Partial Graph Reasoning for Neural Network Regularization

Tiange Xiang 1  Chaoyi Zhang 1  Yang Song 2  Siqi Liu 3  Hongliang Yuan 4  Weidong Cai 1

Abstract
Regularizers help deep neural networks prevent feature co-adaptations. Dropout, as a commonly used regularization technique, stochastically disables neuron activations during network optimization. However, such complete feature disposal can affect the feature representation and network understanding. Toward better descriptions of latent representations, we present DropGraph that learns a regularization function by constructing a stand-alone graph from the backbone features. DropGraph first samples stochastic spatial feature vectors and then incorporates graph reasoning methods to generate feature map distortions. This add-on graph regularizes the network during training and can be completely skipped during inference. We provide intuitions on the linkage between graph reasoning and Dropout with further discussions on how partial graph reasoning method reduces feature correlations. To this end, we extensively study the modeling of graph vertex dependencies and the utilization of the graph for distorting backbone feature maps. DropGraph was validated on 4 tasks with a total of 8 different datasets. The experimental results show that our method outperforms other state-of-the-art regularizers while leaving the base model structure unmodified during inference.

1. Introduction
Dropout (Srivastava et al., 2014) has been widely used for reducing feature co-adaptation in deep neural networks. With the success of using Dropout as a regularization technique, many recent studies have studied the impact of allowing dropout in neural networks theoretically (Baldi & Sadowski, 2013; Wager et al., 2013) and empirically (Simonyan & Zisserman, 2015; Krizhevsky et al., 2012). In a classic paradigm, neurons are dropped randomly following the Bernoulli distribution with a specific dropout rate $\rho$ during training, such that:

$$f_{\text{Dropout}}(X) = \delta \cdot X,$$

(1)

where $X$ is the incoming signals and $\delta \sim \text{Bernoulli}(\rho)$ is a gating 0-1 Bernoulli variable, with probability $\rho$ for $\delta$ to be 0 and thus dropping out neuron activations. However, this classic approach is primarily targeted on fully-connected linear layers and has little effect on multi-dimensional feature maps with strong spatial correlations.

Considering spatial tensors in image-oriented Convolutional Neural Networks (CNNs), Tompson et al. (Tompson et al., 2015) experimented with dropping out the feature vectors across all channels, which they called SpatialDropout. When neuron activations pass through such layers, feature vectors are binary gated with only two possible states: identity or none. Several recent works (Larsson et al., 2017; Ghiasi et al., 2018; Chen et al., 2020; Tang et al., 2020; Zoph et al., 2018) followed the same intuition and applied dropout in more structured forms.

We argue that the gist of Dropout lies in distorting information propagation of co-adapted signals by random weakening inter-neuron dependencies. The neuron activation distribution varies during network optimization and ends up with information spreading over all neurons uniformly. To this end, applying appropriate distortions to the feature maps could accomplish the same regularization effect. This has also been pointed out by (Tang et al., 2020). Instead of zeroing out $X$ directly, signals can also be obscured by binary gating an extra distortion term, which can be formulated as:

$$f_{\text{reg}}(X) = \delta \cdot X + (1 - \delta) \cdot R,$$

(2)

where $R$ is the appended distortions and $\delta \sim \text{Bernoulli}(\rho)$. This equation splits Eq. 1 into two parts: the origin term and the distortion term, which also generalizes dropout-based methods (Srivastava et al., 2014; Tompson et al., 2015) when there is no distortions i.e. $R = 0$.

In this work, we propose to learn the distortions $R$ via partial graph reasoning. While graph reasoning has been commonly employed to enhance feature representation learning in the computer vision domain (Chen et al., 2019; Li &
Gupta, 2018), we reformulate it as a Dropout-like regularizer in deep neural networks. Specifically, we build a standalone graph network that can be inserted into the backbone to learn \( R \) from an input-specific perspective. Graph reasoning with partially sampled feature vectors is designed with residual-based graph convolutions (Kipf & Welling, 2016) to introduce regularizations similar to Eq. 2. With only the most basic operators, our partial graph reasoning module brings little computational overhead during training. During inference, being a stand-alone graph network, we could completely Drop the Graph, and provide performance improvements to the regularized backbone network without any additional computations.

Our main contributions are two-fold: (1) We propose the very first learning-based regularization framework, namely DropGraph, which dynamically regularizes networks based on partial graph reasoning. A stand-alone graph neural network generates distortions based on randomly sampled feature vectors, and is skipped during inference yet the regularized learning provides marked performance improvement. Motivations and intuitions are clearly clarified with preliminary studies and in-depth discussions. (2) The effectiveness of DropGraph is empirically validated on numerous benchmarks, including: image classification, image semantic segmentation, point cloud classification, and graph recognition. DropGraph outperforms its state-of-the-art counterparts numerically and statistically. Detailed ablative experiments were additionally conducted to provide a comprehensive analysis of our proposed method.

2. Related Work

**Dropout-based Network Regularization.** Regularization techniques (Huang et al., 2016; Wan et al., 2013; Ioffe & Szegedy, 2015) in neural networks aim to reduce feature co-adaptation and, therefore, overcome data over-fitting. As one of the most commonly used regularization techniques, Dropout (Srivastava et al., 2014) randomly wipes out neuron information during network training to penalize model complexity. Recent works have studied the feasibility of dropping out different signal types and proposed several structured dropout variants. DropPath (Larsson et al., 2017) and ScheduledDropPath (Zoph et al., 2018) drop layer-to-layer connections in multi-branch building blocks. DropBlock (Ghiasi et al., 2018; Dai et al., 2019), as a close extension to SpatialDropout (Tompson et al., 2015), drops a block of spatial feature vectors simultaneously. Following DropBlock, AutoDropout (Pham & Le, 2021) utilized an external controller to search for the most suitable Dropout patterns for each individual architecture. In a recent work, R-Drop (Wu et al., 2021) proposed a new training strategy to ensure dropout consistency of sub models in transformer (Vaswani et al., 2017) architectures. Rather than wiping out neuron activations completely, Tang et al. (Tang et al., 2020) adopted a similar approach to ours that generates feature map distortions. However, their so-called distortion policy starts from reducing the intermediate Rademacher complexity, while we focus on modeling feature sample correlations through graph reasoning methods. Moreover, unlike most of these existing approaches, which were mainly evaluated on image classification, our DropGraph is verified on a variety of downstream tasks with different data modalities.

