Supplementary Material for "Distilling Causal Effect of Data in Class-Incremental Learning"

This Appendix includes additional explanation of causal graphs (Section A), the detailed implementations for different models (Section B), and more supplementary results (Section C).

A. Additional Explanation of Causal Graph

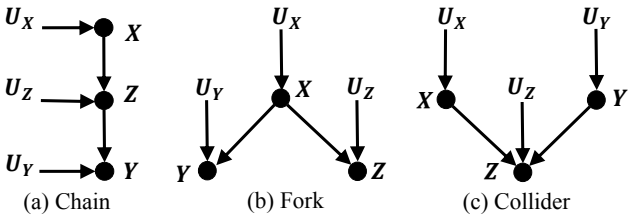
In the main paper, we show in detail that based on our causal graphs for Class-Incremental Learning, (anti-) forgetting can be explained by the causal effect of old knowledge. When the current learning is independent of the old knowledge, the causal effect of old knowledge is zero. In this section, we supplement that the concept of independence, which is defined in the language of probability, can be expressed visually using the causal graph [6, 7].

A.1. Chains, Forks and Colliders

As discussed, the directed path from X to Y is called the causal path, which denotes that X is the cause of Y. For an unspecified causal graph, where the exogenous variables (U_X, U_Y, U_Z) represent the unknown or random effects that may alter the relationship between the endogenous variables X, Y and Z, there are three basic configurations as shown in Figure 1: 1) **Chain**: one arrow directs into, and one arrow directs out of the middle variable. 2) **Fork**: two arrows emanate from the middle variable. 3) **Collider**: one node receives arrows from two other nodes.

A.2. Conditional Independence

We have rules of conditional independence in chains, forks, and colliders, respectively. 1) conditional independence in chains: If there is only a unidirectional path intercepted by Z between X and Y, X and Y are conditionally independent given Z. 2) conditional independence in forks: If X is a common cause of Y and Z, and there is no other path connecting Y and Z, then Y and Z are independent conditional on X. 3) conditional independence in colliders: If Z is the collision node between X and Y, and there is no other path between X and Y, then X and Y are unconditionally independent, but they are dependent when conditional on Z.



(a) Chain (b) Fork (c) Collider

Figure 1. Illustrations of three causal graph configurations.

A.3. *d*-separation

In complex causal graphs, pairs of variables will have multiple possible paths connecting them, containing a variety of chains, forks, and colliders. To predict the dependencies of a graphical causal model of any complexity, we apply a criterion called d-separation (the d stands for "directional"). A general definition of d-separation is that if node Z blocks every path between two nodes X and Y, then X and Y are d-separated conditional on Z, that is to say, X and Y are independent conditional on Z. To see whether a node blocks a path, we have the following rule: node Z blocks a path p if and only if 1) p contains a chain or a fork where Z is the middle node, and 2) p contains a collider node which is neither Z nor the no descendant of Z.

With d-separation, for any pair of nodes, we can determine whether they are d-separated, meaning there exists an unblocked path between them, or d-connected, meaning that all the paths between them are blocked. When a pair of nodes X and Y is d-separated, they are definitely independent, and thus no causal effect is observed; on the contrary, when a pair of nodes is d-connected, we mean that they are most likely dependent, and thus the causal effect is *non-zero*.

B. More Implementation Details

In Section 5 of the main paper, we deployed our methods (DDE) on two strong baseline models: LUCIR [3] and PODnet [2]. This section first shows the detailed implementation of these models and supplemented different weight assignments in Section 5.2 of the main paper.

