REVISITING LOCALLY SUPERVISED TRAINING OF DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Due to the need to store the intermediate activations for back-propagation, end-to-end (E2E) training of deep networks usually suffers from high GPUs memory footprint. This paper aims to address this problem by revisiting the locally supervised learning, where a network is split into gradient-isolated modules and trained with local supervision. We experimentally show that simply training local modules with E2E loss tends to collapse task-relevant information at early layers, and hence hurts the performance of the full model. To avoid this issue, we propose an information propagation (InfoPro) loss, which encourages local modules to preserve as much useful information as possible, while progressively discard task-irrelevant information. As InfoPro loss is difficult to compute in its original form, we derive a feasible upper bound as a surrogate optimization objective, yielding a simple but effective algorithm. In fact, we show that the proposed method boils down to minimizing the combination of a reconstruction loss and a normal cross-entropy/contrastive term. Extensive experiments on five datasets (i.e., CIFAR, SVHN, STL-10, ImageNet and Cityscapes) validate that our method is capable of achieving competitive performance with fewer than 40% memory footprint compared to E2E training, or enables training local modules asynchronously for potential training acceleration.

1 INTRODUCTION

End-to-end (E2E) back-propagation has become a standard paradigm to train deep networks (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2016; Huang et al., 2019). Typically, a training loss is computed at the final layer, and then the gradients are propagated backward layer-by-layer to update the weights. Although being effective, this procedure may suffer from memory/computation inefficiencies. First, the entire computational graph as well as the activations of most, if not all, layers need to be stored, resulting in intensive memory consumption. The GPU memory constraint is usually a bottleneck that inhibits the training of state-of-the-art models with high-resolution inputs and sufficient batch sizes, which arises in many realistic scenarios, such as 2D/3D object detection in autonomous driving, tissue segmentation in medical imaging and object recognition from remote sensing data. Existing work has addressed this issue via gradient checkpointing technique (Salimans & Bulatov, 2017) or special architecture design (Gomez et al., 2017), while they both come at the cost of significantly increased computation. Second, E2E training is a sequential process that impedes model parallelization (Belilovsky et al., 2020; Löwe et al., 2019), as earlier layers need to wait for their successors for error signals.

As an alternative to E2E training, the locally supervised learning paradigm (Hinton et al., 2006; Bengio et al., 2007; Nøkland & Eidnes, 2019; Belilovsky et al., 2019; 2020) by design enjoys higher memory efficiency and allows for model parallelization. In specific, it divides a deep network into several gradient-isolated modules and train them separately under local supervision (see Figure 1 (b)). Since back-propagation is performed only within local modules, one does not need to store all intermediate activations at the same time. Consequently, the memory footprint during training is reduced without involving significant computational overhead. Moreover, by removing the demands for obtaining error signals from later layers, different local modules can potentially be trained in parallel. This approach is also considered more biologically plausible, given that brains are highly modular and predominantly learn from local signals (Crick, 1989; Dan & Poo, 2004; Bengio et al.,

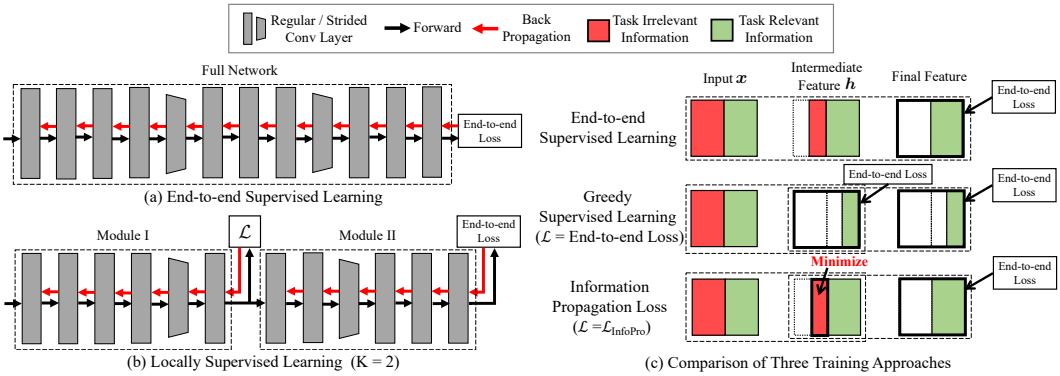

Figure 1: (a) and (b) illustrate the paradigms of end-to-end (E2E) learning and locally supervised learning ($K=2$). "End-to-end Loss" refers to the standard loss function used by E2E training, e.g., softmax cross-entropy loss for classification, etc., while $\mathcal{L}$ denotes the loss function used to train local modules. (c) compares three training approaches in terms of the information captured by features. Greedy supervised learning (greedy SL) tends to collapse some of task-relevant information with the beginning module, leading to inferior final performance. The proposed information propagation (InfoPro) loss, however, alleviates this problem by encouraging local modules to propagate forward all the information, while maximally discard task-irrelevant information.

2015). However, a major drawback of local learning is that they usually lead to inferior performance compared to E2E training (Mostafa et al., 2018; Belilovsky et al., 2019; 2020).

In this paper, we revisit locally supervised training and analyse its drawbacks from the information-theoretic perspective. We find that directly adopting an E2E loss function (i.e., cross-entropy) to train local modules produces more discriminative intermediate features at earlier layers, while it collapses task-relevant information from the inputs and leads to inferior final performance. In other words, locally supervised learning tends to be short sighted, and learns features that only benefit local modules, while ignoring the demands of the rest layers. Once task-relevant information is washed out in earlier modules, later layers cannot take full advantage of their capacity to learn more powerful representations.

Based on the above observations, we hypothesize that a less greedy training procedure that preserves more information about the input might be a rescue for locally supervised training. Therefore, we propose a less greedy information propagation (InfoPro) loss that aims to encourage local modules to propagate forward as much information from the inputs as possible, while progressively abandon task-irrelevant parts (formulated by an additional random variable named nuisance), as shown in Figure 1 (c). The proposed method differentiates itself from existing algorithms (Nøkland & Eidnes, 2019; Belilovsky et al., 2019; 2020) on that it allows intermediate features to retain a certain amount of information which may hurt the short-term performance, but can potentially be leveraged by later modules. In practice, as the InfoPro loss is difficult to estimate in its exact form, we derive a tractable upper bound, leading to surrogate losses, e.g., cross-entropy loss and contrastive loss.

Empirically, we show that InfoPro loss effectively prevents collapsing task-relevant information at local modules, and yields favorable results on five widely used benchmarks (i.e., CIFAR, SVHN, STL-10, ImageNet and Cityscapes). For instance, it achieves comparable accuracy as E2E training using 40% or less GPU memory, or allows using 50% larger batch sizes with the same memory constraints. Additionally, our method enables training different local modules in parallel(even asynchronously).

2 Why locally supervised learning underperforms E2E training?

We start with considering a local learning setting where a deep network is split into multiple successively stacked modules, each with the same depth. The inputs are fed forward in an ordinary way, while the gradients are produced at the end of every module and back-propagated until reaching an earlier module. To generate supervision signals, a straightforward solution is to train all the local modules as independent networks, e.g., in classification tasks, attaching a classifier to each module, and computing the local classification loss such as cross-entropy. However,

Table 1: Test errors of a ResNet-32 using greedy SL on CIFAR-10. The network is divided into K successive local modules. Each module is trained separately with the softmax cross-entropy loss by appending a global-pool layer followed by a fully-connected layer (see Appendix F for details). "$K=1$" refers to end-to-end (E2E) training.

	$K=1$	$K=2$	$K=4$	$K=8$	$K=16$
Test Error	7.37%	10.30%	16.07%	21.19%	24.59%

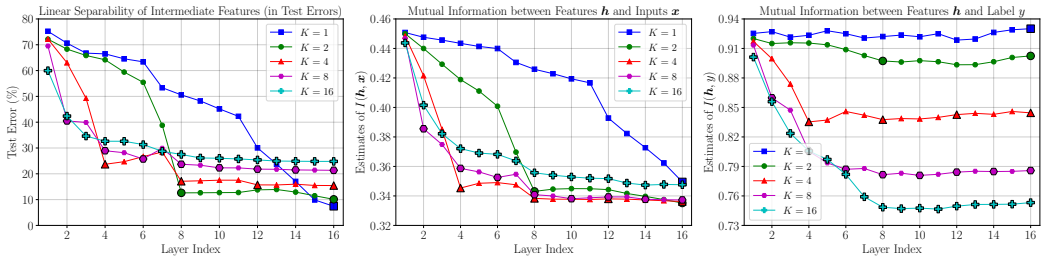

Figure 2: The linear separability (*left*, measured by test errors), mutual information with the input $\boldsymbol{x}$ (*middle*), and mutual information with the label y (*right*) of the intermediate features $\boldsymbol{h}$ from different layers when the greedy supervised learning (greedy SL) algorithm is adopted with K local modules. The ends of local modules are marked using larger markers with black edges. The experiments are conducted on CIFAR-10 with a ResNet-32.

such a *greedy* version of the standard supervised learning algorithm leads to inferior performance of the whole network. For instance, in Table 1, we present the test error of a ResNet-32 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009) when it is greedily trained with K modules. One can observe a severe degradation (even more than 15%) with K growing larger. Plausible as this phenomenon seems, it remains unclear whether it is inherent for local learning and how to alleviate this problem. In this section, we investigating the performance degradation issue of the greedy local training from an information-theoretic perspective, laying the basis for the algorithm proposed in the next section.

