Generating Classification Weights with GNN Denoising Autoencoders for Few-Shot Learning

Spyros Gidaris[1,2] and Nikos Komodakis[1]

[1]University Paris-Est, LIGM, Ecole des Ponts ParisTech
[2]valeo.ai

Hanqing Chao

Few-shot learning in this paper

Train

- Tiger, Car, Dog … (50 classes, called basic classes)
- 500 images for each class

Test

- Tiger, Car, Dog … (50 classes, called basic classes)

  \+ Lion, Train, Desk … (80 classes called, novel classes.)
- 10 images for each novel class

arxiv: 1905.01102

Few-shot learning in this paper

Train

- Tiger, Car, Dog … (50 classes, called basic classes)
- 500 images for each class

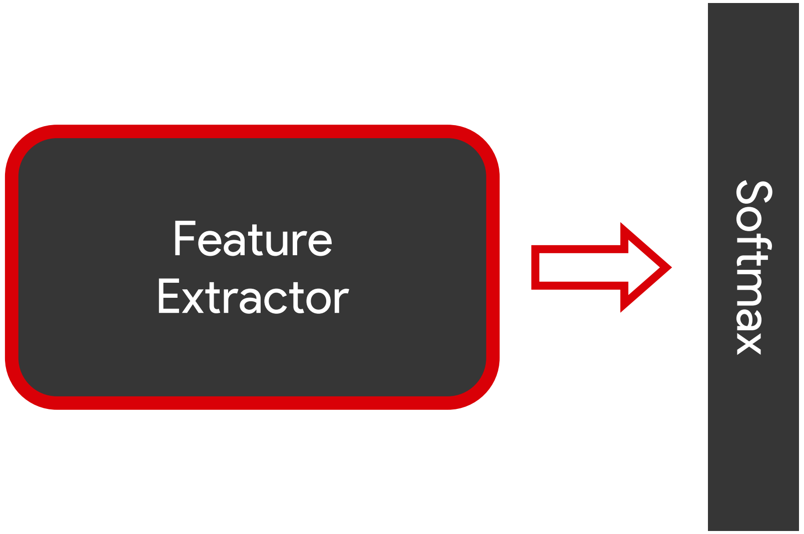
Test

- Tiger, Car, Dog … (50 classes, called basic classes)

  + Lion, Train, Desk … (80 classes called, novel classes.)
- 10 images for each novel class

