CNN2Graph: Building Graphs for Image Classification

Vivek Trivedy, Longin Jan Latecki
Department of Computer and Information Sciences, Temple University, Philadelphia, USA
{vivektrivedy, latecki}@temple.edu

Abstract

Neural Network classifiers generally operate via the i.i.d. assumption where examples are passed through independently during training. We propose CNN2GNN and CNN2Transformer which instead leverage inter-example information for classification. We use Graph Neural Networks (GNNs) to build a latent space bipartite graph and compute cross-attention scores between input images and a proxy set. Our approach addresses several challenges of existing methods. Firstly, it is end-to-end differentiable despite the generally discrete nature of graph construction. Secondly, it allows inductive inference at no extra cost. Thirdly, it presents a simple method to construct graphs from arbitrary datasets that captures both example level and class level information. Finally, it addresses the proxy collapse problem by combining contrastive and cross-entropy losses rather than separate clustering algorithms. Our results increase classification performance over baseline experiments and outperform other methods. We also conduct an empirical investigation showing that Transformer style attention scales better than GAT attention with dataset size.

1. Introduction

Traditionally in image classification, examples are independently passed through a CNN model which is a stack of convolutional and pooling layers followed by fully connected classification layers. The convolutional and pooling layers combine information from local parts of each image with the goal of producing a global vector representation in the fully connected layers. Recent Vision Transformer based methods [1, 2] replace the convolution operation with a self-attention module that operates over patches of an image and which also computes a per-image global representation. Implicit in this paradigm is that each image is processed independently of all other images and thus although CNNs and Vision Transformers learn from each image individually, they do not explicitly capture relationships between images in the global example space. In particular, there is a difference between constructing an intra-image global representation that combines local information of one image and constructing an inter-image global representation which combines information from multiple images. To allow for the latter, we leverage GNNs which provide a natural framework for describing relationships.

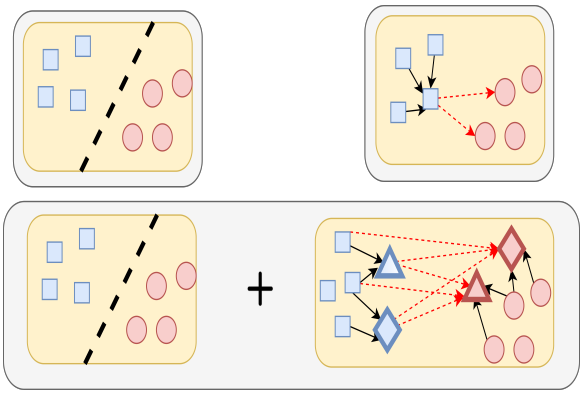


Figure 1: Diagram of cross-entropy loss (top-left), contrastive loss (top-right), and our combined loss (bottom). We use standard cross-entropy loss along with an adapted contrastive loss where we compute losses between training examples and a set of proxies and anchors which are uniformly distributed by class.

The two primary operations in GNNs are propagation and aggregation. The propagation step updates each node's feature representations via a weight matrix, $\mathbf{W}$, and the aggregation step applies a permutation invariant function (*e.g.* sum, max) over each node's neighborhood. Aggregation allows GNNs to learn the relationships between examples through neighborhood information routing. Early GNN variants such as GraphSAGE [3] and Graph Convolutional Network (GCN) [4] use isotropic neighborhood aggregation. Graph Attention Networks (GATs) [5, 6] extend aggregation by introducing an attention computation in the model which allows nodes to preferentially weight the contributions of neighbors. This is particularly important in low-homophily graphs where the majority of a node's neighbors are of a different class [7].

A drawback of many GNN methods is the inability to do inductive inference. Many methods operate only in a transductive setting [4, 8, 9] where test nodes are present in the graph during training. Inductive inference can be especially challenging when the user must define the graph as the graph

generation processes during training and testing may be misaligned, leading to poor generalization. A common scheme for building graphs from arbitrary data is to use a k-nearest neighbors (KNN) graph [10–12] where nodes are connected via a distance metric defined over the initial feature space. There are three main drawbacks to using this method. Firstly, the structure of the graph is determined via some initial feature representation which may be a poor prior for the downstream task. Secondly, this process is non-differentiable due to the nature of the KNN selection rule [13, 14] which disconnects the model that learns the initial features from the model that learns on the graph. Thirdly, inductive inference does not scale because it necessitates a computation of order $O(n \cdot t)$, where n is the number of training nodes and t is the number of test nodes, to place new nodes into the graph before a forward pass. We address these drawbacks by constructing a complete bipartite graph between training examples and a fixed proxy set containing learnable proxy vectors and anchor examples which are chosen uniformly over each class in the training data. Thus the graph is not built based on some initial feature representation, but is instead constructed via class information, directly aligned to our downstream task. Our model is end-to-end differentiable as the graph connectivity circumvents the non-differentiability of the KNN selection rule. Finally, node insertion for inference is an $O(1)$ operation because each test example is simply connected to each element in the proxy set, mirroring the training setting.

We also explore the relationship between GNNs and Transformers as described in [15]. The relationship between GNNs and Transformers is relevant in our setting because the graph structure we propose is complete and bipartite like the self-attention mechanism of Transformers [16]. We discuss these details in Section 3.3. Our main contributions are the following:

- We propose a simple framework for building graphs from arbitrary image data which leverages inter-example information. It improves on existing methods by allowing end-to-end learning and inductive inference and clearly improves classification accuracy by adding a simple module to a backbone CNN.
- We use proxies that contrastively learn class level global information and are also directly incorporated into feature representations for classification. We show that a simple combination of contrastive and cross-entropy losses can prevent proxy collapse (when learnable proxies are ignored by the model).
- We conduct an empiricial investigation of two styles of attention and show the better scalability of Transformer attention in comparison to GAT attention.

