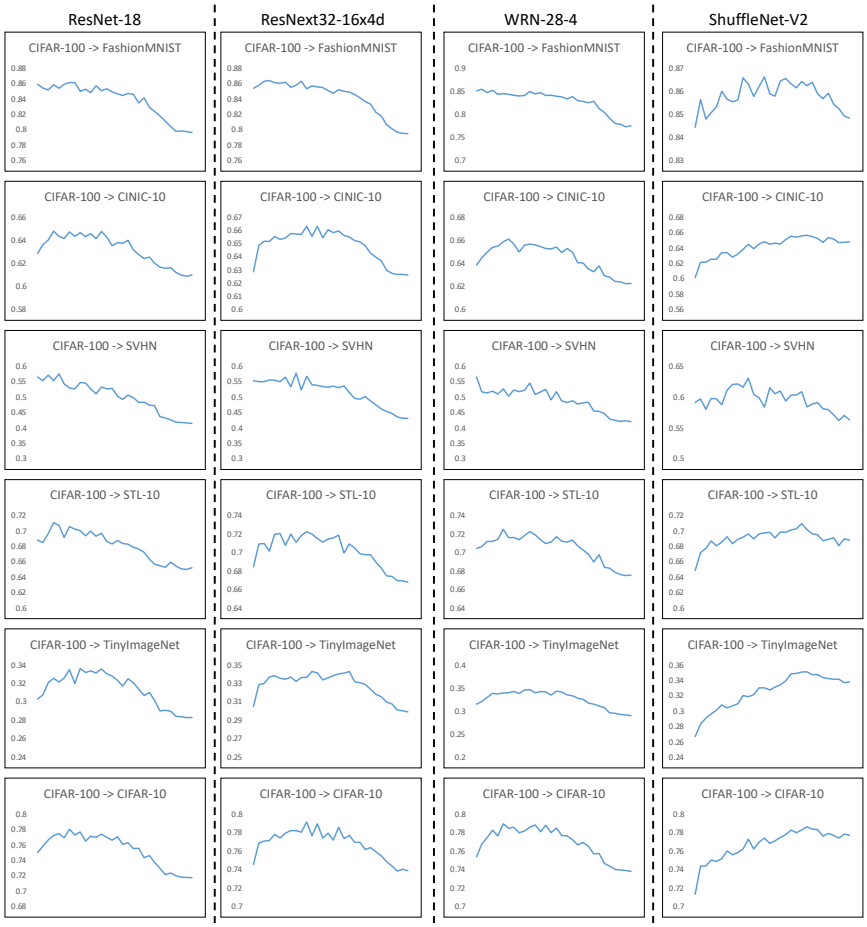

Fig. 1. Temporal transferring results of vanilla training on the CIFAR-10, STL-10, CINIC-10, TinyImageNet, SVHN, and Fashion-MNIST datasets. X and Y axes represent the training process and top-1 accuracy, respectively.

Similar to experiments in Section 3 of the manuscript, we further provide the information dynamics of the aforementioned networks. Specifically, we evaluate the

[1] https://www.kaggle.com/c/tiny-imagenet

$I(X;T)$ and $I(T;Y)$ on source dataset (CIFAR-100) and target datasets (STL-10 and CINIC-10), with ResNet-18, ResNext32-16x4d, WRN-28-4, and ShuffleNetV2. Results are reported in Figure 2, we still observe over-compression since $I(X;T)$ decrease on all experiments.