**Graph Reasoning.** Graph structures are considered to better represent inter-relations of signal embeddings than grid-based data structures. Graph Neural Network (GNN) (Scarselli et al., 2008) was first proposed to resolve a graph by updating node embeddings iteratively. Following GNN, Kipf et al. (Kipf & Welling, 2016) presented GCN that adopts learnable kernels to apply convolutions on graphs. During recent years, subsequent studies have been made on learning more effective message passing rules (Hamilton et al., 2017), incorporating attention (Veličković et al., 2017), and increasing network depth (Li et al., 2019). There are also prior works that combine graph convolution modules, i.e., graph reasoning, with CNNs for assisting vision tasks. For example, the two concurrent works (Li & Gupta, 2018; Chen et al., 2019) have studied the feasibility of learning graph representations from features in 2D-grids. They first fully transform the 2D-grid features into discrete graph embeddings and then reason global information through GCN layers. In this work, we reformulate Dropout-based regularizers as partial graph reasoning to regularize the backbone network. In contrast to other graph reasoning approaches, our proposed module is only activated during training and can be completely skipped during inference.

3. Methods

Consider a graph \( G = (V, A) \) with a vertex set \( V \) sampled from the backbone feature maps and an adjacency matrix \( A \). We first describe the commonly adopted graph reasoning methods in CNNs and then discuss how such graphs can be used for regularization. Subsequently, we introduce DropGraph, a graph-based regularizer that learns distortions.

3.1. Graph Reasoning as Regularization

**Graph reasoning on a partial graph.** In a standard graph reasoning paradigm, image-space feature maps \( X \) at a layer are first projected into discrete graph node embeddings (Li & Gupta, 2018; Chen et al., 2019). A stand-alone graph network consisting of consecutive GCN layers is then built to reason the transformed features and re-project them back to the original feature maps in a residual style, such that:

\[
I_{GCN}(X) = X + AXW, \tag{3}
\]
where $A$ is the adjacency matrix, and $W$ is the learnable weights of one GCN layer for simplicity.

We claim that by constructing $V$ with only a subset of the feature vectors in $X$, applying graph reasoning on such partial graph still learns meaningful representations and can be re-formulated as a backbone regularizer with further modifications (later in this section). A sampling ratio $\alpha$ can be set up to control the portion of feature vectors in $X$ to be selected as the graph vertices such that $V \subseteq X$ ($V \equiv X$ when $\alpha = 100\%$ and $V \equiv \emptyset$ when $\alpha = 0\%$). Such partial graph reasoning can be extended from Eq. 3 as:

$$f_{\text{PartialGCN}}(X) = (X \setminus V) \cup (AVW), \quad (4)$$

where $X \setminus V$ represents the complement set of $V$ in the super set $X$.

**Can partial graph reasoning help network learn better?**

To validate whether the stand-alone graph is able to improve the overall performance of backbone network under our partial reasoning setting, preliminary trials are conducted by incorporating Eq. 4 on a ResNet-50 network for image classification. In Table 1, we investigate constructing $V$ from the feature maps following two heuristics: (i) sampling feature tensors with the top absolute activations; (ii) sampling random feature tensors. The impact of different sampling strategies along with sampling ratio $\alpha$ is quantitatively analyzed by either enabling Eq. 4 during both network training & inference or training only. The preliminary experiments offer us the following insights: (i) The additional graph reasoning module with sampling ratio $\alpha < 100\%$ improves backbone network performance generally (highlighted cells), therefore validating the effectiveness of our formulation in Eq. 4. (ii) Enabling graph reasoning modules during training and disabling such modules during inference enhances performance the most.

### Table 1: CIFAR-100 validation accuracy on sampling the top activated features and random features with different sampling rate $\alpha$. ‘Train&Inf.’ denotes enabling the graph during both network training and inference while ‘Train’ denotes during network training only. Highlighted cells represent results better than the plain backbone (with $\alpha = 0\%$).

| $\alpha$ | Top (%) | Random (%) |
| --- | --- | --- |
| | Train&Inf. | Train | Train&Inf. | Train |
| 0% | 77.85 | 77.85 | 77.85 | 77.85 |
| 12.5% | 78.61 | 77.65 | 77.80 | 79.29 |
| 25.0% | 78.88 | 77.28 | 77.92 | 79.42 |
| 50.0% | 77.95 | 76.26 | 77.75 | 79.03 |
| 100.0% | 75.99 | 75.49 | 75.44 | 2.5 |

**Linking between graph reasoning and Dropout.** Following the above insights, we suggest that by disabling graph reasoning during network inference, the partial graph reasoning resembles Dropout (Eq. 1). In this way, we can easily reformulate Eq. 4 to the same form as Eq. 2 by specifying the distortion term $R$ as the output of graph reasoning:

$$f_{\text{reg}}(X) = (\delta \cdot X \setminus V) \cup ((1 - \delta) \cdot AVW), \quad (5)$$

where $V \sim \text{Bernoulli}(X, \alpha)$, $\delta \sim \text{Bernoulli}(\rho)$, and $\cdot$ denotes the broadcast pointwise multiplication. By skipping the graph during network inference, the behaviour of such partial graph reasoning aligns to Dropout, and it therefore can be reformulated as another form of regularization. However, questions still remain on how best to define a valid adjacency matrix $A$ and how to ensure the regularization effect, which will be discussed in the next section.

**Why partial graph reasoning regularizes networks?** In CNNs, feature maps are generated by sliding fixed convolution kernels across each pixel with limited receptive fields. Such paradigm enforces local correlations in the same neighborhood and easily leads to feature co-adaptations. Eq. 5 alleviates the co-adaptation from two perspectives: first, appended distortions break the conventional convolution pattern and disturb inter-pixel correlations at random positions, which has also been discussed in (Tang et al., 2020); second, local pixel responses are replaced by the enhanced graph vertices, enabling long-range dependencies to be captured out of the convolution neighborhood.

