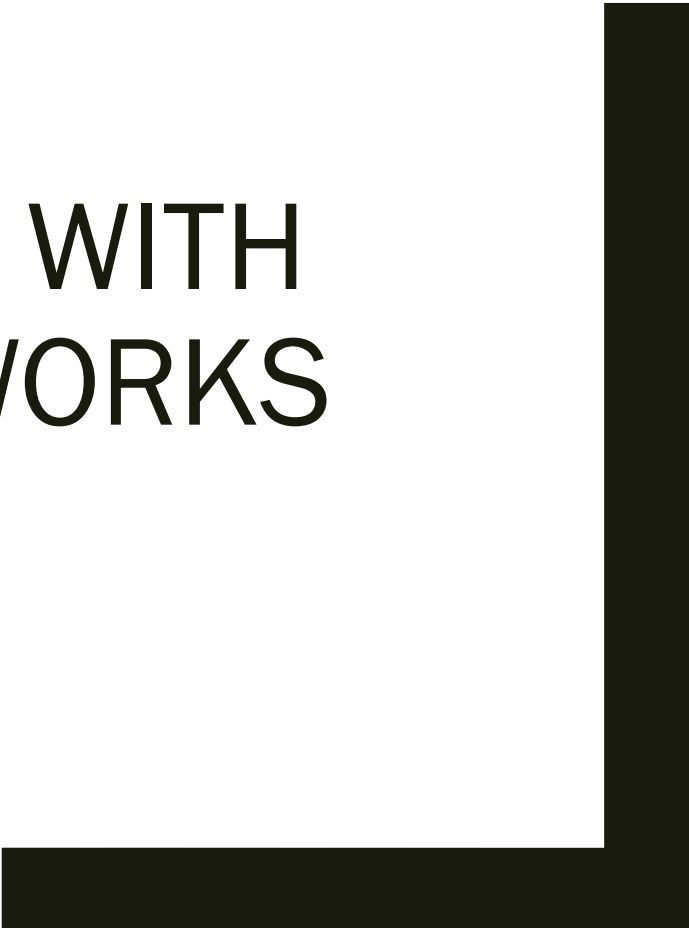




# FEW-SHOT LEARNING WITH GRAPH NEURAL NETWORKS

Victor Garcia, Joan Bruna ICLR 2018

Presented by – Harsha Kokel, Zelun Kong  
in [CS7301](#)



# Problem

- q-shot K-way classification



# Idea

- Present images as fully connected graph
  - *Images as nodes*
  - *Similarity as edges*

] sort of



Dataset  $\{(\mathcal{T}_i, Y_i)_i\}_{i \leq L}$

$$\mathcal{T} = \left\{ \underbrace{\{(x_1, l_1), \dots, (x_s, l_s)\}}_{\text{supervised samples}}, \underbrace{\{\tilde{x}_1, \dots, \tilde{x}_r\}}_{\text{unsupervised samples}}, \underbrace{\{\bar{x}_1, \dots, \bar{x}_t\}}_{\text{test samples}}; \right\}$$

*data* ↑ *label* ↗

\*

$$l_i \in \{1, K\}, x_i, \tilde{x}_j, \bar{x}_j \sim \mathcal{P}_l(\mathbb{R}^N)$$

$$Y = (y_1, \dots, y_t) \in \{1, K\}^t$$

For q-shot K-way classification  
 $r = 0, t = 1$  and  $s = qK$ ,

\* Ignore unsupervised samples for now,  
They will be used for semi-supervised and  
active learning setting



# Objective

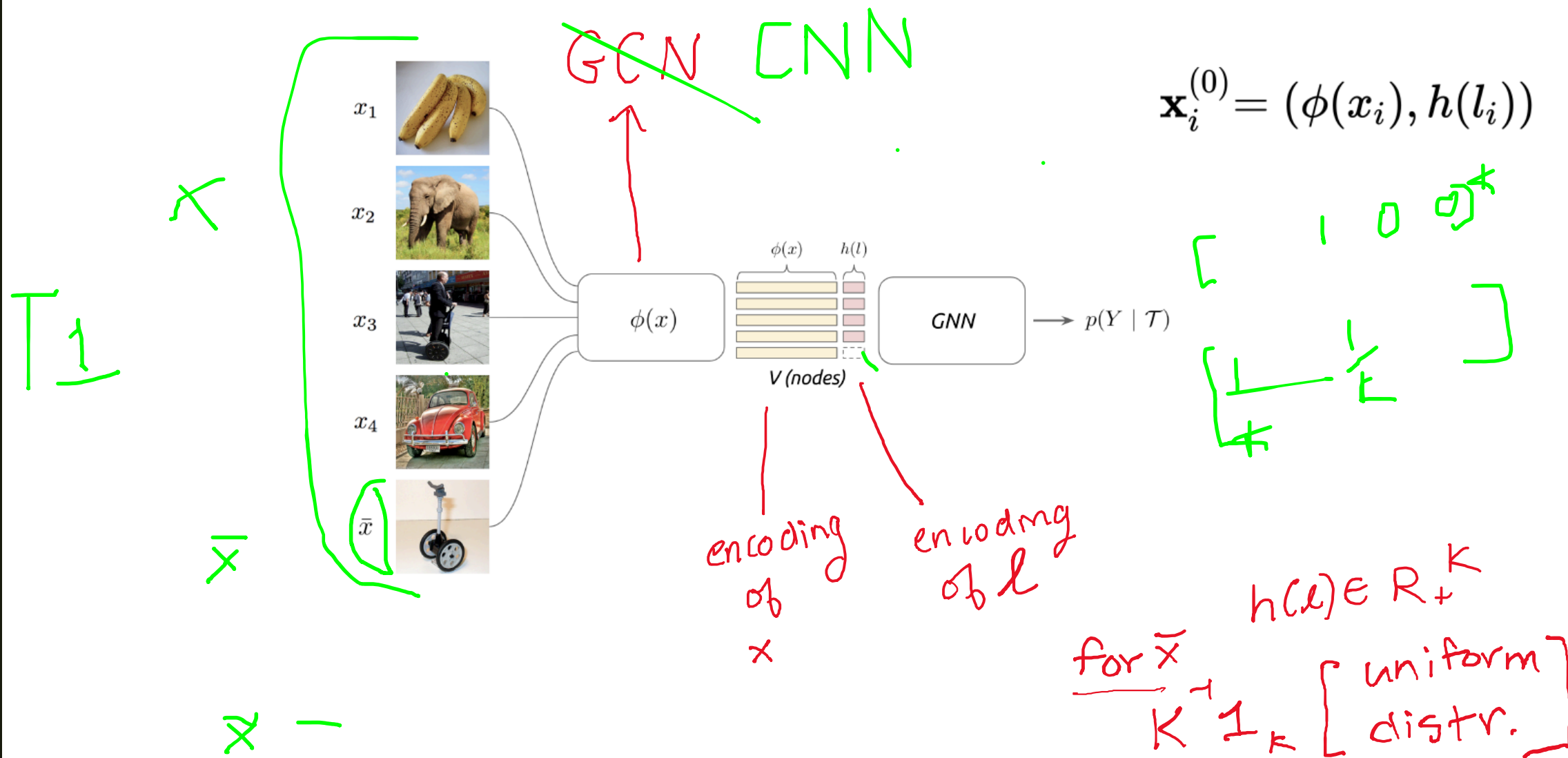
$$\min_{\Theta} \frac{1}{L} \sum_{i \leq L} \underbrace{\ell(\Phi(\mathcal{T}_i; \Theta), Y_i)}_{\text{loss w.r.t. test samples}} + \underbrace{\mathcal{R}(\Theta)}_{\text{regularizer}}$$

