[Submitted on 25 Nov 2016 (v1), last revised 6 Dec 2016 (this version, v3)]

Geometric deep learning on graphs and manifolds using mixture model CNNs

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, Michael M. Bronstein

Deep learning has achieved a remarkable performance breakthrough in several fields, most notably in speech recognition, natural language processing, and computer vision. In particular, convolutional neural network (CNN) architectures currently produce state-of-the-art performance on a variety of image analysis tasks such as object detection and recognition. Most of deep learning research has so far focused on dealing with 1D, 2D, or 3D Euclidean-structured data such as acoustic signals, images, or videos. Recently, there has been an increasing interest in geometric deep learning, attempting to generalize deep learning methods to non-Euclidean structured data such as graphs and manifolds, with a variety of applications from the domains of network analysis, computational social science, or computer graphics. In this paper, we propose a unified framework allowing to generalize CNN architectures to non-Euclidean domains (graphs and manifolds) and learn local, stationary, and compositional task-specific features. We show that various non-Euclidean CNN methods previously proposed in the literature can be considered as particular instances of our framework. We test the proposed method on standard tasks from the realms of image-, graph- and 3D shape analysis and show that it consistently outperforms previous approaches.

Subjects: **Computer Vision and Pattern Recognition (cs.CV)**
Cite as: arXiv:1611.08402 **[cs.CV]**
(or arXiv:1611.08402v3 **[cs.CV]** for this version)
https://doi.org/10.48550/arXiv.1611.08402

A Graph Neural Network for superpixel image classification

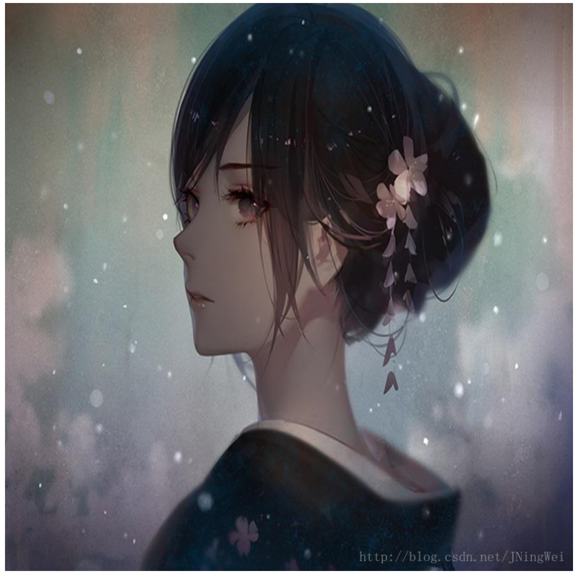
超像素

- 超像素就是把一幅原本是像素级(pixel-level)的图，划分成区域级(district-level)的图。可以将其看做是对基本信息进行的抽象。
- 它利用像素之间特征的相似性将像素分组,用少量的超像素代替大量的像素来表达图片特征,很大程度上降低了图像后处理的复杂度，所以通常作为分割算法的预处理步骤。
- 如果这幅图的大小为480 * 640，那么建立的图(graph)有480 * 640个节点。如果预先对这幅图像使用超像素分割，将其分割为1000个超像素，那么建立的图只有1000个节点。
- 有趣直观并且带有源代码（业界良心）的是SLIC Superpixel，使用K-means的聚类方法，分割的效果很好。

SLIC(simple linear iterativeclustering)

- 它是2010年提出的一种思想简单、实现方便的算法，将彩色图像转化为CIELAB颜色空间和XY坐标下的5维特征向量，然后对5维特征向量构造距离度量标准，对图像像素进行局部聚类的过程。
- 优点
  - 生成的超像素如同细胞一般紧凑整齐，邻域特征比较容易表达。这样基于像素的方法可以比较容易的改造为基于超像素的方法。
  - 不仅可以分割彩色图，也可以兼容分割灰度图。
  - 需要设置的参数非常少，默认情况下只需要设置一个预分割的超像素的数量。
  - 相比其他的超像素分割方法，SLIC在运行速度、生成超像素的紧凑度、轮廓保持方面都比较理想。