### 3.2. DropGraph

After obtaining the derivation in Eq. 5, we here introduce a functional module, namely DropGraph that instantiates such re-formulation in the previous section with a graph reasoning architecture, to be equipped in deep neural networks for feature map regularization. DropGraph is a graph-based regularizer that learns to generate distortions from data. During training, the regularization process of DropGraph (Figure 1 left) is achieved via a mask branch and a graph branch, which can be controlled by the sampling probability $\rho$ and $\alpha$, respectively. The DropGraph algorithm is presented in Figure 1 right.

In the mask branch, we follow Eq. 5 to sample the binary gating variables $\delta$ for each feature vector in the incoming feature map $X$. Instead of gating individual pixels (Srivastava et al., 2014), we sample $\delta$ as spatially contiguous square blocks (Ghiasi et al., 2018; Tang et al., 2020). Therefore, we refer such block-wise gates for the entire feature map as the dropout mask $M$ that is controlled by the block size $s$ and the sampling probability $\rho$.

In the graph branch, we build the graph $G = (V, A)$ from the incoming feature map $X$ and apply partial graph reasoning to generate the distortions as described in Sec. 3.1. Specifically, we construct the graph vertices $V$ by sampling
vertices are assigned with lower values in sim. Strong connections are expected to be built for any pair of non-diagonal values of A. In CNNs, two adjacent feature vectors share similar distributions and hence are very likely to correlate with each other. To ease such adjacency-based correlations, strong connections are expected to be built for any pair of vertices that are apart from each other in the semantic feature space. Specifically, we utilize pair-wise dot product similarity $\text{sim}(\cdot, \cdot)$ for constructing A. Semantically closer vertices are assigned with lower values in A while the less similar ones are allocated with higher values. To achieve this, we minus the softmax gated similarity scores from an all-one matrix and then normalize $A \in (0, 1)$ by scaling down with their cardinality:

$$A = \frac{1 - \text{softmax}(\text{sim}(v_i, v_j))}{\max(||V|| - 1, 1)},$$

(6)

where $|| \cdot ||$ denotes cardinality and all arithmetic operations above are in vectorized form.

**Distortion generation.** Given the sampled vertex set V and the adjacency matrix A, we utilize a set of basic graph message passing rules $\Phi(V, A)$ to generate distortions similarly to Eq. 4, but in a residual style. Towards better efficiency and less computation burdens, we incorporate two GCN bottlenecks at the ends of $\Phi$ with a channel reduction ratio of 4. Another residual GCN is adopted within the bottleneck for better information propagation. All GCN layers in our DropGraph share the same constructed A. After average pooling the distortions, following (Tang et al., 2020), we apply random multipliers within (0, 1) at all sampled units masked by $M$.

**Comparison with state of the art.** DropGraph resembles previous works with probability scheduling and block distortion, but distinguishes them by the novel reformulation of graph reasoning as regularization. Such formulation generalizes other Dropout-based regularizers (Tompson et al., 2015; Ghiasi et al., 2018) as special cases, and is potentially capable of fitting any possible distortion function including Disout (Tang et al., 2020). Note that we are not aiming at

---

**Algorithm 1 DropGraph**

**Input:** $X \in \mathbb{R}^{w \times h \times c}$, mode, $s$, $\rho$, and $\alpha$.

**Output:** $X \in \mathbb{R}^{w \times h \times c}$

1: if mode == Inference then
2: return $X$
3: end if

######initialize######
4: $V = \text{sample}(\alpha, X)$
5: $M = \text{sample}(\rho, X, s)$

######generate distortions######
6: $A \gets \text{Eq. 6}$
7: $R = \text{average}(\Phi(A, V))$
8: $R = \text{expand}(R, M)$

######apply distortions######
9: $X = X \cdot M + R \cdot (1 - M)$
10: return $X$

---

Figure 1: DropGraph learns feature map distortions during network training and can be completely skipped during inference. $w, h, c, s$ denote the feature map width, height, channel, and the block size; sample denotes the Bernoulli sampling; average denotes the average pooling operation; expand denotes the expansion of the pooled distortions to all sampled feature vectors for size consistency; ‘.’ denotes broadcastly multiplication.

*random* tensor vectors in $X$ based on the probability $\alpha$. Then, the adjacency matrix $A$ is constructed among all vertices $\{v\} \in V$ through a similarity measurement function $\text{sim}(\cdot, \cdot)$. $A$ and $V$ are subsequently fed into a distortion generator $\Phi$ to infer vertex-to-vertex distortions, which are then channel-wise average pooled, expanded and applied on the $M$ masked input features. During network training, the graph weights $W$ are jointly optimized with the backbone network by sharing the same gradient flows. To maintain the completeness of feature representations at an initial learning stage, $\rho$ is adjusted from 0% to the target value following a predefined scheduler (Zoph et al., 2018).

**Dependency modeling.** Recall that the purpose of regularizers lies in reducing pixel-to-pixel dependencies. There are two kinds of pixel relationships modeled by the adjacency matrix $A$ in our graph: intra-dependency (modeled as diagonal values of $A$) and inter-dependency (modeled as non-diagonal values of $A$). When $A$ degrades to an all-zero matrix without modeling any dependencies, DropGraph instantiates to DropBlock.

To activate graph reasoning, dependencies have to be embedded in $A$. In CNNs, two adjacent feature vectors share similar distributions and hence are very likely to correlate with each other. To ease such adjacency-based correlations, strong connections are expected to be built for any pair of vertices that are apart from each other in the semantic feature space. Specifically, we utilize pair-wise dot product similarity $\text{sim}(\cdot, \cdot)$ for constructing $A$. Semantically closer vertices are assigned with lower values in $A$ while the less similar ones are allocated with higher values. To achieve this, we minus the softmax gated similarity scores from an all-one matrix and then normalize $A \in (0, 1)$ by scaling down with their cardinality:
chasing state-of-the-art performances on all down-streaming tasks. In this work, we implemented the DropGraph in the most basic form consisting of simple convolutions only. A more careful design of the graph network or the propagation rules will lead to substantially better results, whereas we leave this extension to our future work.

