Factorizing Knowledge in Neural Networks
-Supplementary Materials-

Xingyi Yang, Jingwen Ye, and Xinchao Wang

National University of Singapore
xyang@u.nus.edu,{jingweny,xinchao}@nus.edu.sg

In this supplementary, we first provide the proof of our variational bound for the InfoMax-BottleNeck objective, and then showcase additional results to verify our proposed knowledge factorization. Next, we describe our implementation details, dataset settings, evaluation metrics, and hyper-parameter settings.

1 Variational Lower bound for IMB

1.1 Lower and Upper bound for Mutual Information

For two random variables X and Y, mutual information (MI) describes the independence between each other. It can be formulated as the Kullback Leibler divergence between the joint probability $p(\mathbf{x}, \mathbf{y})$ and the product of marginal distribution $p(\mathbf{x})p(\mathbf{y})$:

$$\mathcal{I}(X,Y) = D_{KL}[p(\mathbf{x},\mathbf{y})||p(\mathbf{x})p(\mathbf{y})] \tag{1}$$
$$= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y},\mathbf{x})}{p(\mathbf{y})p(\mathbf{x})} \tag{2}$$
$$= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})}. \tag{3}$$

Lower bound. Since computing $p(\mathbf{y}|\mathbf{x})$ directly is intractable, we use a variational distribution $q(\mathbf{y}|\mathbf{x})$ to approximate $p(\mathbf{y}|\mathbf{x})$. Because the Kullback Leibler divergence is always positive $D_{KL}[p(\mathbf{y}|\mathbf{x})||q(\mathbf{y}|\mathbf{x})] \geq 0$, we have a lower bound for the MI as:

$$\mathcal{I}(X,Y) = \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})} \tag{4}$$
$$= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})q(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})q(\mathbf{y}|\mathbf{x})} \tag{5}$$
$$= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{q(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})} + D_{KL}[p(\mathbf{y}|\mathbf{x})||q(\mathbf{y}|\mathbf{x})] \tag{6}$$
$$\geq \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{q(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})} \tag{7}$$
$$= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log q(\mathbf{y}|\mathbf{x}) - \int d\mathbf{y} p(\mathbf{y}) \log p(\mathbf{y}) \tag{8}$$
$$= \mathbb{E}_{p(\mathbf{y},\mathbf{x})}\Big[\log q(\mathbf{y}|\mathbf{x})\Big] + H(Y) \tag{9}$$
$$\geq \mathbb{E}_{p(\mathbf{y},\mathbf{x})}\Big[\log q(\mathbf{y}|\mathbf{x})\Big]. \tag{10}$$

Note that the entropy term $H(Y) \geq 0$ and is sometimes discarded when $H(Y)$ is a constant.

In this work, we assume $q(\mathbf{y}|\mathbf{x})$ to be an energy-based function that is parameterized by a critic function $f(\mathbf{x}, \mathbf{y})$:

$$q(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y})}{C} e^{f(\mathbf{x},\mathbf{y})}, \text{where } C = \mathbb{E}_{p(\mathbf{y})}\left[e^{f(\mathbf{x},\mathbf{y})}\right]. \tag{11}$$

We substitute $q(\mathbf{y}|\mathbf{x})$ into Equation 9 and derive an unnormalized lower bound on MI, which we refer to as I_{UBA} for the Barber and Agakov bound [1] and I_{DA} Donsker & Varadhan bound [6]. We write:

$$\begin{aligned}
&\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\log q(\mathbf{y}|\mathbf{x})] + H(Y) && (12)\\
\geq &\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[f(\mathbf{x},\mathbf{y})] - \mathbb{E}_{p(\mathbf{x})}\left[\log C\right] = I_{\text{UBA}} && (13)\\
\geq &\mathbb{E}_{p(\mathbf{x},\mathbf{y})}[f(\mathbf{x},\mathbf{y})] - \log \mathbb{E}_{p(\mathbf{x})}\left[C\right] = I_{\text{DV}} \quad \text{[Jensen's inequality]}. && (14)
\end{aligned}$$

Upper bound. We then consider the upper bound of mutual information. Similarly, we use $q(\mathbf{y})$ as the variational approximation to the marginal distribution of $p(\mathbf{y})$. Using the non-negativity property of KL divergence again, we know that $D_{KL}[p(\mathbf{y})|q(\mathbf{y})] = \int d\mathbf{y} p(\mathbf{y}) \log \frac{p(\mathbf{y})}{q(\mathbf{y})} > 0$. Therefore we get a tractable variational upper bound for $\mathcal{I}(X, Y)$:

$$\begin{aligned}
\mathcal{I}(X,Y) &= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y})} && (15)\\
&= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})q(\mathbf{y})}{p(\mathbf{y})q(\mathbf{y})} && (16)\\
&= \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y})} - \int d\mathbf{y} p(\mathbf{y}) \log \frac{p(\mathbf{y})}{q(\mathbf{y})} && (17)\\
&\leq \int d\mathbf{x}\, d\mathbf{y} p(\mathbf{x},\mathbf{y}) \log \frac{p(\mathbf{y}|\mathbf{x})}{q(\mathbf{y})} && (18)\\
&= \mathbb{E}_{p(\mathbf{y})}\Big[D_{KL}[p(\mathbf{y}|\mathbf{x})||q(\mathbf{y})]\Big]. && (19)
\end{aligned}$$