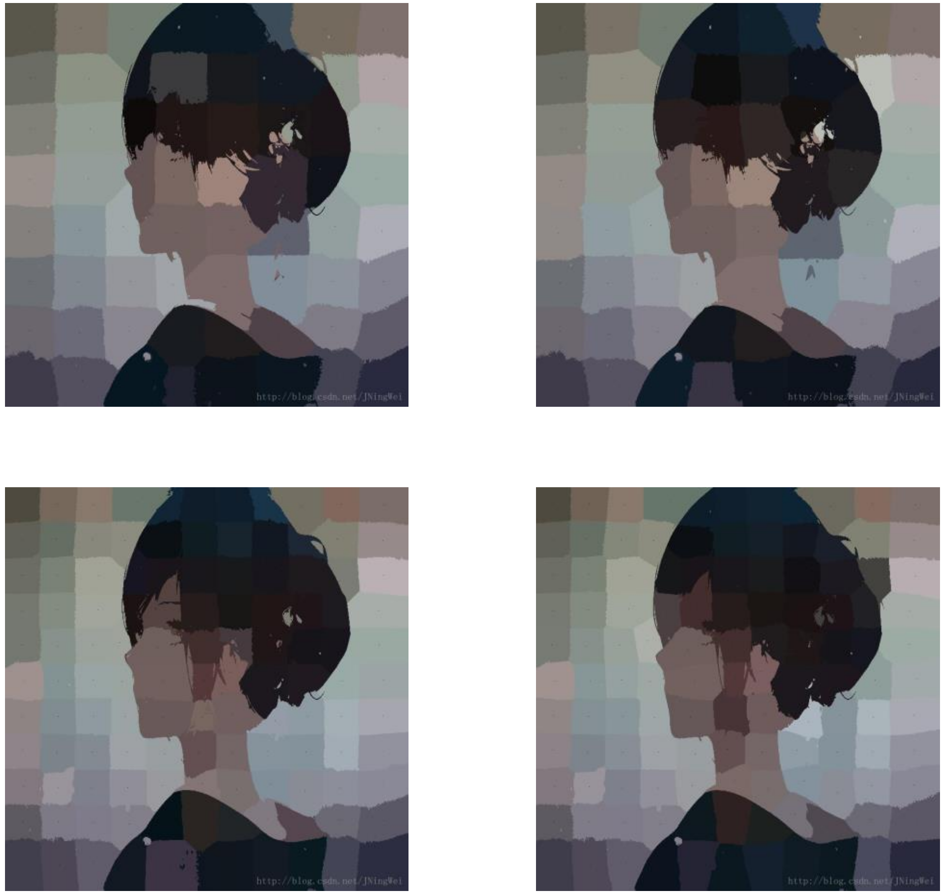
SLIC



Iter=1 Iter=20

K=64

K=128



HGNN

2.1. Superpixel graph

Nowadays, there are many technics that can generate superpixels from images, such as SLIC [9], SEEDS [10] and etc. Like other related works, we use the SLIC to segment superpixels because of its efficiency. After segmentation, we define nodes from superpixels and connect them from K nearest neighbors to generate region adjacency graph (RAG), Figure 1 describes the entire process from the original image to the RAG generation.