Compared to the recent works, R-Drop (Wu et al., 2021) and AutoDropout (Pham & Le, 2021), our plug-and-play DropGraph is able to function on various architectures universally. R-Drop requires multiple forward pass of the same module during training and adds intermediate supervisions that are hard to balance between the original training objectives; AutoDropout demands highly customized controllers to adjust dropout patterns dynamically for different architectures. Moreover, AutoDropout needs pre-defined search spaces with carefully tuned parameters which imposes much more manual effort and exceeds the run time cost of DropGraph and other counterparts. With close performance, DropGraph yields considerably better accessibility and extensibility during network training. However, due to the lack of resources and open-sourced implementation \(^1\), a direct and fair comparison under our experimental setting is impossible.

### 4. Experiments

In this section, experimental results are firstly reported on 4 different tasks across 8 datasets. We repeat three independent runs for image classification, semantic segmentation and point cloud classification tasks. For graph recognition task, 100 independent runs were repeated. Subsequently, we conducted extensive studies to analyze DropGraph under different ablative settings with a fixed random seed. All experiments were implemented in PyTorch framework (Paszke et al., 2019) using Tesla V100 GPUs. More experiments and complete implementation details can be found in the Appendix Sec. A and Sec. B.

#### 4.1. DropGraph for Image Classification

Two commonly adopted benchmarks were used to evaluate our method on image classification task. CIFAR 10/100 are composed of 60000 images of size $32 \times 32$, with the training set containing 50000 images and the validation set containing 10000 images. The images were evenly distributed into 10 and 100 classes. The ImageNet dataset is composed of 1.2M high-resolution training images and 50K validation images, which are distributed into 1K different categories.

To have a comprehensive validation of our DropGraph on both standard and mobile network regimes, we experimented on two network backbones: ResNet-50 (He et al., 2016) and RegNetX-200MF (Radosavovic et al., 2020).

Only the most basic training strategies were used without advanced tricks such as AutoAugment (Cubuk et al., 2019) or Exponential Moving Average (EMA) of network weights. Note that Disout is originally trained for twice as many epochs as other methods and achieves slightly better results. For fair comparisons, we aligned the training procedures and reproduced all results.

For ResNet-50 backbone, we inserted DropGraph at all building blocks of the last two building groups (Tang et al., 2019) or Exponential Moving Average (EMA) of network weights. Note that Disout is originally trained for twice as many epochs as other methods and achieves slightly better results. For fair comparisons, we aligned the training procedures and reproduced all results.

### Table 2: Comparison results on image classification tasks (%)。

|                | CIFAR          | CIFAR 10        | CIFAR 100       |
|----------------|----------------|-----------------|-----------------|
| ResNet-50      | 93.67±0.11     | 77.73±0.18      |                 |
| + Dropout      | 94.81±0.09     | 77.96±0.13      |                 |
| + SpatialDropout| 94.92±0.11     | 78.27±0.10      |                 |
| + DropBlock    | 95.14±0.11     | 78.71±0.13      |                 |
| + Disout       | 95.25±0.13     | 78.91±0.15      |                 |
| + DropGraph    | **95.35±0.14** | **79.58±0.14**  |                 |
| RegNetX-200MF  | 93.10±0.10     | 71.57±0.10      |                 |
| + Dropout      | 92.77±0.08     | 71.19±0.12      |                 |
| + DropBlock    | 93.05±0.13     | 71.21±0.10      |                 |
| + Disout       | 93.15±0.11     | 71.66±0.13      |                 |
| + DropGraph    | **93.76±0.13** | **71.94±0.15**  |                 |

|                | ImageNet       | Top-1           | Top-5           |
|----------------|----------------|-----------------|-----------------|
| ResNet-50      | 76.51±0.07     | 93.20±0.05      |                 |
| + Dropout      | 76.80±0.04     | 93.41±0.04      |                 |
| + SpatialDropout| 77.41±0.04     | 93.74±0.02      |                 |
| + DropBlock    | 78.13±0.05     | 94.02±0.02      |                 |
| + Disout       | 78.33±0.06     | 93.98±0.04      |                 |
| + DropGraph    | **78.43±0.04** | **94.05±0.04**  |                 |
| RegNetX-200MF  | 67.54±0.08     | 88.29±0.05      |                 |
| + Dropout      | 67.59±0.06     | 88.28±0.04      |                 |
| + DropBlock    | 67.94±0.05     | 88.44±0.04      |                 |
| + Disout       | 68.25±0.06     | 88.56±0.04      |                 |
| + DropGraph    | **68.77±0.06** | **88.65±0.03**  |                 |

\(^1\)https://github.com/google-research/google-research/issues/727
Table 3: Comparison results on semantic segmentation tasks (%).

| Method               | mIoU | mAcc |
|----------------------|------|------|
| FCN-32S              | 67.6 | 68.1 |
| + DropBlock          | 50.6 | 56.9 |
| + Disout             | 57.2 | 67.2 |
| + DropGraph          | 51.0 | 68.3 |
| DeepLabV3            | 53.8 | 83.5 |
| + DropBlock          | 56.9 | 83.7 |
| + Disout             | 57.2 | 84.0 |
| + DropGraph          | 58.0 | 84.6 |

Table 4: Results on point cloud classification task (%).

| Method                | mIoU | DICE |
|-----------------------|------|------|
| U-Net                 | 68.2 | 80.7 |
| + DropBlock           | 67.6 | 80.2 |
| + Disout              | 68.1 | 80.0 |
| + DropGraph           | 68.5 | 81.4 |
| Attention U-Net       | 68.3 | 81.1 |
| + DropBlock           | 68.0 | 81.0 |
| + Disout              | 68.3 | 81.2 |
| + DropGraph           | 68.7 | 81.5 |

![Table 4: Results on point cloud classification task (%).](image)