2. Related Work

GNNs Many GNN methods differ primarily in the aggregation function [3–6, 17–19]. Splitting up computation is also a vital part of using GNNs for large graphs. Methods such as GCN [4] originally required the entire graph adjacency matrix during each forward pass. GraphSAGE [3] addresses this problem through neighborhood sampling. Our approach makes neighborhood sampling simple via a complete bipartite graph structure with a fixed number of proxy set nodes, and we use one-hop neighborhoods during forward propagation.

In most GNN settings, a graph structure is given. Misraa *et al.* [10] explore cases where the graph structure is not available and must be constructed using a combination of class based connectivity and KNN based connectivity. Our method constructs the graph at the dataset level rather than the mini-batch level like in [20]. Zhu *et al.* [7] explore the effect of homophily on node classification and highlight how many GNN methods struggle with heterophily. We overcome this through attention and the construction of our loss function.

Image Classification We use the ResNet architecture [21] as the backbone of our model as ResNets have been shown to be competitive with the state of the art for image classification. We replace the fully connected layers used in classic CNN architectures [21–24] with our cross-attention module and apply a linear projection for classification.

Contrastive Loss and Proxies Contrastive methods as described in [25–28] seek to construct an embedding space such that *similar* examples cluster together. This can be done by generating pairs or triplets of examples. Common approaches include using the same image with different augmentations to construct positive pairs [29, 30] or using label information to construct pairs or triplets of objects [26]. Selecting pairs or triplets can be computationally expensive. Solutions to this problem include picking particularly hard [31] or semi-hard negative examples [32] or ignoring negative examples altogether [33]. Methods such as [34–36] introduce proxies or anchors to reduce pair and triplet selection. Again our method operates at the dataset level rather than the mini-batch level in choosing anchors and proxies. Proxies can be

susceptible to collapse. Methods such as [30] handle this by enforcing an equipartition constraint. We show that a combination of contrastive and cross-entropy losses can prevent proxy collapse as described in Section 3.4.

SVMs, Kernel Methods, and Prototypical Networks Support Vector Machines (SVMs) [37] and kernel methods [38, 39] are similar to our approach in that they capture relationships between data points. SVMs use support points which are analogous to our anchors while kernel methods capture similarity scores between all data points via a kernel function making them difficult to scale with more examples. A downside to both methods is that the feature transformation for classification is found by experimenting with a choice of kernel. Neural networks construct a latent space directly optimized for the downstream classification task. We leverage the idea that computing similarities and using support examples is useful, but we instead use neural networks as an embedding function, use attention for similarity computations, and learn the support examples to be directly optimized for classification. Non-parametric approaches such as [40] also learn relationships between data points via attention, but our parametric approach is directly tuned for downstream classification and scales better because attention is applied over a proxy set which represents the broader data distribution rather than over the entire dataset.

We also adapt ideas from few-shot learning through Prototypical Networks [41] which assume a prototype representation for each class can be captured by taking the mean of embedded support set examples. Classification is done by computing distances to the prototypes. We use embedded anchor examples as class representatives, but also include fully learnable proxy vectors which are not constrained by the embedding function. Furthermore, rather than computing distances to prototypes, we use cross-attention and aggregate information between input image embeddings, anchor examples, and proxies before performing a linear classification. This differs from other methods that generally use proxies/prototypes only contrastively rather than incorporating them directly into feature representations.

3. Method

We build a proxy set and add a cross-attention module on top of a backbone CNN model to learn relationships between data points for better image classification. Our approach can be summarized as a series of five steps:

1. Select c anchor images from dataset $\mathcal{D}$ - one per class
2. Initialize c learnable proxy vectors $\mathbf{P} \in \mathbb{R}^{c \times F}$
3. Pass n training images and c fixed anchor images through an encoder CNN, $\mathbf{\Phi}$ - giving train embeddings, $\mathbf{X} \in \mathbb{R}^{n \times F}$, and anchor embeddings, $\mathbf{L} \in \mathbb{R}^{c \times F}$
4. Compute separate cross-attentions: $CA(\mathbf{X}, \mathbf{L}) = \mathbf{L}'$ and $CA(\mathbf{X}, \mathbf{P}) = \mathbf{P}'$. The style of attention differs between CNN2GNN and CNN2Transformer.
5. $\mathbf{X}' = Aggregate(\mathbf{X}, \mathbf{L}', \mathbf{P}')$. Pass $\mathbf{X}'$ through a **linear classification layer**.

3.1. Graph and Proxy Set Construction

The proxy set consists of two types of examples: proxy examples and anchor examples. The proxy examples are learnable parameters that are initialized with a dimension equal to the embedding dimension of a backbone CNN. This allows an attention coefficient to be computed between each training example and proxy. We select c proxies corresponding to c classes in the dataset because the proxies are meant to serve as global class representatives that partition the latent space. Anchors remain fixed throughout the training and inference settings and are sampled uniformly over each class:

$$\mathbf{L} = \{\ell^i \in_U \mathbf{X}^i : \mathbf{X}^i \subseteq \mathbf{X}\},\ i = 1 \dots c \tag{1}$$

where $\mathbf{L}$ is the set of anchors, $\in_U$ is the uniform sampling operation, c is the number of classes in the dataset, $\mathbf{X}$ is the training set, and $\mathbf{X}^i$ is a subset of $\mathbf{X}$ containing elements of class i. We discuss the role of anchors in the loss function in Section 3.4. In all settings, a complete bipartite graph is constructed between a mini-batch of training examples and the proxy set.