For q-shot K-way classification

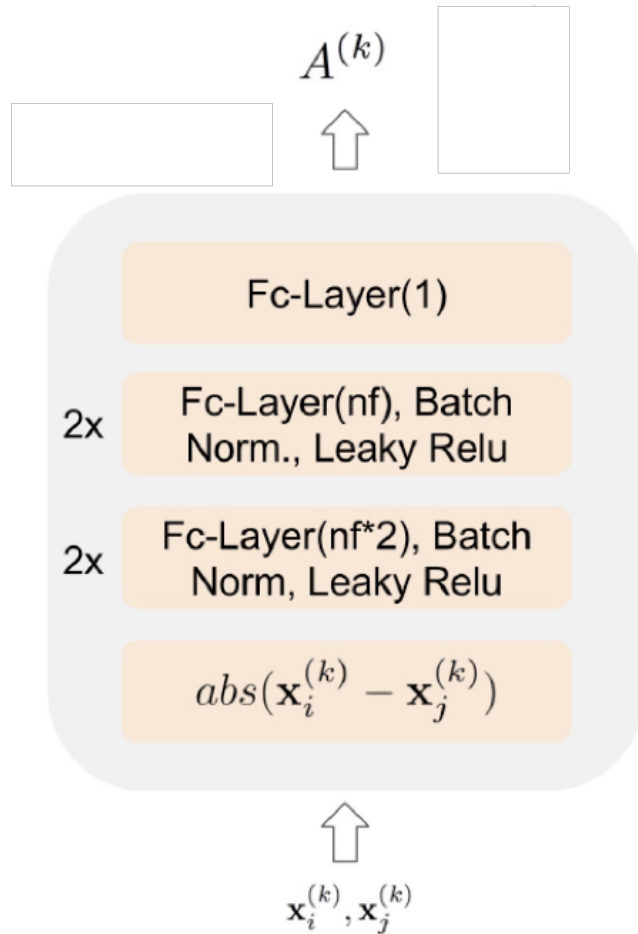
Cross entropy loss with t=1,

$$\ell(\Phi(\mathcal{T}; \Theta), Y) = - \sum_k y_k \log P(Y = y_k | \mathcal{T})$$

# Nodes - embedding



# Edge – distance metric



$$\tilde{A}_{i,j}^{(k)} = \varphi_{\tilde{\theta}}(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(k)}) = \text{MLP}_{\tilde{\theta}}(abs(\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}))$$

*→ learnt*

Identity  $\varphi_{\tilde{\theta}}(a, a) = 0$

Symmetry  $\varphi_{\tilde{\theta}}(a, b) = \varphi_{\tilde{\theta}}(b, a)$

*↳ allowed by def<sup>n</sup>*

# Proposed model

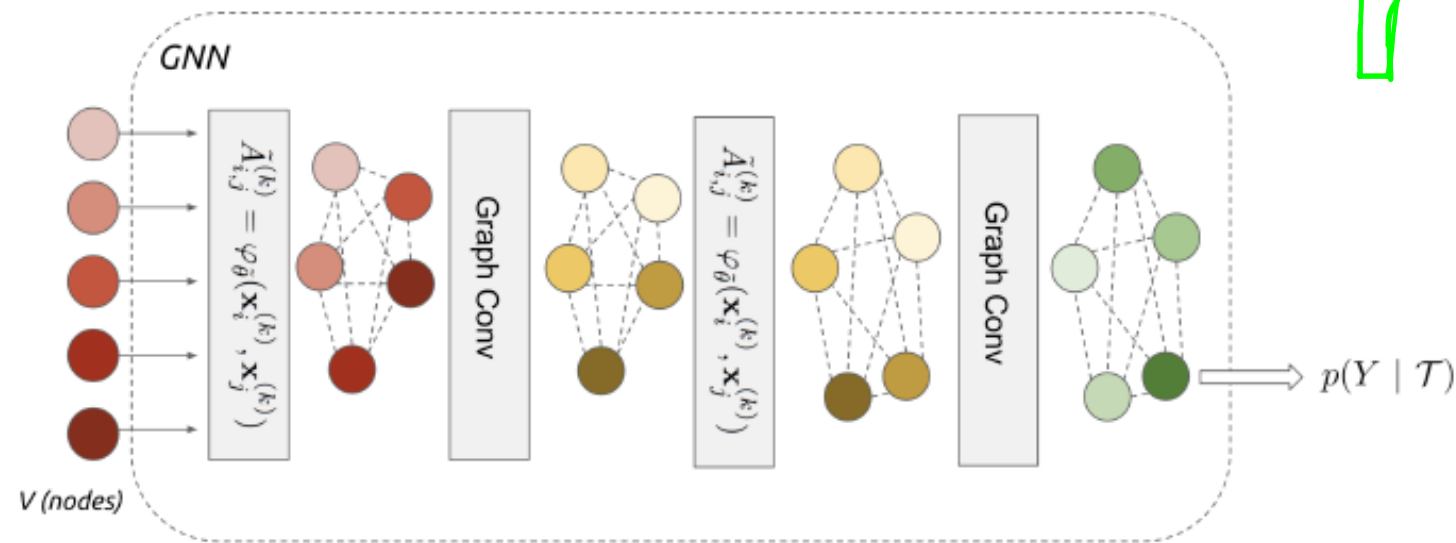


Figure 2: Graph Neural Network illustration. The Adjacency matrix is computed before every Convolutional Layer.