4.2. DropGraph for Semantic Segmentation

DropGraph is then benchmarked on two semantic segmentation datasets: PASCAL VOC 2012 (Everingham et al., 2015) and MoNuSeg (Kumar et al., 2017). The PASCAL VOC benchmark consists of 21 classes with 20 foreground object classes and one background class. There are 1464 training samples and 1449 validation samples in the original dataset. Following (Hariharan et al., 2011; Chen et al., 2018), we used the extra annotations to enrich the dataset to contain 10582 training images. The regularization methods were evaluated on two of the most commonly used segmentation networks, FCN (Long et al., 2015) and DeepLabV3 (Chen et al., 2018) with randomly initialized ResNet-50 backbone. The quantitative metrics include intersection-over-union (mIoU) and pixel accuracy (mAcc) averaged across the 21 classes. The nuclei segmentation dataset MoNuSeg contains 30 training and 14 testing microscopy images in size 1000². The scans were sampled from different whole slide histopathology images of multiple organs. Following (Xiang et al., 2020), we enrich the dataset by extracting 512² patches at 4 corners. mIoU along with Dice coefficient (DICE) scores are reported for this task. U-Net (Ronneberger et al., 2015) and Attention U-Net (Oktay et al., 2018) were adopted as the backbone networks for nuclei segmentation.

For FCN and DeepLabV3, regularizers were applied at the last two building groups of the ResNet-50 backbone and retain the same hyper-parameters used in image classification tasks, such that $\alpha = 0.2, \rho = 0.1$ and $s = 7$. For U-Net based networks, we inserted the regularizers at the last encoder level only with $\alpha = 0.1, \rho = 0.1$ and $s = 7$.

The segmentation results are presented in Table 3. Consistent performance improvements can be observed on both backbone networks and both datasets, hence proving the universal applicability of our DropGraph. Noteworthy, by zeroing out neuron activations, DropBlock leads to even poorer results on many metrics. However, distortion-based methods appeared to be more suitable for such pixel-wise prediction task. Among all the competing methods, DropGraph provides the greatest performance improvements on the randomly initialized backbone networks and shrinks the gap with the ImageNet pre-trained ones.

4.3. DropGraph for Point Cloud Classification

Apart from studies on image data, we conducted subsequent evaluations of our DropGraph on graph-like data. Since DropBlock is not intrinsically applicable for graph signals, we thus compared DropGraph with Dropout, SpatialDropout and Disout. The ModelNet40 dataset (Wu et al., 2015) was first adopted to verify the potential of DropGraph on 3D point cloud classification task. This 40-class dataset is composed of 9843 training samples and 2468 testing samples. We adopted the same pre-processing instructions as introduced in (Qi et al., 2017; Xiang et al., 2021) to uniformly sample 1024 points from each raw 3D model. The sampled coordinates were further normalized into unit spheres for better network understanding. We utilized DGCNN (Wang et al., 2019) as the backbone network for all experiments. All regularization methods were equipped after each EdgeConv module with $\alpha = 0.15, \rho = 0.1$ and $s = 1$.

As demonstrated in Table 4, most of the regularizers impact the backbone network positively, except for the plain Dropout that hurts feature representations and leads to inferior results. Among all the competing methods, our DropGraph stands out with the greatest average accuracy improvement (0.6%) that even outperforms more advanced and complex networks (Liu et al., 2019; Yan et al., 2020).

4.4. DropGraph for Graph Recognition

Lastly, DropGraph was evaluated on the semi-supervised node classification task with the Cora dataset (Sen et al., 2008) and on the graph classification task with the Protein dataset (Dobson & Doig, 2003). Cora is a citation network dataset, consisting of one graph with 2708 nodes (docu-
we studied four alternatives of $A$ that models distinct intra- and inter-dependencies: (a) $A$ is learned jointly with backbone network and hence is independent of $V$. (b) Similar vertices are assigned with high values, such that $A = \text{softmax}(\text{sim}(V, V))$. (c) Only intra-dependencies are modeled, such that $A = I$ becomes an identity matrix. (d) Both intra-dependencies and inter-dependencies are equally modeled with $A = \frac{1}{||V||}$. 

Table 6: Comparisons on designs of the adjacency matrix $A$.

| Designs of $A$ | Top-1 (%) | Top-5 (%) |
|----------------|-----------|-----------|
| Backbone       | 76.51     | 93.20     |
| (a)            | 77.74     | 93.68     |
| (b)            | 78.13     | 93.88     |
| (c)            | 78.02     | 93.89     |
| (d)            | 78.25     | 93.98     |
| Eq. 6          | 78.40     | 94.07     |

Partial Graph Reasoning for Neural Network Regularization

Table 5: Results on graph recognition tasks (%).

| Methods        | Cora       | Protein    |
|----------------|------------|------------|
| GCN            | 82.9 ± 0.5 | 81.5 ± 0.9 |
| + Dropout      | 83.1 ± 0.5 | 80.8 ± 2.1 |
| + SpatialDropout | 83.2 ± 0.6 | 81.6 ± 1.3 |
| + Disout       | 82.7 ± 0.6 | 81.7 ± 1.5 |
| + DropGraph    | 83.3 ± 0.5 | 82.1 ± 1.7 |

Cluster-GCN

| Methods        | products Val | products Test |
|----------------|--------------|---------------|
| + Dropout      | 92.1 ± 0.0   | 79.0 ± 0.0   |
| + DropGraph    | 92.9 ± 0.0   | 80.2 ± 0.0   |

In these experiments, $\alpha$ is set to 0.15 for all the tasks while $\rho$ is set to 0.1 for Cora, OGB datasets, and 0.25 for Protein.

The results reported in Table 5 demonstrate that among all competing methods, DropGraph yields the greatest average accuracy gains with 0.4% on the Cora dataset and 0.6% on the Protein dataset. DropGraph also achieves superior performances on both validation and testing set of the two large-scale OGB benchmarks.

4.5. Ablation Studies and Analysis

Unless explicitly mentioned, we adopted ResNet-50 as the backbone network and conducted ablative experiments on ImageNet dataset with the same training configurations specified in Sec. 4.1. The linear distortion probability scheduler (Ghiasi et al., 2018) was used for all ablative studies.