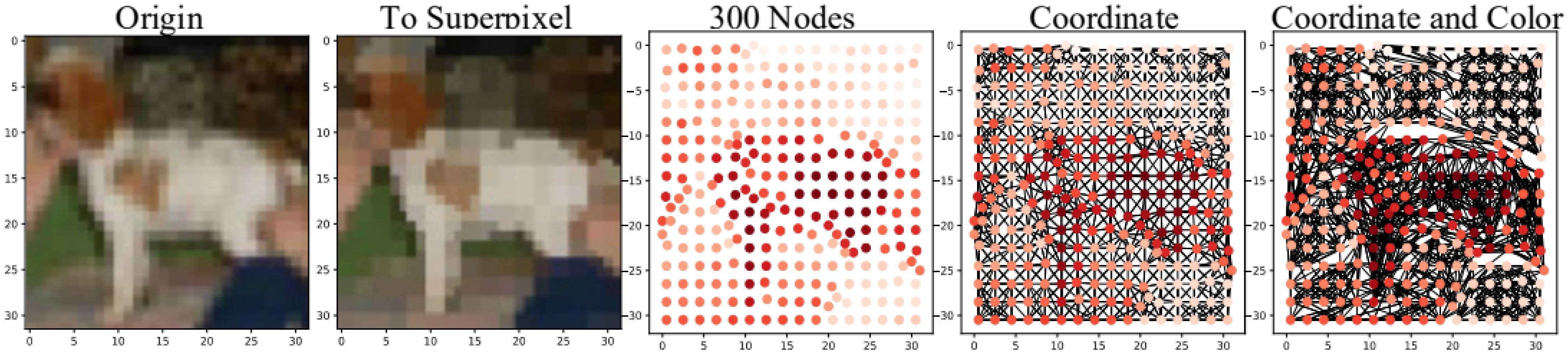


Figure 1. Image to graph.

Node feature

$$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n)^T \quad (1)$$

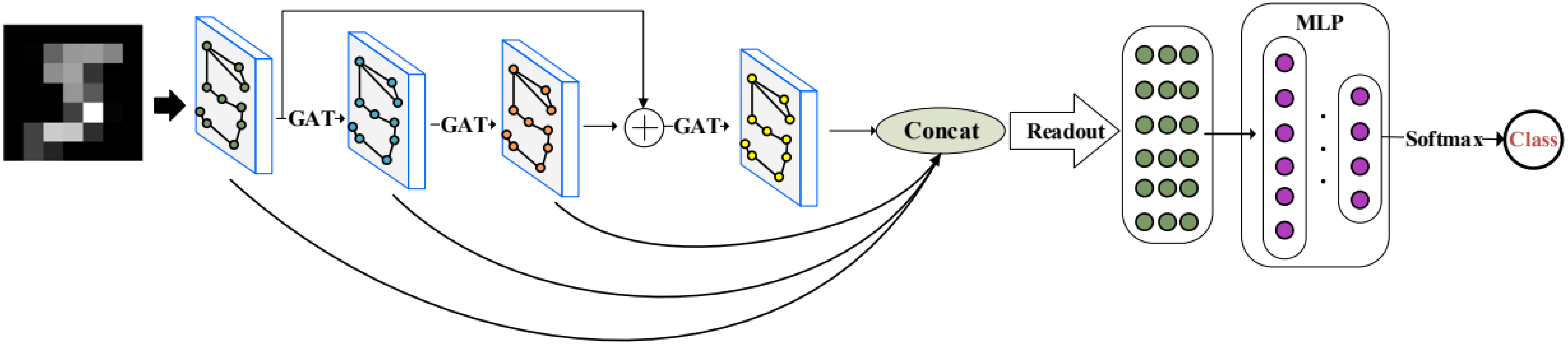
$$\boldsymbol{x}_i = \left[\frac{1}{N_i}\sum_{j=1}^{N_i}(R_j, G_j, B_j, a_j, b_j)\right]^T \quad (2)$$

where $\boldsymbol{x_i}$ is the node feature with superpixel label is i, and N_i is the number of pixels which label is i, a_j and b_j is pixel's position in the image.