# Proposed model

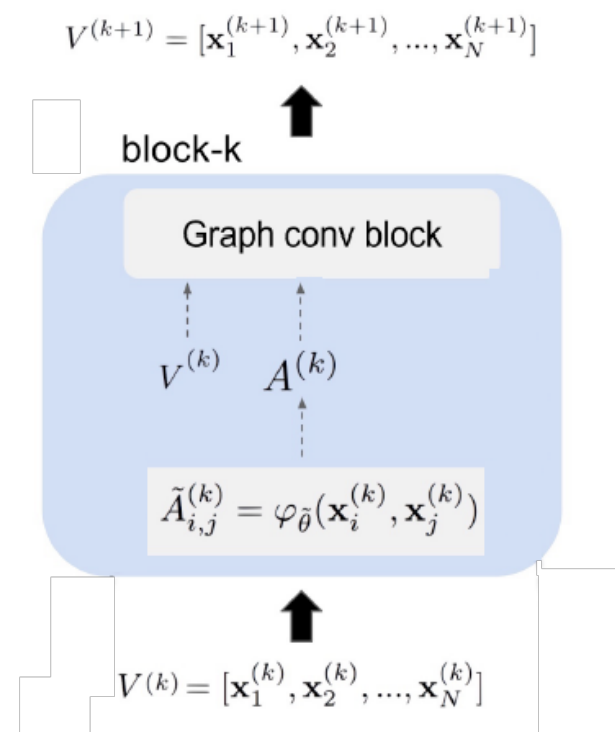
$$\mathbf{x}_l^{(k+1)} = \text{Gc}(\mathbf{x}^{(k)}) = \rho \left( \sum_{B \in \mathcal{A}} B \mathbf{x}^{(k)} \theta_{B,l}^{(k)} \right), l = d_1 \dots d_{k+1}$$

$$\mathcal{A} = \{ \tilde{A}^{(k)}, \mathbf{1} \}$$

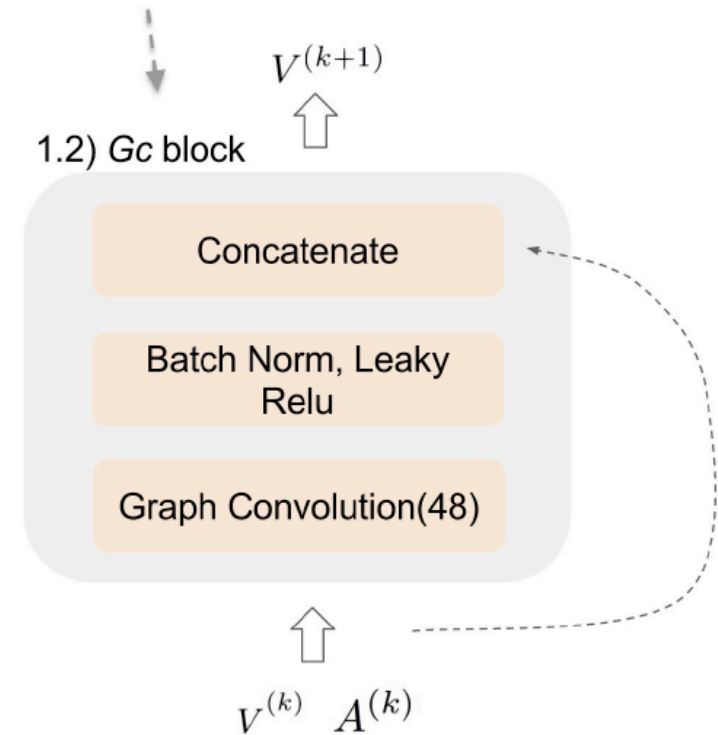
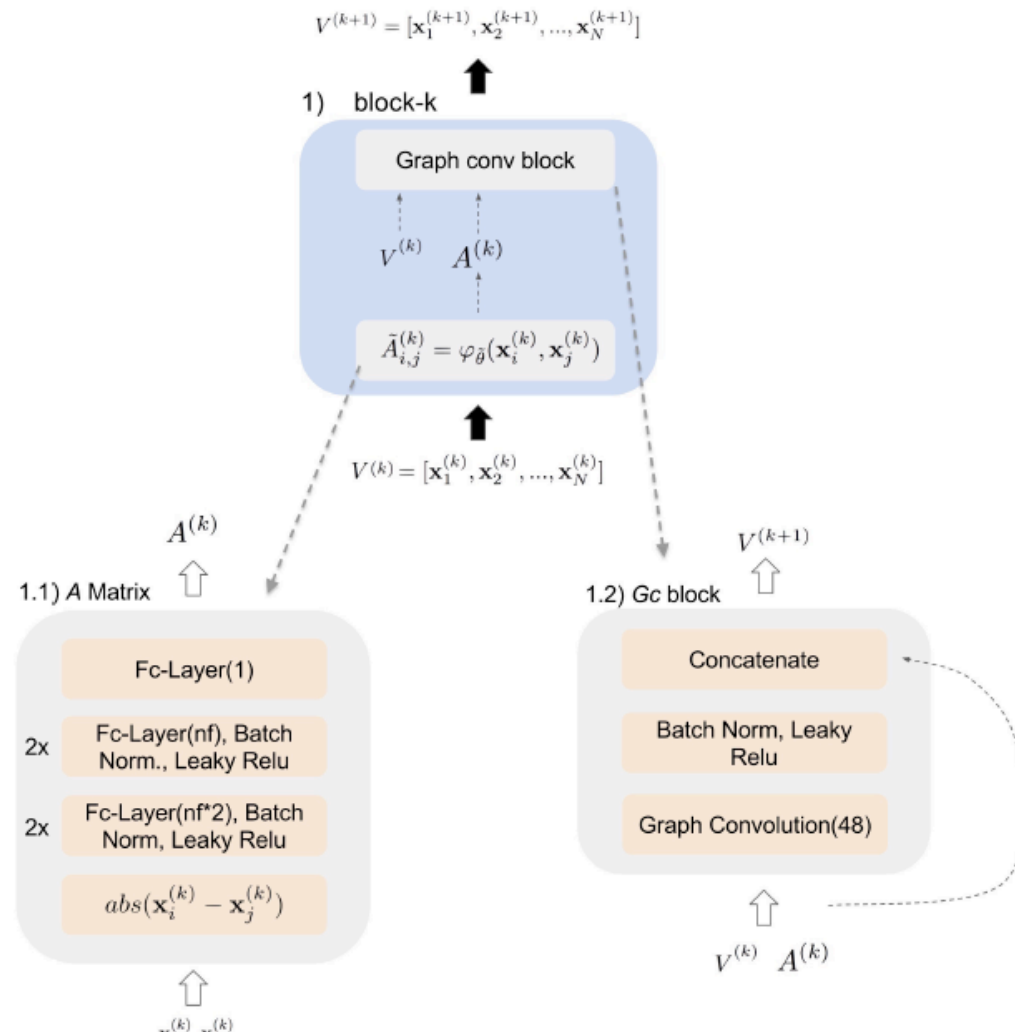
$$\Theta = \{ \theta_1^{(k)}, \dots, \theta_{|\mathcal{A}|}^{(k)} \}_k, \theta_B^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}}$$