3.2. Isotropic Aggregation

We explore isotropic aggregation functions as used in previous GNN methods. A forward propagation step for a mini-batch with meanpooling aggregation is defined as:

$$\mathbf{z_B} = \mathbf{D}^{\frac{-1}{2}}\mathbf{A}\mathbf{D}^{\frac{-1}{2}}\left[\boldsymbol{\phi}(\mathbf{X}) \parallel \mathbf{P} \parallel \boldsymbol{\phi}(\mathbf{L})\right] \tag{2}$$

where $\mathbf{z_B}$ is the updated representation for the images in the mini-batch, $\mathbf{X}$ is the set of input images, $\mathbf{D}^{\frac{-1}{2}}$ is the inverse degree matrix after an element-wise square root, $\mathbf{A}$ is the mini-batch adjacency matrix, $\boldsymbol{\phi}$

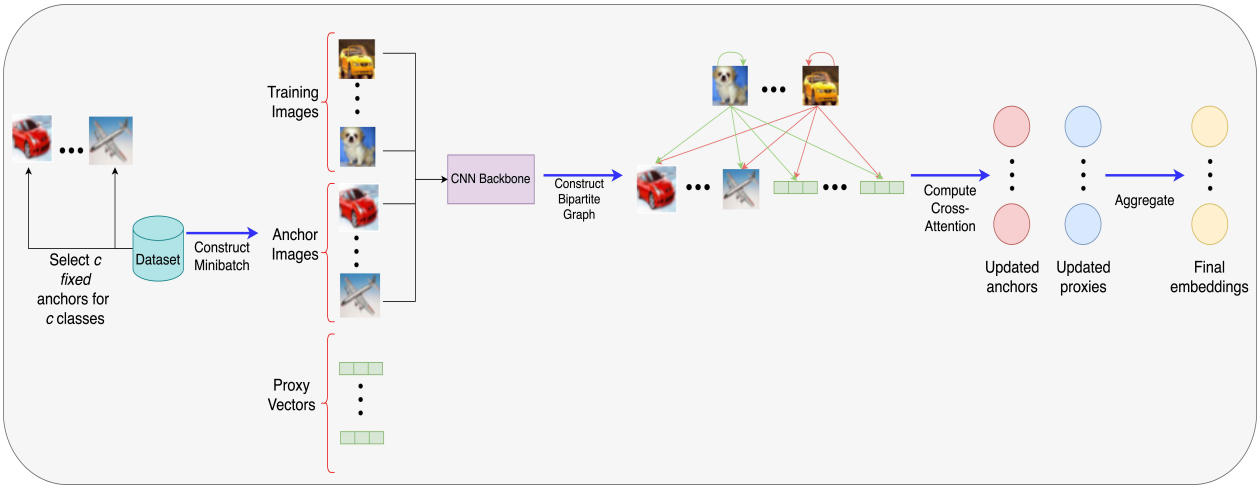


Figure 2: Forward propagation steps in the CNN2GNN with attention.

is a backbone CNN, $\mathbf{P}$ is the set of proxies, $\mathbf{L}$ is the set of anchors, and $\|$ is the concatenation operation.

The maxpooling aggregation is similar to meanpooling, but multiplication by the adjacency matrix is replaced by a feature-wise maximum over the concatenated training node and proxy set representations. The maxpooling aggregation for an input image, $\mathbf{X_i}$ is computed as: $\mathbf{z_i} = max\left[\boldsymbol{\phi}(\mathbf{X_i}) \parallel \mathbf{P} \parallel \boldsymbol{\phi}(\mathbf{L})\right]$.

3.3. Attention Based Aggregation

Anisotropic attention based aggregation allows the model to weight proxy set elements when producing new training example representations. Proxy set elements serve as global class representatives so each input image should most greatly attend to the element corresponding to its class rather than equally over each proxy set element. Furthermore, our artificially constructed graph has high heterophily with an *edge homophily ratio* of $\frac{1}{c}$ as described in [7], where c is the number of classes. Because GNNs can struggle in heterophilic settings [7, 42] anisotropic aggregation allows for a more refined neighborhood context.

We experiment with two approaches for applying attention.

3.3.1 CNN2GNN

In CNN2GNN, the attention computation can be written as follows [6]:

$$\mathbf{e}(\mathbf{h_i}, \mathbf{h_j}) = \mathbf{a}^T LeakyReLU(\mathbf{W}[\mathbf{h_i} \parallel \mathbf{h_j}]) \qquad (3)$$

where $\mathbf{e}(\mathbf{h_i}, \mathbf{h_j})$ is the attention weight between a source image, $\mathbf{h_i}$, and a member of the proxy set, $\mathbf{h_j}$, $\mathbf{a}^T$ is a learnable attention vector, and $\mathbf{W}$ is a learnable weight matrix.

Each training node attends over each proxy set element, producing a complete bipartite graph with optional self-loops as seen in Figure 2. We adapt the implementation from [43] and present the forward propagation method for an input image in Algorithm 1. For brevity of notation, we show the single-headed attention case for a single example.

Algorithm 1: CNN2GNN Forward

Input : image $\mathbf{X_i}$, anchors $\mathbf{L}$, proxies $\mathbf{P}$, backbone CNN $\boldsymbol{\phi}$, adjacency matrix $\mathbf{A}$
Output: Context-aware image embeddings $\mathbf{z}_{out}$

$\mathbf{g} = [\boldsymbol{\phi}(\mathbf{X_i}) \parallel \mathbf{P} \parallel \boldsymbol{\phi}(\mathbf{L})]$
$\mathbf{g}_1, \mathbf{g}_2 = \mathbf{W}_1\mathbf{g}, \mathbf{W}_2\mathbf{g}$
$\mathbf{g}_{sum}[i, j] = \vec{\mathbf{g}}^i + \vec{\mathbf{g}}^j \quad \forall\, \vec{\mathbf{g}}^i, \vec{\mathbf{g}}^j \in \mathbf{g}_1, \mathbf{g}_2$
$\mathbf{e} = \mathbf{a}^T \cdot LeakyReLU\left(\mathbf{g}_{sum}\right)$
$\mathbf{e}_{ij} = -\infty$ if $\mathbf{A}_{ij} == 0$ else $\mathbf{e}_{ij}$
$\boldsymbol{\alpha} = Softmax(\mathbf{e})$
$\mathbf{z}_{out} = \boldsymbol{\alpha} \cdot \mathbf{g}_2$

In Algorithm 1, $\boldsymbol{\phi}$ is a backbone CNN and $\mathbf{W}_1, \mathbf{W}_2$ are learnable weight matrices. Note that in the implementation, we apply Softmax separately over the anchor and proxy indices in $\mathbf{e}$ to ensure that the normalization of the anchor attention weights does not affect the proxy attention weights and vice-versa. Multi-headed attention is used for greater expressivity and stability as described in [5, 16].