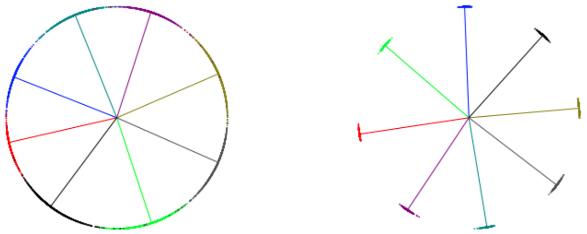
Model Structure

2.2. Hierarchical Graph Neural Network

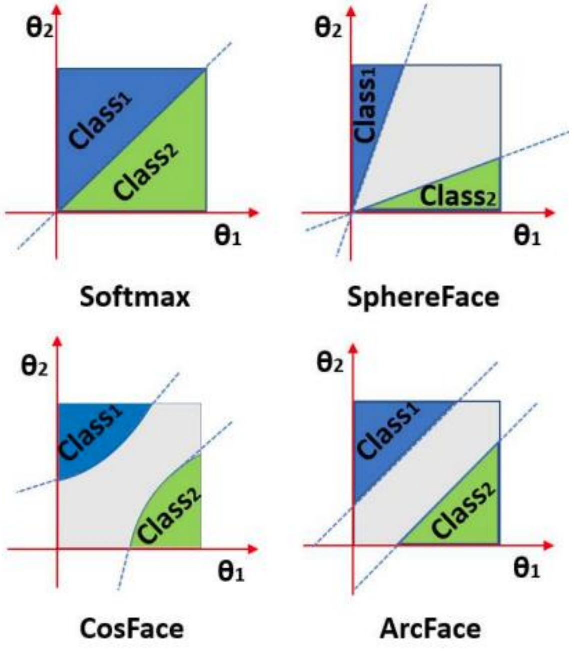
The nodes in graph convolutional neural network usually tend to over-smooth (OS) as the increasing iteration and deeper layers, that is the nodes of the same subgraph have the same values or features. We use two aspects to solve OS. First, residual and concat structure are used for the node graph neural network to alleviate OS situation. Second, GAT is the operater that pays attention to the information of adjacent nodes, which can suppress the over-smooth through aggregating neibourhood features.



ArcFace Loss



(a) Softmax (b) ArcFace



https://zhuanlan.zhihu.com/p/60747096

Arcface: Additive angular margin loss for deep face recognition
J Deng, J Guo, N Xue... - Proceedings of the IEEE ..., 2019 - openaccess.thecvf.com
... (**ArcFace**) to obtain highly discriminative features for face recognition. The proposed **ArcFace** has a ... We show that **ArcFace** consistently outperforms the state of the art and can be easily ...
☆ Save 99 Cite Cited by 4238 Related articles All 18 versions »

2.3. Loss function

The loss function of HGNN is ArcFace loss (Additive Angular Margin Loss, Additive Angular Margin Loss) [11] and cross-entropy (CE) loss. Among them, the CE loss is as the equation (3).

$$L_{CE} = -\sum_{i=1}^{n} p_i \log(q_i) \tag{3}$$

where p_i is ground truth and q_i is prediction.

ArcFace loss used for face recognition at the beginning, which improves the additive angle interval on the basis of Softmax loss (equation (4)) to increases the distance between classes and compactness within classes.

$$L_{Softmax} = -\frac{1}{m}\sum_{i=1}^{m} \log\left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}\right) \tag{4}$$

where $\boldsymbol{W}_j^T \boldsymbol{x}_i + b_j$ is the output of the fully connection (FC). When calculating $L_{softmax}$, as the $\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i + b_{y_i}$ increases, more ith samples can be within the decision boundary. However, $L_{softmax}$ ignores the intra-class and inter-class constraint, so ArcFace loss adds an angle interval t to the angle θ between $\boldsymbol{x}_i$ and $\boldsymbol{W}_{ji}$, and penalizes the angle between the features and weights in an additive manner, so that GNN can learn information more efficiently. $L_{ArcFace}$ such as equation (5).

$$L_{ArcFace} = -\frac{1}{m}\sum_{i=1}^{m} \log\left(\frac{\boxed{e^{s\times(\cos(\theta_{y_i}+t))}}}{e^{s\times(\cos(\theta_{y_i}+t))} + \sum_{j=1, j\neq y_i}^{n} e^{s\times\cos\theta_j}}\right) \tag{5}$$

where θ_j is the angle of the weight $\boldsymbol{W}_j$ and the feature $\boldsymbol{x}_i$, is calculated by $\boldsymbol{W}_j^T \boldsymbol{x}_i = ||\boldsymbol{W}_j||\ ||\boldsymbol{x}_i|| \cos\theta_j$, so is θ_{y_i}.In summary, the final loss function L used in this paper is as follows equation (6):