Linear separability of intermediate features. In the common case that greedy SL operates directly on the features output by internal layers, a natural intuition is to investigate how these locally learned features differ from their E2E learned counterparts in task-relevant behaviors. To this end, we fix the networks in Table 1, and train a linear classifier using the features from each layer. The test errors of these classifiers are presented in the left plot of Figure 2, where the horizontal axis denotes the indices of layers. The plot shows an intriguing trend: greedy SL contributes to dramatically more discriminative features with the first one (or few) local module, but is only able to slightly improve the performance with all the consequent modules. In contrast, the E2E learned network progressively boosts the linear separability of features throughout the whole network with even more significant effects in the later layers, surpassing greedy SL eventually. This raises an interesting question: *why does the full network achieve inferior performance in greedy SL compared to the E2E counterparts, even the former is based on more discriminative earlier features?* This observation appears incompatible with prior works like deeply-supervised nets (Lee et al., 2015a).

Information in features. Since we use the same training configuration for both greedy SL and E2E learning, we conjecture that the answer to the above question might lie in the differences of features apart from merely separability. To test that, we look into the information captured by the intermediate features. In specific, given intermediate feature $\boldsymbol{h}$ corresponding to the input data $\boldsymbol{x}$ and the label y (all of them are treated as random variables), we use the mutual information $I(\boldsymbol{h}, \boldsymbol{x})$ and $I(\boldsymbol{h}, y)$ to measure the amount of retained information and the task-relevant information in $\boldsymbol{h}$, respectively. As these metrics cannot be directly computed, we estimate the former by training a decoder with binary cross-entropy loss to reconstruct $\boldsymbol{x}$ from $\boldsymbol{h}$, and for the latter we train a CNN using $\boldsymbol{h}$ as inputs to correctly classify $\boldsymbol{x}$, and estimate $I(\boldsymbol{h}, y)$ with its performance. Details are deferred to Appendix G.

The estimates of $I(\boldsymbol{h}, \boldsymbol{x})$ and $I(\boldsymbol{h}, y)$ at different layers are shown in the middle and right plots of Figure 2. We note that in E2E learned networks, $I(\boldsymbol{h}, y)$ remains unchanged when the features pass through all the layers, while $I(\boldsymbol{h}, \boldsymbol{x})$ reduces gradually, revealing that the models progressively discard task-irrelevant information. However, greedily trained networks collapse both $I(\boldsymbol{h}, \boldsymbol{x})$ and $I(\boldsymbol{h}, y)$ in their first few modules. We attribute this to the short sighted optimization objective of earlier modules, which have relatively small capacity compared with full networks and are not capable of extracting and leveraging all the task-relevant information in $\boldsymbol{x}$, as the E2E learned networks do. As a consequence, later modules, even introduce additional parameters and increased capacity, lack necessary information about the original inputs $\boldsymbol{x}$ to construct more discriminative features.

Information collapse hypothesis. The above observations suggest that greedy SL induces local modules to collapse some of the task-relevant information that may be useless for short-term performance. However, these information is useful for the full model. We postulate that, although E2E training is incapable of extracting all task-relevant information at earlier layers as well, it alleviates this phenomenon by allowing a larger amount of task-irrelevant information to be kept, even though it

may not be ideal for short-term performance. More empirical validation of our hypothesis is provided in Appendix A.

3 INFORMATION PROPAGATION (INFOPRO) LOSS

In this section, we propose an information propagation (InfoPro) loss to address the issue of information collapse in locally supervised training. The key idea is to enforce local modules to retain as much information about the input as possible, while progressively discard task-irrelevant parts. As it is difficult to estimate InfoPro loss in its exact form, we derive an easy-to-compute upper bound as the surrogate loss, and analyze its tightness.

3.1 LEARNING TO DISCARD USELESS INFORMATION

Nuisance. We first model the task-irrelevant information in the input data $\boldsymbol{x}$ by introducing the concept of nuisance. A nuisance is defined as an arbitrary random variable that affects $\boldsymbol{x}$ but provides no helpful information for the task of interest (Achille & Soatto, 2018). Take recognizing a car in the wild for example. The random variables determining the weather and the background are both nuisances. Formally, given a nuisance r, we have $I(r, \boldsymbol{x}) > 0$ and $I(r, y) = 0$, where y is the label. Without loss of generality, we suppose that y, r, $\boldsymbol{x}$ and $\boldsymbol{h}$ form the Markov chain $(y, r) \to \boldsymbol{x} \to \boldsymbol{h}$, namely $p(\boldsymbol{h}|\boldsymbol{x}, y, r) = p(\boldsymbol{h}|\boldsymbol{x})$. As a consequence, for the intermediate feature $\boldsymbol{h}$ from any layer, we obviously have $I(r, \boldsymbol{x}) \geq I(r, \boldsymbol{h})$. Nevertheless, we postulate that $\operatorname{argmax}_r I(r, \boldsymbol{h}) > 0$. This assumption is mild since it does not hold only when $\boldsymbol{h}$ strictly contains no task-irrelevant information.

Information Propagation (InfoPro) Loss. Thus far, we have been ready to introduce the proposed InfoPro loss. Instead of overly emphasizing on learning highly discriminative features at local modules, we also pay attention to preventing collapsing useful information in the feed-forward process. A simple solution to achieve this is maximizing the mutual information $I(\boldsymbol{h}, \boldsymbol{x})$. Ideally, if there is no information loss, all useful information will be retained. However, it goes to another extreme case where the local modules do not learn any task-relevant feature, and is obviously dispensable. By contrast, in E2E training, intermediate layers progressively discard useless (task-irrelevant) information as well as shown above. Therefore, to model both effects simultaneously, we propose the following combined loss function:

$$\mathcal{L}_{\text{InfoPro}}(\boldsymbol{h}) = \alpha[-I(\boldsymbol{h}, \boldsymbol{x}) + \beta I(r^*, \boldsymbol{h})],\ \alpha, \beta > 0, \quad s.t.\ r^* = \underset{r,\ I(r,\boldsymbol{x})>0,\ I(r,y)=0}{\operatorname{argmax}} I(r, \boldsymbol{h}), \tag{1}$$

where the nuisance r^* is formulated to capture as much task-irrelevant information in $\boldsymbol{h}$ as possible, and the coefficient β controls the amount of useful information that is propagated forward (first term) and task-irrelevant information that is discarded (second term). Notably, we assume that the final module is always trained using the normal E2E loss (e.g. softmax cross-entropy for classification) weighted by the constant 1, such that α is essential to balance the intermediate loss and the final one. In addition, $\mathcal{L}_{\text{InfoPro}}(\boldsymbol{h})$ is used to train the local module outputting $\boldsymbol{h}$, whose inputs are not required to be $\boldsymbol{x}$. The module may stack over another local module trained with the same form of $\mathcal{L}_{\text{InfoPro}}(\boldsymbol{h})$ but (possibly) different α and β.

Our method differs from existing works (Nøkland & Eidnes, 2019; Belilovsky et al., 2019; 2020) in that it is a non-greedy approach. The major effect of minimizing $\mathcal{L}_{\text{InfoPro}}(\boldsymbol{h})$ can be described as maximally discarding the task-irrelevant information under the constraint of retaining all task-relevant information. Obtaining high short-term performance is not necessarily required. As we explicitly facilitate information propagation, we refer to $\mathcal{L}_{\text{InfoPro}}(\boldsymbol{h})$ as the InfoPro loss.

3.2 UPPER BOUND OF $\mathcal{L}_{\text{InfoPro}}$

The objective function in Eq.(1) is difficult to be directly optimization, since it is usually intractable to estimate r^*, which is equivalent to disentangle all task-irrelevant information from intermediate features. Therefore, we derive an easy-to-compute upper bound of $\mathcal{L}_{\text{InfoPro}}$ as an surrogate loss. Our result is summarized in Proposition 1, with the proof in Appendix B.

Proposition 1. *Suppose that the Markov chain $(y, r) \to \boldsymbol{x} \to \boldsymbol{h}$ holds. Then an upper bound of $\mathcal{L}_{\text{InfoPro}}$ is given by*

$$\mathcal{L}_{\text{InfoPro}} \leq -\lambda_1 I(\boldsymbol{h}, \boldsymbol{x}) - \lambda_2 I(\boldsymbol{h}, y) \triangleq \bar{\mathcal{L}}_{\text{InfoPro}}, \tag{2}$$

where $\lambda_1 = \alpha(1 - \beta)$, $\lambda_2 = \alpha\beta$.

For simplicity, we integrate α and β into two mutually independent hyper-parameters, λ_1 and λ_2. Although we do not explicitly restrict $\lambda_1 > 0$, we find in experiments that the performance of

networks is significantly degraded with $\lambda_1 \to 0^+$ (see Figure 4), or say, $\beta \to 1^-$, where models tend to reach local minima by trivially minimizing $I(r^*, \boldsymbol{h})$ in Eq. (1). Thus, we assume $\lambda_1, \lambda_2 > 0$.

With Proposition 1, we can optimize the upper bound $\bar{\mathcal{L}}_{\text{InfoPro}}$ as an approximate, circumventing dealing with the intractable term $I(r^*, \boldsymbol{h})$ in $\mathcal{L}_{\text{InfoPro}}$. To ensure that the approximation is accurate, the gap between the two should be reasonably small. Below we give an analysis of the tightness of $\bar{\mathcal{L}}_{\text{InfoPro}}$, as shown by Proposition 2, (the proof is given in Appendix C). We also empirically check its values in Appendix H. Proposition 2 provides a useful tool to examine the discrepancy between $\mathcal{L}_{\text{InfoPro}}$ and its upper bound.

Proposition 2. *Given that* $r^* = \operatorname{argmax}_{r,\ I(r,\boldsymbol{x})>0,\ I(r,y)=0} I(r, \boldsymbol{h})$ *and that* y *is a deterministic function with respect to* $\boldsymbol{x}$*, the gap* $\epsilon = \bar{\mathcal{L}}_{\text{InfoPro}} - \mathcal{L}_{\text{InfoPro}}$ *is upper bounded by*

$$\epsilon \leq \lambda_2 \left[I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y) \right]. \tag{3}$$

3.3 Mutual Information Estimation

In the following, we describe the specific techniques we use to obtain the mutual information $I(\boldsymbol{h}, \boldsymbol{x})$ and $I(\boldsymbol{h}, y)$ in $\bar{\mathcal{L}}_{\text{InfoPro}}$. Both of them are estimated using small auxiliary networks. However, we note that the involved additional computational costs are minimal or even negligible (see Tables 3, 4).