3.3.2 CNN2Transformer

In CNN2Transformer, our approach incorporates anchors and proxies as shown in Figure 3. The equations are as follows:

$$\mathbf{X}_{emb}, \mathbf{L}_{emb} = \boldsymbol{\phi}(\mathbf{X}), \boldsymbol{\phi}(\mathbf{L}) \quad (4)$$

$$\mathbf{L}_{mha} = S\left(\frac{\mathbf{W}_q\mathbf{X}_{emb} \cdot \mathbf{W}_{k_1}\mathbf{L}_{emb}}{\sqrt{d}}\right)\mathbf{W}_{v_1}\mathbf{L}_{emb} \quad (5)$$

$$\mathbf{P}_{mha} = S\left(\frac{\mathbf{W}_q\mathbf{X}_{emb} \cdot \mathbf{W}_{k_1}\mathbf{P}}{\sqrt{d}}\right)\mathbf{W}_{v_2}\mathbf{P} \quad (6)$$

$$\mathbf{z}_{out} = \boldsymbol{\omega}\left(\mathbf{X}_{emb}, \mathbf{L}_{mha}, \mathbf{P}_{mha}\right) \quad (7)$$

where $\mathbf{X}$ is the set of input images, $\mathbf{L}$ is the set of anchors, $\mathbf{P}$ is the set of proxies, $\boldsymbol{\phi}$ is the CNN backbone, $\boldsymbol{\omega}$ is an aggregation function (*e.g.* cat, max), S is the softmax function, and d is the embedding dimension of the images. $\mathbf{W}_q$ is the learnable weight for the queries. $\mathbf{W}_{k_1}, \mathbf{W}_{k_2}$ and $\mathbf{W}_{v_1}, \mathbf{W}_{v_2}$ are the learnable key and value matrices respectively. Our approach uses two cross-attention modules where the queries are input images and the keys and values are the anchors or proxies. Thus the output of the attention modules provides two separate weighted sums of the anchors and proxies, $\mathbf{L}_{mha}$ and $\mathbf{P}_{mha}$, which we then aggregate with the original image embeddings to produce a final representation. In this way, the final representation for each image has information from a fixed set of other images uniformly distributed by class (*i.e.* the anchors) and with parameters trained to globally represent each class (*i.e.* the proxies).

3.4. Loss Function and Proxy Collapse

Proxies can collapse without a penalizing term in the loss function or some sort of regularization [30]. Ideally, we want each proxy to cluster with one of the classes, acting as a global class token. We combine two methods to avoid proxy collapse: *classifying proxies* and *using contrastive style losses on proxies*.

Classifying elements of the proxy set is straightforward where each proxy is assigned a class uniformly over the number of classes and passed through a shared classification layer to be incorporated in each mini-batch loss. The anchors are similarly classified. This explicitly pushes each proxy and anchor to be representatives for their class through a hard classification constraint.

We also apply both triplet [25] and contrastive losses [28] to prevent collapse. We first uniformly assign each proxy to a class label. We construct triplets and from training examples $\mathbf{X}$, anchors $\mathbf{L}$, and proxies $\mathbf{P}$. Anchors and proxies are as described in Section 3.1. To avoid confusion in terminology between the anchors we use in our method and the term "anchors" generally used in triplet loss, we will refer to the "anchors" in triplet loss as "sources" and denote them $\mathbf{S}$. The general construction of triplet loss [25] is:

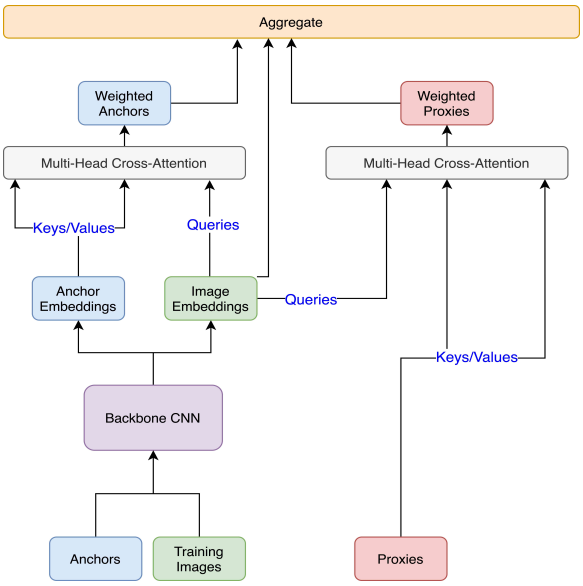


Figure 3: Transformer encoder module that produces new image representations via cross-attention over anchors and proxies.

$$\mathcal{L}_{triplet}(\mathbf{S}, \mathbf{G}, \mathbf{N}) = max\Big(\|f(\mathbf{S}) - f(\mathbf{G})\|_2^2 - \|f(\mathbf{S}) - f(\mathbf{N})\|_2^2 + \alpha, 0\Big) \quad (8)$$

where $\mathbf{S}$ is the set of sources, $\mathbf{G}$ is the set from which positives are sampled, $\mathbf{N}$ is the set from which negatives are sampled, f is an embedding function, $\|\|_2$ is the $L2$ norm, and α is the margin parameter. Note that $\mathbf{S}$, $\mathbf{G}$, and $\mathbf{N}$ are sets not elements. The construction for contrastive loss [28] is:

$$D(\mathbf{X}) = \|f(X_1) - f(X_2)\|_2, \;\; X_1, X_2 \in \mathbf{X} \quad (9)$$

$$\mathcal{L}_{contrastive}(\mathbf{X}) = (1 - Y)\frac{1}{2}(D(\mathbf{X})^2 + (Y)\frac{1}{2}\left(max(0, \alpha - D(\mathbf{X})\right)^2 \quad (10)$$

where X_1, X_2 are instances in a larger set $\mathbf{X}$, f is an embedding function, $\|\|_2$ is the $L2$ norm, α is the margin parameter, and Y is the similarity between X_1 and X_2 (*i.e.* whether or not they are of the same class).

