Supplemental Materials

A. KNN Graph Experiments

In this section, we present results for experiments on CIFAR-10 that use a KNN graph structure, as discussed in Section 1, rather than the fully-connected bipartite graph structure we propose. An important benefit of the CNN2Transformer and CNN2GNN methods is that learning can be done in an end-to-end fashion even though there is an intermediate graph construction step which is usually a non-differentiable process, but is made differentiable in our approach as described in Section 1. In these experiments, we explore whether end-to-end learning is beneficial in our setting.

We use a ResNet34 (without the classification layer) pretrained on ImageNet as a feature extractor and store these features. We then compute a KNN graph in the following way:

$$\mathbf{X}_{emb} = \boldsymbol{\phi}(\mathbf{X}) \in \mathbb{R}^{n \times \mathcal{F}} \quad (18)$$

$$\mathbf{D} = \mathbf{X}_{emb}\mathbf{X}_{emb}^{T} \in \mathbb{R}^{n \times n}, \quad (19)$$

where $\mathbf{X}$ is our original training data, $\mathbf{X}_{emb}$ is the training data after feature extraction, $\mathcal{F}$ is the embedding dimension, $\boldsymbol{\phi}$ is the pretrained ResNet34, and $\mathbf{D}$ is the similarity matrix where entry $\mathbf{D}_{ij}$ gives the similarity between examples i and j. We use $\mathbf{D}$ to store the k-nearest neighbors for each example in an adjacency list from which we construct a KNN graph, $\mathbf{G}$. We experiment with several choices of k as shown in Table 7. We also test both one-hop and two-hop neighborhoods. Note that in the Neighborhood Size column in Table 7, $[15, 5]$ means that we sample 15 one-hop neighbors and 5 two-hop neighbors. We experiment with two models for the task of node classification, GraphSAGE [3] and GAT [5]. For GraphSAGE, we use the maxpooling aggregation scheme. For GAT, we use multihead attention with 8 heads. During inference, we extract the features of the new image, and place it into $\mathbf{G}$ by finding its k-nearest training examples. We then do a forward pass through the model to predict the label of the new image.

Table 7 shows the results of our experiments. We find that end-to-end training is valuable as both CNN2Transformer and CNN2GNN (results shown in Table 2) significantly outperform methods without end-to-end learning while using the same backbone network. We find it interesting that in the KNN graph setting, the maxpooling aggregation method from GraphSAGE outperforms the attention based aggregation in GAT, but both methods in the KNN graph setting are outperformed by our end-to-end method that uses maxpooling (results shown in Table 1). This leads us to believe that the benefit of end-to-end learning in the graph setting is more important than the choice of aggregation function. We point to the fact that attention-based aggregation in CNN2Transformer and CNN2GNN outperforms all other methods, but attention-based aggregation without end-to-end learning performs the worst for all experiments on CIFAR-10 with a pretrained ResNet34 backbone. We hypothesize that this is partly because the refinement of image features by the backbone network enhances the ability of the attention mechanism to focus on particular neighborhood examples. However, we acknowledge that the use of proxies and anchors also play a role in the performance of CNN2Transformer and CNN2GNN. We offer the following inequality as an intuitive summary for the benefits of end-to-end learning and attention in our experiments:

$$\begin{aligned} &\text{end-to-end learning with attention} > \\ &\text{end-to-end learning without attention} > \\ &\text{non-end-to-end learning without attention} \geq \\ &\text{non-end-to-end learning with attention} \end{aligned} \quad (20)$$

Model	k	Neighborhood Size	Accuracy
GraphSAGE [3]	5	[5]	88.96
	10	[10]	88.98
	15	[15]	88.51
	15	[15,5]	88.56
GAT [5]	5	[5]	88.12
	10	[10]	87.91
	15	[15]	87.71
	15	[15,5]	86.19

Table 7: Results for GAT and GraphSage models on CIFAR-10. The experiments vary in how many closest neighbors are used to construct the graph from the initial image features and how to do neighborhood sampling (*i.e.* one or two hop neighborhoods).