Estimating $I(\boldsymbol{h}, \boldsymbol{x})$. It has been widely known that the expected reconstruction error $\mathcal{R}(\boldsymbol{x}|\boldsymbol{h})$ follows $I(\boldsymbol{h}, \boldsymbol{x}) = H(\boldsymbol{x}) - H(\boldsymbol{x}|\boldsymbol{h}) \geq H(\boldsymbol{x}) - \mathcal{R}(\boldsymbol{x}|\boldsymbol{h})$, where $H(\boldsymbol{x})$ denotes the marginal entropy of $\boldsymbol{x}$, as a constant (Vincent et al., 2008; Rifai et al., 2012; Kingma & Welling, 2013; Makhzani et al., 2015; Hjelm et al., 2019). Therefore, we estimate $I(\boldsymbol{h}, \boldsymbol{x})$ by training a decoder parameterized by $\boldsymbol{\omega}$ to obtain the minimal reconstruction loss, namely $I(\boldsymbol{h}, \boldsymbol{x}) \approx \operatorname{argmax}_{\boldsymbol{\omega}} [H(\boldsymbol{x}) - \mathcal{R}_{\boldsymbol{w}}(\boldsymbol{x}|\boldsymbol{h})]$. In practice, we use the binary cross-entropy reconstruction loss for $\mathcal{R}_{\boldsymbol{w}}(\boldsymbol{x}|\boldsymbol{h})$.

Estimating $I(\boldsymbol{h}, y)$. We propose two ways to estimate $I(\boldsymbol{h}, y)$. Since $I(\boldsymbol{h}, y) = H(y) - H(y|\boldsymbol{h}) = H(y) - \mathbb{E}_{(\boldsymbol{h},y)}[-\log p(y|\boldsymbol{h})]$, a straightforward approach is to train an auxiliary classifier $q_{\boldsymbol{\psi}}(y|\boldsymbol{h})$ with parameters $\boldsymbol{\psi}$ to approximate $p(y|\boldsymbol{h})$, such that we have $I(\boldsymbol{h}, y) \approx \operatorname{argmax}_{\boldsymbol{\psi}} \{H(y) - \mathbb{E}_{\boldsymbol{h}}[\sum_y -p(y|\boldsymbol{h}) \log q_{\boldsymbol{\psi}}(y|\boldsymbol{h})]\}$. Hence, $q_{\boldsymbol{\psi}}(y|\boldsymbol{h})$ can be trained in a normal classification fashion.

In addition, motivated by recent advances in contrastive representation learning (Chen et al., 2020; Khosla et al., 2020; He et al., 2020), we formulate a contrastive style loss function $\mathcal{L}_{\text{contrast}}$, and prove in Appendix D that minimizing $\mathcal{L}_{\text{contrast}}$ is equivalent to maximizing a lower bound of $I(\boldsymbol{h}, y)$. Empirical results indicate that adopting $\mathcal{L}_{\text{contrast}}$ leads to better performance if a large batch size is available. In specific, considering a mini-batch of intermediate features $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\}$ corresponding to the labels $\{y_1, \ldots, y_N\}$, $\mathcal{L}_{\text{contrast}}$ is given by:

$$\mathcal{L}_{\text{contrast}} = \frac{1}{\sum_{i \neq j} \mathbb{1}_{y_i = y_j}} \sum_{i \neq j} \left[\mathbb{1}_{y_i = y_j} \log \frac{\exp(\boldsymbol{z}_i^\top \boldsymbol{z}_j / \tau)}{\sum_{k=1}^{N} \mathbb{1}_{i \neq k} \exp(\boldsymbol{z}_i^\top \boldsymbol{z}_k / \tau)} \right], \quad \boldsymbol{z}_i = f_{\boldsymbol{\phi}}(\boldsymbol{h}_i). \tag{4}$$

Herein, $\mathbb{1}_{\mathbf{A}} \in \{0, 1\}$ returns 1 only when $\mathbf{A}$ is true, $\tau > 0$ is a pre-defined hyper-parameter, temperature, and $f_{\boldsymbol{\phi}}$ is a projection head parameterized by $\boldsymbol{\phi}$ that maps the feature $\boldsymbol{h}_i$ to a representation vector $\boldsymbol{z}_i$ (this design follows Chen et al. (2020); Khosla et al. (2020)).

Implementation details. We defer the details on the network architecture of $\boldsymbol{w}$, $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ to Appendix E. Briefly, $\boldsymbol{w}$ is a two layer decoder with up-sampled inputs (if not otherwise noted), with $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ sharing the same architecture consisting of a single convolutional layer followed by two fully-connected layers. On ImageNet and Cityscapes, we use relatively larger auxiliary nets, but they are small compared with the primary network. Empirically, we find that these simple architectures are capable of achieving competitive performance consistently. Moreover, in implementation, we train $\boldsymbol{w}$, $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ collaboratively with the main network. Formally, let $\boldsymbol{\theta}$ denote the parameters of the local module to be trained, and then the final optimization objective of our method is

$$\min_{\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{\psi}} \lambda_1 \mathcal{R}_{\boldsymbol{w}}(\boldsymbol{x}|\boldsymbol{h}) + \lambda_2 \mathbb{E}_{\boldsymbol{h}}[\sum_y -p(y|\boldsymbol{h}) \log q_{\boldsymbol{\psi}}(y|\boldsymbol{h})] \ \text{ or } \ \min_{\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{\phi}} \lambda_1 \mathcal{R}_{\boldsymbol{w}}(\boldsymbol{x}|\boldsymbol{h}) + \lambda_2 \mathcal{L}_{\text{contrast}},$$

which corresponds to using the cross-entropy and contrast loss to estimate $I(\boldsymbol{h}, y)$, respectively. Such an approximation is acceptable as we do not need to acquire the exact approximation of mutual information, and empirically it performs well in various experimental settings.

4 Experiments

Setups. Our experiments are based on five widely used datasets (i.e., CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), STL-10 (Coates et al., 2011), ImageNet (Deng et al., 2009)

and Cityscapes (Cordts et al., 2016)) and two popular network architectures (i.e., ResNet (He et al., 2016) and DenseNet (Huang et al., 2019)) with varying depth. We split each network into K local modules with the same (or approximately the same) number of layers, where the first $K-1$ modules are trained using $\bar{\mathcal{L}}_{\text{InfoPro}}$, and the last module is trained using the standard E2E loss, as aforementioned. Due to spatial limitation, details on data pre-processing, training configurations and local module splitting are deferred to Appendix F. The hyper-parameters λ_1 and λ_2 are selected from $\{0, 0.1, 0.5, 1, 2, 5, 10, 20\}$. Notably, to avoid involving too many tunable hyper-parameters when K is large (e.g., $K=16$), we assume that λ_1 and λ_2 change linearly from 1st to $(K-1)^{\text{th}}$ local module, and thus we merely tune λ_1 and λ_2 for these two modules. We always use $\tau=0.07$ in $\mathcal{L}_{\text{contrast}}$.

Two training modes are considered: (1) *simultaneous training*, where the back-propagation process of all local modules is sequentially triggered with every mini-batch of training data; and (2) *asynchronous training*, where local modules are isolatedly learned given cached outputs from completely trained earlier modules. Both modes enjoy high memory efficiency since only the activations within a single module require to be stored at a time. The second mode removes the dependence of local modules on their predecessors, enabling the fully decoupled training of network components. In addition, it can be easily extended to training different local modules in parallel by dynamically caching the outputs of the earlier module. The experiments using asynchronous training are referred to as "Asy-InfoPro", while all other results are based on simultaneous training.

4.1 Main Results

Comparisons with other local learning methods. We first compare the proposed InfoPro method with three recently proposed algorithms, decoupled greedy learning (Belilovsky et al., 2020) (DGL), BoostResNet (Huang et al., 2018) and deep incremental boosting (Mosca & Magoulas, 2017) (DIB) in Figure 3. Our method yields the lowest test errors with all values of K. Notably, DGL can be viewed as a special case of InfoPro where $\lambda_1=0$. Hence, we use the same architecture of auxiliary networks as us in DGL for fair comparison. In addition, we present the estimates of mutual information between intermediate features and labels in the *right* plot of Figure 3. One can observe that DGL suffers from a severe collapse of task-relevant information at early modules, since it optimizes local modules greedily for merely short-term performance. By contrast, our method effectively alleviates this problem, retaining a larger amount of task-relevant information within intermediate features.

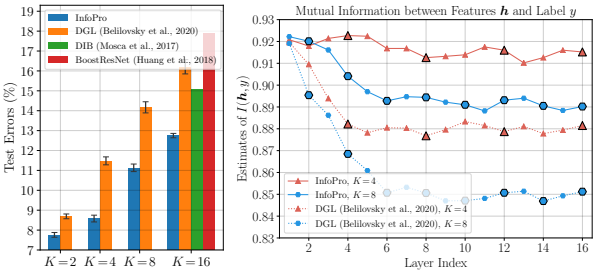

Figure 3: Comparisons of InfoPro and state-of-the-art local learning methods in terms of the test errors at the final layer (*left*) and the task-relevant information capture by intermediate features, $I(\boldsymbol{h}, y)$ (*right*). Results of ResNet-32 on CIFAR-10 are reported. We use the contrastive loss in $\bar{\mathcal{L}}_{\text{InfoPro}}$.

Results on various image classification benchmarks are presented in Table 2. It can be observed that InfoPro outperforms greedy SL by large margins consistently across different networks, especially when K is large. For example, on CIFAR-10, InfoPro achieves a test error of 12.75% with $K=16$, surpassing greedy SL by 11.84%. For ResNet-110, we note that our method performs on par with E2E training with $K=2$, while degrading the performance by up to 0.8% with $K=4$.