We use a four contrastive style loss terms:

$$\mathcal{L}_{at} = \mathcal{L}_{triplet}(\mathbf{L}, \mathbf{X}, \mathbf{X}), \quad (11)$$
$$\mathcal{L}_{pt} = \mathcal{L}_{triplet}(\mathbf{P}, \mathbf{X}, \mathbf{X}) \quad (12)$$
$$\mathcal{L}_{ap} = \mathcal{L}_{triplet}(\mathbf{L}, \mathbf{P}, \mathbf{P}), \quad (13)$$
$$\mathcal{L}_{p} = \mathcal{L}_{contrastive}(\mathbf{P}) \quad (14)$$

Each term either helps prevent proxy collapse or pushes training examples to attend to the correct proxy set element. See Figure 4 on how the triplet losses affect the latent space. $\mathcal{L}_{at}$ and $\mathcal{L}_{pt}$ partition the latent space such that training images move towards the anchors and proxies corresponding to their class. $\mathcal{L}_{ap}$ provides a consistent support set [44] to the proxies via the anchors. Anchors also act as class representatives but because of their shared embedding with training examples through the backbone CNN, their representation is similar to that of other training examples. The anchors are also classified during each backpropagation step which stabilizes their representation by being repeatedly seen by the model. The proxies are free parameters in the model, and we found that to avoid collapse, they require a stable grounding (provided by the anchors) which balances the stochasticity of $\mathcal{L}_{pt}$ for an arbitrary mini-batch. Finally, $\mathcal{L}_{p}$ enforces a margin between each proxy to explicitly penalize collapse.

Both the contrastive style and cross-entropy losses can be summarized as a sum of individual losses:

$$\mathcal{L}_{total\_contrastive} = \mathcal{L}_{at} + \mathcal{L}_{pt} + \mathcal{L}_{ap} + \mathcal{L}_{p} \quad (15)$$

$$\mathcal{L}_{classification} = \mathcal{L}_{ce}(\mathbf{X}) + \mathcal{L}_{ce}(\mathbf{L}) + \mathcal{L}_{ce}(\mathbf{P}) \quad (16)$$

where $\mathcal{L}_{ce}$ is a standard cross-entropy loss which classifies instances from a set. The total loss can be summarized as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{total\_contrastive} + \mathcal{L}_{classification} \quad (17)$$

We note that our method works out of the box without careful heuristical weighting of loss terms, but that generally it can be difficult to balance many terms. We highlight some points on optimization that we believe helps our method work. Different loss magnitutudes can be problematic during optimization so embeddings are first $L2$ normalized. We also observe that optimization follows roughly three stages. In **Stage 1**, classification losses decrease relatively early in training. In **Stage 2**, classification losses stabilize and contrastive losses continue to decrease as the latent space reorganizes so proxies move to the "correct" classes - seen in UMAP [45] plots over time (Figure 4). In **Stage 3**, contrastive losses converge during which the classification loss also decreases to a minimum as the network is fully optimized. Our method takes roughly 20% longer to train over baselines (which overfit sooner) to manage these losses but consistently converges to a better accuracy as seen in Table 2.

4. Evaluation

4.1. Implementation Details and Datasets

In our experiments we use the standard Imagenet pretrained ResNet18 and ResNet34 models as baselines. To compare, we add our module on top of the ResNets which then serve as the backbone networks. Proxies are initialized via He initialization [50]. All experiments use the SGD optimizer with momentum of 0.9 and learning rate of 3×10^{-4} for 100 epochs. We use a batch size of 256 for all datasets

Backbone	Aggregation	Accuracy
ResNet18	Meanpool	92.64
	Maxpool	92.79
ResNet34	Meanpool	92.82
	Maxpool	92.90

Table 1: Isotropic Aggregation Accuracy CIFAR-10

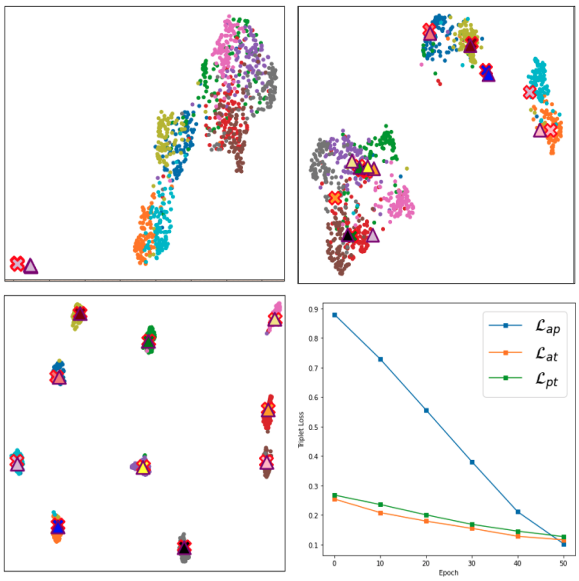
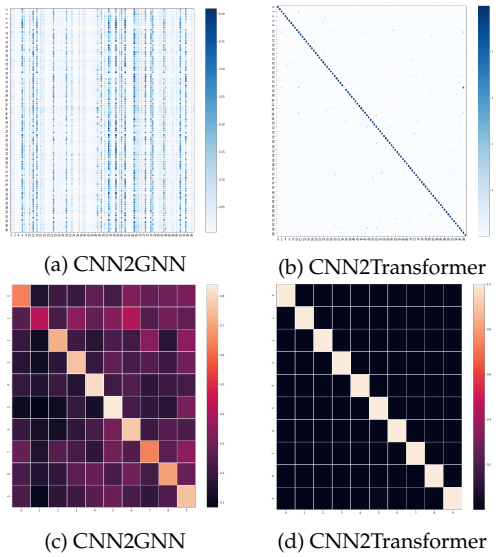


Figure 4: CIFAR-10 image, anchor (X), and proxy (triangles) embeddings at the start (top left), middle (top right), and end (bottom left) of training via UMAP plots with CNN2Transformer (ResNet34). The points are colored by label and the graph (bottom right) shows how the decrease of the contrastive losses corresponds with each anchor and proxy clustering with a class.