arxiv: 1905.01102

Main Idea

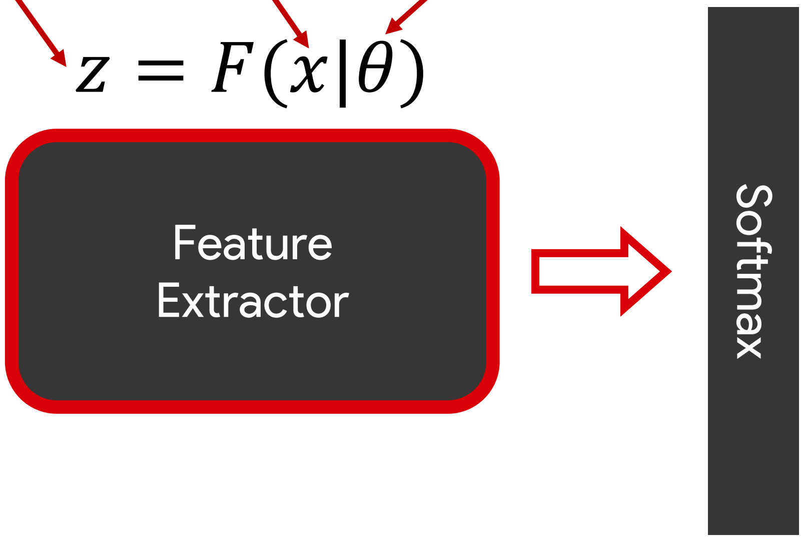


arxiv: 1905.01102

Main Idea

Feature image parameters

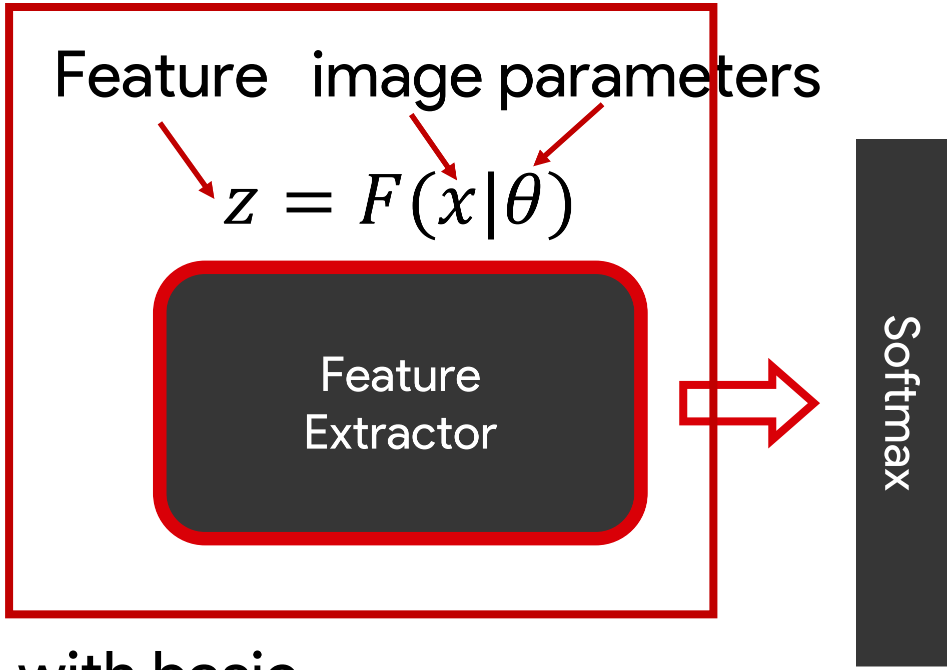
$$z = F(x|\theta)$$



$$y = Softmax(w \cdot z)$$

arxiv: 1905.01102

Main Idea

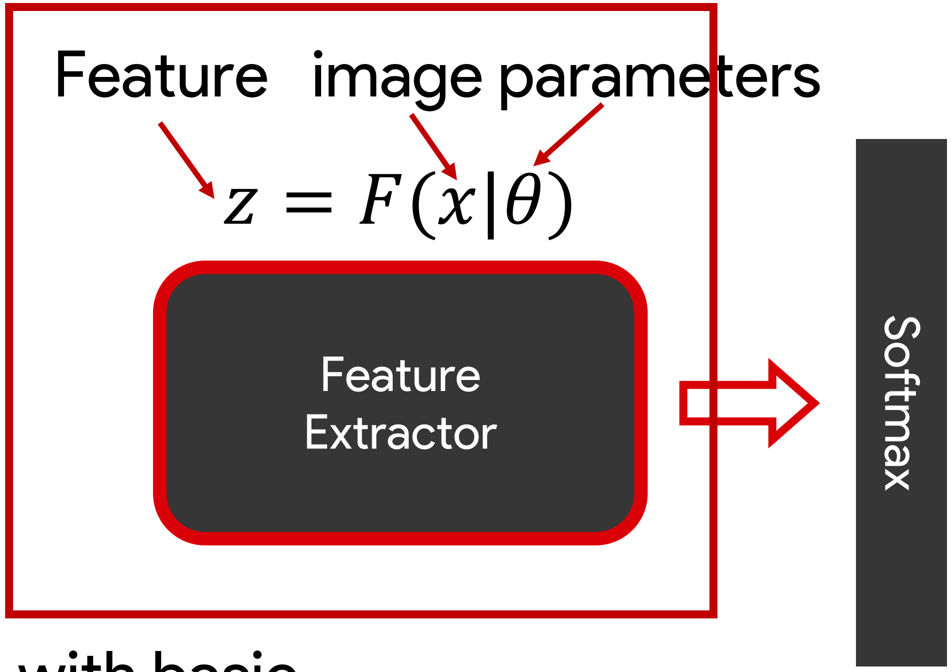


Train with basic
classes, then **fixed**

$$y = Softmax(w \cdot z)$$

arxiv: 1905.01102

Main Idea



Train with basic classes, then **fixed**

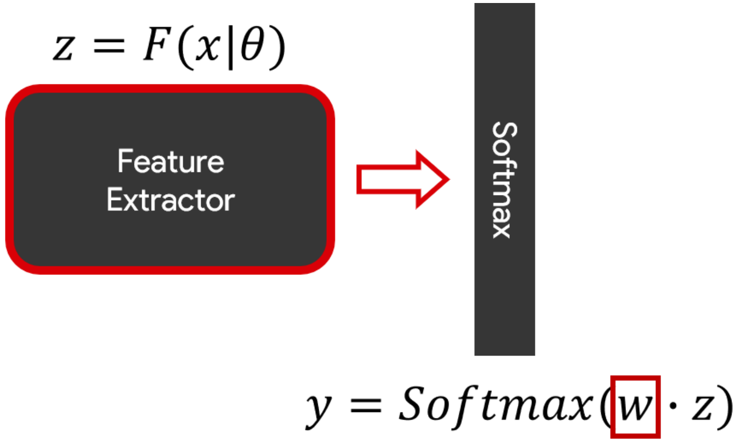
To learn with few shots

$$y = Softmax(\boxed{w} \cdot z)$$

classification weights
(principle vectors of classes)

arxiv: 1905.01102

Denoising Autoencoder



- Use Denoising Autoencoder (DAE) to learn w

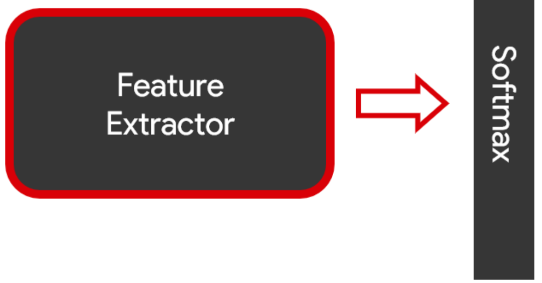
Object: $r(w + \eta) = w$

r: DAE
η: injected Gaussian noise $\eta \sim \mathcal{N}(0, \sigma)$

arxiv: 1905.01102

Denoising Autoencoder

$z = F(x|\theta)$



$y = Softmax(\boxed{w} \cdot z)$

- Use Denoising Autoencoder (DAE) to learn w

Object: $r(w + \eta) = w$

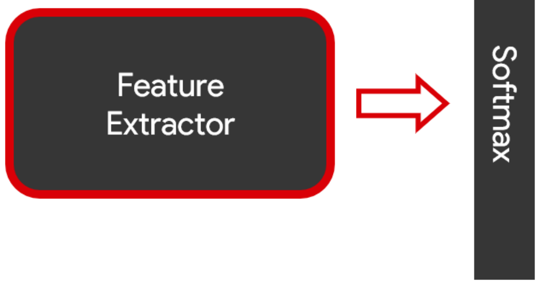
r: DAE
η: injected Gaussian noise $\eta \sim \mathcal{N}(0, \sigma)$

- First guess a w, then feed it to DAE
- Use DAE to generate better w

arxiv: 1905.01102

Why it is possible?

$z = F(x|\theta)$



$y = Softmax(\boxed{w} \cdot z)$

- According to Yoshua Bengio:

$$\frac{\partial \log p(\mathbf{w})}{\partial \mathbf{w}} \approx \frac{1}{\sigma^2} \cdot \left(r(\mathbf{w}) - \mathbf{w}\right)$$

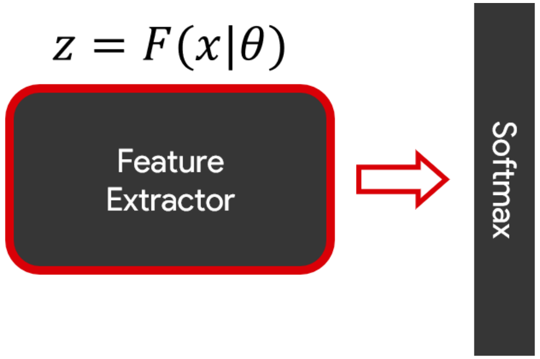
Cite: *What regularized auto-encoders learn from the data-generating distribution*