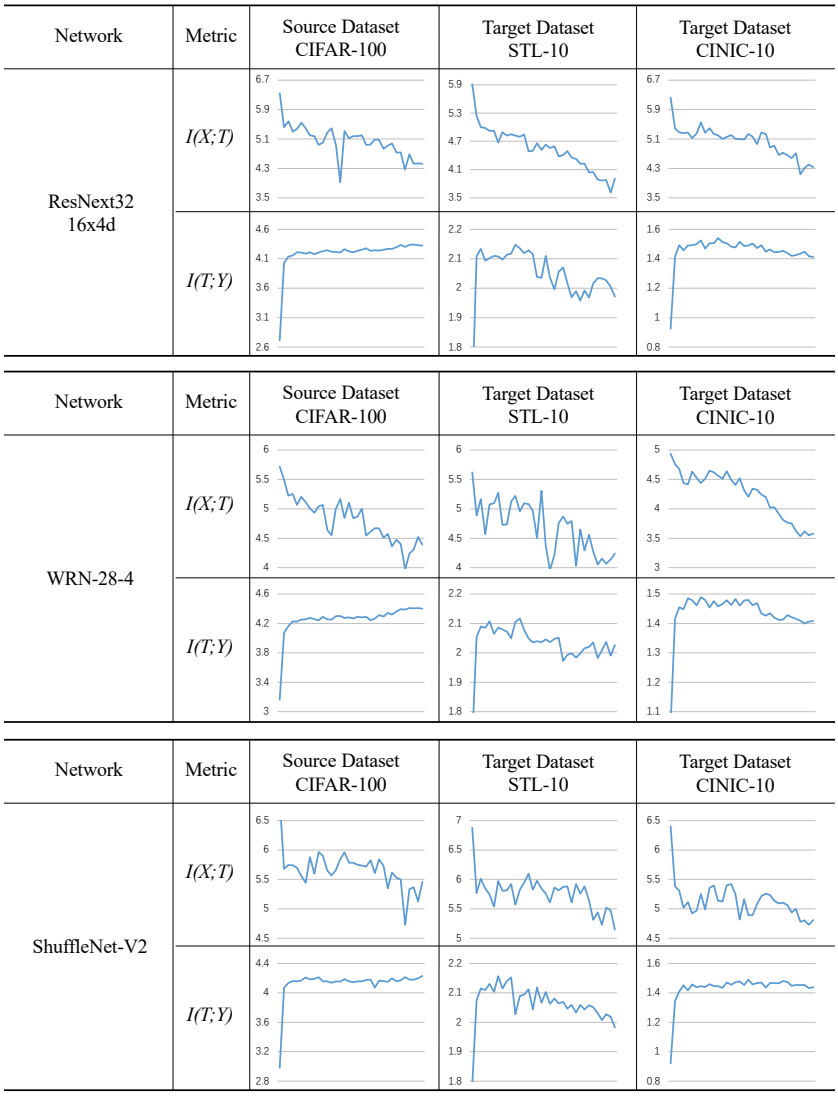

Fig. 2. Mutual information dynamics of vanilla training on CIFAR-100 and transferring to STL-10 and CINIC-10.

A.3 Alleviating Over-Compression on More Datasets and Networks

In this part, we demonstrate out proposed CTC alleviates over-compression on all evaluated datasets and networks mentioned in Sec. A.2. Especially, in the later training, the transferability does not decrease. It can be reflected that over-compression could be a general problem in the transfer learning, and deserves further explorations. Our proposed method provides a practical strategy for achieving this goal.