Table 2: Attention Models Validation Accuracy

		CIFAR-10 [46]	CIFAR-100 [46]	STL-10 [47]	SVHN [48]	ImageNet-1k [49]
ResNet18	Baseline	94.07	76.95	95.38	95.29	69.42
	CNN2GNN	95.51 ±0.42	74.80 ±0.81	95.70 ±0.20	**96.62** ±0.55	60.12 ±1.02
	CNN2Transformer	**95.79** ±0.24	**77.39** ±0.20	**95.74** ±0.19	96.35 ±0.22	**71.12** ±0.35
ResNet34	Baseline	95.24	79.32	95.89	95.56	73.03
	CNN2GNN	96.39 ±0.41	77.87 ±0.91	96.89 ±0.26	**97.01** ±0.31	61.02 ±0.77
	CNN2Transformer	**96.73** ±0.37	**80.10** ±0.45	**97.21** ±0.19	96.54 ±0.07	**75.42** ±0.15



(a) CNN2GNN (b) CNN2Transformer

(c) CNN2GNN (d) CNN2Transformer

Figure 5: Attention matrices for validation examples with a ResNet34 backbone where entry (i, j) shows the normalized attention scores between images with label i and proxies with label j. Top row is for CIFAR100 and bottom row is for SVHN. For CNN2Transformer, examples attend most greatly to proxies corresponding to their label shown by the diagonal line. CNN2GNN correctly attends to the correct proxy for the SVHN dataset but fails to do so on CIFAR-100 which has more classes as shown in (a).

except CIFAR-100 and ImageNet-1k where we use a batch size of 100. For each run, anchors are selected uniformly at random over the number of classes. For data augmentation, we use Random Crop [51] and color jitter on both the baseline and experimental models. For the anchors, we fix the augmentation within an epoch and apply a new augmentation for each new epoch. We use a linear classification protocol for all experiments. We evaluate on the CIFAR-10 [46], CIFAR-100 [46], STL-10 [47], and SVHN [48], and ImageNet-1k [49] datasets and use the splits given by Torchvision.

4.2. Discussion of Results

Table 1 shows the importance of attention as isotropic aggregation with our graph construction performs worse than baselines. This is consistent with other methods where isotropic aggregation results in lower accuracy for heterophilic graphs.

Table 2 shows that CNN2Transformer consistently outperforms baselines and does especially well on ImageNet-1k. Baselines are pretrained on ImageNet and fine-tuned for other datasets. We hypothesize that CNN2GNN performs worse on the CIFAR-100 and ImageNet-1k datasets because the

Model	CIFAR-10	SVHN	CIFAR100
CNN2GNN	96.21	96.49	79.63
CNN2Transformer	96.82	97.10	81.49

Table 3: Images not aggregated with anchors/proxies (ResNet34).

Model	Accuracy
BYOL (ResNet50) [52]	91.3
SimCLR (ResNet50) [29]	90.5
NNCLR (ResNet50) [44]	93.7
SpinalNet (VGG19_bn) [53]	96.00
ConvMixer-256/8 [54]	96.03
Mixer-S/16-SAM [55]	96.10
DenseNet-BC (k=24) [56]	94.81
CNN2GNN (ResNet34)	**96.39**
CNN2Transformer (ResNet34)	**96.73**

Table 4: Comparisons on CIFAR-10.

Model	Accuracy
ConViT-Ti [57]	73.1
MobileNetV2 [58]	74.7
LocalViT-T [59]	74.8
SimCLR (Resnet 50 2x) [29]	74.2
CMC (Resnet 50 2x) [60]	70.6
DenseNet-121 [56]	74.98
CNN2Transformer (ResNet34)	**75.42**
CNN2GNN (ResNet34)	61.02

Table 5: Comparisons on ImageNet-1k.

Num Proxies/Num Anchors	1	3	5
1	96.73	**96.89**	96.81
3	96.12	96.62	96.33
5	96.18	96.21	96.42

Table 6: CIFAR10 results varying number of anchors and proxies per class for CNN2Transformer (ResNet34). Entry (i, j) is a run with i proxies and j anchors.

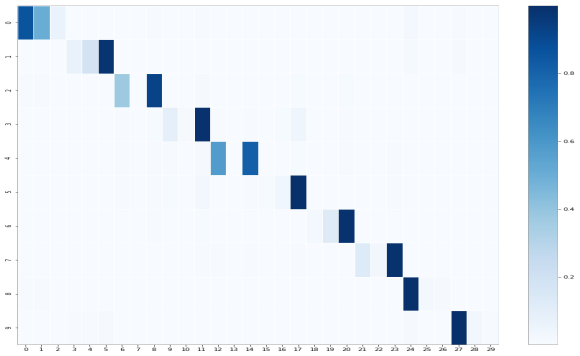


Figure 6: Attention matrix for CIFAR-10 validation examples with CNN2Transformer (ResNet34 backbone) where entry (i, j) shows the normalized attention scores between images with label i and anchors with label j. This run uses 3 anchors and 1 proxy. We find that attention is divided over the anchors.

attention mechanism of GAT is weaker than that of Transformers, particularly for large scale data with many classes. GATv2 [6] mentions that GAT attention [5] can collapses over a few nodes (*i.e.* the attention is not conditioned on the query nodes), and we find that the GATv2 attention also collapses when neighborhood sizes grow as shown in Figure 5. Further results on this finding can be found in the Appendix. Our method is also robust to different anchor choices as shown by the relatively small standard deviations over three runs.