Studies on different designs of $A$. Different dependency modeling strategies may impact the regularization effect to different extents. To benchmark our design in Eq. 6, we further examined our proposed method on two large scale datasets: ogpa-ppa and ogbn-products from the Open Graph Benchmark (OGB) (Hu et al., 2020). We followed the official implementations of the backbone networks and inserted our DropGraph between every pair of GCN layers. We studied four alternatives of $A$ that models distinct intra- and inter-dependencies: (a) $A$ is learned jointly with backbone network and hence is independent of $V$. (b) Similar vertices are assigned with high values, such that $A = \text{softmax}(\text{sim}(V, V))$. (c) Only intra-dependencies are modeled, such that $A = I$ becomes an identity matrix. (d) Both intra-dependencies and inter-dependencies are equally modeled with $A = \frac{1}{||V||}$.

The ablative results of the four designs are numerically compared in Table 6. DropGraph brings consistent performance gains to the backbone network with all adjacency matrix designs. We observed from experiment (a) that learning a static $A$ that is insensitive to the incoming feature distributions hurts the overall regularization effect provided by DropGraph. Inferior performances are also obtained when modeling stronger dependencies for similar vertices, as indicated by experiments (b) and (c). Surprisingly, experiment (d) reveals that constructing equal connections between all pairs of vertices also leads to effective regularization and ends up with even higher results than DropBlock.

In our hypothesis, dissimilar features in $V$ should be strongly correlated to weaken the locally similar ones. Therefore, we expect to see the strong modeling of similar features (b) and identity features (c) to be inferior to a weak modeling of similar features (d) and strong modeling of dissimilar features (Eq. 6). The experimental results agreed with our hypothesis.

Studies on $\alpha$ and $\rho$. DropGraph is controlled by the sampling ratio $\alpha$ and the distortion probability $\rho$. A higher $\alpha$ provides a more complete feature distribution and a higher $\rho$ imposes stronger regularizations.

In Figure 2 top, we showed that different settings of $\alpha$ bring consistent accuracy improvement to the backbone network that universally outperforms DropBlock. When $\alpha = 0.2$, DropGraph reaches the highest top-1 accuracy of 78.4%. Our hypothesis is two-fold: (i) After softmax normalization (Eq. 6), $A$ with more samples will have much closer one-to-all similarity. In this way, dependencies between any two samples will become hardly discriminated when the sample number increases. Poorer dependency modeling will directly impact the overall performance,
also pointed out in Table 4. (ii) More samples require better reasoning ability of the graph network. Since our graph network is constructed with only few basic GCN layers, it cannot guarantee effective distortions for large numbers of samples at a time. The experiments also validated that the sampling rate should be not too small nor too big, and 0.2 sampling rate yields the best performance.

With the best $\alpha$ found, in Figure 2 bottom, we demonstrated that with distortion (drop) probability $\in (0.0, 0.2)$, DropGraph significantly outperforms the simplest Dropout and SpatialDropout. Under the same set of $\rho$ values studied in (Ghiasi et al., 2018), DropGraph consistently surpasses the DropBlock baseline with identical inference behaviour.

Grad-CAM masking classification. Grad-CAM (Selvaraju et al., 2017) is widely used to visualize the focus of networks. We are curious whether the focus of backbone network would change when applying different regularization methods. Therefore, we here conduct a set of quantitative experiments to inspect the network focus using the CIFAR 10 dataset, which were set up as follows: (i) Grad-CAM heatmaps were generated using the regularized backbones. (ii) The heatmaps were binarized with a 50% threshold for each image instance. (iii) The original images were then masked by the thresholded binary heatmaps and we trained an additional ResNet-56 on the masked RGB images.

Table 7: Grad-CAM heatmaps masked CIFAR-10 classification.

| Methods | +Dropout | +Disout | +DropGraph |
|---------|---------|---------|-----------|
| Acc (%) | 51.25±0.22 | 58.69±0.21 | 65.64±0.17 |

In Table 7, we found that masking the images with our DropGraph yields the highest classification accuracy, thus validating the heatmaps indeed focus on the primary objects better.

**Distortion generation methods.** As mentioned in Sec. 3.2, we hypothesize that GCN is able to aggregate long range uncorrelated signals with specific underlying patterns. However, other perturbation methods, such as standard convolution, feature average pooling, and noise injection, either cannot model meaningful long range dependencies or yield no capturable patterns for the backbone network. To validate our hypothesis empirically and claim the necessity of using the stand-alone graph network to generate distortions, we examined a series of different methods in Table 8.

Our conclusion from this experiment is two-fold: (i) Generating distortions with short range features provides weak regularization effects. This aligns with our claim in Sec. 3.1. (ii) For dilated conv generated distortions, using the one average pooled distortion to override all original features leads to slightly better results, whereas using GCN-based generation method to generate different distortions for different samples is able to achieve the best results. Therefore, such empirical study validates that our proposed DropGraph framework along with the partial graph reasoning strategy is the most effective method among all other possible designs.

**Limitations.** With the additional graph, DropGraph yields a bit more complex framework than the DropBlock baseline: (i) Same as Disout, DropGraph requires three hyper-parameters ($\alpha, \rho, s$) which are one more than DropBlock. However, in Sec. 4.5, we demonstrate that DropGraph generally surpasses DropBlock without explicit tuning of the hyper-parameters; (ii) DropGraph puts additional computations to the backbone network during training. Nevertheless, the training computation overhead is trivial that DropGraph only imposes less than $1 \times 10^{-3}$ G extra MACs on the ResNet-50 backbone and merely 4% more training time compared to Disout.

**5. Conclusion**

In this paper, we introduce the very first learning-based framework for neural network regularization, namely DropGraph. Instead of zeroing out information as in Dropout,
we build a stand-alone graph neural network to generate feature map distortions. DropGraph provides Dropout-based regularization by sampling graph nodes from partial feature maps with dependencies modeled via the adjacency matrix and distortions generated through a set of graph reasoning layers. Our DropGraph outperforms other state-of-the-art regularizers on a variety of tasks and datasets.

References

Baldi, P. and Sadowski, P. J. Understanding dropout. In Advances in Neural Information Processing Systems (NeurIPS), pp. 2814–2822, 2013.