	Methods	CIFAR-100		ImageNet-Sub		ImageNet-Full	
		T=5	*10*	*5*	*10*	*5*	*10*
R=5	LUCIR	50.76	53.60	66.44	60.04	62.61	58.01
	+DCE	58.53+5.77	60.06+6.46	70.30+3.86	67.81+7.77	63.81+1.20	60.56+2.55
	+MER	53.55+2.79	57.25+3.65	69.81+3.37	65.54+5.50	65.52+2.91	62.15+4.14
	+All	59.82+9.06	60.53+6.93	70.86+4.42	68.30+8.26	66.18+3.57	62.89+4.88
	PODnet	53.38	55.97	71.43	64.90	61.01	55.36
	+DCE	60.25+6.87	58.58+2.61	73.07+1.64	68.02+3.12	62.98+1.97	58.05+2.69
	+MER	55.43+2.05	58.56+2.59	73.49+2.06	66.14+1.24	62.87+1.76	57.43+2.07
	+All	61.47+8.09	60.08+4.11	74.59+3.16	69.26+4.36	63.15+2.14	58.34+2.98
R=10	LUCIR	61.68	58.30	68.13	64.04	65.21	61.60
	+DCE	64.04+2.36	61.69+3.39	70.91+2.79	68.66+4.62	66.00+0.79	63.28+1.68
	+MER	63.40+1.72	60.27+1.97	70.74+2.62	67.56+3.52	66.68+1.47	64.13+2.53
	+All	64.41+2.73	62.00+3.70	71.20+3.07	69.05+5.01	67.04+1.83	64.98+3.38
	PODnet	61.40	58.92	74.5	70.40	62.88	59.56
	+DCE	62.43+0.83	60.19+1.27	75.76+1.26	71.31+0.91	63.56+0.68	61.32+1.76
	+MER	63.40+2.00	60.44+1.52	75.26+0.76	72.53+2.13	64.00+1.12	61.67+2.11
	+All	62.90+1.50	60.52+1.60	75.76+1.26	73.00+2.60	64.41+1.53	62.09+2.53
R=20	LUCIR	63.57	60.95	70.71	67.60	66.84	64.17
	+DCE	65.18+1.61	62.04+1.09	72.22+1.51	70.14+2.54	67.05+0.21	65.28+1.11
	+MER	64.43+0.86	61.88+0.93	72.13+1.42	69.81+2.21	67.22+0.38	65.08+0.91
	+All	65.27+1.70	62.36+1.41	72.34+1.63	70.20+2.60	67.51+0.67	65.77+1.60
	PODnet	64.7	62.72	75.58	73.48	65.59	63.27
	+DCE	65.07+0.37	64.13+1.41	76.36+0.78	74.19+0.71	66.20+0.61	64.63+1.36
	+MER	65.32+0.62	62.93+0.21	76.50+0.92	74.88+1.40	65.88+0.29	63.78+0.51
	+All	65.42+0.72	63.33+0.61	76.71+1.13	75.41+1.93	66.42+0.83	64.71+1.44

Table 1. More results.

B.1. Implementation of LUCIR

Settings. We adopt a 32-layer ResNet for CIFAR100 [4] and an 18-layer ResNet for ImageNet [1]. When adopting cosine normalization in the last layer, the ReLU layer in the penultimate layer is removed to allow the features to take both positive and negative values. For all datasets, images are augmented with random crops and flips, where the images of CIFAR100 are of size 32×32, and images of ImageNet are of size 224×224, and no more data augmentation is used.

Loss Functions. For original LUCIR, the total loss L is composed of three items — a standard cross-entropy loss L_{ce}, a distillation loss on the features L_{dist} and a margin ranking loss L_{mr}. In each incremental step, the distillation loss is scaled by an adaptive scaling factor $\lambda = \lambda_{base}\sqrt{C_n/C_o}$, where λ_{base} is a hyper-parameter and C_o and C_n are the number of old and current observed classes. λ increases over time. For our distillation of colliding effect (DCE), we change the calculation of L_{ce} as changing the predicted logits to the weighted sum of the logits predicted by the current image and its neighbor images. The backward process is preserved as the prediction logits of neighbor images are detached from the calculation graph.

For the incremental momentum effect removal (MER), in the class-balance finetuning stage, we add a cross-entropy loss for bias-removed logits to learn the hyper-parameters α and β. This loss is only used for training the weights of bias term and does not update the network.