- Means, with a DAE, we can get the gradient of $p(w)$ (probability density function of w)

arxiv: 1905.01102

Denoising Autoencoder

$z = F(x|\theta)$



$y = Softmax(\boxed{w} \cdot z)$

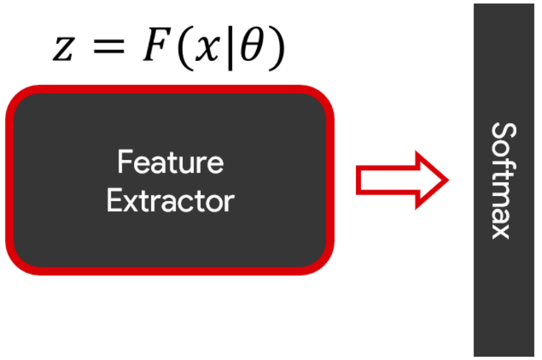
- Thus, given data D_{tr}, following equation can give us a w with a higher probability (a better w):

$$\mathbf{w} \leftarrow \mathbf{w} + \epsilon \cdot \frac{\partial \log p(\mathbf{w}|D_{tr})}{\partial \mathbf{w}} = \mathbf{w} + \epsilon \cdot (r(\mathbf{w}) - \mathbf{w})$$

arxiv: 1905.01102

Denoising Autoencoder

$$z = F(x|\theta)$$



$$\mathbf{w} \leftarrow \mathbf{w} + \epsilon \cdot \frac{\partial \log p(\mathbf{w}|D_{tr})}{\partial \mathbf{w}} = \mathbf{w} + \epsilon \cdot (r(\mathbf{w}) - \mathbf{w})$$

$$y = Softmax(\boxed{w} \cdot z)$$

- Initialization:

$$\mathbf{w}_i = \begin{cases} \mathbf{w}_i^{bs}, & \text{if } i \text{ is a base class} \\ \frac{1}{K}\sum_{k=1}^{K} F(\mathbf{x}_{k,i}|\theta), & \text{otherwise} \end{cases}$$

arxiv: 1905.01102

How to train a DAE

- Randomly split basic classes N_{bs} into “fake” novel classes $\widetilde{N}_{nv}$ and regard $\widetilde{N}_{bs} = N_{bs} - \widetilde{N}_{nv}$ as basic classes.

- Input: $w_i + \eta$

$$\mathbf{w}_i = \begin{cases} \mathbf{w}_i^{bs}, & \text{if } i \text{ is a base class} \\ \frac{1}{K}\sum_{k=1}^{K} F(\mathbf{x}_{k,i}|\theta), & \text{otherwise} \end{cases}$$

- Output: w^*
- Loss function:

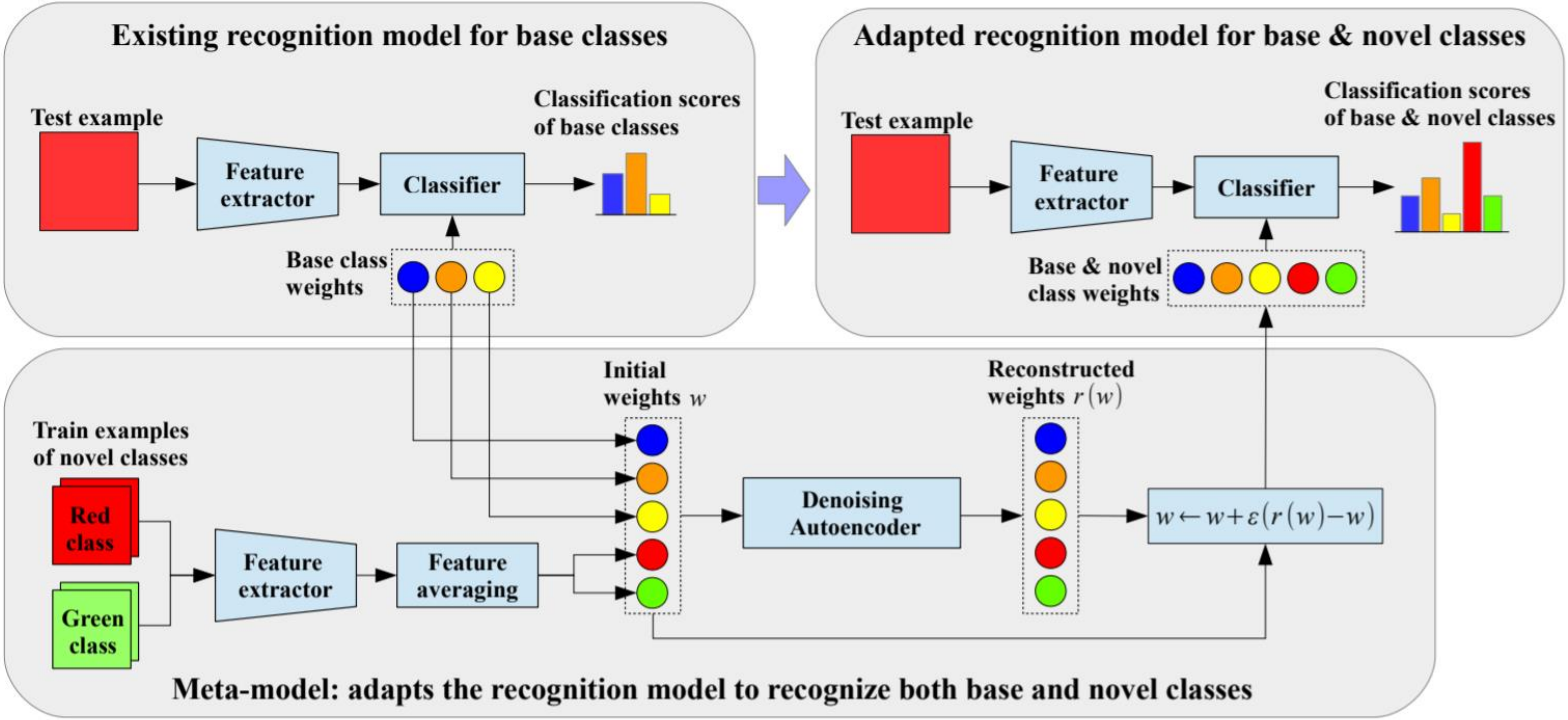
CE Loss

$$\boxed{\frac{1}{N}\sum_{i=1}^{\tilde{N}} \|\hat{\mathbf{w}}_i - \mathbf{w}^*{}_i\|^2} + \boxed{\frac{1}{M}\sum_{m=1}^{M} loss(\mathbf{x}_m, y_m|\hat{\mathbf{w}})}$$