In addition, given that our method introduces auxiliary networks, we increase the depth of networks with greedy SL to match the computational cost of ours, named as greedy SL^{+}. However, this only slightly ameliorates the performance since the problem of information collapse still exists. Another interesting phenomenon is that InfoPro (Contrast) achieves better performance than InfoPro (Softmax) on CIFAR-10 and SVHN, yet fail to do so on STL-10. We attribute this to the larger batch size we use on the former two datasets, because previous observations have revealed that using larger mini-batches is beneficial for contrastive loss (Chen et al., 2020; He et al., 2020; Khosla et al., 2020).

Asynchronous and parallel training. The results of asynchronous training are presented in Table 2 as "Asy-InfoPro", and it appears to slightly hurt the performance. This might be ascribed to lacking regularizing effects from the noisy outputs of earlier modules during training (Löwe et al., 2019). However, it is still considerably better than greedy SL and approaching the E2E baseline. We also preliminarily test training two local modules in parallel on 2 GPUs when $K=2$, using the same experimental protocols as Huo et al. (2018) (train ResNet-110 with a batch size of 128 on CIFAR-10) and their public codes. Our method gives a $1.5\times$ speedup over the standard parallel paradigm of E2E

Table 2: Performance of different networks with varying numbers of local modules. The averaged test errors and standard deviations of 5 independent trials are reported. InfoPro (Softmax/Contrast) refers to two approaches to estimating $I(\boldsymbol{h}, y)$, The results of Asy-InfoPro is obtain by asynchronous training, while others are based on simultaneous training. Greedy SL$^+$ adopts deeper networks to have the same computational costs as InfoPro.

Dataset	Network	Method	$K = 2$	$K = 4$	$K = 8$	$K = 16$
CIFAR-10	ResNet-32 (E2E: 7.37 $\pm$ 0.10%)	Greedy SL	10.30 $\pm$ 0.20%	16.07 $\pm$ 0.46%	21.19 $\pm$ 0.52%	24.59 $\pm$ 0.83%
		InfoPro (Softmax)	8.13 $\pm$ 0.23%	8.64 $\pm$ 0.25%	11.40 $\pm$ 0.18%	14.23 $\pm$ 0.42%
		InfoPro (Contrast)	**7.76 $\pm$ 0.12%**	**8.58 $\pm$ 0.17%**	**11.13 $\pm$ 0.19%**	**12.75 $\pm$ 0.11%**
	ResNet-110 (E2E: 6.50 $\pm$ 0.34%)	Greedy SL	8.21 $\pm$ 0.24%	13.16 $\pm$ 0.28%	15.61 $\pm$ 0.57%	18.92 $\pm$ 1.27%
		Greedy SL$^+$	8.00 $\pm$ 0.11%	12.47 $\pm$ 0.17%	14.58 $\pm$ 0.36%	17.35 $\pm$ 0.31%
		InfoPro (Softmax)	7.01 $\pm$ 0.34%	7.96 $\pm$ 0.06%	9.40 $\pm$ 0.27%	10.78 $\pm$ 0.28%
		Asy-InfoPro (Contrast)	7.34 $\pm$ 0.11%	8.39 $\pm$ 0.15%	–	–
		InfoPro (Contrast)	**6.42 $\pm$ 0.08%**	**7.30 $\pm$ 0.14%**	**8.93 $\pm$ 0.40%**	**9.90 $\pm$ 0.19%**
	DenseNet-BC-100-12 (E2E: 4.61 $\pm$ 0.08%)	Greedy SL	5.10 $\pm$ 0.05%	6.07 $\pm$ 0.21%	8.21 $\pm$ 0.31%	10.41 $\pm$ 0.42%
		InfoPro (Softmax)	4.79 $\pm$ 0.07%	5.69 $\pm$ 0.21%	6.44 $\pm$ 0.11%	7.47 $\pm$ 0.21%
		InfoPro (Contrast)	**4.74 $\pm$ 0.04%**	**5.24 $\pm$ 0.25%**	**5.86 $\pm$ 0.18%**	**6.92 $\pm$ 0.16%**
SVHN	ResNet-110 (E2E: 3.07 $\pm$ 0.23%)	Greedy SL	3.71 $\pm$ 0.16%	5.39 $\pm$ 0.22%	5.75 $\pm$ 0.10%	6.37 $\pm$ 0.42%
		InfoPro (Softmax)	3.41 $\pm$ 0.08%	3.72 $\pm$ 0.03%	4.67 $\pm$ 0.07%	5.14 $\pm$ 0.08%
		InfoPro (Contrast)	**3.15 $\pm$ 0.03%**	**3.28 $\pm$ 0.11%**	**3.62 $\pm$ 0.11%**	**3.91 $\pm$ 0.16%**
STL-10	ResNet-110 (E2E: 22.27 $\pm$ 1.61%)	Greedy SL	25.56 $\pm$ 1.37%	27.97 $\pm$ 0.75%	29.07 $\pm$ 0.76%	30.38 $\pm$ 0.39%
		InfoPro (Softmax)	21.02 $\pm$ 0.51%	**21.28 $\pm$ 0.27%**	**23.60 $\pm$ 0.49%**	**26.05 $\pm$ 0.71%**
		InfoPro (Contrast)	**20.99 $\pm$ 0.64%**	22.73 $\pm$ 0.40%	25.15 $\pm$ 0.52%	26.27 $\pm$ 0.48%

Table 3: Trade-off between GPU memory requirements during training and test errors. Results of training ResNet-110 on a single Nvidia Titan Xp GPU are reported.

Methods	CIFAR-10 (batch size = 1024)			STL-10 (batch size = 128)		
	Test Error	Memory Cost	Computational Overhead	Test Error	Memory Cost	Computational Overhead
E2E Training	6.50 $\pm$ 0.34%	9.40 GB	–	22.27 $\pm$ 1.61%	10.77 GB	–
InfoPro*, $K = 2$	**6.41 $\pm$ 0.13%**	5.38 GB (↓42.8%)	1.3%	**20.95 $\pm$ 0.57%**	6.15 GB (↓42.9%)	1.3%
InfoPro*, $K = 3$	6.74 $\pm$ 0.12%	4.22 GB (↓55.1%)	3.3%	21.00 $\pm$ 0.52%	4.96 GB (↓53.9%)	3.3%
InfoPro*, $K = 4$	6.93 $\pm$ 0.20%	**3.52 GB** (↓62.6%)	5.9%	21.22 $\pm$ 0.72%	**4.08 GB** (↓62.1%)	5.9%

Table 4: Performance of InfoPro* on ImageNet. The results are obtained with 8 Tesla V100 GPUs.

Models	Methods	Batch Size	Top-1 Error	Top-5 error	Memory Cost (per GPU)	Computational Overhead
ResNet-101	E2E Training	1024	22.03%	5.93%	19.71 GB	–
	InfoPro*, $K = 2$	1024	**21.85%**	**5.89%**	**12.06 GB** (↓38.8%)	5.7%
ResNet-152	E2E Training	1024	21.60%	5.92%	26.29 GB	–
	InfoPro*, $K = 2$	1024	**21.45%**	**5.84%**	**15.53 GB** (↓40.9%)	3.9%
ResNeXt-101, 32×8d	E2E Training	512	20.64%	5.40%	19.22 GB	–
	InfoPro*, $K = 2$	512	**20.35%**	**5.28%**	**11.55 GB** (↓39.9%)	2.7%

Table 5: Results of semantic segmentation on Cityscapes with single-scale inference. 4 Nvidia Titan Xp GPUs are used for training. We present the results reported by the original paper in the "original" row.

Models	Backbone	Methods	Batch Size	mIoU	Memory Cost (per GPU)	Computational Overhead
DeepLab-V3 (Chen et al., 2017b)	ResNet-101	E2E Training (original)	8	77.8%	11.76 GB	–
		E2E Training (ours)	8	78.3%	11.76 GB	–
		InfoPro*, $K = 2$	8	78.1%	**7.56 GB** (↓35.7%)	12.8%
		InfoPro*, $K = 2$	**12** (↑50%)	**78.4%**	10.91 GB (↓7.2%)	12.8%

training (the DataParallel toolkit in Pytorch). Note that parallel training has the same performance as simultaneous training since their training processes are identical except for the parallelism.

Reducing GPUs memory requirements. We note that deep networks are usually designed with unbalanced memory consumption at each stage. For example, with the down-sampling operations in ResNets, the size of feature maps is reduced to 1/4, but the numbers of channels merely double, halving the memory usage. Therefore, we re-split local modules to make them share the same memory footprint during training, which is denoted by InfoPro*. We present the trade-off between memory consumption and test errors in Table 3, where the contrastive and softmax loss are used in InfoPro* on CIFAR-10 and STL-10 respectively. One can observe that our method significantly improves the memory efficiency of CNNs. For instance, on STL-10, InfoPro* ($K = 4$) outperforms the E2E baseline by 1.05% with 37.9% of the GPUs memory requirements.

Results on ImageNet are reported in Table 4. The softmax loss is used in InfoPro* since the batch size is relatively small. The proposed method reduces the memory cost by 40%, and achieves slightly better performance. Notably, our method enables training these large networks using 16 GB GPUs.

Results of semantic segmentation on Cityscapes are presented in Table 5. Here we use an ASPP head (Chen et al., 2017a) as $\boldsymbol{\psi}$ and the same decoder $\boldsymbol{w}$ as on ImageNet. Our method boosts the performance of the DeepLab-V3 (Chen et al., 2017b) network and allows training the model with 50% larger batch sizes ($2 \to 3$ per GPU) under the same memory constraints. This contributes to more accurate statistics for batch normalization, which is a practical requirement for tasks with high resolution inputs.