$$\mathbf{x}^{(k)} \in \mathbb{R}^{V \times d_k}$$

$$\mathbf{x}^{(k+1)} \in \mathbb{R}^{V \times d_{k+1}}$$

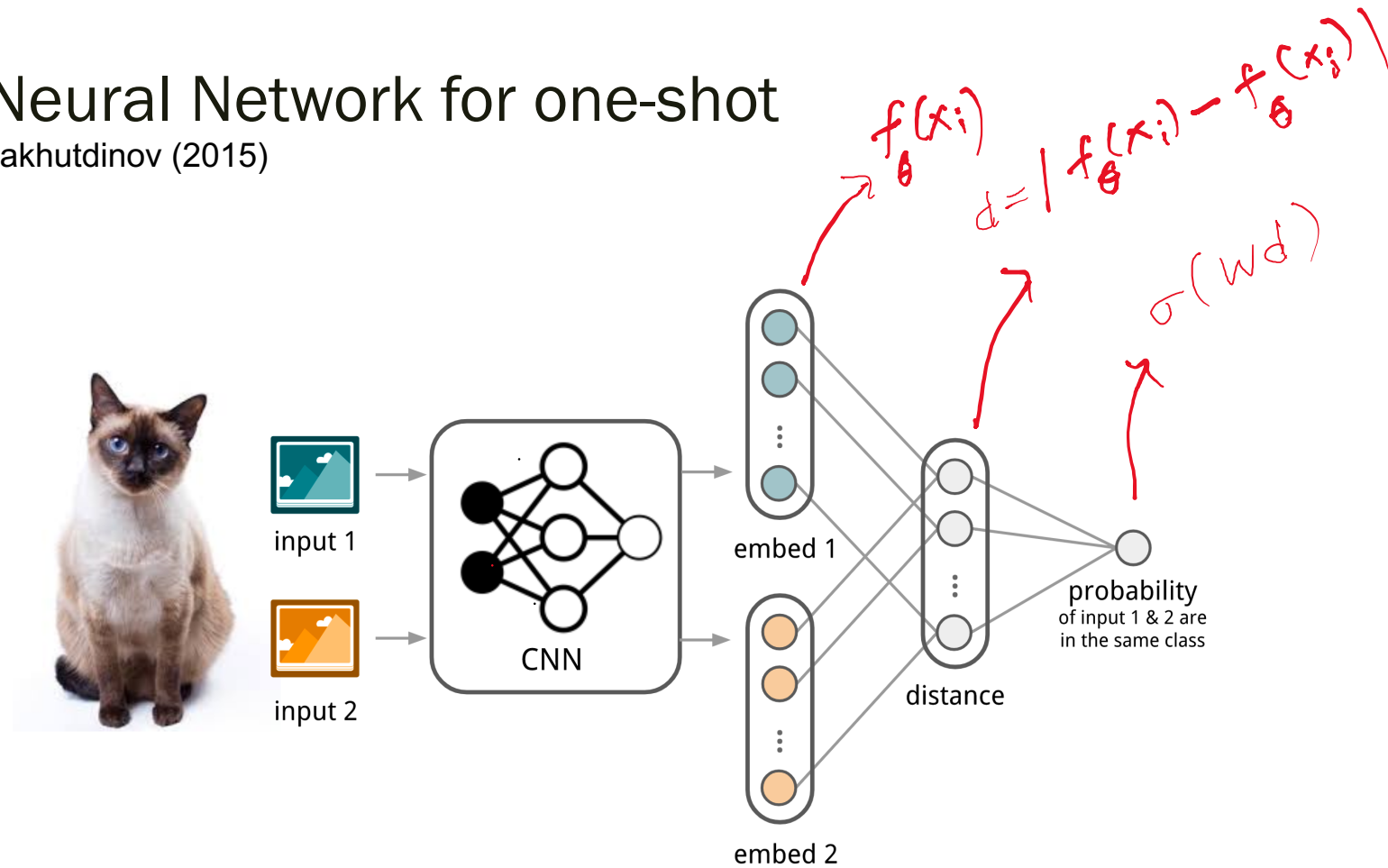


# Proposed model



# Siamese Neural Network for one-shot

Koch, Zemel & Salakhutdinov (2015)



Training  $\rightarrow$

$$\mathcal{L}(B) = \sum_{(\mathbf{x}_i, \mathbf{x}_j, y_i, y_j) \in B} \mathbf{1}_{y_i=y_j} \log p(\mathbf{x}_i, \mathbf{x}_j) + (1 - \mathbf{1}_{y_i=y_j}) \log(1 - p(\mathbf{x}_i, \mathbf{x}_j))$$

testing  $\rightarrow$

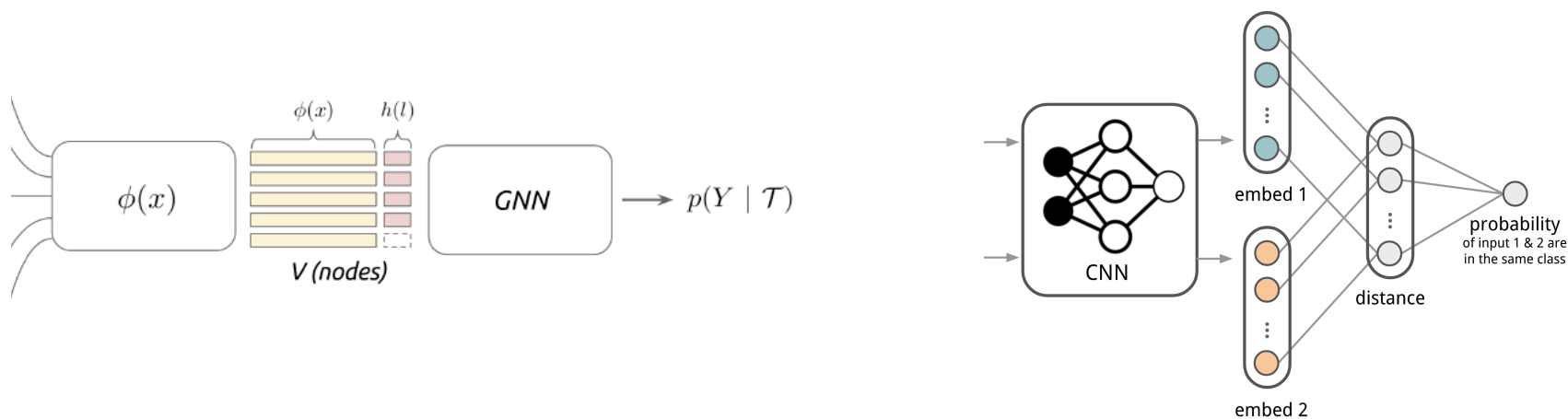
$$\hat{c}_S(\mathbf{x}) = c(\arg \max_{\mathbf{x}_i \in S} P(\mathbf{x}, \mathbf{x}_i))$$