Reconstruction Loss

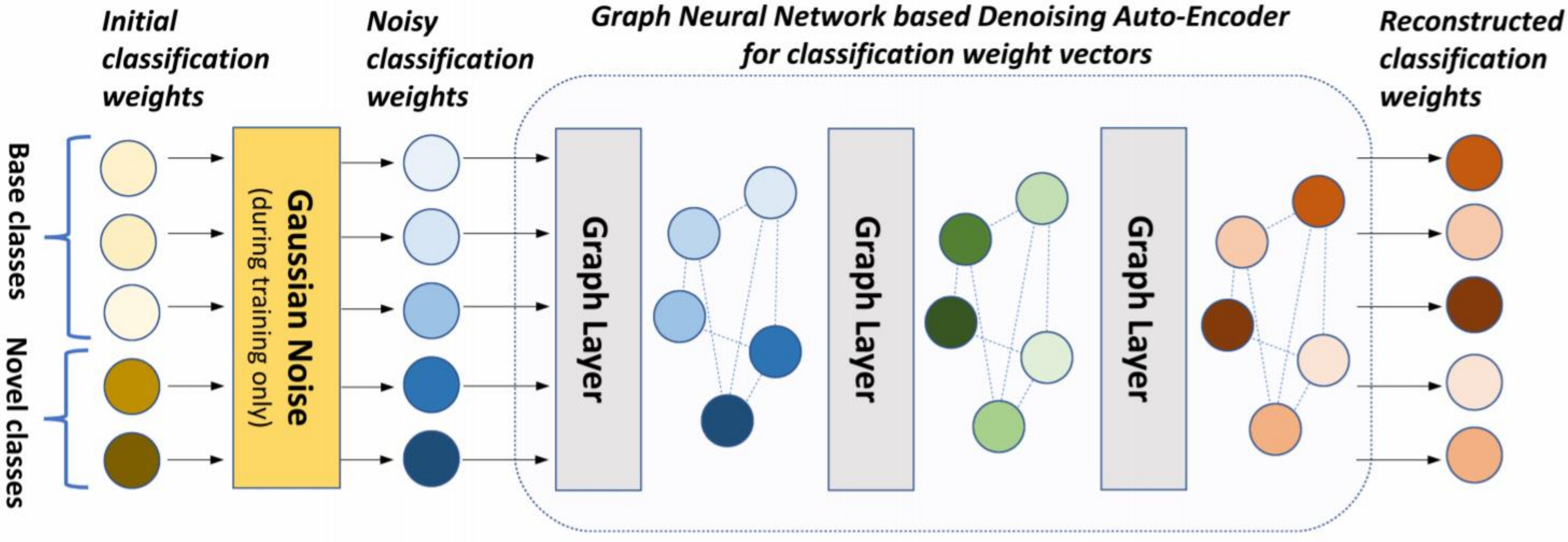
arxiv: 1905.01102

Pipeline



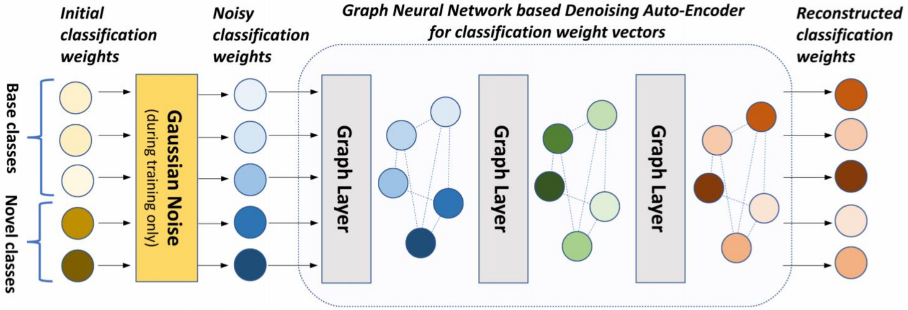
arxiv: 1905.01102

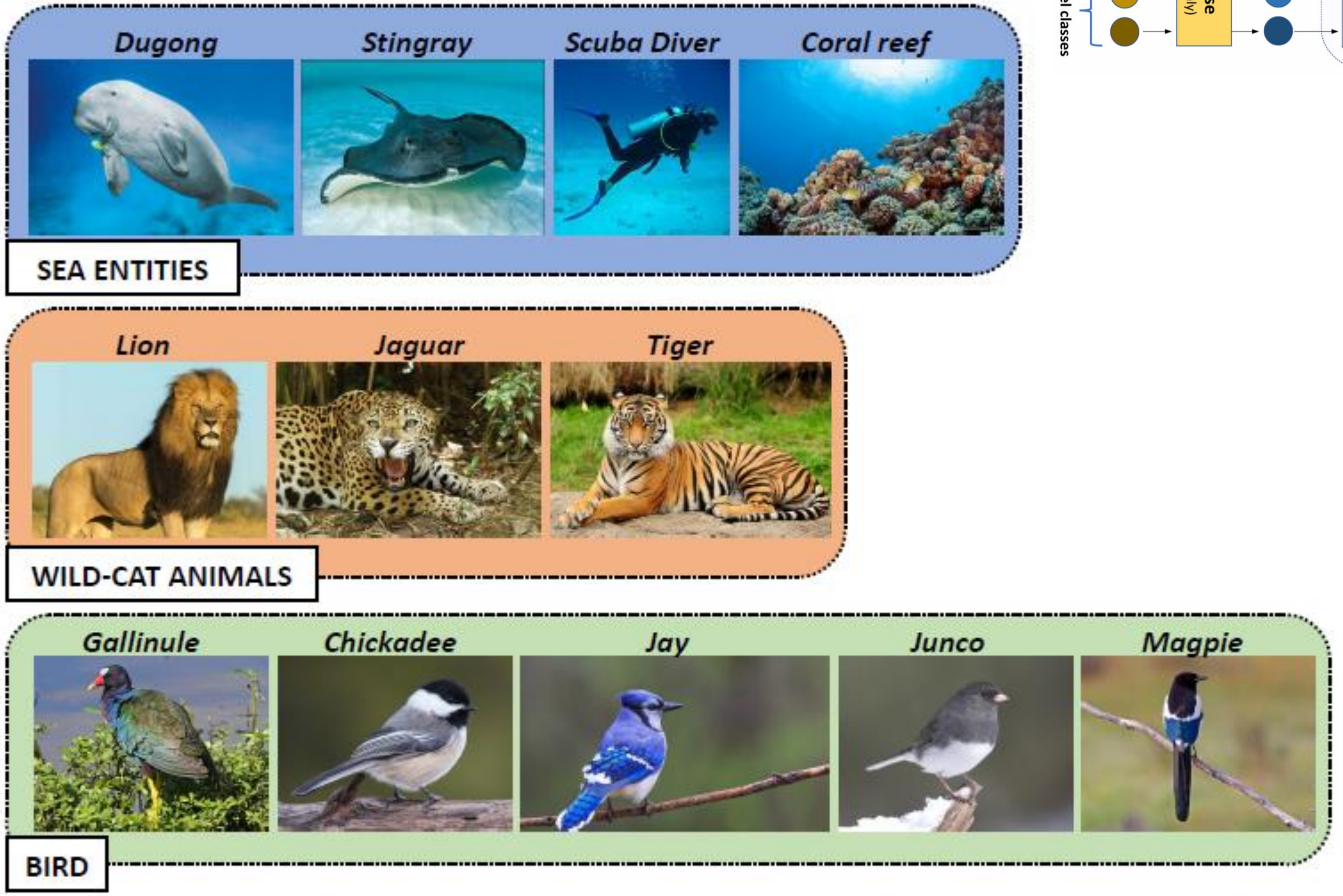
GNN based DAE



arxiv: 1905.01102

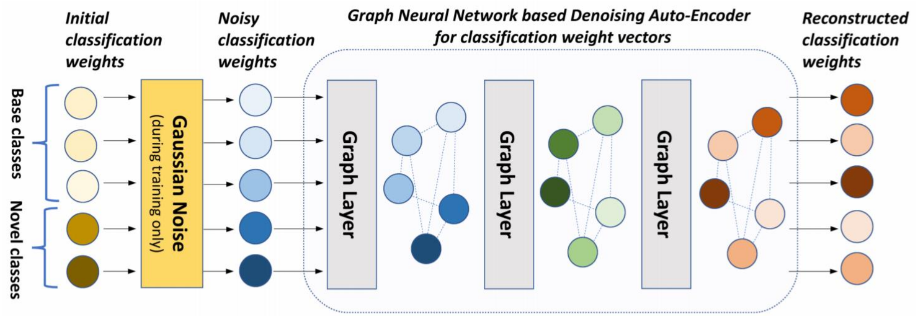
Why

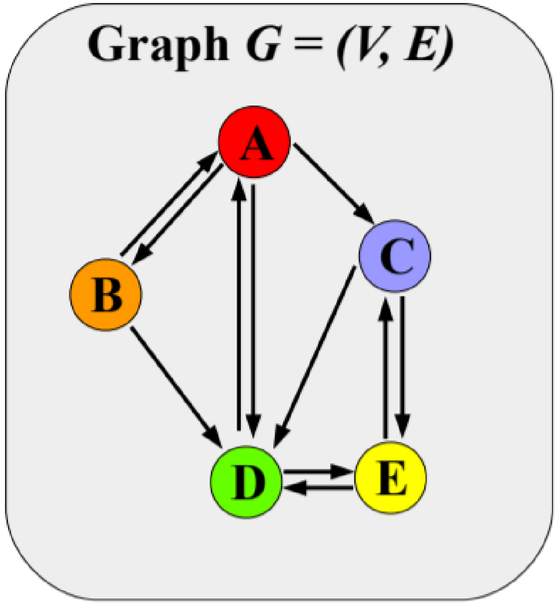




arxiv: 1905.01102