1.2 Approximate IMB with Variational bounds

For each task, we assume the three random variables X, Y_j, T_j follow a Markov chain that $X \rightarrow T_j \rightarrow Y_j$. Because computing IMB objective directly is intractable, we resort to maximizing a variational lower bound:

$$\begin{aligned}
\hat{\mathcal{L}}_I &= \mathcal{I}(T_j, Y_j) + \alpha\mathcal{I}(X,Z) - \beta\mathcal{I}(X,T_j)\\
&\geq \mathbb{E}_{p(\mathbf{y}_j,\mathbf{t}_j)}[\log q(\mathbf{y}_j|\mathbf{t}_j)] + H(Y)\\
&\quad + \alpha\big(\mathbb{E}_{p(\mathbf{z},\mathbf{x})}[\log q(\mathbf{z}|\mathbf{x})] + H(Z)\big)\\
&\quad - \beta\mathbb{E}_{p(\mathbf{t}_j)}\Big[D_{KL}[p(\mathbf{t}_j|\mathbf{x})||q(\mathbf{t}_j)]\Big]\\
&\geq \mathbb{E}_{p(\mathbf{y}_j,\mathbf{t}_j)}[\log q(\mathbf{y}_j|\mathbf{t}_j)]\\
&\quad + \alpha\big(\mathbb{E}_{p(\mathbf{z},\mathbf{x})}[\log q(\mathbf{z}|\mathbf{x})] + H(Z)\big)\\
&\quad - \beta\mathbb{E}_{p(\mathbf{t}_j)}\Big[D_{KL}[p(\mathbf{t}_j|\mathbf{x})||q(\mathbf{t}_j)]\Big],
\end{aligned}$$

where D_{KL} denotes the KL divergence between two distributions and $q(\cdot)$ denotes the variational distributions. The entropy term $H(Y) \geq 0$ is canceled because it is a constant that is irrelevant to the optimization.

2 Additional Experiments

In this section, we provide additional experiments to further verify the utility of our proposed KF.

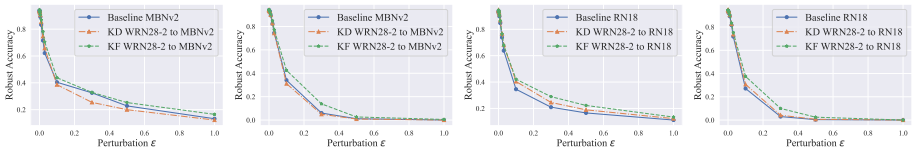
2.1 Factor Networks are Interpretable and Robust

We testify if our factorized model embraces better interpretability and robustness by applying adversarial attacks and visualizing its attribution map.

Robustness Setup. We apply adversarial perturbations to MBNv2 and ResNet-18 to examine their robustness. For each backbone, we use three models that are trained with different strategies: (1) Baseline without teacher (2) Distilled from WRN28-2 model and (3) Single-task network factorized from WRN28-2. We apply Fast Gradient Sign Method (FGSM) [8] and Projected Gradient Descent (PGD) [14] with 40 iterations as our attacks. Attacks are conducted with different magnitude ϵ in L_∞ norm ball.

Attribution Setup. We apply Grad-Cam [17] to visualize the attribution map for ResNet-18 trained on Shape3D and MBNv2 trained on CIFAR-10 datasets. The yellow area highlights the most informative region in the image space. For each factor network, we visualize the Grad-Cam with respect to the last layer of its TSN.

Robustness and Attribution Results. Figure 1 illustrates the accuracy under different attacks. For both attacks and two backbones, factor networks (green dashed plot with star) perform the marginal better than the KD model and the baselines. Meanwhile, factor networks provide sharp and noiseless explanation for the input compared with its teacher and KD counterpart, as shown in Figure 2 and 3. In fact, minimizing $\mathcal{I}(X, T_j)$ can be regarded as posing regularization on task-specific feature $\mathbf{t}^j$. It enforces $\mathbf{t}^j$ to be compact and sparse, which naturally gives rise to interpretability to different tasks and robustness to input noise.



(a) MBNv2 FGSM (b) MBNv2 PGD-40 (c) R18 FGSM (d) R18 PGD-40

Fig. 1: Adversarial robustness accuracy across backbones and attack. R18 stands for ResNet-18.