B. Relation to KNN

As $|\mathbf{L}| \rightarrow n$, where $\mathbf{L}$ is the set of anchors and n is the number of training examples, our approach mirrors an adapted version of the k-Nearest Neighbors (KNN) algorithm. In the KNN algorithm, new points are classified by computing the distance to each point in the training set and then using a majority vote of the k closest points. In our setting, rather than having each anchor vote directly with its class, each anchor votes with its feature representation. Each feature representation

is weighted by its corresponding attention score which is the analog of the distance metric in KNN. Then this feature representation is used as one information component in determining the final class along with the weighted proxy feature, and the image feature itself. By choosing $|\mathbf{L}| \ll n$, we greatly reduce the computational complexity of our method in comparison to KNN. Rather than use the entire training set as anchors, we use an embedding function, $\boldsymbol{\phi}$, to update the representation of the selected anchors such that they act as good representatives for all other possible anchors of their class.

C. Varying Loss Function and Aggregation Scheme

In this section, we present results on experiments that vary the loss function and aggregation scheme. The experiments are on CIFAR-10 for CNN2Transformer with a ResNet34 backbone pretrained on ImageNet. The overall loss function presented in Equation 17 is a summation of multiple terms which serve different purposes. Also, after the forward propagation of images, anchors, and proxies, our original method aggregates these three features to produce a final representation for each image. The details of the loss function can be found in Section 3.4 and details of the CNN2Transformer aggregation can be found in Section 3.3. Table 8 shows results for six experiments that vary different components of the loss function and aggregation scheme. All experiments outperform the baseline (shown in Table 2). Each row in Table 2 represents shows an experiment which omits certain loss terms for backpropagation or omits certain representations during aggregation.

Experiments 1 and 2 are the same as those presented in Table 2 and Table 3 respectively. These experiments use every term in the loss function but vary which features are used to produce the final representation for each image before the classification layer. We see that these experiments outperform all other tested variants. Surprisingly, we find that excluding image features during aggregation yields the best results.

Experiment 3 removes the $\mathcal{L}_{ce}(\mathbf{P})$ term from the loss function. This means that although proxies are assigned a class label, they are not subjected to a hard classification constraint. The result is that accuracy decreases compared to Experiment 1 where proxies are classified.

Experiment 4 also excludes the $\mathcal{L}_{ce}(\mathbf{P})$ term from the loss function along with the $\mathcal{L}_{ap}$ term. This means that proxies are not classified and there is also no triplet loss computed between the anchors and proxies. This experiment yields lower accuracy than Experiment 3 meaning that removing the $\mathcal{L}_{ce}(\mathbf{P})$ and $\mathcal{L}_{ap}$ terms hurts overall performance.

Experiment 5 excludes the $\mathcal{L}_{ce}(\mathbf{P})$ and $\mathcal{L}_{ap}$ terms from the loss function and also uses just the proxy representations during the aggregation step. This means that proxy features are the only piece of information used to represent each image before classification. This experiment performs worse than including the anchor and image features during aggregation.

Experiment 6 excludes the $\mathcal{L}_{ce}(\mathbf{P})$ and $\mathcal{L}_{ap}$ terms from the loss function and this time uses just the anchor representations during the aggregation step. This experiment performs better than using just the proxy representations during the aggregation step.

Although we do not run every permutation of loss terms and aggregation schemes, overall we find that using all of the loss terms is most beneficial for our method. We mostly experiment with altering the loss terms related to the proxies and find that a a hard classification of the proxies is beneficial for downstream classification. Interestingly, we also find that including image features during aggregation is not necessary for good results and that in general, using either or both of the anchors and proxies during aggregation performs reasonably well (above the baseline).

D. Training Models from Scratch

In this section, we show results on models trained from scratch on the CIFAR-10 and CIFAR-100 datasets [46] using a ResNet32 backbone [21]. The training procedure is similar to that described in Section 4.1. The changes are that we do not resize the images to be compatible with networks pretrained on ImageNet and we only use Random Crop for data augmentation. Table 9 shows our results.