Detail of DAE-GNN

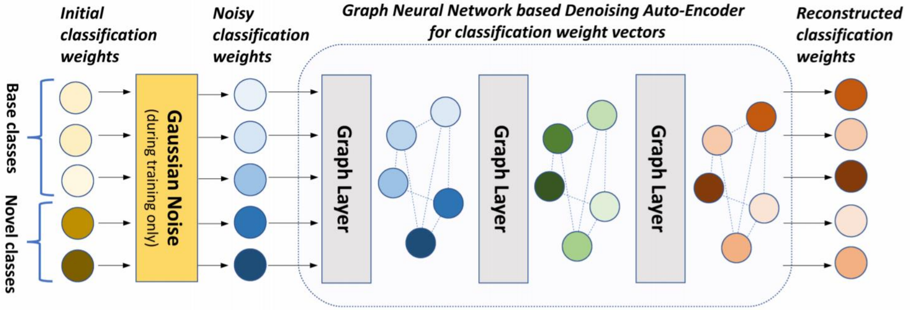


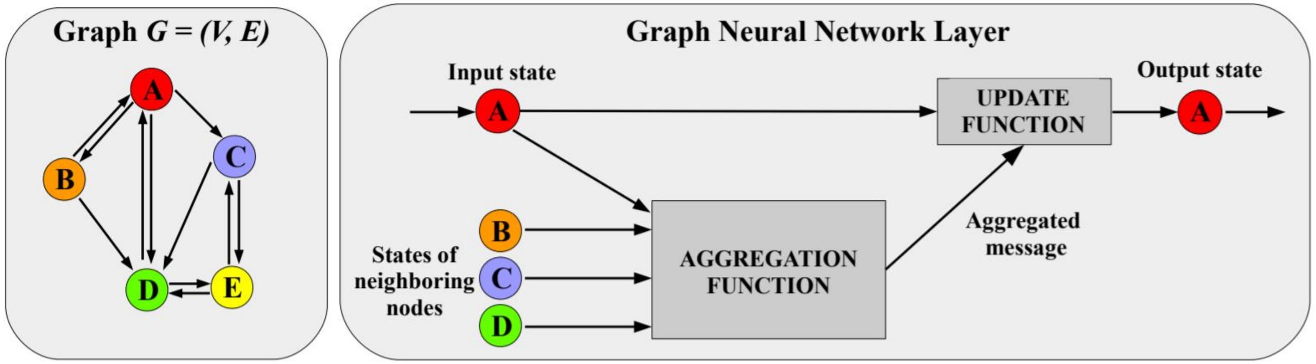


- $V: w$
- $E:$
  - Weight of edge: cos similarity
  - Which to connect: 10 nearest neighbors

arxiv: 1905.01102

Detail of DAE-GNN





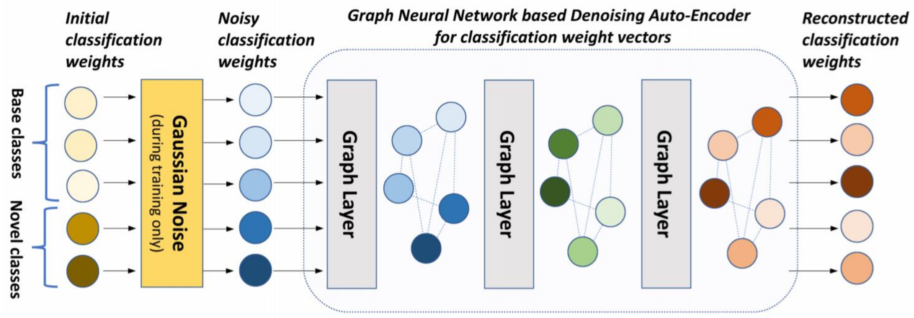
(a) The general architecture of a GNN layer.