$B = \text{training Batch}$

$S = \text{support set}$

# Siamese Neural Network for one-shot

Koch, Zemel & Salakhutdinov (2015)



$$\mathbf{x}_i^{(0)} = (\phi(x_i), h(l_i))$$

$$f_{\theta}(x_i) = \phi(x_i)$$

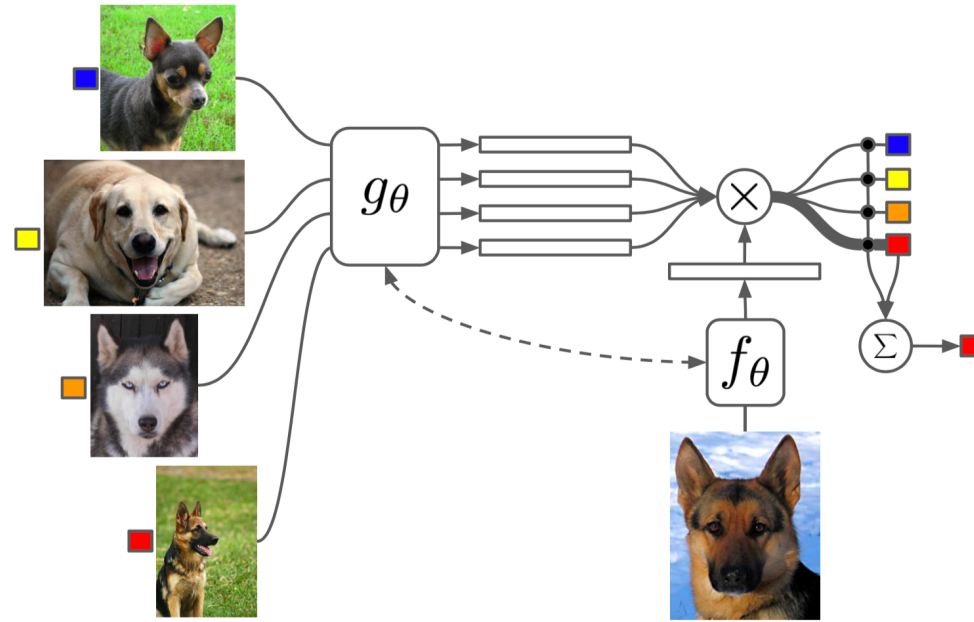
$$\tilde{A}_{i,j} = \varphi_{\tilde{\theta}}(\mathbf{x}_i, \mathbf{x}_j) = \|\phi(x_i) - \phi(x_j)\|, \quad \tilde{A} = \text{softmax}(-\varphi)$$

$$\hat{Y}_* = \sum_j \tilde{A}_{*,j}^{(0)} \langle \mathbf{x}_j^{(0)}, u \rangle$$



# Matching Network

Vinyals et al., 2016



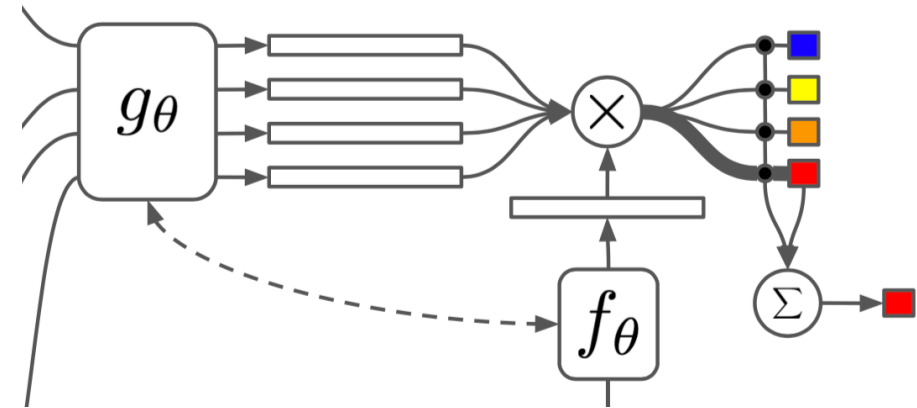
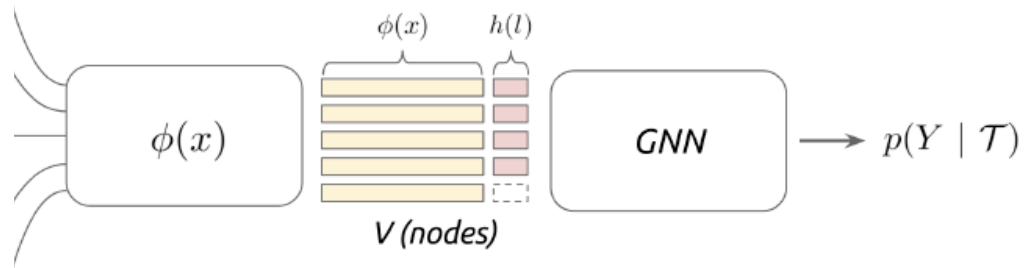
$$c_S(\mathbf{x}) = P(y|\mathbf{x}, S) = \sum_{i=1}^k a(\mathbf{x}, \mathbf{x}_i) y_i, \text{ where } S = \{(\mathbf{x}_i, y_i)\}_{i=1}^k$$

$$a(\mathbf{x}, \mathbf{x}_i) = \frac{\exp(\text{cosine}(f(\mathbf{x}), g(\mathbf{x}_i)))}{\sum_{j=1}^k \exp(\text{cosine}(f(\mathbf{x}), g(\mathbf{x}_j)))}$$

$$\theta = \arg \max_{\theta} E_{L \sim T} \left[ E_{S \sim L, B \sim L} \left[ \sum_{(x,y) \in B} \log P_{\theta}(y|x, S) \right] \right]$$

# Matching Network

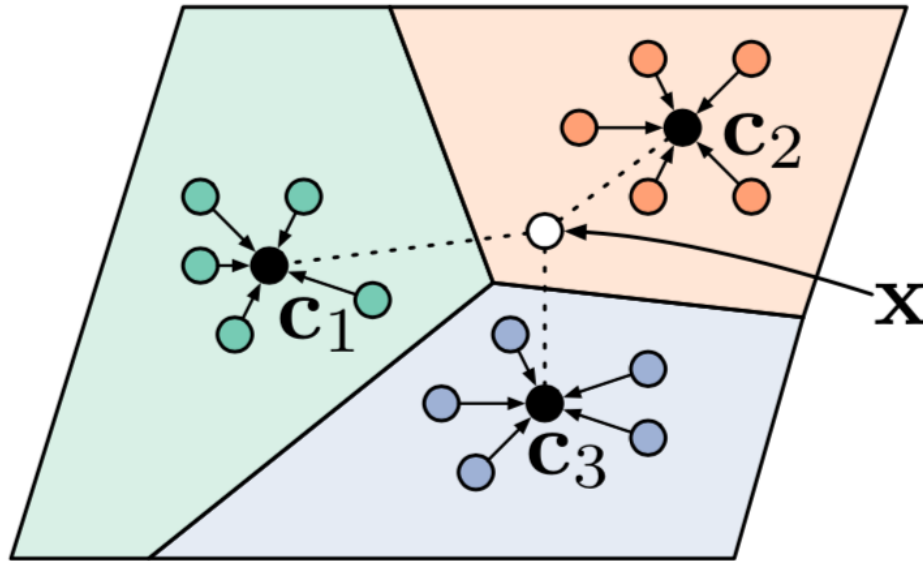
Vinyals et al., 2016



- matching networks consider attention mechanisms of the form  $\tilde{A}_{*,j}^{(k)} = \varphi(\mathbf{x}_*^{(k)}, \mathbf{x}_j^{(T)})$ , where  $\mathbf{x}_j^{(T)}$  is the encoding function for the elements of the support set, obtained with bidirectional LSTMs – independently of the target image
- the label and image fields are treated separately throughout the model, with a final step that aggregates linearly the labels using a trained kernel