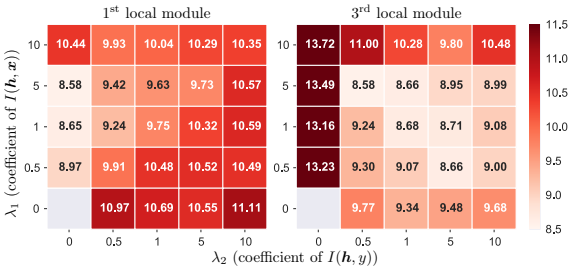

Figure 4: Sensitivity tests. The CIFAR-10 test errors of ResNet-32 trained using InfoPro ($K = 4$) are reported. We vary λ_1 and λ_2 for 1st and 3rd local modules respectively, with all other modules unchanged. We do not consider $\lambda_1 = \lambda_2 = 0$, where we obviously have $\bar{\mathcal{L}}_{\text{InfoPro}} = \mathcal{L}_{\text{InfoPro}} \equiv 0$.

4.2 Hyper-parameter Sensitivity and Ablation Study

To study how λ_1 and λ_2 affect the performance, we change them for the 1st and 3rd local modules of a ResNet-32 trained using InfoPro (Contrast), $K = 4$, with the results shown in Figure 4. We find that the earlier module benefits from small λ_2 to propagate more information forward, while larger λ_2 helps the later module to boost the final accuracy. This is compatible with previous works showing that removing earlier layers in ResNets has a minimal impact on performance (Veit et al., 2016). For ablation, we also test removing the decoder $\boldsymbol{w}$ or replacing the contrastive head $\boldsymbol{\phi}$ by the linear classifier used in greedy SL, as shown in Table 6.

Table 6: Ablation studies. Test errors of ResNet-32 on CIFAR-10 are reported.

$\boldsymbol{w}$	$\boldsymbol{\phi}$	$K = 2$	$K = 8$
		$10.30 \pm 0.20\%$	$21.19 \pm 0.52\%$
	✓	$8.90 \pm 0.17\%$	$15.82 \pm 0.34\%$
✓		$8.49 \pm 0.16\%$	$14.13 \pm 0.22\%$
✓	✓	$\mathbf{7.76 \pm 0.12\%}$	$\mathbf{11.13 \pm 0.19\%}$

5 Related Work

Greedy training of deep networks is initially proposed to learn unsupervised deep generative models, or to obtain a proper initialization for E2E supervised training (Hinton et al., 2006; Bengio et al., 2007). However, later advances reveal that this initialization is indeed unnecessary (He et al., 2016; Huang et al., 2019). Some earlier contributions (Kulkarni & Karande, 2017; Malach & Shalev-Shwartz, 2018; Marquez et al., 2018; Huang et al., 2018) also attempt to learn supervised models greedily, but they mainly focus on the theoretical analysis. More recently, several works have pointed out the inefficiencies of back-propagation and revisited this problem in deep learning (Nøkland & Eidnes, 2019; Belilovsky et al., 2019; 2020). These methods mostly train local modules with a greedy objective, and hence suffer from the information collapse issue we have discussed in this paper.

Alternatives of back-propagation have been widely studied in recent years. Some biologically-motivated algorithms including target propagation (Lee et al., 2015b; Bartunov et al., 2018) and feedback alignment (Lillicrap et al., 2014; Nøkland, 2016) avoid back-propagation by directly propagating backward optimal activations or error signals with auxiliary networks. Decoupled Neural Interfaces (DNI) (Jaderberg et al., 2017) learn auxiliary networks to produce synthetic gradients. Nevertheless, these methods differ from us fundamentally as they are explicitly or implicitly designed for optimizing a global objective, while we exclusively consider *local* objectives.

Information-theoretic analysis in deep learning has received increasingly more attention in the past few years. Shwartz-Ziv & Tishby (2017) and Saxe et al. (2019) study the information bottleneck (IB) principle (Tishby et al., 2000) to explain the training dynamics of deep networks. Achille & Soatto (2018) decompose the cross-entropy loss and propose a novel IB for weights. There are also efforts towards fulfilling efficient training with IB (Alemi et al., 2016). In the context of unsupervised learning, a variety of methods have been proposed based on mutual information maximization (Oord et al., 2018; Tian et al., 2020; Hjelm et al., 2019; Chen et al., 2020; He et al., 2020).

6 Conclusion

This work studied locally supervised deep learning from the information-theoretic perspective. We demonstrated that training local modules greedily results in collapsing task-relevant information at earlier layers, degrading the final performance. To address this issue, we proposed an information propagation (InfoPro) loss that encourages local modules to preserve more information about the input, while progressively discard task-irrelevant information. Extensive experiments validated that InfoPro significantly reduced GPUs memory footprints during training without sacrificing accuracy. It also enabled model parallelization in an asynchronous fashion. InfoPro may open new avenues for developing more efficient and biologically plausible deep learning algorithms.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *NeurIPS*, pp. 9368–9378, 2018.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *ICML*, pp. 583–593. PMLR, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *ICML*, 2020.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017a.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pp. 215–223, 2011.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pp. 3213–3223, 2016.

Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

Yang Dan and Mu-ming Poo. Spike timing-dependent plasticity of neural circuits. *Neuron*, 44(1): 23–30, 2004.

J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *ICML*, pp. 248–255, 2009.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. In *NeurIPS*, pp. 2214–2224, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9729–9738, 2020.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

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. In *ICLR*, 2019.

Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. In *ICML*, pp. 2058–2067, 2018.

Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S. Huang. Ccnet: Criss-cross attention for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.3007032.

Zhouyuan Huo, Bin Gu, Qian Yang, and Heng Huang. Decoupled parallel backpropagation with convergence guarantee. In *ICML*, 2018.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *ICML*, pp. 1627–1635. PMLR, 2017.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pp. 1097–1105, 2012.

Mandar Kulkarni and Shirish Karande. Layer-wise training of deep networks using kernel similarity. *arXiv preprint arXiv:1703.07115*, 2017.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *AISTATS*, pp. 562–570, 2015a.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pp. 498–515. Springer, 2015b.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.

Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *NeurIPS*, pp. 3039–3051, 2019.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Eran Malach and Shai Shalev-Shwartz. A provably correct algorithm for deep learning that actually works. *arXiv preprint arXiv:1803.09522*, 2018.

Enrique S Marquez, Jonathon S Hare, and Mahesan Niranjan. Deep cascade learning. *IEEE transactions on neural networks and learning systems*, 29(11):5475–5485, 2018.

Alan Mosca and George D Magoulas. Deep incremental boosting. *arXiv preprint arXiv:1708.03704*, 2017.

Hesham Mostafa, Vishwajith Ramesh, and Gert Cauwenberghs. Deep supervised learning using local errors. *Frontiers in neuroscience*, 12:608, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *NeurIPS*, pp. 1037–1045, 2016.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. *arXiv preprint arXiv:1901.06656*, 2019.

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

Salah Rifai, Yoshua Bengio, Aaron Courville, Pascal Vincent, and Mehdi Mirza. Disentangling factors of variation for facial expression recognition. In *ECCV*, pp. 808–822. Springer, 2012.

Tim Salimans and Yaroslav Bulatov. Gradient checkpointing, 2017.

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pp. 1–9, 2015.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, pp. 1195–1204, 2017.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NeurIPS*, pp. 550–558, 2016.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pp. 1096–1103, 2008.

Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. In *NeurIPS*, pp. 12635–12644, 2019.

Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *CVPR*, pp. 2369–2378, 2020.

APPENDIX

A A TOY EXAMPLE

To further validate the proposed information collapse hypothesis, we visualize the "information flow" within deep networks using a toy example. First, we establish a MNIST-STL10 dataset via placing MNIST digits on a certain position (randomly selected from 64 candidates) of a background image from STL-10. Then, three specific tasks can be defined on MNIST-STL10, namely classifying digits, backgrounds and positions of numbers. We refer to the labels of them as y_1, y_2 and y_3, respectively, as illustrated by Figure 5.

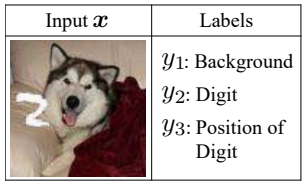

Figure 5: Illustration of the MNIST-STL10 dataset.

We train ResNet-32 networks for the three tasks with greedy SL ($K=4$) and end-to-end training. The estimates of mutual information $I(\boldsymbol{h}, y_1)$, $I(\boldsymbol{h}, y_2)$ and $I(\boldsymbol{h}, y_3)$ is shown in Figure 6, with the same estimating approach as Figure 2 (details in Appendix G). Note that when one label (take y_1 for example) is adopted for training, the information related to other labels ($I(\boldsymbol{h}, y_2)$ and $I(\boldsymbol{h}, y_3)$) is task-irrelevant. From the plots, one can clearly observe that end-to-end training retains all task-relevant information throughout the feed-forward process, while greedy SL usually yields less informative intermediate representations in terms of the task of interest. This phenomenon confirm the proposed information collapse hypothesis empirically. In addition, we postulate that the end-to-end learned early layers prevent collapsing task-relevant information by being allowed to keep larger amount of task-irrelevant information, which, however, may lead to the inferior classification performance of intermediate features, and thus cannot be achieved by greedy SL.

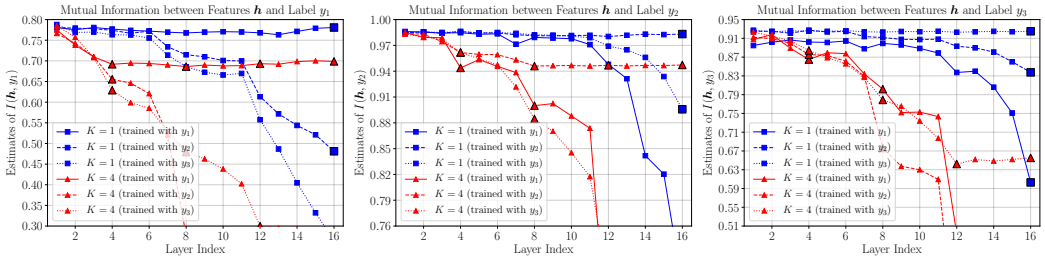

Figure 6: The estimates of mutual information between the intermediate features $\boldsymbol{h}$ and the three labels of MNIST-STL10 (see: Figure 5), i.e. y_1 (*left*, background), y_2 (*middle*, digit) and y_3 (*right*, position of digit). Models are trained using greedy SL ($K=1, 4$) supervised by one of the three labels, and the results are shown with respect to layer indices. "$K=1$" refers to end-to-end training.