$$L = L_{CE} + L_{ArcFace} \tag{6}$$

Experiment

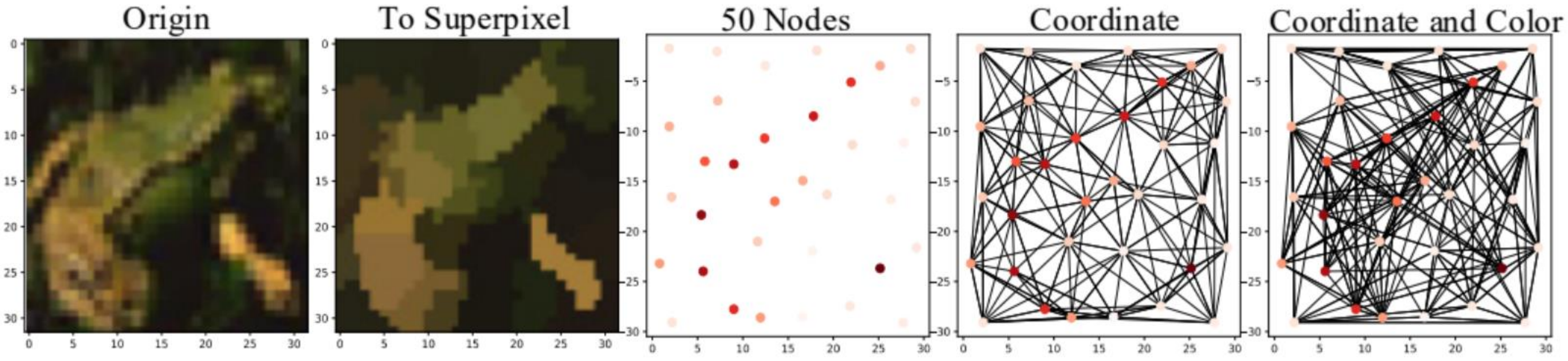


Figure 3. Cifar10 with 50 nodes.

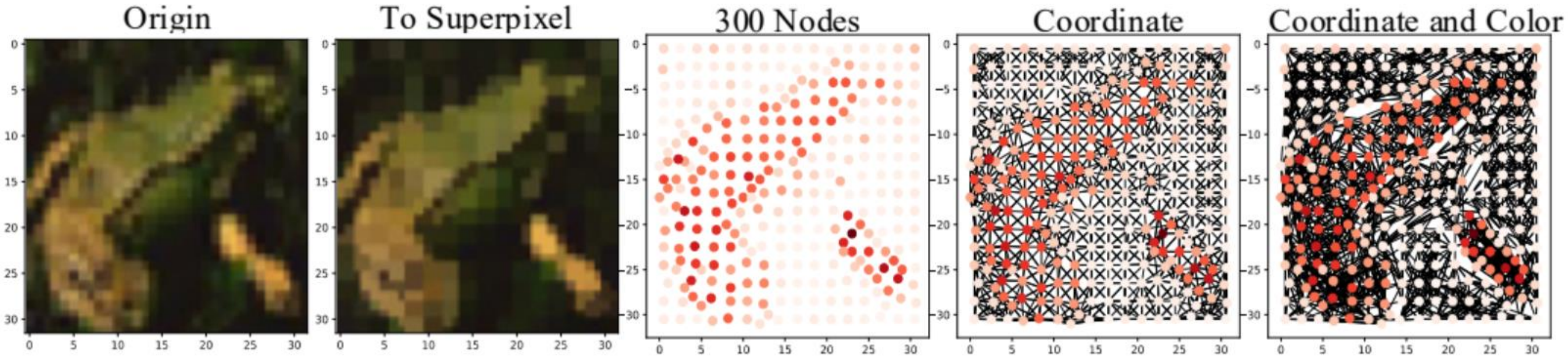


Figure 4. Cifar10 with 300 nodes.