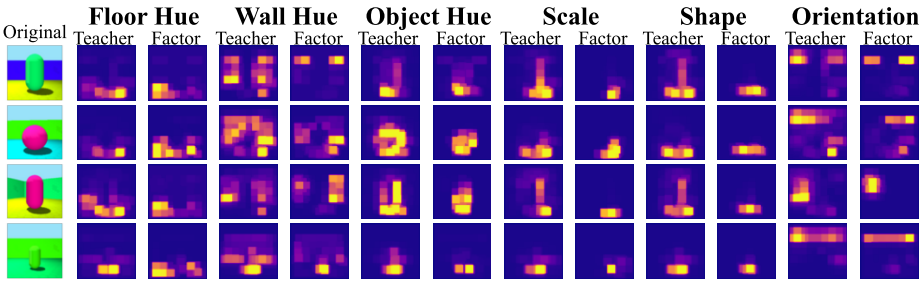


Fig. 2: Grad-Cam visualization for ResNet-18 on Shape3D datasets. Given the input in (column 1), we compare the attribution obtained from the teacher network and its factorized students on 6 tasks.

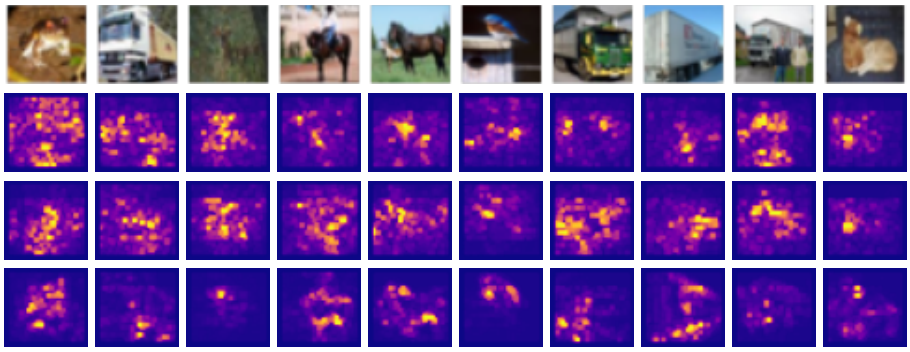


Fig. 3: Grad-Cam visualization for MBNv2 on CIFAR-10. (row 1) Input (row 2) Baseline MBNv2 (row 3) MBNv2 distilled from ResNet-18 and (row 4) MBNv2 factorized from ResNet-18.

2.2 Sub-Task Integration

We show here how KF performs when constructing new sub-task prediction models by integrating factor networks.

Experiment Setup. We construct 5 binary-class classification tasks and 5 trinary-class classification tasks from the CIFAR-10. We deliberately selected some indistinguishable categories into a group to increase the challenge. The binary classification problems include *cat-dog*, *deer-horse*, *automobile-truck*, *bird-airplane* and *automobile-ship*. Each binary classification task contains 10,000 images for training (5,000 per class) and 1,000 for testing (500 per class). The trinary-class classification problems include *cat-dog-frog*, *deer-horse-dog*, *automobile-truck-ship*, *bird-airplane-frog* and *automobile-ship-airplane*. Each trinary-class problem contains 15,000 images for training (5,000 per class) and 1,500 for testing (500 per class). We compare the classification accuracy for 4 models

- A ResNet-18 trained on full CIFAR-10,
- A MobileNetv2 trained on sub-task without teacher,
- A MobileNetv2 trained on sub-task, distilled from full-CIFAR-10 ResNet-18,
- Ten factor networks (MobileNetv2 as CKN, MobileNetv2x0.5 as TSN) trained on full CIFAR-10, factorizied from full-CIFAR-10 ResNet-18.

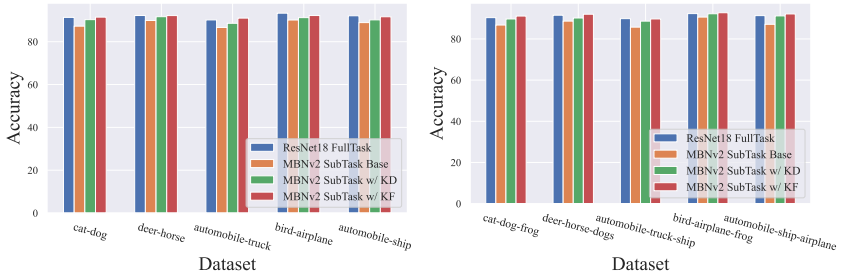
For the KF experiments, we first factorize 10 factor networks from the teacher model, with each network corresponding to a single category prediction. Then we evaluate the combined prediction of the 2 or 3 factor networks based on the sub-task requirement. We use the soft-target [9] as our knowledge transfer loss function for both KD and KF, with temperature $T = 10$. We set the initial learning rate to 0.1, momentum to 0.9, and weight-decay to 0.0001. The models' weights are optimized with SGD for 200 epochs. The learning rate is reduced by 0.1 at the 100-th and 150-th epoch.