$$\mathbf{h}_{\mathcal{N}(i)}^{(l)} = \texttt{AGGREGATE}\left(\{\mathbf{h}_j^{(l)}, \forall j \in \mathcal{N}(i)\}\right) ,$$

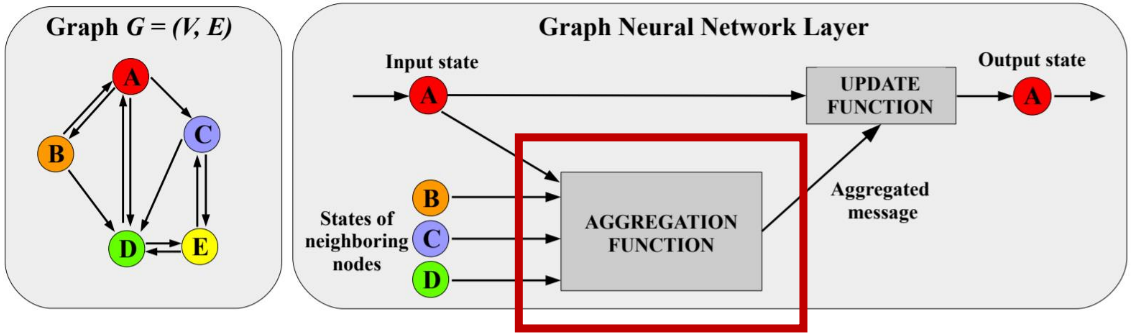
$$\mathbf{h}_i^{(l+1)} = \texttt{UPDATE}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_{\mathcal{N}(i)}^{(l)}\right) ,$$

arxiv: 1905.01102

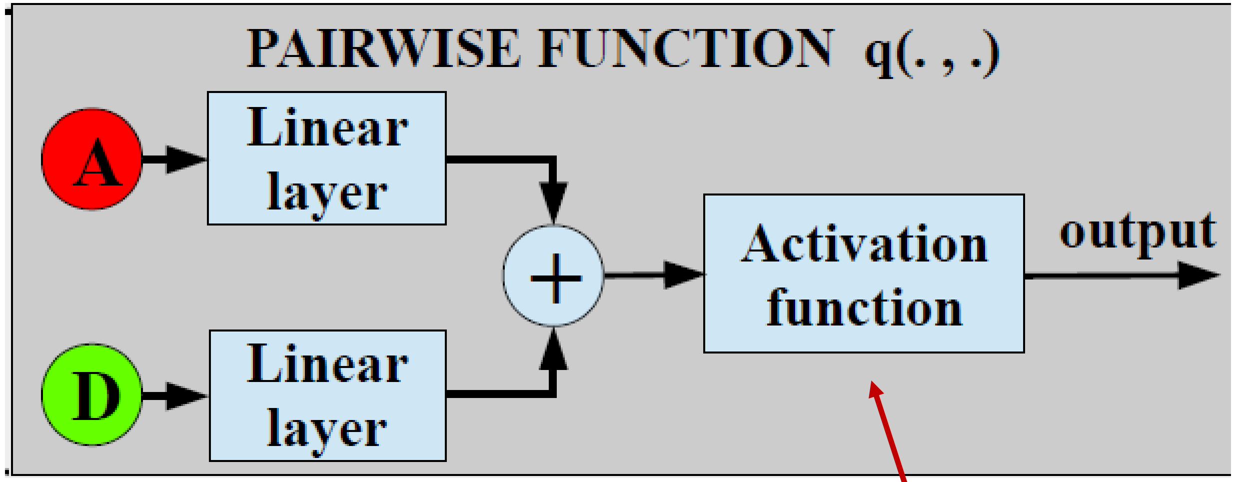
Aggregation



$$\mathrm{h}_{\mathcal{N}(i)}^{(l)} = \sum_{j \in \mathcal{N}(i)} a_{ij} \cdot q^{(l)}\left(\mathrm{h}_i^{(l)}, \mathrm{h}_j^{(l)}\right)$$



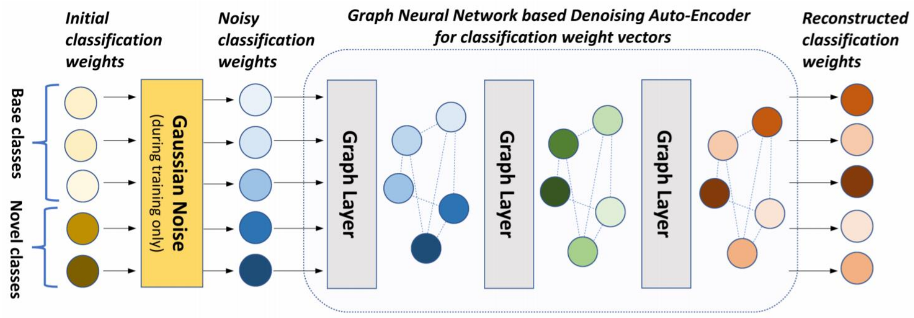
(a) The general architecture of a GNN layer.