Experiment

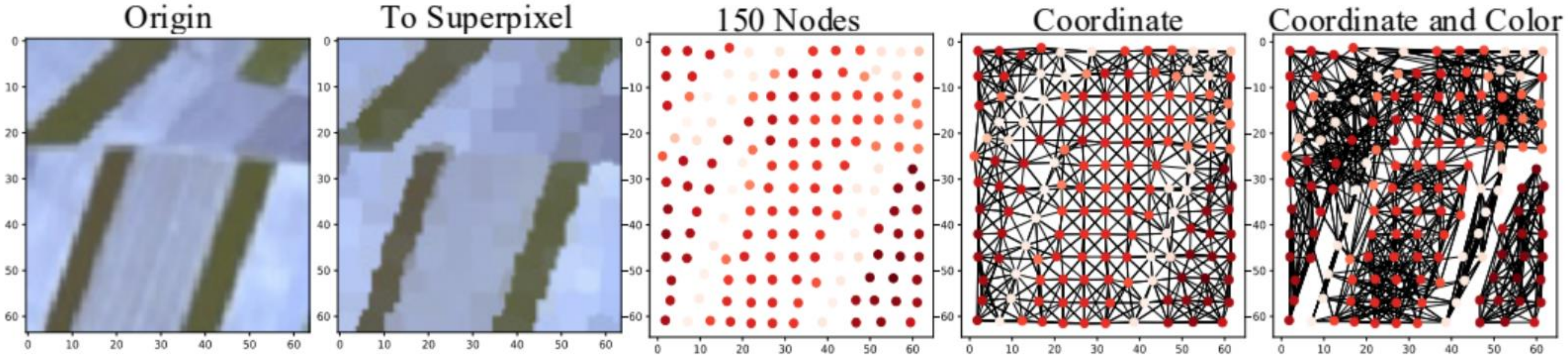


Figure 5. Euro_SAT with 150 nodes.

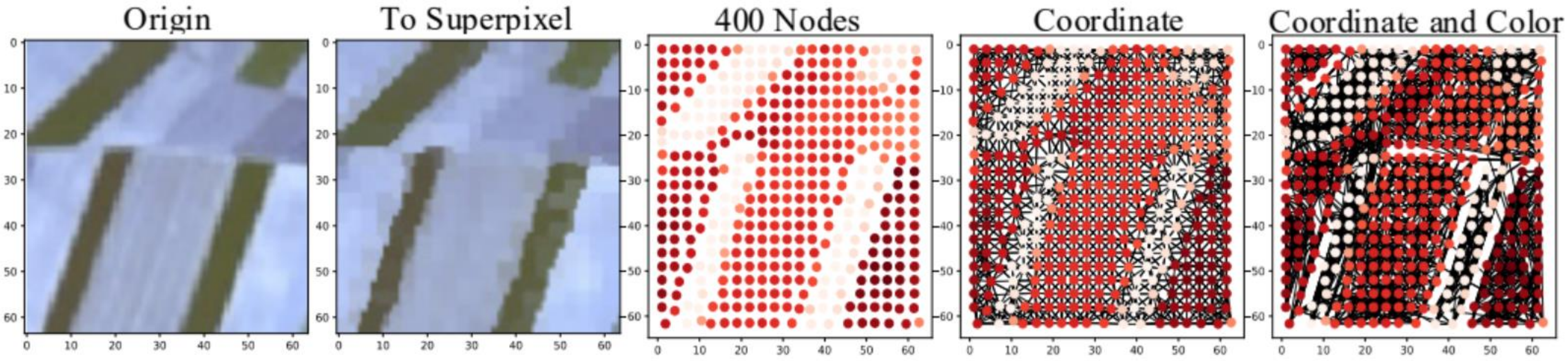


Figure 6. Euro_SAT with 400 nodes.

Experiment

Table 1. Experiments in Mnist with different number of nodes.

Method	Acc		
	Mnist-75	Mnist-150	Mnist-200
GCN	89.99	91.12	93.24
GAT	95.62	97.01	98.23
GraphSage	97.09	98.07	-
MoNet	90.36	92.24	-
GIN	93.91	96.49	-
SplineCNN	95.22	97.48	98.11
Ours	**97.11**	**98.27**	**98.50**

Table 2. Experiments in Cifar10 with different number of nodes.

Method	Acc	
	Cifar10-150	Cifar10-200
GCN	54.46	58.28
GAT	65.40	69.09
GraphSage	65.93	-
MoNet	53.42	-
GIN	53.28	-
Ours	**66.24**	**70.61**

Table 3. Experiments in Euro_SAT with different number of nodes.

Method	Acc		
	Euro_SAT-75	Euro_SAT-150	Euro_SAT-200
GCN	59.54	62.33	62.25
GAT	66.12	66.74	67.66
SplineCNN	66.00	70.12	71.24
Ours	**75.22**	**78.13**	**77.88**