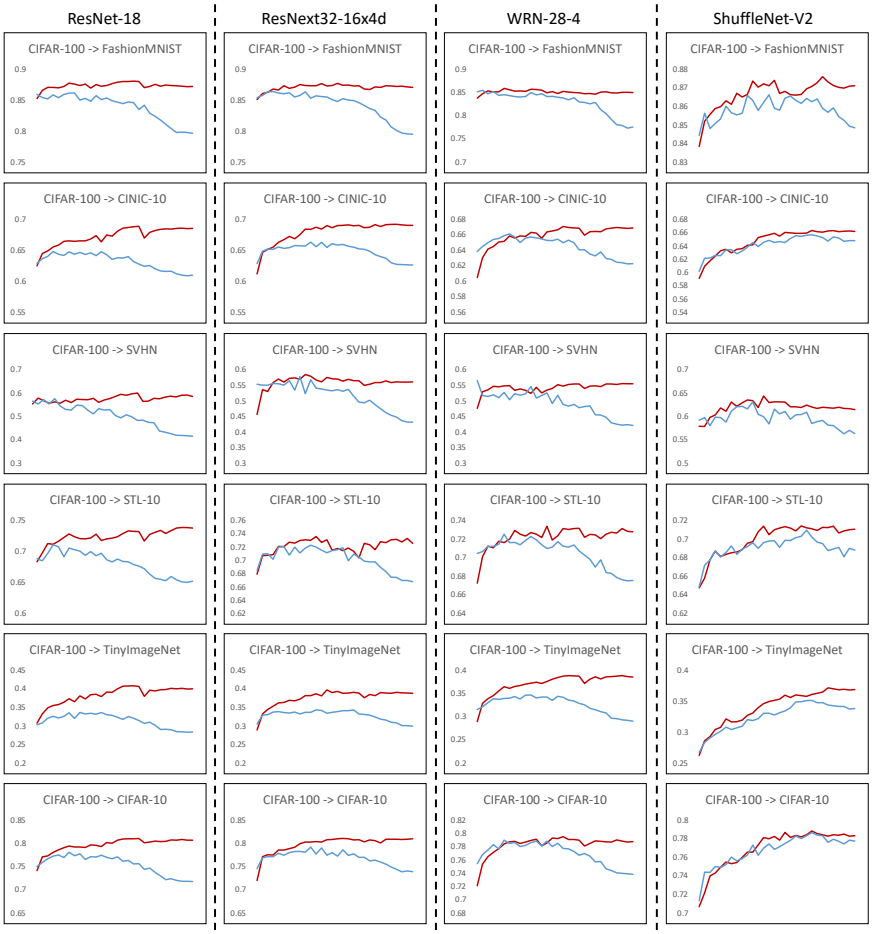

Fig. 3. Temporal transferring results of vanilla and CTC training on the CIFAR-10, STL-10, CINIC-10, TinyImageNet, SVHN, and Fashion-MNIST datasets. X and Y axes represent the training process and top-1 accuracy, respectively. Here we use $\alpha = 0.5$ for proving that CTC is able to achieve much better tranferability than the vanilla training.

A.4 Does over-fitting happen?

As defined in Deep Learning book [2], over-fitting happens when the generalization error on the *source test* set decreases, but we did not observe such decreases on source datasets. As shown in Figure 4, we provide the training and test errors of training ResNet-18 on CIFAR-100, and no decrease of generalization error is observed. Over-compression is related to the deteriorating transferring results while the performance on the source test set keeps improving.

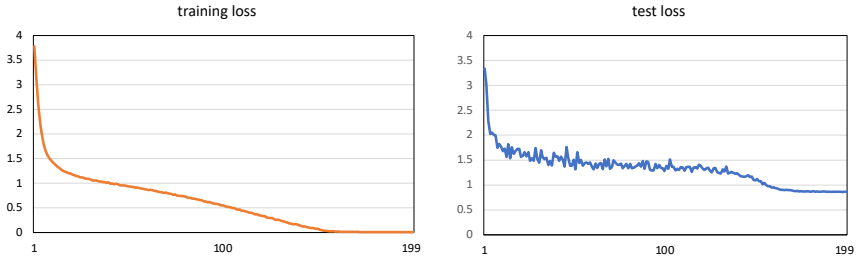

Fig. 4. Training and test loss curves of ResNet-18 on CIFAR-100.

A.5 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 [3]. 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 1K 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 10K steps. The Adam [7] optimizer is used.

A.6 Implementation Details

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

[2] https://www.deeplearningbook.org/contents/ml.html, pp. 108–114.
[3] https://github.com/sungyubkim/MINE-Mutual-Information-Neural-Estimation-

Sec. 3.1: Training on CIFAR-100. We optimize a ResNet18 [6] on the source dataset CIFAR-100 [8], 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. For avoiding effects of over-fitting on information dynamics, we set a minimum learning rate 1e-2 and early stop the training at the 190-th epoch.

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 [14].

Sec. 5.1: Benchmarking on CIFAR-100. We mainly follow the baseline settings in Sec. A.6. 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 temperature parameters for the first and second stages are 0.5 and 0.4. The α and β are set to 0.1 and 1.0, respectively.

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

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)
```

In Sec. 5.3 of the manuscript, we transfer representations to various tasks, *i.e.*, object detection on COCO [10] and fine-grained visual categorization (FGVC) on CUB200 [16], Aircraft [12], and iNaturalist-18 [15].

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 [9] 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.7 Pseudo Code of CTC

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

References

1. Belghazi, M.I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., Hjelm, R.D.: MINE: mutual information neural estimation. arXiv:1801.04062 (2018)
2. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: AISTATS (2011)
3. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: CVPR (2019)
4. Darlow, L.N., Crowley, E.J., Antoniou, A., Storkey, A.J.: Cinic-10 is not imagenet or cifar-10. arXiv:1810.03505 (2018)
5. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV (2017)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
8. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
9. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
10. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)
11. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: ECCV. pp. 116–131 (2018)
12. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv:1306.5151 (2013)
13. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
14. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: ImageNet large scale visual recognition challenge. IJCV (2015)
15. Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., Belongie, S.: The inaturalist species classification and detection dataset. In: CVPR (2018)
16. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
17. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747 (2017)
18. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR. pp. 1492–1500 (2017)
19. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)