Table 3 shows results of an experiment that tests the quality of the learned anchor and proxy representations. Here we do not aggregate the image representations before doing classification, meaning that the final representation for each image is an attention-weighted sum of anchors and proxies conditioned on the input image. This is equivalent to removing self-connections in the graph for CNN2GNN and altering Equation 7 to $\mathbf{z}_{out} = \boldsymbol{\omega}\left(\mathbf{L}_{mha}, \mathbf{P}_{mha}\right)$ for CNN2Transformer. We find that despite removing information about the image itself, the learned representations of the anchors and proxies outperform baselines.

In Tables 4 and 5, we compare our method to other methods. We compare against a wide range of architectures and model sizes and find that our method performs better than bigger models across CNN and pure Transformer architectures with relatively less data augmentation. We also outperform self-supervised methods such as BYOL, SimCLR, and NNCLR, although we use a smaller ResNet and do not use any self-supervision in pretraining.

In Table 6, we show results for different numbers of anchors and proxies. We find that increasing the number of anchors increases model performance, but increasing the number of proxies decreases model performance. We hypothesize that this is because each anchor introduces new information about its class, while additional proxies do not add new information on the underlying image data distribution. For example, having several anchors for the car class, with each anchor being a car of a different color, trains the model to be invariant to such differences. This idea is validated by Figure 6 which shows that examples divide attention over the anchors rather than collapsing on a single anchor. We also note that the $\mathcal{L}_p$ loss term encourages proxy representations per assigned class to collapse onto each other which also explains why multiple proxies per class decreases classification accuracy.

5. Conclusion

We propose a method for bringing graph structure into image classification that allows for learning local and global representations between data points via a proxy set and cross-attention module. The motivation behind the work is to bring together ideas from kernel methods, graph representation learning, and vision. Our models show clear improvements in classification accuracy and our graph construction framework for arbitrary non-graph data addresses common problems of end-to-end learning and inductive inference. We further show that a simple combination of cross-entropy and contrastive losses is enough to prevent proxy collapse. Finally, we present an empirical study showing that Transformer attention scales better than GAT attention with an increasing number of classes.

For future work, we are interested in adapting novel graph structures, aggregation schemes, and proxy learning schemes for downstream tasks. Our cross-attention over anchors and proxies as independent sets gives a simple recipe to incorporate multiple pieces of information (*i.e.* multimodal data). A current drawback of our approach is that the number of proxies scales linearly with the number of classes which can require significant computational resources. We leave addressing this for future work, perhaps by relaxing the class constraint imposed on proxies.

Acknowledgments

This work was in part supported by the NSF Grant IIS-1814745. The calculations were carried out on HPC resources at Temple University supported in part by the NSF through grant number 1625061 and by the U.S. ARL under contract number W911NF-16-2-0189.

References

[1] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020 (cit. on p. i).

[2] Hugo Touvron et al. *Training data-efficient image transformers & distillation through attention*. 2021. arXiv: 2012.12877 [cs.CV] (cit. on p. i).

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI] (cit. on pp. i, ii, xii).

[4] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG] (cit. on pp. i, ii).

[5] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML] (cit. on pp. i, ii, iv, viii, xii).

[6] Shaked Brody, Uri Alon, and Eran Yahav. *How Attentive are Graph Attention Networks?* 2021. arXiv: 2105.14491 [cs.LG] (cit. on pp. i, ii, iv, viii).

[7] Jiong Zhu et al. *Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs*. 2020. arXiv: 2006.11468 [cs.LG] (cit. on pp. i, ii, iv).

[8] W. Liu and Shih-Fu Chang. "Robust multi-class transductive learning with graphs". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 381–388 (cit. on p. i).

[9] Thorsten Joachims. "Transductive Learning via Spectral Graph Partitioning". In: *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. ICML'03. Washington, DC, USA: AAAI Press, 2003, pp. 290–297. ISBN: 1577351894 (cit. on p. i).

[10] Aashish Kumar Misraa et al. *Multi-Modal Retrieval using Graph Neural Networks*. 2020. arXiv: 2010.01666 [cs.IR] (cit. on p. ii).

[11] Seokho Kang. "k-Nearest Neighbor Learning with Graph Neural Networks". In: *Mathematics* 9.8 (2021). ISSN: 2227-7390. DOI: 10.3390/math9080830. URL: https://www.mdpi.com/2227-7390/9/8/830 (cit. on p. ii).

[12] Luca Franceschi et al. *Learning Discrete Structures for Graph Neural Networks*. 2020. arXiv: 1903.11960 [cs.LG] (cit. on p. ii).

[13] Tobias Plötz and Stefan Roth. "Neural Nearest Neighbors Networks". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/f0e52b27a7a5d6a1a87373dffa53dbe5-Paper.pdf (cit. on p. ii).

[14] Anees Kazi et al. *Differentiable Graph Module (DGM) for Graph Convolutional Networks*. 2020. arXiv: 2002.04999 [cs.LG] (cit. on p. ii).

[15] Chaitanya Joshi. "Transformers are Graph Neural Networks". In: *The Gradient* (2020) (cit. on p. ii).

[16] Ashish Vaswani et al. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964 (cit. on pp. ii, iv).

[17] Felix Wu et al. *Simplifying Graph Convolutional Networks*. 2019. arXiv: 1902.07153 [cs.LG] (cit. on p. ii).

[18] Christopher P. Burgess et al. *MONet: Unsupervised Scene Decomposition and Representation*. 2019. arXiv: 1901.11390 [cs.CV] (cit. on p. ii).

[19] Jiani Zhang et al. "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs". In: *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*. 2018, pp. 339–349 (cit. on p. ii).

[20] Jenny Seidenschwarz, Ismail Elezi, and Laura Leal-Taixé. "Learning Intra-Batch Connections for Deep Metric Learning". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 9410–9421 (cit. on p. ii).

[21] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV] (cit. on pp. ii, xiii).

[22] Geoffrey Hinton Alex Krizhevsky Ilya Sutskever. "ImageNet Classification with Deep Convolutional Neural Networks". In: (2012). URL: https://paperswithcode.com/method/alexnet (cit. on p. ii).

[23] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV] (cit. on p. ii).