B PROOF OF PROPOSITION 1

Proposition 1. *Suppose that the Markov chain $(y, r) \to \boldsymbol{x} \to \boldsymbol{h}$ holds. Then an upper bound of $\mathcal{L}_{\text{InfoPro}}$ is given by*

$$\mathcal{L}_{\text{InfoPro}} \leq -\lambda_1 I(\boldsymbol{h}, \boldsymbol{x}) - \lambda_2 I(\boldsymbol{h}, y) \triangleq \bar{\mathcal{L}}_{\text{InfoPro}}, \tag{5}$$

where $\lambda_1 = \alpha(1-\beta)$, $\lambda_2 = \alpha\beta$.

Proof. Note that, $\mathcal{L}_{\text{InfoPro}}$ is given by

$$\mathcal{L}_{\text{InfoPro}}(\boldsymbol{h}) = \alpha[-I(\boldsymbol{h}, \boldsymbol{x}) + \beta I(r^*, \boldsymbol{h})],\ \alpha, \beta > 0, \quad s.t.\ r^* = \underset{r,\ I(r,\boldsymbol{x})>0,\ I(r,y)=0}{\operatorname{argmax}} I(r, \boldsymbol{h}). \tag{6}$$

Due to the Markov chain $(y, r) \to \boldsymbol{x} \to \boldsymbol{h}$, we have $I(\boldsymbol{h}, (y, r^*)) \leq I(\boldsymbol{h}, \boldsymbol{x})$. Given that

$$\begin{aligned} I(\boldsymbol{h}, (y, r^*)) &= H(\boldsymbol{h}) - H(\boldsymbol{h}|y, r^*) \\ &= [H(\boldsymbol{h}) - H(\boldsymbol{h}|r^*)] + [H(\boldsymbol{h}|r^*) - H(\boldsymbol{h}|y, r^*)] \\ &= I(\boldsymbol{h}, r^*) + I(\boldsymbol{h}, y|r^*), \end{aligned} \tag{7}$$

we have

$$I(\boldsymbol{h}, r^*) \leq I(\boldsymbol{h}, \boldsymbol{x}) - I(\boldsymbol{h}, y|r^*). \tag{8}$$

By the definition of nuisance, we note that r^* and y are mutually independent, and thus we obtain

$$\begin{aligned} I(\boldsymbol{h}, y|r^*) &= H(y|r^*) - H(y|\boldsymbol{h}, r^*) \\ &= H(y) - H(y|\boldsymbol{h}, r^*) \\ &\geq H(y) - H(y|\boldsymbol{h}) = I(\boldsymbol{h}, y). \end{aligned} \tag{9}$$

Combining Eqs. (8) and (9), we have

$$I(\boldsymbol{h}, r^*) \leq I(\boldsymbol{h}, \boldsymbol{x}) - I(\boldsymbol{h}, y). \tag{10}$$

Finally, Proposition 1 is proved by combining Eq. (6) and Inequality (9). □

C Proof of proposition 2

We first introduce a Lemma proven by Achille & Soatto (2018).

Lemma 1. *Given a joint distribution $p(\boldsymbol{x}, y)$, where y is a discrete random variable, we can always find a random variable r independent of y such that $\boldsymbol{x} = f(y, r)$, for some deterministic function f.*

Proposition 2. *Given that $r^* = \operatorname{argmax}_{r,\ I(r,\boldsymbol{x})>0,\ I(r,y)=0} I(r, \boldsymbol{h})$ and that y is a deterministic function with respect to $\boldsymbol{x}$, the gap $\epsilon = \bar{\mathcal{L}}_{\text{InfoPro}} - \mathcal{L}_{\text{InfoPro}}$ is upper bounded by*

$$\epsilon \leq \lambda_2 \left[I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y) \right]. \tag{11}$$

Proof. Let $\tilde{r}$ be the random variable in Lemma 1, and then, since we can find a deterministic function that maps $\boldsymbol{x}$ to y, we have

$$I(\boldsymbol{h}, \boldsymbol{x}) = I(\boldsymbol{h}, (y, \tilde{r})) = I(\boldsymbol{h}, \tilde{r}) + I(\boldsymbol{h}, y|\tilde{r}). \tag{12}$$

Assume $\zeta = [I(\boldsymbol{h}, \boldsymbol{x}) - I(\boldsymbol{h}, y)] - I(\boldsymbol{h}, r^*)$, in terms that $r^* = \operatorname{argmax}_{r,\ I(r,\boldsymbol{x})>0,\ I(r,y)=0} I(r, \boldsymbol{h})$, we obtain

$$\begin{aligned} \zeta &\leq [I(\boldsymbol{h}, \boldsymbol{x}) - I(\boldsymbol{h}, y)] - I(\boldsymbol{h}, \tilde{r}) \\ &= I(\boldsymbol{h}, \tilde{r}) + I(\boldsymbol{h}, y|\tilde{r}) - I(\boldsymbol{h}, y) - I(\boldsymbol{h}, \tilde{r}) \\ &= I(\boldsymbol{h}, y|\tilde{r}) - I(\boldsymbol{h}, y) \\ &= I(\boldsymbol{h}, y|\tilde{r}) - I(\boldsymbol{x}, y) + I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y) \\ &= H(y|\tilde{r}) - H(y|\tilde{r}, \boldsymbol{h}) - H(y) + H(y|\boldsymbol{x}) + I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y). \end{aligned} \tag{13}$$

Since y and $\tilde{r}$ are mutually independent, namely $H(y|\tilde{r}) = H(y)$, we have

$$\begin{aligned} \zeta &\leq H(y) - H(y|\tilde{r}, \boldsymbol{h}) - H(y) + H(y|\boldsymbol{x}) + I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y) \\ &= H(y|\boldsymbol{x}) - H(y|\tilde{r}, \boldsymbol{h}) + I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y) \\ &\leq H(y|\boldsymbol{x}) + I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y). \end{aligned} \tag{14}$$

When considering y as a deterministic function with regards to $\boldsymbol{x}$, we obtain $H(y|\boldsymbol{x}) = 0$, and therefore

$$\zeta \leq I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y). \tag{15}$$

Given that $\epsilon = \bar{\mathcal{L}}_{\text{InfoPro}} - \mathcal{L}_{\text{InfoPro}} = \lambda_2 \zeta$, we have

$$\epsilon \leq \lambda_2 \left[I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y) \right], \tag{16}$$

for which we have proven Proposition 2. □

D Why minimizing the contrastive loss maximizes the lower bound of task-relevant information?

In this section, we show that minimizing the proposed contrastive loss, namely

$$\mathcal{L}_{\text{contrast}} = \frac{1}{\sum_{i \neq j} \mathbb{1}_{y_i = y_j}} \sum_{i \neq j} \left[\mathbb{1}_{y_i = y_j} \log \frac{\exp(\boldsymbol{z}_i^\top \boldsymbol{z}_j / \tau)}{\sum_{k=1}^{N} \mathbb{1}_{i \neq k} \exp(\boldsymbol{z}_i^\top \boldsymbol{z}_k / \tau)} \right], \quad \boldsymbol{z}_i = f_{\boldsymbol{\phi}}(\boldsymbol{h}_i), \tag{17}$$

actually maximizes an lower bound of task-relevant information $I(\boldsymbol{h}, y)$. We start by considering a simplified but equivalent situation. Suppose that we have a query sample $\boldsymbol{z}^+$, together with a set

$\boldsymbol{X} = \{\boldsymbol{z}_1, \dots, \boldsymbol{z}_N\}$ consisting of N samples with one positive sample $\boldsymbol{z}^{\mathrm{p}}$ from the same class as $\boldsymbol{z}^{+}$ definitely, and other negative samples are randomly sampled, namely $\boldsymbol{X} = \{\boldsymbol{z}^{+}\} \cup \boldsymbol{X}_{\text{neg}}$. Then, the expectation of $\mathcal{L}_{\text{contrast}}$ can be written as

$$\mathbb{E}[\mathcal{L}_{\text{contrast}}] = \mathbb{E}_{\boldsymbol{z}^{+}, \boldsymbol{X}} \left[-\log \frac{\exp({\boldsymbol{z}^{+}}^{\top} \boldsymbol{z}^{\mathrm{p}}/\tau)}{\sum_{i=1}^{N} \exp({\boldsymbol{z}^{+}}^{\top} \boldsymbol{z}_i/\tau)} \right]. \tag{18}$$