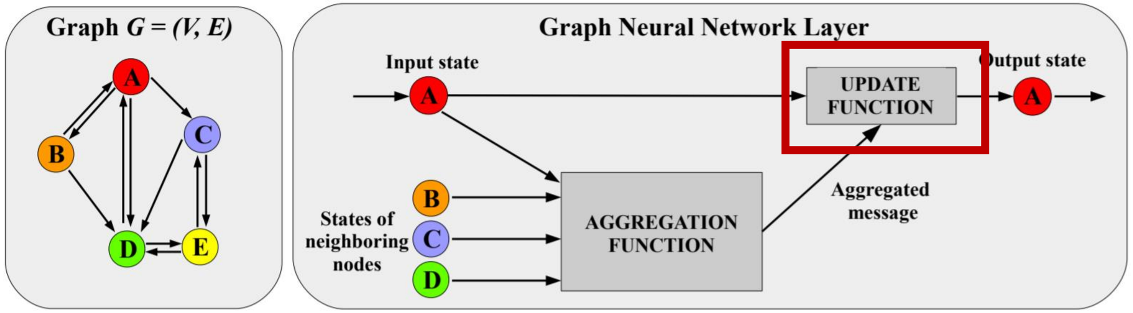
BatchNorm + Dropout + LeakyReLU

arxiv: 1905.01102

Update



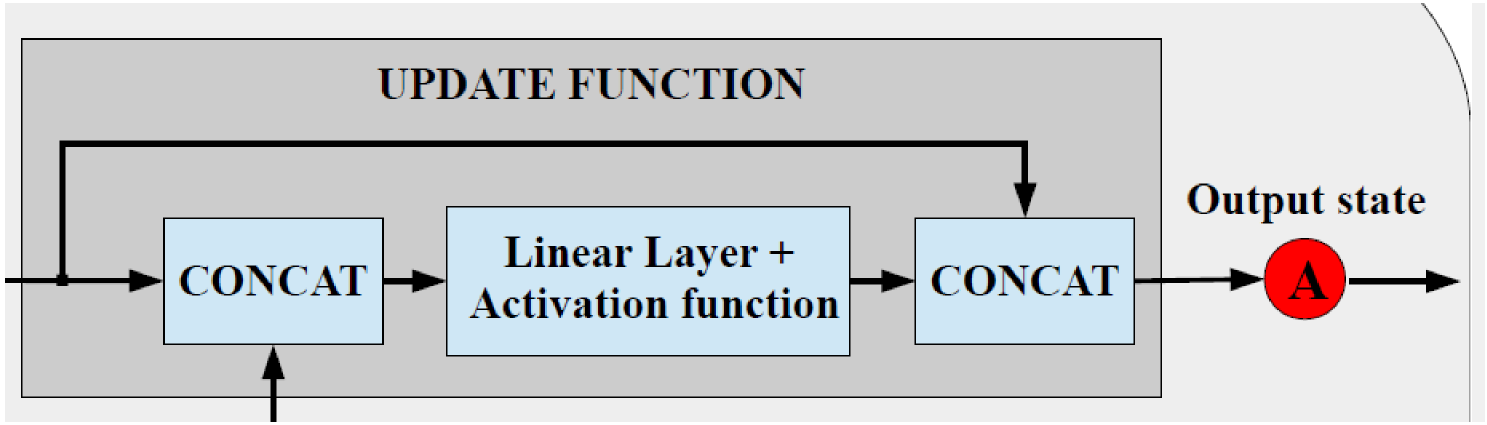
$$\mathbf{h}_i^{(l+1)} = \left[\mathbf{h}_i^{(l)};\ u^{(l)}\left(\left[\mathbf{h}_i^{(l)};\ \mathbf{h}_{\mathcal{N}(i)}^{(l)}\right]\right)\right]$$



(a) The general architecture of a GNN layer.

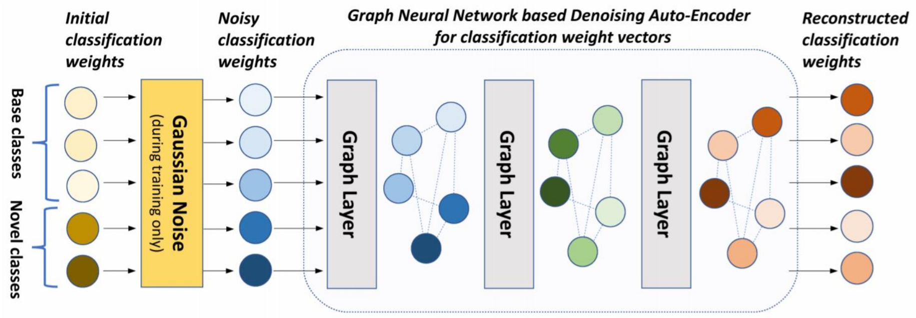
$[\alpha; \beta]$: concatenation

$u(\cdot)$: FC+BN+DO+LeakyReLU



arxiv: 1905.01102

Update

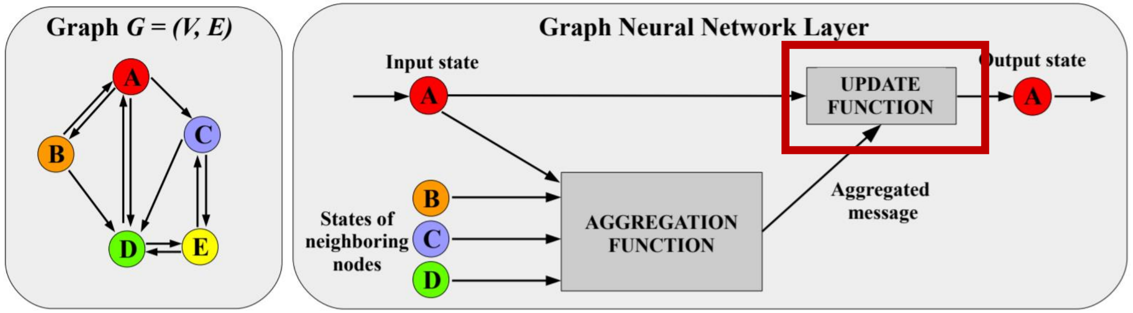


Specially, in the last layer:

$$\delta \mathbf{w}_i,\ \mathbf{o}_i = u^{(L-1)}\left(\left[\mathbf{h}_i^{(L-1)};\ \mathbf{h}_{\mathcal{N}(i)}^{(L-1)}\right]\right)$$

$u^{L-1}(\cdot)$: FC+
L2_norm (on δw_i) / Sigmoid (on o_i)

$$\hat{\mathbf{w}}_i = \mathbf{w}_i + \mathbf{o}_i \odot \delta \mathbf{w}_i.$$



(a) The general architecture of a GNN layer.