[24] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV] (cit. on p. ii).

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015). DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682 (cit. on pp. ii, v).

[26] Prannay Khosla et al. *Supervised Contrastive Learning*. 2021. arXiv: 2004.11362 [cs.LG] (cit. on p. ii).

[27] Kilian Q. Weinberger and Lawrence K. Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification". In: *J. Mach. Learn. Res.* 10 (June 2009), pp. 207–244. ISSN: 1532-4435 (cit. on p. ii).

[28] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press, 2006 (cit. on pp. ii, v).

[29] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG] (cit. on pp. ii, vii).

[30] Mathilde Caron et al. *Unsupervised Learning of Visual Features by Contrasting Cluster Assignments*. 2021. arXiv: 2006.09882 [cs.CV] (cit. on pp. ii, iii, v).

[31] Joshua Robinson et al. *Contrastive Learning with Hard Negative Samples*. 2021. arXiv: 2010.04592 [cs.LG] (cit. on p. ii).

[32] Yannis Kalantidis et al. *Hard Negative Mixing for Contrastive Learning*. 2020. arXiv: 2010.01028 [cs.CV] (cit. on p. ii).

[33] Jean-Bastien Grill et al. *Bootstrap your own latent: A new approach to self-supervised Learning*. 2020. arXiv: 2006.07733 [cs.LG] (cit. on p. ii).

[34] Q. Qian et al. "SoftTriple Loss: Deep Metric Learning Without Triplet Sampling". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2019, pp. 6449–6457. DOI: 10.1109/ICCV.2019.00655. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00655 (cit. on p. ii).

[35] Sungyeon Kim et al. "Proxy Anchor Loss for Deep Metric Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020 (cit. on p. ii).

[36] Yair Movshovitz-Attias et al. *No Fuss Distance Metric Learning using Proxies*. 2017. arXiv: 1703.07464 [cs.CV] (cit. on p. ii).

[37] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 144–152. ISBN: 089791497X. DOI: 10.1145/130385.130401. URL: https://doi.org/10.1145/130385.130401 (cit. on p. iii).

[38] Youngmin Cho and Lawrence Saul. "Kernel Methods for Deep Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Y. Bengio et al. Vol. 22. Curran Associates, Inc., 2009. URL: https://proceedings.neurips.cc/paper/2009/file/5751ec3e9a4feab575962e78e006250d-Paper.pdf (cit. on p. iii).

[39] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. "Kernel methods in machine learning". In: *The Annals of Statistics* 36.3 (June 2008). ISSN: 0090-5364. DOI: 10.1214/009053607000000677. URL: http://dx.doi.org/10.1214/009053607000000677 (cit. on p. iii).

[40] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. iii).

[41] Jake Snell, Kevin Swersky, and Richard S. Zemel. *Prototypical Networks for Few-shot Learning*. 2017. arXiv: 1703.05175 [cs.LG] (cit. on p. iii).

[42] Yao Ma et al. *Is Homophily a Necessity for Graph Neural Networks?* 2021. arXiv: 2106.06134 [cs.LG] (cit. on p. iv).

[43] *Graph attention networks V2 (GATV2)*. URL: https://nn.labml.ai/graphs/gatv2/index.html#section-0 (cit. on p. iv).

[44] Debidatta Dwibedi et al. *With a Little Help from My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations*. 2021. arXiv: 2104.14548 [cs.CV] (cit. on pp. vi, vii).

[45] Leland McInnes et al. "UMAP: Uniform Manifold Approximation and Projection". In: *Journal of Open Source Software* 3.29 (2018), p. 861. DOI: 10.21105/joss.00861. URL: https://doi.org/10.21105/joss.00861 (cit. on p. vi).

[46] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". In: (2009), pp. 32–33. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (cit. on pp. vii, xiii).

[47] Adam Coates, Andrew Ng, and Honglak Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 215–223 (cit. on p. vii).

[48] Yuval Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: 2011 (cit. on p. vii).

[49] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. vii).

[50] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV] (cit. on p. vi).

[51] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. "Data Augmentation Using Random Image Cropping and Patching for Deep CNNs". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.9 (Sept. 2020), pp. 2917–2931. ISSN: 1558-2205. DOI: 10.1109/tcsvt.2019.2935128. URL: http://dx.doi.org/10.1109/TCSVT.2019.2935128 (cit. on p. vii).

[52] Jean-Bastien Grill et al. *Bootstrap your own latent: A new approach to self-supervised Learning*. 2020. arXiv: 2006.07733 [cs.LG] (cit. on p. vii).

[53] H M Dipu Kabir et al. *SpinalNet: Deep Neural Network with Gradual Input*. 2020. arXiv: 2007.03347 [cs.CV] (cit. on p. vii).

[54] Asher Trockman and J. Zico Kolter. *Patches Are All You Need?* 2022. DOI: 10.48550/ARXIV.2201.09792. URL: https://arxiv.org/abs/2201.09792 (cit. on p. vii).

[55] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. "When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations". In: *arXiv preprint arXiv:2106.01548* (2021) (cit. on p. vii).

[56] Gao Huang et al. "Densely Connected Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243 (cit. on p. vii).

[57] Stéphane d'Ascoli et al. *ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases*. 2021. DOI: 10.48550/ARXIV.2103.10697. URL: https://arxiv.org/abs/2103.10697 (cit. on p. vii).

[58] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: (2018). DOI: 10.48550/ARXIV.1801.04381. URL: https://arxiv.org/abs/1801.04381 (cit. on p. vii).

[59] Yawei Li et al. *LocalViT: Bringing Locality to Vision Transformers*. 2021. DOI: 10.48550/ARXIV.2104.05707. URL: https://arxiv.org/abs/2104.05707 (cit. on p. vii).

[60] Yonglong Tian, Dilip Krishnan, and Phillip Isola. *Contrastive Multiview Coding*. 2019. DOI: 10.48550/ARXIV.1906.05849. URL: https://arxiv.org/abs/1906.05849 (cit. on p. vii).