Chen, L., Gautier, P., and Aydore, S. Dropcluster: A structured dropout for convolutional networks. arXiv preprint arXiv:2002.02997, 2020.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European conference on computer vision (ECCV), pp. 801–818, 2018.

Chen, Y., Rohrbach, M., Yan, Z., Shuicheng, Y., Feng, J., and Kalantidis, Y. Graph-based global reasoning networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 433–442, 2019.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 113–123, 2019.

Dai, Z., Chen, M., Gu, X., Zhu, S., and Tan, P. Batch dropblock network for person re-identification and beyond. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 3691–3701, 2019.

Dobson, P. D. and Doig, A. J. Distinguishing enzyme structures from non-enzymes without alignments. Journal of molecular biology, 330(4):771–783, 2003.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. International journal of computer vision, 111(1):98–136, 2015.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. Dropblock: A regularization method for convolutional networks. In Advances in Neural Information Processing Systems (NeurIPS), pp. 10727–10737, 2018.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035, 2017.

Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. Semantic contours from inverse detectors. In 2011 International Conference on Computer Vision, pp. 991–998. IEEE, 2011.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. Neural Information Processing Systems (NeurIPS), 2020.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In European Conference on Computer Vision (ECCV), pp. 646–661. Springer, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pp. 448–456. PMLR, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NeurIPS), pp. 1097–1105, 2012.

Kumar, N., Verma, R., Sharma, S., Bhargava, S., Vahadane, A., and Sethi, A. A dataset and a technique for generalized nuclear segmentation for computational pathology. IEEE Transactions on Medical Imaging, 36(7):1550–1560, 2017.

Larsson, G., Maire, M., and Shakhnarovich, G. Ultra-deep neural networks without residuals. In International Conference on Learning Representations (ICLR), volume 1605, 2017.

Li, G., Muller, M., Thabet, A., and Ghanem, B. Deepcns: Can gcns go as deep as cnns? In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9267–9276, 2019.

Li, Y. and Gupta, A. Beyond grids: Learning graph representations for visual recognition. In Advances in Neural Information Processing Systems (NeurIPS), pp. 9225–9235, 2018.
Liu, Y., Fan, B., Xiang, S., and Pan, C. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8895–8904, 2019.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., et al. Attention u-net: Learning where to look for the pancreas. *1st Conference on Medical Imaging with Deep Learning (MIDL)*, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

Pham, H. and Le, Q. V. Autodropout: Learning dropout patterns to regularize deep networks. 2021. URL https://arxiv.org/abs/2101.01761.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10428–10436, 2020.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and 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.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

Tang, Y., Wang, Y., Xu, Y., Shi, B., Xu, C., Xu, C., and Xu, C. Beyond dropout: Feature map distortion to regularize deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bergler, C. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 648–656, 2015.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Wager, S., Wang, S., and Liang, P. S. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 351–359, 2013.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015.
Xiang, T., Zhang, C., Liu, D., Song, Y., Huang, H., and Cai, W. Bio-net: Learning recurrent bi-directional connections for encoder-decoder architecture. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 74–84. Springer, 2020.

Xiang, T., Zhang, C., Song, Y., Yu, J., and Cai, W. Walk in the cloud: Learning curves for point clouds shape analysis. *arXiv preprint arXiv:2105.01288*, 2021.

Yan, X., Zheng, C., Li, Z., Wang, S., and Cui, S. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5589–5598, 2020.

Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., and Wang, C. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8697–8710, 2018.
A. Extensive Experiments on Scheduler

As mentioned in (Ghiasi et al., 2018; Zoph et al., 2018), increasing $\rho$ while training proceeds helps network understanding at the initial learning stage. We here provide an additional study on 5 different $\rho$ scheduling functions that are shown in Figure 3 top. According to the quantitative results reported in Figure 3 bottom, using $f_2$ leads to the poorest results. This is because $f_2$ applies the weakest regularization effect as it is the only scheduler that yields the lowest distortion probabilities at all training steps. On the other hand, DropGraph achieves the best top-1 accuracy with scheduler $f_5$ and the best top-5 accuracy with the linear scheduler. Note that with the same linear scheduler, DropGraph already outperforms DropBlock and Disout by a considerable margin.

![Graph showing the distortion probabilities for different schedulers](image)

**Figure 3:** Top Five candidate schedulers that adjust the distortion probability from 0 to $\rho$. Bottom: ImageNet validation results on the five candidate schedulers.

| Scheduler | Top-1 (%) | Top-5 (%) |
|-----------|-----------|-----------|
| $f_1$     | 78.40     | 94.07     |
| $f_2$     | 77.36     | 93.58     |
| $f_3$     | 77.94     | 93.85     |
| $f_4$     | 77.86     | 93.83     |
| $f_5$     | **78.43** | 94.05     |

B. Experimental Details

B.1. Detailed Settings for Image Classification

**CIFAR.** We used the open-source PyTorch implementation for all experiments on the CIFAR datasets. SGD with weight decay of 0.0005 and momentum of 0.9 was used as the optimizer. The batch size was set to 128 for both ResNet-50 and RegNetX-200MF backbones. The learning rate initially began at 0.1 and reduced by a factor of 0.1 at epochs 150 and 250 while all models were trained for 300 epochs from scratch. Standard augmentation techniques were adopted during training, including random cropping of a 32 × 32 sample from the 4-pixel padded images and random horizontal flipping with a probability of 0.5. All images were normalized by their mean and standard deviation in the pre-processing step.

**ImageNet.** For fair comparisons, we borrowed the implementations provided by (Tang et al., 2020)³ and slightly change the total training epochs and optimizer scheduler to align with (Ghiasi et al., 2018). Specifically, we used SGD optimizer with weight decay of 0.0001 and momentum of 0.9 for optimizations. The batch size was set to 1024 for all experiments that were equally distributed on 8 V100 GPUs. All models were trained for 270 epochs starting with an initial learning rate of 0.1 and reduced by 0.1 at epoch 125, 200 and 250 without learning rate warming up. During training, input images are augmented by cropping into 224 × 224 patches and randomly horizontal flip. During validation, images are first scaled into 256 × 256 and then center cropped into 224 × 224 before being fed into the networks. The above training configurations are consistent to the ones used in (He et al., 2016; Ghiasi et al., 2018; Tang et al., 2020).