Results. Figure 4 provides the test accuracy comparison on 10 CIFAR-10 sub-tasks when trained with different strategies. For all binary-class and trinary-class sub-tasks, we observe that KD considerably improves the baseline MBNv2 accuracy by 2%. Meanwhile, KF further boosts the performance by 1%. Ideally, in order to obtain all binary-class and trinary-class sub-task models of CIFAR-10, KD needs to train $\binom{10}{2} = 45$ binary-class models and $\binom{10}{3} = 120$ trinary-class models, while our proposed KF only needs to train 10 models to be competent for all subtasks. KF significantly reduces training time and model numbers compared with KD.

2.3 Model Size

Despite that factorizing a large model into multiple small sub-networks may in theory increase the total number of parameters, we conduct experiments to show that KF, in practise, achieves better performance even with less parameters.

Experiment Setup. We compare the number of parameters and the test performance on CIFAR-10. We compare the baseline MBNv2, ResNet-18, WRN-28-2, and WRN-28-10 models and their factor networks with WRN-28-10 in terms



(a) Binary-class Classification Task Result

(b) Trinary-class Classification Task Result

Fig. 4: Test Accuracy on binary-class and trinary-class sub-tasks contrasted on CIFAR-10 with different training strategies.

of parameter number and accuracy. The experimental setup is the same as the "Image Classification" section in the main paper.

Results. We show the number of parameters and the test performance on CIFAR-10 in Table 1. As illustrated, factor network with 1 CKN WRN-28-2 and 1 TSN MBNv2x0.5 improves MBNv2 by 1.22% with slightly less parameter. Similarly, the factor network with 1 CKN ResNet-18 and 1 TSN MBNv2x0.5 marginally improves WRN-28-10 with only 2/5 model parameters. These results verify that KF is able to achieve promising test accuracy even with less parameters.

Table 1: Model Parameter Number and Accuracy on CIFAR-10. KF yields better classification accuracy with less parameters.

Method	Network	Params(M)	Acc(%)
Baseline	MBNv2	3.50	93.58
KF	CKN WRN-28-2 + 1 TSN MBNv2x0.5	**3.44**	**94.80**
Baseline	WRN-28-10	36.48	95.32
KF	CKN ResNet-18 + 1 TSN MBNv2x0.5	**13.66**	**95.40**
Baseline	ResNet-18	11.69	94.45
KF	CKN WRN-28-2 + 2 TSN MBNv2x0.5	**5.41**	**95.03**

2.4 Visualization for the Multi-task Dense Prediction

We visualize the prediction results for the multi-task dense prediction on NYUDv2 dataset. We compare 3 methods with ground-truth labels

- Multi-Task HRNet-18, without teacher,
- Multi-Task HRNet-18, distilled from Multi-Task HRNet-48,

– Multi-Task HRNet-18 as CKN and MBNv2 as TSN, factorized from Multi-Task HRNet-48.

Results. Figure 5 shows the semantic segmentation (row 2-5) and depth estimation (row 6-9) results on NYUDv2 dataset. We observe that factor networks produce more accurate depth estimation with sharper edges (column 2 and 4) compared with the distilled students. On top of this, the knowledge factorization enables students to better learn the pattern of the minority categories. For example, the general KD or baseline models fail to classify the "whiteboard" category, which is a rare class in NYUDv2. However, the segmentation factor network succeeds in roughly sketching the area of the whiteboard.

Similar results are also visualized for PASCAL-Context in Figure 6, in which the factor networks produce more accurate prediction results than those of KD. For example, on the column 2 and 3, the segmentation factor network correctly predicts the "people" masks whereas the KD fails. Likewise, the normal-task factor network succeeds to estimate the architectural decoration at the top of the building roof (column 5).

3 Experiment Details

3.1 Datasets

ImageNet1K. ImageNet [16] is a large-scale image classification dataset. The publicly released dataset contains 1,280,000 training with image-level images with 1,000 object categories. In this work, we also deem it as a 11-task dataset based on the category semantic subtrees. The 11 subtrees summarize 11 super-classes: (`n01466257`, chordate), (`n01905661`, invertebrate), (`n02152991`,game), (`n01317541`, domestic animal, domesticated animal), (`n00021939`,artifact, artefact), (`fa11misc`, Misc), (`n00019128`, natural object), (`n09287968`, geological formation, formation), (`n00007846`,person, individual, someone, somebody, mortal, soul), (`n00017222`, plant, flora, plant life),(`n12992868`, fungus). During training, we apply data augmentation to each sample. We randomly resize and crop a 224×224 patch from the image and horizontally flip it with a probability of 0.5. For each test image, we resize it to 256×256 and center-crop a 224×224 patch.

CIFAR10. The CIFAR-10 [12] dataset consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. In this work, we deem it as a 10-task dataset, with 10 binary classification tasks on each category, and also as a 2-task dataset, with a vehicle super-class (airplane, automobile, ship, truck) and an animal super-class (bird, cat, deer, dog, frog, horse). Each training sample is randomly cropped to a 32×32 sample and randomly flipped with a probability of 0.5. We do not apply any data augmentation on the test set.

dSprites. dSprites [15] is a dataset of 2D shapes procedurally generated from 6 independent latent factors. These factors are *color*, *shape*, *scale*, *rotation*, *x positions* and *y positions* of a sprite. The latent factor values are shown in Table 2.