# Prototypical Network

Snell, Swersky & Zemel, 2017



$$\mathbf{v}_c = \frac{1}{|S_c|} \sum_{(\mathbf{x}_i, y_i) \in S_c} f_{\theta}(\mathbf{x}_i)$$

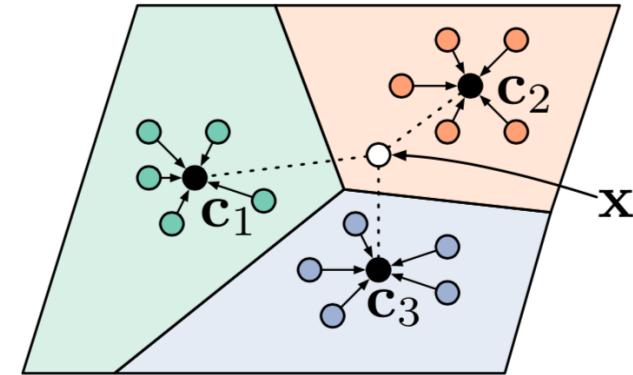
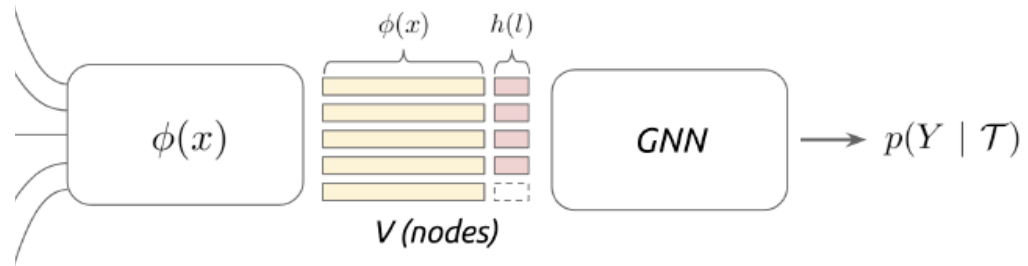
$$\mathcal{L}(\theta) = -\log P_{\theta}(y = c|\mathbf{x}).$$

$$P(y = c|\mathbf{x}) = \text{softmax}(-d_{\varphi}(f_{\theta}(\mathbf{x}), \mathbf{v}_c)) = \frac{\exp(-d_{\varphi}(f_{\theta}(\mathbf{x}), \mathbf{v}_c))}{\sum_{c' \in \mathcal{C}} \exp(-d_{\varphi}(f_{\theta}(\mathbf{x}), \mathbf{v}_{c'}))}$$

distance = squared euclidean distance

# Prototypical Network

Snell, Swersky & Zemel, 2017



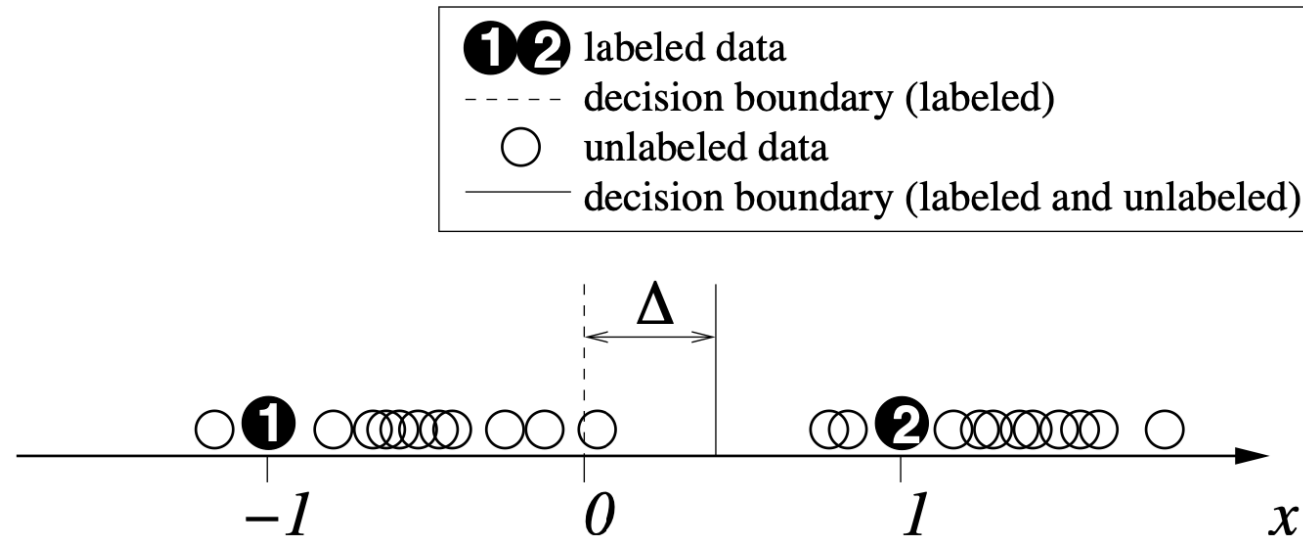
$$\tilde{A}_{i,j}^{(0)} = \begin{cases} q^{-1} & \text{if } l_i = l_j \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{x}_i^{(1)} = \sum_j \tilde{A}_{i,j}^{(0)} \mathbf{x}_j^{(0)}$$

$$\hat{Y}_* = \sum_j \tilde{A}_{*,j}^{(0)} \langle \mathbf{x}_j^{(0)}, u \rangle$$