Eq. (18) can be viewed as a categorical cross-entropy loss of classifying the positive sample $\boldsymbol{z}^{\mathrm{p}}$ correctly. Hence, we define the optimal probability of this classification problem as $P^{\text{pos}}(\boldsymbol{z}_i|\boldsymbol{X})$, which denotes the true probability of $\boldsymbol{z}_i$ being the positive sample. Assuming that the label of $\boldsymbol{z}^{\mathrm{p}}$ is y, the positive and negative samples can be viewed as being sampled from the true distributions $p(\boldsymbol{z}|y)$ and $p(\boldsymbol{z})$, respectively. As a consequence, $P^{\text{pos}}(\boldsymbol{z}_i|\boldsymbol{X})$ can be derived as

$$P^{\text{pos}}(\boldsymbol{z}_i|\boldsymbol{X}) = \frac{p(\boldsymbol{z}_i|y) \prod_{l \neq i} p(\boldsymbol{z}_l)}{\sum_{j=1}^{N} p(\boldsymbol{z}_j|y) \prod_{l \neq j} p(\boldsymbol{z}_l)} = \frac{\frac{p(\boldsymbol{z}_i|y)}{p(\boldsymbol{z}_i)}}{\sum_{j=1}^{N} \frac{p(\boldsymbol{z}_j|y)}{p(\boldsymbol{z}_j)}}, \tag{19}$$

which indicates that an optimal value for $\exp({\boldsymbol{z}^{+}}^{\top} \boldsymbol{z}^{\mathrm{p}}/\tau)$ is $\frac{p(\boldsymbol{z}^{\mathrm{p}}|y)}{p(\boldsymbol{z}^{\mathrm{p}})}$. Therefore, by assuming that $\boldsymbol{z}^{+}$ is uniformly sampled from all classes, we have

$$\begin{aligned}
\mathbb{E}[\mathcal{L}_{\text{contrast}}] \geq \mathbb{E}[\mathcal{L}_{\text{contrast}}^{\text{optimal}}] &= \mathbb{E}_{y, \boldsymbol{X}} \left[-\log \frac{\frac{p(\boldsymbol{z}^{\mathrm{p}}|y)}{p(\boldsymbol{z}^{\mathrm{p}})}}{\sum_{j=1}^{N} \frac{p(\boldsymbol{z}_j|y)}{p(\boldsymbol{z}_j)}} \right] && (20) \\
&= \mathbb{E}_{y, \boldsymbol{X}} \left[-\log \frac{\frac{p(\boldsymbol{z}^{\mathrm{p}}|y)}{p(\boldsymbol{z}^{\mathrm{p}})}}{\frac{p(\boldsymbol{z}^{\mathrm{p}}|y)}{p(\boldsymbol{z}^{\mathrm{p}})} + \sum_{\boldsymbol{z}_j \in \boldsymbol{X}_{\text{neg}}} \frac{p(\boldsymbol{z}_j|y)}{p(\boldsymbol{z}_j)}} \right] && (21) \\
&= \mathbb{E}_{y, \boldsymbol{X}} \left\{ \log \left[1 + \frac{p(\boldsymbol{z}^{\mathrm{p}})}{p(\boldsymbol{z}^{\mathrm{p}}|y)} \sum_{\boldsymbol{z}_j \in \boldsymbol{X}_{\text{neg}}} \frac{p(\boldsymbol{z}_j|y)}{p(\boldsymbol{z}_j)} \right] \right\} && (22) \\
&\approx \mathbb{E}_{y, \boldsymbol{X}} \left\{ \log \left[1 + \frac{p(\boldsymbol{z}^{\mathrm{p}})}{p(\boldsymbol{z}^{\mathrm{p}}|y)} (N-1) \mathbb{E}_{\boldsymbol{z}_j \sim p(\boldsymbol{z}_j)} \frac{p(\boldsymbol{z}_j|y)}{p(\boldsymbol{z}_j)} \right] \right\} && (23) \\
&= \mathbb{E}_{y, \boldsymbol{z}^{\mathrm{p}}} \left\{ \log \left[1 + \frac{p(\boldsymbol{z}^{\mathrm{p}})}{p(\boldsymbol{z}^{\mathrm{p}}|y)} (N-1) \right] \right\} && (24) \\
&\geq \mathbb{E}_{y, \boldsymbol{z}^{\mathrm{p}}} \left\{ \log \left[\frac{p(\boldsymbol{z}^{\mathrm{p}})}{p(\boldsymbol{z}^{\mathrm{p}}|y)} (N-1) \right] \right\} && (25) \\
&= -I(\boldsymbol{z}^{\mathrm{p}}, y) + \log(N-1) \geq -I(\boldsymbol{h}, y) + \log(N-1). && (26)
\end{aligned}$$

In the above, Inequality (23) follows from Oord et al. (2018), which quickly becomes more accurate when N increases. Inequality (26) follows from the data processing inequality Shwartz-Ziv & Tishby (2017). Finally, we have $\mathbb{E}[\mathcal{L}_{\text{contrast}}] \geq \log(N-1) - I(\boldsymbol{h}, y)$, and thus minimizing $\mathcal{L}_{\text{contrast}}$ under the stochastic gradient descent framework maximizes an lower bound of $I(\boldsymbol{h}, y)$.

E ARCHITECTURE OF AUXILIARY NETWORKS

Here, we introduce the network architecture of $\boldsymbol{w}$, $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ we use in our experiments. Note that, $\boldsymbol{w}$ is a decoder that aims to reconstruct the input images from deep features, while $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ share the same architecture except for the last layer. The architectures used on CIFAR, SVHN and STL-10 are shown in Table 7 and Table 8. Architectures on ImageNet are shown in Table 9 and Table 10. Note that, for the semantic segmentation experiments on Cityscapes, we use an ASPP head (Chen et al., 2017a) as $\boldsymbol{\psi}$ and the same decoder $\boldsymbol{w}$ as on ImageNet (except for the size of feature maps).

F DETAILS OF EXPERIMENTS

Datasets. (1) The CIFAR-10 (Krizhevsky et al., 2009) dataset consists of 60,000 32x32 colored images of 10 classes, 50,000 for training and 10,000 for test. We normalize the images with channel

Table 7: Architecture of the decoder $\boldsymbol{w}$ on CIFAR, SVHN and STL-10.

Input: 32×32 / 16×16 / 8×8 feature maps (96×96 / 48×48 / 24×24 on STL-10)
Bilinear Interpolation to 32×32 (96×96 on STL-10)
3×3 conv., stride=1, padding=1, output channels=12, BatchNorm+ReLU
3×3 conv., stride=1, padding=1, output channels=3, Sigmoid

Table 8: Architecture of $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ on CIFAR, SVHN and STL-10.

Input: 32×32 / 16×16 / 8×8 feature maps (96×96 / 48×48 / 24×24 on STL-10)
32×32 (96×96) input features: 3×3 conv., stride=2, padding=1, output channels=32, BatchNorm+ReLU 16×16 (48×48) input features: 3×3 conv., stride=2, padding=1, output channels=64, BatchNorm+ReLU 8×8 (24×24) input features: 3×3 conv., stride=1, padding=1, output channels=64, BatchNorm+ReLU
Global average pooling
Fully connected 32 / 64→128, ReLU
Fully connected 128→10 for $\boldsymbol{\psi}$ or 128→128 for $\boldsymbol{\phi}$

Table 9: Architecture of the decoder $\boldsymbol{w}$ on ImageNet.

Input: 28×28 feature maps
1×1 conv., stride=1, padding=0, output channels=128, BatchNorm+ReLU
Bilinear Interpolation to 56×56
3×3 conv., stride=1, padding=1, output channels=32, BatchNorm+ReLU
Bilinear Interpolation to 112×112
3×3 conv., stride=1, padding=1, output channels=12, BatchNorm+ReLU
Bilinear Interpolation to 224×224
3×3 conv., stride=1, padding=1, output channels=3, Sigmoid

Table 10: Architecture of $\boldsymbol{\phi}$ on ImageNet

Input: 28×28 feature maps
1×1 conv., stride=1, padding=0, output channels=128, BatchNorm+ReLU
3×3 conv., stride=2, padding=1, output channels=256, BatchNorm+ReLU
3×3 conv., stride=2, padding=1, output channels=512, BatchNorm+ReLU
1×1 conv., stride=1, padding=0, output channels=2048, BatchNorm+ReLU
Global average pooling
Fully connected 2048→1000

means and standard deviations for pre-processing. Then data augmentation is performed by 4x4 random translation followed by random horizontal flip (He et al., 2016; Huang et al., 2019; Wang et al., 2019). (2) SVHN (Netzer et al., 2011) consists of 32x32 colored images of digits. 73,257 images for training and 26,032 images for test are provided. Following Tarvainen & Valpola (2017), we perform random 2x2 translation to augment the training set. (3) STL-10 (Coates et al., 2011) contains 5,000 training examples divided into 10 predefined folds with 1000 examples each, and 8,000 images for test. We use all the labeled images for training and test the performance on the provided test set. Data augmentation is performed by 4x4 random translation followed by random horizontal flip. (4) ImageNet is a 1,000-class dataset from ILSVRC2012 (Deng et al., 2009), with 1.2 million images for training and 50,000 images for validation. We adopt the same data augmentation and pre-processing configurations as Huang et al. (2019); Yang et al. (2020). (5) Cityscapes dataset Cordts et al. (2016) contains 5,000 1024x2048 pixel-level finely annotated images (2,975/500/1,525 for training, validation and testing) and 20,000 coarsely annotated images from 50 different cities. Each pixel of the image is categorized among 19 classes. Following Chen et al. (2017b), we conduct our experiments on the finely annotated dataset and report the performance on the validation set. The training images are augmented by randomly scaling (from 0.75 to 2.0) followed by randomly cropping high-resolution patches (769×769).

Networks and training hyper-parameters. Our experiments on CIAFR-10, SVHN and STL-10 are based on three popular networks, namely ResNet-32/110 (He et al., 2016) and DenseNet-BC-100-12

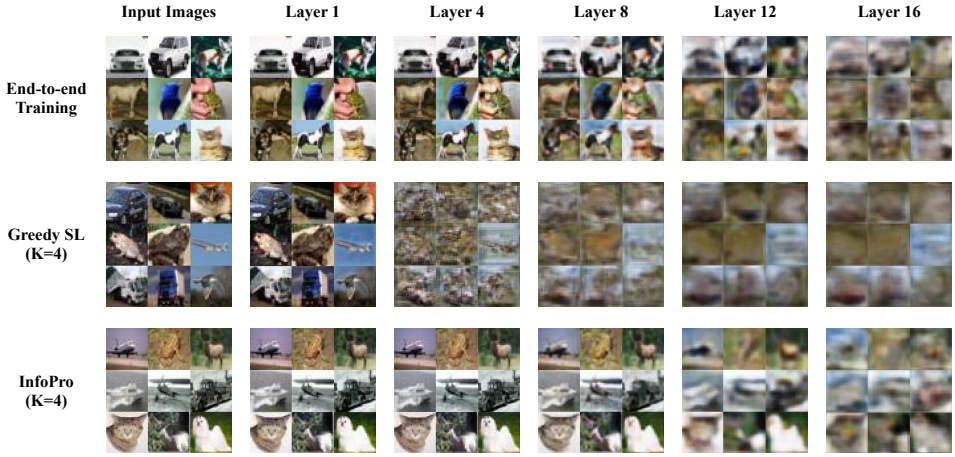

Figure 7: Visualization of the reconstruction results obtained from the decoder $\boldsymbol{\omega}$.