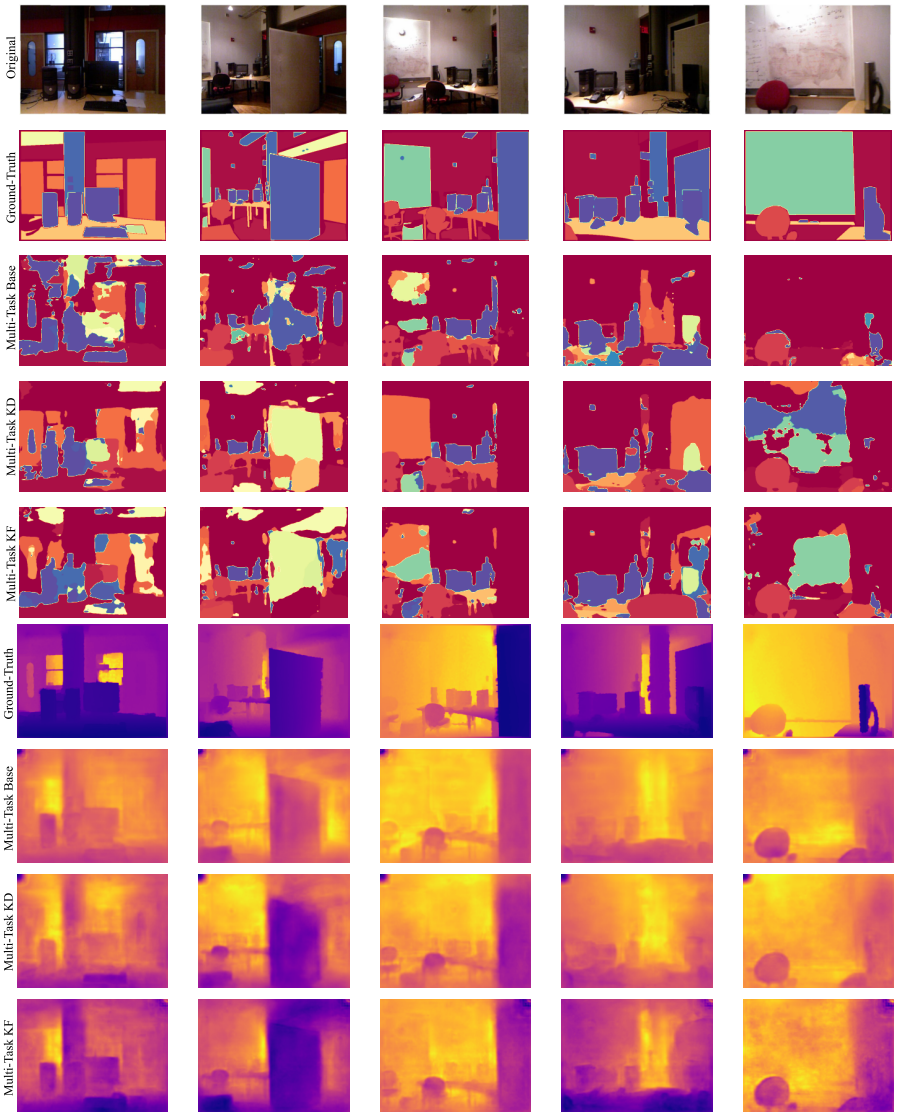


Fig. 5: Qualitative results on scene parsing and depth estimation on NYUDv2.