CNN2Transformer continues to outperform baseline experiments for both CIFAR-10 and CIFAR-100. Note that the CNN2Transformer (NoImgAgg) model mirrors the setting of experiments in Table 3 where image features are not aggregated with anchors and proxies prior to classification. This is equivalent to altering Equation 7 to $\mathbf{z}_{out} = \boldsymbol{\omega}\left(\mathbf{L}_{mha}, \mathbf{P}_{mha}\right)$.

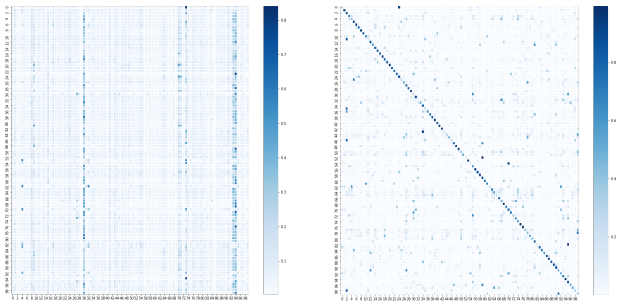
CNN2GNN performs very well on CIFAR-10 but achieves significantly lower results on CIFAR-100. In the pretrained experiments shown in Table 2, CNN2GNN performs reasonably well, although still significantly below the baseline. However when

Experiment	$\mathcal{L}_{at}$	$\mathcal{L}_{pt}$	$\mathcal{L}_{ap}$	$\mathcal{L}_{p}$	$\mathcal{L}_{ce}(\mathbf{X})$	$\mathcal{L}_{ce}(\mathbf{L})$	$\mathcal{L}_{ce}(\mathbf{P})$	$\mathbf{X}_{emb}$	$\mathbf{L}_{mha}$	$\mathbf{P}_{mha}$	Accuracy
1	x	x	x	x	x	x	x	x	x	x	96.73
2	x	x	x	x	x	x	x		x	x	**96.82**
3	x	x	x	x	x	x		x	x	x	96.31
4	x	x		x	x	x		x	x	x	96.16
5	x	x		x	x	x				x	95.99
6	x	x		x	x	x			x		96.11

Table 8: Experiments on CIFAR-10 for CNN2Transformer with a ResNet34 backbone pretrained on Imagenet. Each row is an experiment that shows which loss terms were used for the run and which features were used during the aggregation step.

Model	CIFAR-10	CIFAR-100
Baseline (ResNet32)	87.49	60.01
CNN2Transformer	90.31	61.15
CNN2Transformer (NoImgAgg)	89.82	60.34
CNN2GNN	91.59	12.33

Table 9: Models trained from scratch with a ResNet32 backbone.



(a) CNN2Transformer Epoch 5

(b) CNN2Transformer Epoch 100

(c) CNN2GNN Epoch 5

(d) CNN2GNN Epoch 100

Figure 7: Attention matrices for CIFAR-100 validation examples with a ResNet32 backbone trained from scratch where entry (i, j) shows the normalized attention scores between images with label i and proxies with label j. Just as in the pretrained experiments, CNN2GNN suffers from proxy collapse. CNN2Transformer has collapsed proxies early in training but learns to recover from this through training.

training from scratch, CNN2GNN appears to get stuck during training as we find both training and validation accuracies are low. This is in part due to severe proxy collapse which can be seen in Figure 7. We hypothesize that CNN2GNN performs much better with a pretrained network because the image features are much better at the beginning of training. However for a network trained from scratch, both image and anchor features are not suited for classification at the beginning of training. Thus the model gets stuck in trying to jointly optimize the images, anchors, and proxies. Interestingly, we find that while both CNN2Transformer and CNN2GNN have collapsed attention on proxies at the beginning of training, as shown in Figure 7, CNN2Transformer is able to recover from this while CNN2GNN is not. This provides further evidence that the CNN2Transformer attention mechanism is well suited to datasets with both a small and large number of classes while the CNN2GNN attention mechanism struggles on datasets with a large number of classes.