(Huang et al., 2019). The networks are trained using a SGD optimizer with a Nesterov momentum of 0.9 for 160 epochs. The L2 weight decay ratio is set to 1e-4. For ResNets, the batch size is set to 1024 and 128 for CIFAR-10/SVHN and STL-10, associated with an initial learning rate of 0.8 and 0.1, respectively. For DenseNets, we use a batch size of 256 and an initial learning rate of 0.2. The cosine learning rate annealing is adopted. Note that, the results of greedy supervised learning presented in Table 1 follows exactly the same experimental configurations stated here. On ImageNet, we train ResNet-101 and ResNet-152 with a batch size of 1024 and an initial learning rate of 0.4. For ResNeXt-101, 32×8d, we use a batch size of 512 and an initial learning rate of 0.2. The number of training epochs is set to 90. Other hyper-parameters are the same as CIAFR-10. For the DeepLab-V3 model used in semantic segmentation, we follow the training configurations of Huang et al. (2020), except for using a batch size of 12 and an initial learning rate of 0.015 in InfoPro*, $K = 2$.

Local module splitting. Since ResNets consist of a cascade of residual blocks, which naturally should not be further divided into smaller parts, we view each residual block as a minimal indivisible unit, or say, a basic layer (distinguished from a single convolutional layer). Particularly, the first convolutional layer of the network is individually viewed as a basic layer. As a consequence, ResNet-32 has 16 basic layers, and ResNet-110 has 55 basic layers. If ResNet-32 is split into $K = 8$ local modules, then each module will have 2 basic layers. In the cases where the number of basic layers is not divisible by K, we assign one less basic layer to earlier modules. For example, if ResNet-110 is split into $K = 4$ modules, the corresponding numbers of basic layers will be $\{13, 14, 14, 14\}$. If ResNet-110 is split into $K = 16$ modules, the corresponding numbers of basic layers will be $\{3\} \times 9$ modules $+ \{4\} \times 7$ modules. For DenseNet-BC, similarly, we view each dense layer (the composite function BN-ReLU-1×1Conv-BN-ReLU-3×3Conv) as a basic layer following their paper (Huang et al., 2019). The first convolutional layer and the transition layers are views as individual basic layers. The splitting criteria is the same as ResNets. Notably, for InfoPro*, the networks are divided to make sure that each local module has the same memory consumption during training, and hence the aforementioned splitting criteria based on the same numbers of basic layers is not applicable, as we discuss in Section 4.1.

G Details of Mutual Information Estimation

In this section, we describe the details on obtaining the estimates of $I(\boldsymbol{h}, \boldsymbol{x})$ and $I(\boldsymbol{h}, y)$ we present in Figures 2, 3 and 6.

As we have discussed in Section 3.3, the expected reconstruction error $\mathcal{R}(\boldsymbol{x}|\boldsymbol{h})$ follows $I(\boldsymbol{h}, \boldsymbol{x}) = H(\boldsymbol{x}) - H(\boldsymbol{x}|\boldsymbol{h}) \geq H(\boldsymbol{x}) - \mathcal{R}(\boldsymbol{x}|\boldsymbol{h})$ (Vincent et al., 2008; Rifai et al., 2012; Kingma & Welling, 2013; Makhzani et al., 2015; Hjelm et al., 2019). Therefore, similar to Section 3.3, we estimate $I(\boldsymbol{h}, \boldsymbol{x})$ by training a decoder parameterized by $\boldsymbol{\omega}$ to obtain the minimal reconstruction loss, namely $I(\boldsymbol{h}, \boldsymbol{x}) \approx \operatorname{argmax}_{\boldsymbol{\omega}} [H(\boldsymbol{x}) - \mathcal{R}_{\boldsymbol{w}}(\boldsymbol{x}|\boldsymbol{h})]$. Note that, ideally, this bound can be arbitrarily tight provided that $\boldsymbol{\omega}$ has sufficient capacity. In specific, we use the same network architecture as Table

7, and train it for 10 epochs to minimize the averaged binary cross-entropy reconstruction loss of each pixel. An adam (Kingma & Ba, 2014) optimizer with default hyper-parameters (lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0) is adopted. Naive as this procedure might seems, for one thing, we find that it is sufficient to reconstruct the input images well given enough information, and meanwhile distinguish different values of $I(\boldsymbol{h}, \boldsymbol{x})$ via the quality of reconstructed images, as shown in Figure 7. For another, we are primarily concerned with the comparisons of $I(\boldsymbol{h}, \boldsymbol{x})$ between end-to-end training and various cases of greedy supervised learning rather than obtaining the exact values of $I(\boldsymbol{h}, \boldsymbol{x})$. The same training process is applied to all the intermediate features $\boldsymbol{h}$, and hence the comparisons are fair. Finally, since $H(\boldsymbol{x})$ is a constant, for the ease of understanding, we simply present $1 - AverageBinaryCrossEntropyLoss(\boldsymbol{x}|\boldsymbol{h})$ as the estimates of $I(\boldsymbol{h}, \boldsymbol{x})$, equivalent to adding the constant $1 - H(\boldsymbol{x})$ to the real estimates of $I(\boldsymbol{h}, \boldsymbol{x})$.

For $I(\boldsymbol{h}, y)$, as we have discussed in Section 3.3 as well, since $I(\boldsymbol{h}, y) = H(y) - H(y|\boldsymbol{h}) = H(y) - \mathbb{E}_{(\boldsymbol{h},y)}[-\log\, p(y|\boldsymbol{h})]$, we train an auxiliary classifier $q_{\boldsymbol{\psi}}(y|\boldsymbol{h})$ with parameters $\boldsymbol{\psi}$ to approximate $p(y|\boldsymbol{h})$, such that we have $I(\boldsymbol{h}, y) \approx \operatorname{argmax}_{\boldsymbol{\psi}}\{H(y) - \mathbb{E}_{\boldsymbol{h}}[\sum_y -p(y|\boldsymbol{h}) \log\, q_{\boldsymbol{\psi}}(y|\boldsymbol{h})]\}$. Here we simply adopt the test accuracy of $q_{\boldsymbol{\psi}}(y|\boldsymbol{h})$ as the estimates of $I(\boldsymbol{h}, y)$, which is highly correlated to the value of $-\mathbb{E}_{\boldsymbol{h}}[\sum_y -p(y|\boldsymbol{h}) \log\, q_{\boldsymbol{\psi}}(y|\boldsymbol{h})]$. This can be view as the highest generation performance that a classifier based on $\boldsymbol{h}$ is able to reach. Notably, we use a ResNet-32 as $q_{\boldsymbol{\psi}}$. For the inputs of $q_{\boldsymbol{\psi}}$, we up-sample $\boldsymbol{h}$ to 32×32 and map $\boldsymbol{h}$ to 16 channels at the first layer. All training hyper-parameters of the ResNet-32 are the same as Appendix F.

H EMPIRICAL STUDY ON THE TIGHTNESS OF THE UPPER BOUND

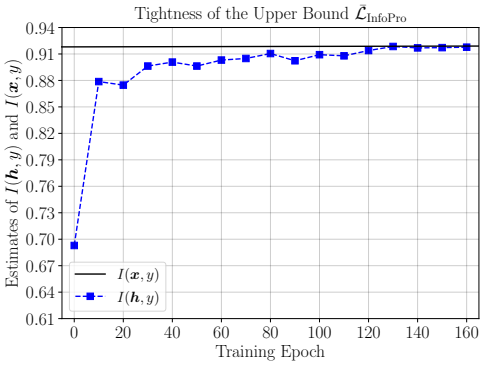

Figure 8: The estimates of $I(\boldsymbol{x}, y)$ and $I(\boldsymbol{h}, y)$.

In this section, we empirically study the gap between of the upper bound $\bar{\mathcal{L}}_{\text{InfoPro}}$ and $\mathcal{L}_{\text{InfoPro}}$. Since the gap $\epsilon = \bar{\mathcal{L}}_{\text{InfoPro}} - \mathcal{L}_{\text{InfoPro}}$ is upper bounded by $\epsilon \leq \lambda_2 \left[I(\boldsymbol{x}, y) - I(\boldsymbol{h}, y)\right]$, we simply need to check the gap between $I(\boldsymbol{x}, y)$ and $I(\boldsymbol{h}, y)$. To this end, we consider training a ResNet-110 on CIFAR-10 using InfoPro (Contrast) with $K = 2$, and estimate the mutual information $I(\boldsymbol{h}, y)$ between the outputs of the first local module and the label. In addition, to obtain a comparable estimate of $I(\boldsymbol{x}, y)$, we assume that in end-to-end learned networks, the intermediate features retains all the task-relevant information within the original inputs (which is empirically observed in this paper). Hence, we can estimate $I(\boldsymbol{x}, y)$ by training the network in an end-to-end fashion, and estimating the mutual information $I(\boldsymbol{h}, y)$ in the same position as the end of the first module. The comparisons between the estimates of $I(\boldsymbol{x}, y)$ and $I(\boldsymbol{h}, y)$ are presented in Figure 8. One can observe that the gap ϵ shrinks gradually during the training process.