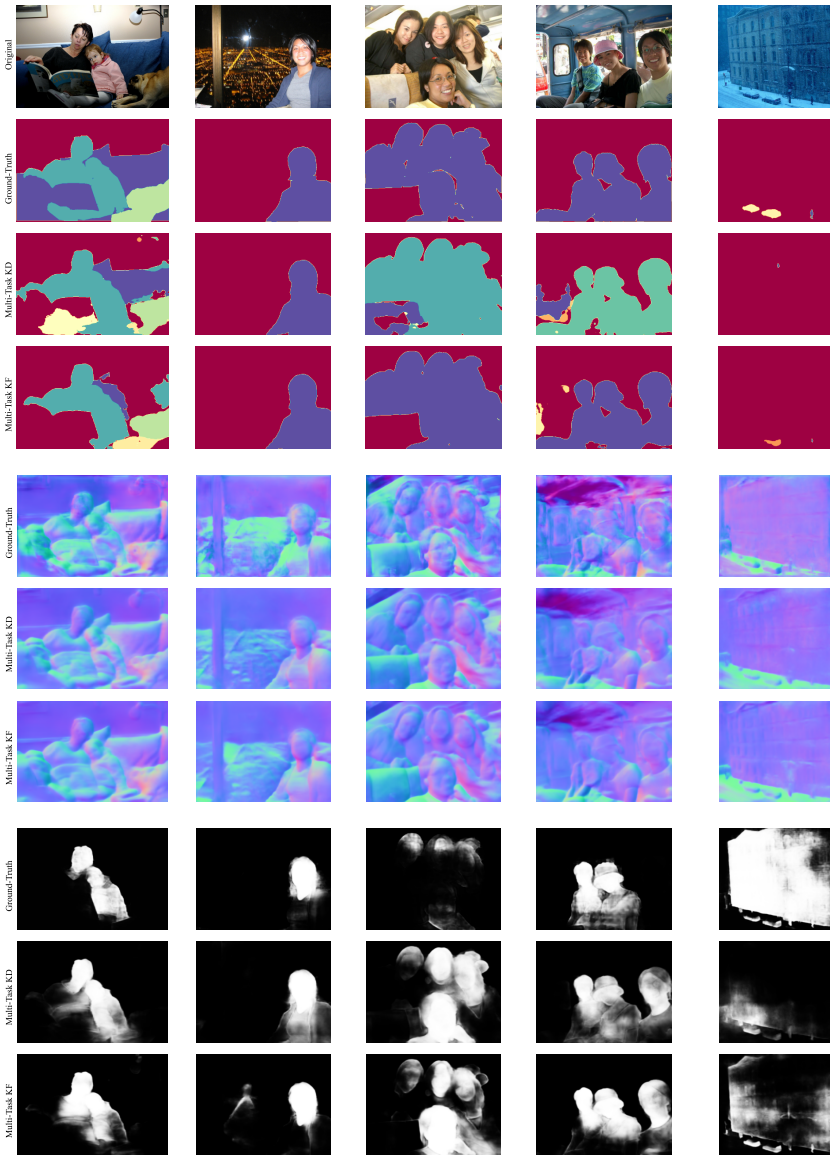


Fig. 6: Qualitative results on segmentation, normal estimation, and salient detection on PASCAL-Context dataset.

All possible combinations of these latents variable are present exactly once, generating N = 737,280 total images. We do not apply any data augmentation when training or testing on dSprites.

Latent factor	Value
Color	White
Shape	square, ellipse, heart
Scale	6 values linearly spaced in [0.5, 1]
Orientation	40 values in [0, 2π]
Position X	32 values in [0, 1]
Position Y	32 values in [0, 1]

Table 2: Latent factor values for dSprites Dataset

Shape3D. Shape3D [2] is a dataset of 3D shapes procedurally generated from 6 independent latent factors. These factors are *floor colour*, *wall colour*, *object colour*, *scale*, *shape*, and *orientation*. The latent factor values are shown in Table 3. All possible combinations of these latents are present exactly once, generating N = 480,000 total images. Each image is of the scale 64×64. We do not apply any data augmentation when training or testing on Shape3D.

Latent factor	Value
floor hue	10 values linearly spaced in [0, 1]
wall hue	10 values linearly spaced in [0, 1]
object hue	10 values linearly spaced in [0, 1]
scale	8 values linearly spaced in [0, 1]
shape	4 values in [0, 1, 2, 3]
orientation	15 values linearly spaced in [-30, 30]

Table 3: Latent factor values for Shape3D Dataset

NYU-Depth V2. The NYU-Depth V2 (NYUDv2) [18] dataset comprises video sequences from a variety of indoor scenes as recorded by both the RGB and Depth cameras from the Microsoft Kinect. It consist of 795 training and 654 testing images of indoor scenes, with annotations for 40-class semantic segmentation ("Seg."), depth estimation ("Depth."), surface normal estimation and boundary detection. We only include the semantic segmentation and depth estimation tasks in our implementation. Each training sample is horizontally flipped with a probability of 0.5 and re-scaled by a factor of $s \in \{1.0, 1.2, 1.5\}$. Finally, the training images are resized to 480×640. All the test images are also resized to 480×640, without other data augmentations.

PASCAL-Context. The PASCAL-Context [5] is a split of the larger PASCAL dataset [5], providing 4,998 training and 5,105 testing images, labeled for 20-class semantic segmentation ("Seg."), human parts segmentation ("H.Part."), saliency estimation ("Sal."), surface normal estimation ("Norm."), and boundary detection. We only include the first 4 tasks in our work. Each training sample is horizontally flipped with a probability of 0.5, randomly rotated with a degree sampled from $\theta \in [-20°, 20°]$ and re-scaled by a factor of $s \in [0.75, 1.25]$. Finally, the training images are resized to 512×512. All the test images are also resized to 512×512, without other data augmentations.
CUB-200-2011. CUB-200-2011 [19] is an extended version of the CUB-200 dataset [21], with roughly double the number of images per class and new part location annotations. The dataset contains 11,788 images of 200 bird species, including 5,994 training samples. Each training sample is randomly cropped to 256×256 and randomly horizontal flipped with the probability of 0.5 for augmentation.
Indoor Scene Dataset. The MIT-67 Indoor Scene Recognition Dataset is an indoor scene imagary classification dataset. It has 15,620 images in total amongst 67 classes including airport, train station, kitchen, and library. We randomly select 70% for training and the rest for testing. Each training sample is randomly flipped with a probability of 0.25 horizontal and 0.25 vertical, randomly rotated with a degree $\theta \in \{90°, 180°, 270°\}$, and randomly cropped to 224×224.