$q$  is the number of examples per class

# Semi-supervised



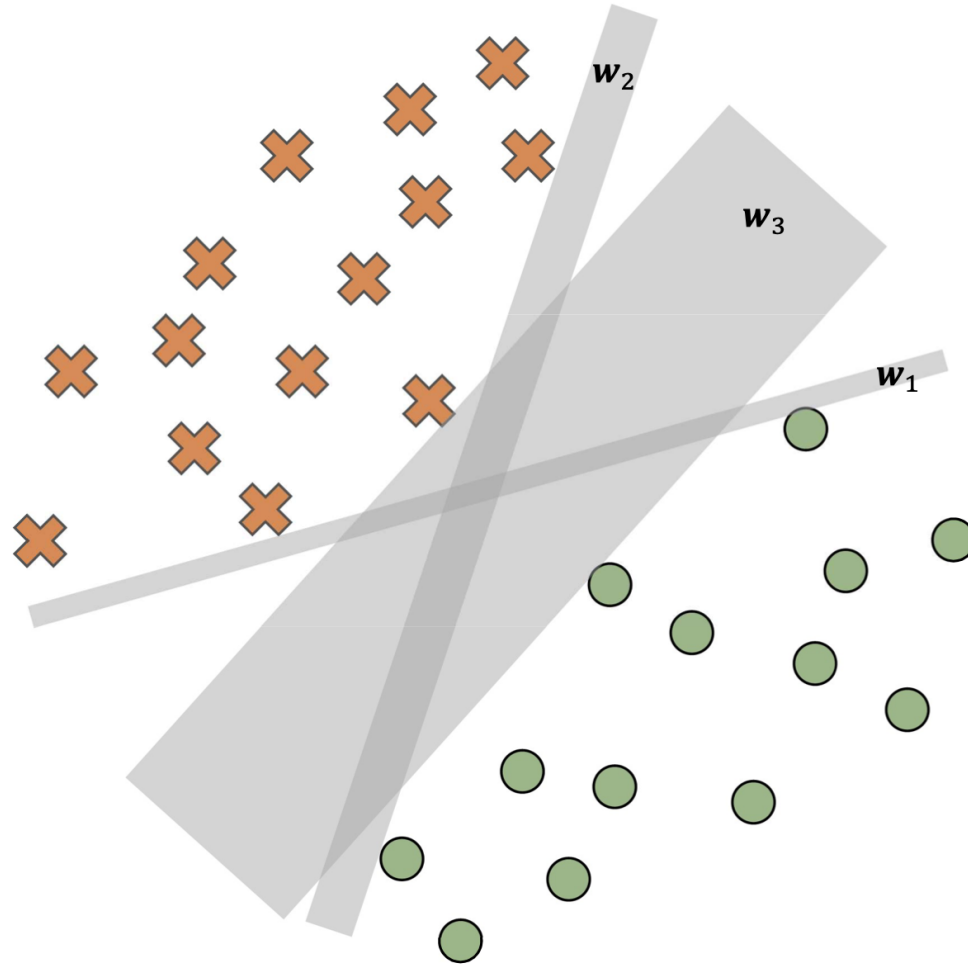
<http://pages.cs.wisc.edu/~jerryzhu/pub/sslicml07.pdf>

# Semi-supervised

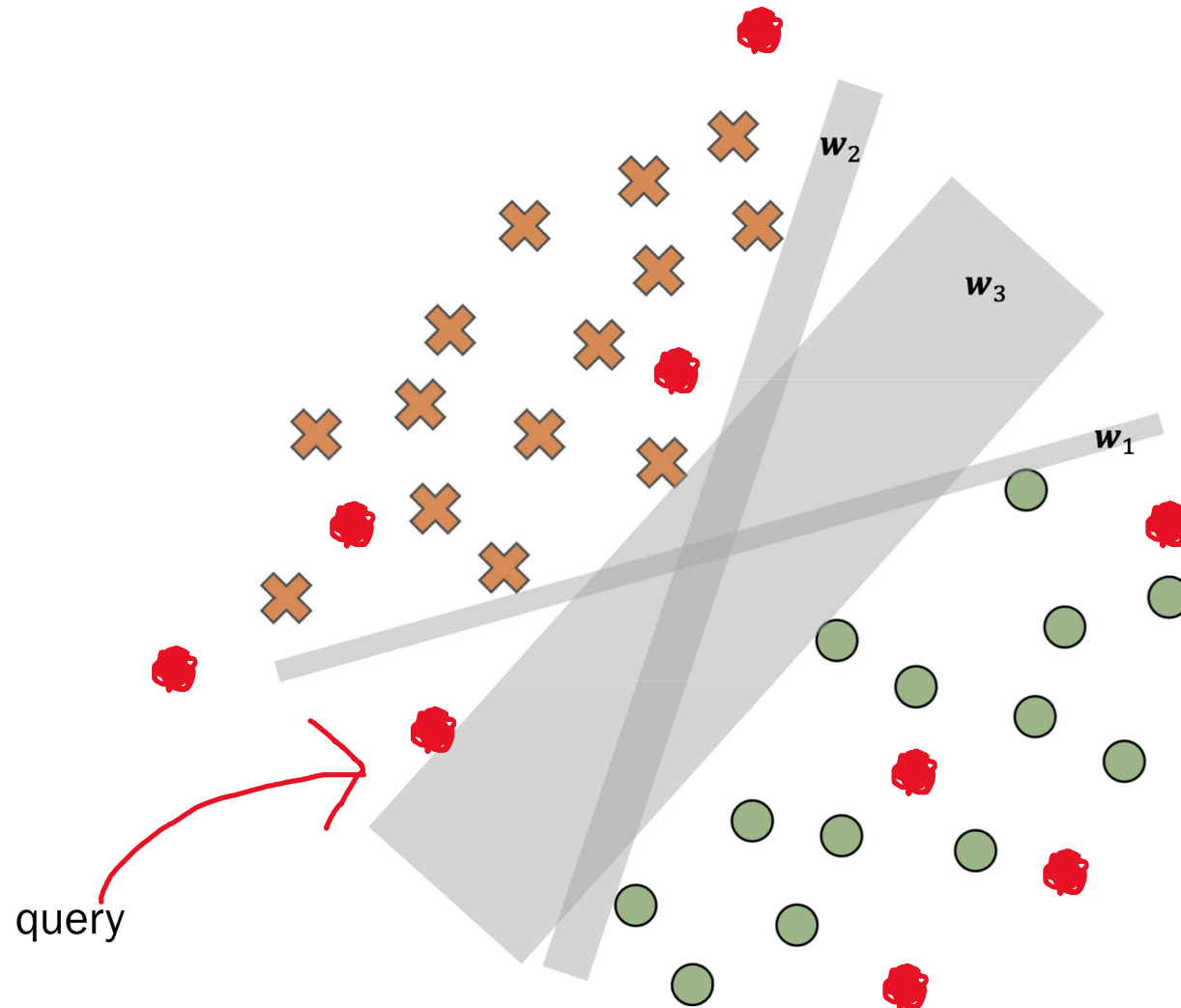
$$\mathcal{T} = \left\{ \underbrace{\{(x_1, l_1), \dots, (x_s, l_s)\}}_{\text{supervised samples}}, \underbrace{\{\tilde{x}_1, \dots, \tilde{x}_r\}}_{\text{unsupervised samples}}, \underbrace{\{\bar{x}_1, \dots, \bar{x}_t\}}_{\text{test samples}}; \right\}$$

- $r > 0$  and  $t = 1$
- Same as few shot
- $h(\ell)$  : Uniform distribution for all  $\{\tilde{x}_1, \dots, \tilde{x}_r\}$

# Active Learning



# Active Learning





# Active Learning

$$\mathcal{T} = \left\{ \underbrace{\{(x_1, l_1), \dots, (x_s, l_s)\}}_{\text{supervised samples}}, \underbrace{\{\tilde{x}_1, \dots, \tilde{x}_r\}}_{\text{unsupervised samples}}, \underbrace{\{\bar{x}_1, \dots, \bar{x}_t\}}_{\text{test samples}}; \right\}$$

- $r > 0$  and  $t = 1$
- Can query label for  $\{\tilde{x}_1, \dots, \tilde{x}_r\}$  after 1<sup>st</sup> layer of GNN using softmax attention node
- Random sampling based on attention multinomial probability is used to choose the queried label during training. Argmax is used at the test time.
- Intuition: The network will learn to ask for the most informative label in order to classify the target sample.