arxiv: 1905.01102

Experiments ImageNet-FS

Approach	Novel classes K=1	2	5	10	20	All classes K=1	2	5	10	20
Prior work										
Prototypical-Nets (from [41])	39.3	54.4	66.3	71.2	73.9	49.5	61.0	69.7	72.9	74.6
Matching Networks (from [41])	43.6	54.0	66.0	72.5	76.9	54.4	61.0	69.0	73.7	76.5
Logistic regression [13]	38.4	51.1	64.8	71.6	76.6	40.8	49.9	64.2	71.9	76.9
Logistic regression w/ H [13]	40.7	50.8	62.0	69.3	76.5	52.2	59.4	67.6	72.8	76.9
SGM w/ H [13]	-	-	-	-	-	54.3	62.1	71.3	75.8	78.1
Batch SGM [13]	-	-	-	-	-	49.3	60.5	71.4	75.8	78.5
Prototype Matching Nets w/ H [41]	45.8	57.8	69.0	74.3	77.4	57.6	64.7	71.9	75.2	77.5
LwoF [9]	46.2	57.5	69.2	74.8	**78.1**	58.2	65.2	72.7	**76.5**	**78.7**
Ours										
wDAE-GNN	**48.0**	**59.7**	**70.3**	**75.0**	77.8	**59.1**	**66.3**	**73.2**	76.1	77.5
wDAE-MLP	47.6	59.2	70.0	74.8	77.7	59.0	66.1	72.9	75.8	77.4
Ablation study on wDAE-GNN										
Initial estimates	45.4	56.9	68.9	74.5	77.7	57.0	64.3	72.3	75.6	77.3
wDAE-GNN - No Noise	47.6	59.0	70.0	74.9	77.8	60.0	66.0	72.9	75.8	77.4
wDAE-GNN - Noisy Targets as Input	47.8	59.4	70.1	74.8	77.7	58.7	66.0	73.1	76.0	77.5
wDAE-GNN - No Cls. Loss	47.7	59.1	69.8	74.6	77.6	58.4	65.5	72.7	75.8	77.5
wDAE-GNN - No Rec. Loss	47.8	59.4	70.1	75.0	77.8	58.7	66.0	73.1	76.1	77.6

arxiv: 1905.01102

Experiments MiniImageNet

Models	Backbone	1-shot	5-shot
Prior work			
MAML [7]	Conv-4-64	$48.70 \pm 1.84\%$	$63.10 \pm 0.92\%$
Prototypical Nets [36]	Conv-4-64	$49.42 \pm 0.78\%$	$68.20 \pm 0.66\%$
LwoF [9]	Conv-4-64	$56.20 \pm 0.86\%$	$72.81 \pm 0.62\%$
RelationNet [42]	Conv-4-64	$50.40 \pm 0.80\%$	$65.30 \pm 0.70\%$
GNN [8]	Conv-4-64	50.30%	66.40%
R2-D2 [3]	Conv-4-64	$48.70 \pm 0.60\%$	$65.50 \pm 0.60\%$
R2-D2 [3]	Conv-4-512	$51.20 \pm 0.60\%$	$68.20 \pm 0.60\%$
TADAM [24]	ResNet-12	$58.50 \pm 0.30\%$	$76.70 \pm 0.30\%$
Munkhdalai *et al.* [23]	ResNet-12	$57.10 \pm 0.70\%$	$70.04 \pm 0.63\%$
SNAIL [33]	ResNet-12	$55.71 \pm 0.99\%$	$68.88 \pm 0.92\%$
Qiao *et al.* [26]†	WRN-28-10	$59.60 \pm 0.41\%$	$73.74 \pm 0.19\%$
LEO [31]†	WRN-28-10	$61.76 \pm 0.08\%$	$77.59 \pm 0.12\%$
LwoF [9] (our implementation)	WRN-28-10	$60.06 \pm 0.14\%$	$76.39 \pm 0.11\%$
Ours			
wDAE-GNN	WRN-28-10	$61.07 \pm 0.15\%$	$76.75 \pm 0.11\%$
wDAE-MLP	WRN-28-10	$60.61 \pm 0.15\%$	$76.56 \pm 0.11\%$
wDAE-GNN†	WRN-28-10	$\mathbf{62.96} \pm 0.15\%$	$\mathbf{78.85} \pm 0.10\%$
wDAE-MLP†	WRN-28-10	$62.67 \pm 0.15\%$	$78.70 \pm 0.10\%$

Ablation study on wDAE-GNN			
Initial estimate	WRN-28-10	$59.68 \pm 0.14\%$	$76.48 \pm 0.11\%$
wDAE-GNN - No Noise	WRN-28-10	$60.29 \pm 0.14\%$	$76.49 \pm 0.11\%$
wDAE-GNN - Noisy Targets as Input	WRN-28-10	$60.92 \pm 0.15\%$	$76.69 \pm 0.11\%$
wDAE-GNN - No Cls. Loss	WRN-28-10	$60.96 \pm 0.15\%$	$76.75 \pm 0.11\%$
wDAE-GNN - No Rec. Loss	WRN-28-10	$60.76 \pm 0.15\%$	$76.64 \pm 0.11\%$
Ablation study on wDAE-MLP			
wDAE-MLP - No Noise	WRN-28-10	$60.16 \pm 0.15\%$	$76.50 \pm 0.11\%$
wDAE-MLP - Noisy Targets as Input	WRN-28-10	$60.43 \pm 0.15\%$	$76.49 \pm 0.11\%$
wDAE-MLP - No Cls. Loss	WRN-28-10	$60.55 \pm 0.15\%$	$76.62 \pm 0.11\%$
wDAE-MLP - No Rec. Loss	WRN-28-10	$60.45 \pm 0.15\%$	$76.50 \pm 0.11\%$

Table 2: Top-1 accuracies on the novel classes of MiniImageNet test set with 95% confidence intervals. †: using also the validation classes for training.

arxiv: 1905.01102

Experiments tiered-MiniImageNet

Models	Backbone	1-shot	5-shot
MAML [7] (from [21])	Conv-4-64	$51.67 \pm 1.81\%$	$70.30 \pm 0.08\%$
Prototypical Nets [36]	Conv-4-64	$53.31 \pm 0.89\%$	72.69 ± 0.74 %
RelationNet [42] (from [21])	Conv-4-64	$54.48 \pm 0.93\%$	$71.32 \pm 0.78\%$
Liu *et al.* [21]	Conv-4-64	$57.41 \pm 0.94\%$	71.55 ± 0.74
LEO [31]	WRN-28-10	$66.33 \pm 0.05\%$	81.44 ± 0.09 %
LwoF [9] (our implementation)	WRN-28-10	$67.92 \pm 0.16\%$	**83.10** $\pm 0.12\%$
wDAE-GNN (Ours)	WRN-28-10	**68.18** $\pm 0.16\%$	$83.09 \pm 0.12\%$

arxiv: 1905.01102

Thanks!