3.2 Feature Similarity

On dsprites and Shape3D, we utilize a 6-layer CNN as teacher and 3-layer CNNs for the rest models. On CIFAR10, we report results with a ResNet18 teacher and ResNet18 students/CKN, alongside with MBNv2x0.5 TSNs. On NYUDv2, we adopt the ResNet50 teacher, and ResNet18 students or CKA, with MBNv2x0.5 TSNs.

3.3 Common Knowledge Network and Task specific network Implementation

As mentioned in the main paper, each factor network is composed of two network modules, namely Common Knowledge Network (CKN) and Task Specific Network (TSN).
TSN. Based on the derivation for the IMB bound, $\mathbf{t}_j$ is stochastically sampled from a Gaussian distribution parameterized by the TSN. Specifically, the channel-wise mean and variance of the output feature vector of a TSN is computed. At the same time, $\mathbf{t}_j$ should be powerful to make the individual task prediction with another task head $\mathcal{H}'_{T_j}$:

$$\mathbf{t}_j = \mathcal{S}_{T_j}(\mathbf{x}; \Theta_{\mathcal{S}_{T_j}}); \mu_{t_j} = \mathbb{E}[\mathbf{t}_j], \sigma^2_{t_j} = \mathrm{Var}[\mathbf{t}_j], \tag{20}$$

$$\hat{y}'_j = \mathcal{H}'_j(\mathbf{t}_j; \Theta_{\mathcal{H}'_j}). \tag{21}$$

Final Prediction. Each input data is fed into two networks in parallel and two feature vectors $\mathbf{z}$ and $\mathbf{t}_j$ from the last layer of CKN and TSN are added as the full task representation. The final task prediction is made by the full task representation:

$$\mathbf{z} = \mathcal{S}_C(\mathbf{x}; \Theta_{\mathcal{S}_C}); \mathbf{t}_j = \mathcal{S}_{T_j}(\mathbf{x}; \Theta_{\mathcal{S}_{T_j}}), \tag{22}$$

$$\hat{y}_j = \mathcal{H}_j(\mathbf{z} + \mathbf{t}_j; \Theta_{\mathcal{H}_j}). \tag{23}$$

3.4 Implementation Details and Hyper-parameters

Disentanglement Experiments. We optimize the model with the following hyper-parameter settings:

- We adopt the Adam [11] method, with initial learning rate set to be 1e-4, weight decay to be 1e-4, and mini-batch size to be 128.
- We train the networks for 20 and 5 epochs on dSprites and 3dshapes respectively. The learning rate is reduced by 0.1 at 10-th and 15-th epoch on dSprites.
- The network structure is shown in Table 5.

We use the evaluation code from *disentanglement_lib*[1].

Layer	Parameter
Input	$64 \times 64 \times c$
Conv-ReLU	4×4, $c = 32$, stride= 2
Conv-ReLU	4×4, $c = 32$, stride= 2
Conv-ReLU	4×4, $c = 64$, stride= 2
Conv-ReLU	4×4, $c = 128$, stride= 2
Conv-ReLU	4×4, $c = 256$, stride= 2
Conv-ReLU	4×4, $c = 256$, stride= 2
FC	$c = 10$

Table 4: 6-layer network architecture for the disentanglement experiment.

Layer	Parameter
Input	$64 \times 64 \times c$
Conv-ReLU	4×4, $c = 32$, stride= 2
Conv-ReLU	4×4, $c = 32$, stride= 2
Conv-ReLU	4×4, $c = 64$, stride= 2
Global Avg-Fool	to 1×1
FC	$c = 10$

Table 5: 3-laye network architecture for the disentanglement experiment.

Image Classification Experiments.

[1] https://github.com/google-research/disentanglement_lib

- On CIFAR-10, we adopt SGD to optimize the objectives, with 0.1 initial learning rate and a momentum term of 0.9. We train the network for 200 epochs and the learning rate is reduced by 0.1 at 100-th and 150-th epoch.
- On ImageNet1K, we adopt SGD to train the model and set the initial learning rate to 0.1 for ResNet-students and 0.05 for MBNv2, with cosine annealing policy and batch-size of 256. The networks are trained for 150 epochs.
- The knowledge transfer loss is set to a soft-target loss with temperature $T = 10$. We use binary cross-entropy as our supervised loss for all experiments.

The classification implementation is based on the *mmclassification*[2] framework.

Multi-Task Dense Experiments.