B.2. Detailed Settings for Semantic Segmentation

**Pascal VOC 2012.** The experiments were conducted in a public framework. We used the Adam optimizer (Kingma & Ba, 2014) with 0.0001 weight decay to minimize the cross entropy loss. Learning rate starts from 0.001 and cosinanealing scheduled to $1e^{-5}$ in 400 epochs. The batch size was set to 64 for both backbones. Training images are randomly horizontal flipped, randomly scaled by a factor within [0.5, 1.5] and then center cropped leaving with 224 × 224 patches to fit the ResNet-50 backbone. Note that all backbone networks were trained from scratch rather than being pre-trained on ImageNet as suggested in (Ghiasi et al., 2018).

**MoNuSeg.** Implementations from (Xiang et al., 2020)⁵ were adopted for this experiment. Cross entropy loss was minimized by the Adam optimizer with an initial learning rate of 0.01 and a decay rate of 0.00003 at per step. The training dataset was augmented through random rotation (with the angles from [-15, +15]), random x-y shifting (with the angles from [-5%, 5%]), random shearing, random zooming (within [0, 0.2]), and random flipping (both horizontally and vertically). The batch size was set to 2.

³https://github.com/huawei-noah/Disout
⁴https://github.com/warmspringwinds/pytorch-segmentation-detection
⁵https://github.com/tiangexiang/BiO-Net
Figure 4: **Left:** Accuracy curves. **Right:** Loss curves. Results at the last 70 epochs are zoomed in.

Figure 5: **Deployment position of DropGraph on ResNet and U-Net.**

**B.3. Detailed Settings for Point Cloud Analysis**

We used the open-source implementation \(^6\) for the point cloud classification experiments. We set the height and width of the 2D CNN feature maps of DropGraph and Disout to be equal to the number of points. SGD optimizer with weight decay of 0.0001 and momentum of 0.9 were used for the optimization. We utilized the cosine annealing scheduler to adjust the learning rate from 0.1 to 0.0001 in 250 epochs. The batch size was set to 32 for training and 16 for validation. Raw point clouds were first normalized into unit spheres followed by random scaling with a multiplier within [0.66, 1.5], random translation along the three directions by displacements within [-0.2, 0.2]. The number of neighbors in KNN was set to 20 for all experiments.

**B.4. Detailed Settings for Graph Recognition**

**Cora.** The official GCN implementation \(^7\) was used for experiments on the Cora dataset. Adam optimizer with learning rate of 0.01 and weight decay of 0.0005 was used for optimization. For each run, we trained the models for 200 epochs without batch processing.

**Protein.** We used the open-source implementation \(^8\) for experiments on the Protein dataset with the same training and validation split suggested in (Zhang et al., 2019). The optimizer and network were set identically to the ones used for Cora experiments, except that we placed an additional linear layer at the end of the network. Learning rate was set to 0.02 for training on the entire dataset for 200 epochs.

**Open Graph Benchmark.** We used the exact official settings \(^9\), including dataloaders, network implementations, and optimizations without any modifications.

**C. ImageNet Training Curves**

Here we compare the accuracy and loss curves between DropGraph and Disout during ImageNet training in Figure 4. Validation and training statistics are plotted in solid lines and dotted lines respectively, with red denote our DropGraph and blue Disout.

An effective regularizer is able to alleviate over-fitting on the training set and improve the generalization ability on the validation set. According to Figure 4 left, DropGraph yields generally lower training accuracy and higher validation accuracy. As shown in Figure 4 right, DropGraph yields higher training loss with lower validation loss. Therefore, our DropGraph demonstrates stronger regularization effects than Disout.

\(^6\)https://github.com/WangYueFt/dgcnn
\(^7\)https://github.com/tkipf/pygcn
\(^8\)https://github.com/cszhangzhen/HGP-SL
\(^9\)https://github.com/snap-stanford/ogb
Partial Graph Reasoning for Neural Network Regularization

Noteworthy, without modifying the backbone network, all existing regularization methods yield very similar learning curves. Our proposed method is the very first learnable regularizer in the community. Without any advanced designs, the most basic DropGraph implementation already achieves on par and even better results than all existing state-of-the-arts.

D. Deployment Positions of DropGraph

In Figure 5, we show where to apply DropGraph on ResNet and U-Net style networks. As mentioned in Sec. 3.2 and Sec. 4, in general, we apply DropGraph after each activation layer. Note that we also apply distortions to skip features in all networks with residual connections.

E. Visualizations

E.1. Grad-CAM Visualizations

Additional to the quantitative experiments conducted in Sec. 4.5, we visualize the Grad-CAM heatmaps here for more intuitive demonstrations. The gradients flowed through last convolutional layer are collected for visualization. Figure 6 compares the heatmaps on two input images with different regularizers. Without regularization, backbone network can be easily distracted and is therefore hard to focus on the primary object. The same problem still exists when employing DropBlock and Disout in the backbone network. However, our DropGraph greatly alleviates such distraction and forces the backbone network to apply more attention to the major object features.

E.2. t-SNE of Feature Representations

Regularization methods enable the backbone network to better discriminate input samples. We here adopt t-SNE (Van der Maaten & Hinton, 2008) to visualize the feature representations in CIFAR 10 validation set in Figure 7. The representations are collected before the last linear layer of the ResNet-50 model. Compared to DropBlock and Disout, the features generated by DropGraph regularized network have less false predictions and more centered clustered based on their GT labels. Hence, it is validated that our method can achieve better feature representation via a more explainable manner.

E.3. Qualitative Semantic Segmentation Results

We present several qualitative segmentation results in Figure 8 to intuitively demonstrate the impacts of different regularization methods to the backbone networks. With the network architecture unchanged and identical inference behaviour, DropGraph regularized network infers better segmentation masks that are closer to the references.
Figure 8: Qualitative results on semantic segmentation.