Training Details. The training process comprises 1) network training stage where the backbone and new classifiers are trained, and 2) class-balance finetuning stage where only the new classifiers are trained using balanced subsets. In the network training stage, for CIFAR100, the learning rate starts from 0.1 and is divided by 10 after 80 and 120 epochs (160 epochs in total). For ImageNet, the learning rate also starts from 0.1 and is divided by 10 every 30 epochs (90 epochs in total). In the class-balance finetuning stage, for both datasets, the learning rate starts from 0.01 and is divided by 10 per 10 epochs (20 epochs in total). The networks are trained by SGD [5] with the batch size 128 and momentum 0.9, the weight decay is $5 \cdot 10^{-4}$ for CIFAR100 and $1 \cdot 10^{-4}$ for ImageNet, and λ_{base} is set to 5 for CIFAR100 and 10 for ImageNet.

B.2. Implementation of PODnet

Settings. Same as LUCIR, the ConvNet backbones are ResNet-32 and ResNet-18 for CIFAR-100 and ImageNet

and the last ReLU activation is removed. Specifically, the multi-proxies classifier [2] are composed of M original classifiers. The image augmentation and the size of images are the same as LUCIR.

Loss Functions. For original PODnet, the final loss L is composed of two items — an NCA loss with the multi-modal classifier L_{LSC} and its proposed distillation loss $L_{POD-final}$ combining two components $L_{POD-spatial}$ and $L_{POD-flat}$ on the feature with different types of pooling, where the hyper-parameters λ_c and λ_f are used to balance the two terms. $L_{POD-final}$ is also scaled by the same adaptive scaling factor λ. The implementation of our DDE is the same as in LUCIR.

Training Details. Similarly, the training process comprises network training stage and class-balance finetuning stage. For network training stage, the model is trained for 160 epochs for CIFAR100, and 90 epochs for ImageNet. The learning starts from 0.1 and decreases following cosine annealing scheduling. In the class-balance finetuning stage, for both datasets, the learning rate starts from 0.01 and decreases following cosine annealing scheduling. The networks are trained by SGD [5] with the batch size 64 and momentum 0.9, the weight decay is $5 \cdot 10^{-4}$ for CIFAR100 and $1 \cdot 10^{-4}$ for ImageNet, λ_c and λ_f is set to 3 and 1 for CIFAR100 and 8 and 10 for ImageNet.

B.3. Different Weight Assignments

As discussed in Section 4.1 and Section 5 of the main paper, we used a weight assignment for summing the prediction logits of the current image i and its n neighbor images. For the *Top n* strategy, we can treat the $n+1$ weights as a series as:

$$\{W_i, W_1, \cdots, W_n\} = \{\frac{1}{2}, \underbrace{\frac{1}{2n}, \ldots, \frac{1}{2n}}_{n}\}, \qquad (1)$$

which is simple as the neighbor images sharing the same weights and follows the rule Eq.7. For *Variant 1*, we lower the first term as $1/3$, and then the remaining $2/3$ is split equally for each neighbor image. For *Variant2*, the weight of each neighbor image is not the same but is calculated using their similarity with image i — we first calculate the cosine similarity of their old features, and then use the softmax of the similarity as the weights.

C. Additional Results

We include additional results on the effect of proposed components individually (Table 1).

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei Fei Li. ImageNet: a Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 2

[2] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. PODNet: Pooled Outputs Distillation for Small-Tasks Incremental Learning. In *ECCV*, 2020. 1, 3

[3] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a Unified Classifier Incrementally via Rebalancing. In *CVPR*, 2019. 1

[4] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Tech. Rep., University of Toronto*, 2012. 2

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *NeurIPS*, 2012. 2, 3

[6] Judea Pearl. Causality: Models, reasoning, and inference, second edition. *Causality*, 2000. 1

[7] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016. 1