- On ResNet-18 and ResNet-50 backbone, we use the Deeplabv3 [3] as our decoder. For the HRNet backbone, we concatenate the (upsampled) representations that are from all the resolutions to make the final prediction [20].
- We use the Adam [11] optimizer to train the model, with initial learning rate of 1e-4, weight decay of 1e-4, and batch-size of 12. The learning rate is updated using polynomial policy. We train all models for 80 epochs on NYUDv2 and 100 epochs on PASCAL-Context.
- All network are trained to minimize their cross-entropy loss on semantic segmentation task or L1 distance on other tasks with respect to the ground-truth labels. All task weights are set to be 1.
- We distill the knowledge from pretrained teacher network by minimizing the feature difference between the teacher and students with a dense L1 norm function.

The multi-task dense prediction is implemented base on the *Multi-Task-Learning-PyTorch*[3] open source repository.

Attribution Experiments. We visualize the Grad-CAM map by using the pretrained classification model on CIFAR-10 and 3dshapes.

- On the 3dshapes experiments with ResNet-18, we visualize the Grad-CAM with respect to the output of layer LAYER4.RELU.
- On the CIFAR-10 experiments with MBNv2, we visualize the Grad-CAM that corresponds to the layer CONV2.BN, which is the last batch-norm layer of the MBNv2.

The Grad-CAM is implemented base on the *Captum*[4] library.

Adversarial Experiments. We apply 2 adversarial attacks to the model trained on CIFAR-10 to testify their robustness to adversarial samples.

- The attack magnitude ϵ is defined as the relative pixel perturbation level $k/255$.
- We transverse all $\epsilon \in \{0.0, 0.0005, 0.001, 0.0015, 0.002, 0.003, 0.005, 0.01, 0.02, 0.03, 0.1, 0.3, 0.5, 1.0\}$ and report the test accuracy.

All attacks are implemented using the open-sourced *foolbox*[5] library.

[2] https://github.com/open-mmlab/mmclassification
[3] https://github.com/SimonVandenhende/Multi-Task-Learning-PyTorch
[4] https://captum.ai
[5] https://github.com/bethgelab/foolbox

Transfer Learning Experiments. On both CUB-2011-bird and Indoor Scene dataset, we use stochastic gradient descent with momentum of 0.9 and learning rate of 0.1 for 200 epochs for training both from scratch and with distillation. For applying distillation, we set $T = 20$.

3.5 Disentanglement Metrics

In the main paper, we utilize 4 data evaluation metrics to quantify how well the learned representations disentangle with latent factors. We introduce the definition of each metrics in detail.

FactorVAE. FactorVAE [10] metric measures disentanglement as the accuracy of a majority vote classifier on a different feature vector that predicts the index of a fixed factor of variation.

Mutual Information Gap. For each ground-truth factor of variation, the Mutual Information Gap (MIG) [4] measures the mutual information difference between the top two latent variables with highest mutual information. In our implementation, we instead compute the mutual information difference between two feature dimension with highest MI.

Disenanglement-Completness-Informativeness (DCI). DCI [7] computes the entropy of the distribution obtained by normalizing the importance of each dimension of the learned representation for predicting the value of a factor of variation. We only include the disentanglement score in our evaluations.

SAP. The SAP score [13] is the average difference of the prediction error of the two most predictive latent dimensions for each factor.

References

1. Agakov, D.B.F.: The im algorithm: a variational approach to information maximization. Advances in neural information processing systems **16**(320), 201 (2004)
2. Burgess, C., Kim, H.: 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/ (2018)
3. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
4. Chen, R.T., Li, X., Grosse, R., Duvenaud, D.: Isolating sources of disentanglement in variational autoencoders. arXiv preprint arXiv:1802.04942 (2018)
5. Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., Yuille, A.: Detect what you can: Detecting and representing objects using holistic models and body parts. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1971–1978 (2014)
6. Donsker, M.D., Varadhan, S.S.: Asymptotic evaluation of certain markov process expectations for large time, i. Communications on Pure and Applied Mathematics **28**(1), 1–47 (1975)
7. Eastwood, C., Williams, C.K.: A framework for the quantitative evaluation of disentangled representations. In: International Conference on Learning Representations (2018)
8. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015), `http://arxiv.org/abs/1412.6572`

9. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. ArXiv **abs/1503.02531** (2015)
10. Kim, H., Mnih, A.: Disentangling by factorising. ArXiv **abs/1802.05983** (2018)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
13. Kumar, A., Sattigeri, P., Balakrishnan, A.: Variational inference of disentangled latent concepts from unlabeled observations. arXiv preprint arXiv:1711.00848 (2017)
14. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=rJzIBfZAb`
15. Matthey, L., Higgins, I., Hassabis, D., Lerchner, A.: dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/ (2017)
16. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
17. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
18. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: European conference on computer vision. pp. 746–760. Springer (2012)
19. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
20. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. IEEE transactions on pattern analysis and machine intelligence (2020)
21. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-UCSD Birds 200. Tech. Rep. CNS-TR-2010-001, California Institute of Technology (2010)