$$\text{Attention} = \text{Softmax}(g(\mathbf{x}_{\{1, \dots, r\}}^{(1)}))$$

$$w \cdot h(l_{i^*}) = \langle \text{Attention}', h(l_{\{1, \dots, r\}}) \rangle$$

$$\mathbf{x}_{i^*}^{(1)} = [\mathbf{Gc}(\mathbf{x}_{i^*}^{(0)}), \mathbf{x}_{i^*}^{(0)}] = [\mathbf{Gc}(\mathbf{x}_{i^*}^{(0)}), (\phi(x_{i^*}), h(l_{i^*}))]$$

# Experiments

## Omniglot: Few shot Learning

| Model   | 5-Way              |                    | 20-Way             |                    |
|---|--------------------|--------------------|--------------------|--------------------|
|   | 1-shot             | 5-shot             | 1-shot             | 5-shot             |
| <b>Pixels</b> Vinyals et al. (2016)               | 41.7%              | 63.2%              | 26.7%              | 42.6%              |
| <b>Siamese Net</b> Koch et al. (2015)             | 97.3%              | 98.4%              | 88.2%              | 97.0%              |
| <b>Matching Networks</b> Vinyals et al. (2016)    | 98.1%              | 98.9%              | 93.8%              | 98.5%              |
| <b>N. Statistician</b> Edwards & Storkey (2016)   | 98.1%              | 99.5%              | 93.2%              | 98.1%              |
| <b>Res. Pair-Wise</b> Mehrotra & Dukkipati (2017) | -                  | -                  | 94.8%              | -                  |
| <b>Prototypical Networks</b> Snell et al. (2017)  | 97.4%              | 99.3%              | 95.4%              | 98.8%              |
| <b>ConvNet with Memory</b> Kaiser et al. (2017)   | 98.4%              | 99.6%              | 95.0%              | 98.6%              |
| <b>Agnostic Meta-learner</b> Finn et al. (2017)   | 98.7 $\pm$ 0.4%    | 99.9 $\pm$ 0.3%    | 95.8 $\pm$ 0.3%    | 98.9 $\pm$ 0.2%    |
| <b>Meta Networks</b> Munkhdalai & Yu (2017)       | 98.9%              | -                  | 97.0%              | -                  |
| <b>TCML</b> Mishra et al. (2017)                  | 98.96% $\pm$ 0.20% | 99.75% $\pm$ 0.11% | 97.64% $\pm$ 0.30% | 99.36% $\pm$ 0.18% |
| <b>Our GNN</b>                                    | 99.2%              | 99.7%              | 97.4%              | 99.0%              |

- The TCML approach from Mishra et al. (2017) is in the same confidence interval for 3 out of 4 experiments, but it is slightly better for the 20-Way 5-shot, although the number of parameters is reduced from  $\sim 5\text{M}$  (TCML) to  $\sim 300\text{K}$  (3 layers GNN).

# Experiments

Mini-Imagenet: Few shot Learning

| Model   | 5-Way              |                    |
|---|--------------------|--------------------|
|   | 1-shot             | 5-shot             |
| <b>Matching Networks</b> Vinyals et al. (2016)        | 43.6%              | 55.3%              |
| <b>Prototypical Networks</b> Snell et al. (2017)      | 46.61% $\pm$ 0.78% | 65.77% $\pm$ 0.70% |
| <b>Model Agnostic Meta-learner</b> Finn et al. (2017) | 48.70% $\pm$ 1.84% | 63.1% $\pm$ 0.92%  |
| <b>Meta Networks</b> Munkhdalai & Yu (2017)           | 49.21% $\pm$ 0.96  | -                  |
| <b>Ravi &amp; Larochelle</b> Ravi & Larochelle (2016) | 43.4% $\pm$ 0.77%  | 60.2% $\pm$ 0.71%  |
| <b>TCML</b> Mishra et al. (2017)                      | 55.71% $\pm$ 0.99% | 68.88% $\pm$ 0.92% |
| <b>Our metric learning + KNN</b>                      | 49.44% $\pm$ 0.28% | 64.02% $\pm$ 0.51% |
| <b>Our GNN</b>  | 50.33% $\pm$ 0.36% | 66.41% $\pm$ 0.63% |

- Our metric learning + KNN – no aggregation of nodes
- the TCML architecture in Mini-Imagenet, the number of parameters is reduced from 11M (TCML) to 400K (3 layers GNN).

# Experiments

| Model                           | 20%-labeled | 5-Way 5-shot |              |
|---------------------------------|-------------|--------------|--------------|
|                                 |             | 40%-labeled  | 100%-labeled |
| GNN - Trained only with labeled | 99.18%      | 99.59%       | 99.71%       |
| GNN - Semi supervised           | 99.59%      | 99.63%       | 99.71%       |

Table 3: Semi-Supervised Learning — Omniglot accuracies.

| Model                           | 20%-labeled        | 5-Way 5-shot       |                    |
|---------------------------------|--------------------|--------------------|--------------------|
|                                 |                    | 40%-labeled        | 100%-labeled       |
| GNN - Trained only with labeled | 50.33% $\pm$ 0.36% | 56.91% $\pm$ 0.42% | 66.41% $\pm$ 0.63% |
| GNN - Semi supervised           | 52.45% $\pm$ 0.88% | 58.76% $\pm$ 0.86% | 66.41% $\pm$ 0.63% |

Mini-Imagenet: semi-supervised Learning

- labeled samples are balanced among classes
- In 5-Way 5-shot 20%-labeled setting, the GNN receives as input 1 labeled sample per class and 4 unlabeled samples per class, where as "Trained only with labeled" is equivalent to the 5-way 1-shot learning

# Experiments

Active learning

| Method       | 5-Way 5-shot 20%-labeled | Method       | 5-Way 5-shot 20%-labeled |
|--------------|--------------------------|--------------|--------------------------|
| GNN - AL     | 99.62%                   | GNN - AL     | 55.99% $\pm$ 1.35%       |
| GNN - Random | 99.59%                   | GNN - Random | 52.56% $\pm$ 1.18%       |

Omniglot

Mini-Imagenet

- GNN-AL queries sample for label by Active Learning, GNN - Random queries sample randomly.
- 20% of the samples are labeled

THANKS



# Omniglot

**Dataset:** Omniglot is a dataset of 1623 characters from 50 different alphabets, each character/class has been drawn by 20 different people. Following Vinyals et al. (2016) implementation we split the dataset into 1200 classes for training and the remaining 423 for testing. We augmented the dataset by multiples of 90 degrees as proposed by Santoro et al. (2016).

**Architectures:** Inspired by the embedding architecture from Vinyals et al. (2016), following Mishra et al. (2017), a CNN was used as an embedding  $\phi$  function consisting of four stacked blocks of  $\{3 \times 3$ -convolutional layer with 64 filters, batch-normalization,  $2 \times 2$  max-pooling, leaky-relu $\}$  the output is passed through a fully connected layer resulting in a 64-dimensional embedding. For the GNN we used 3 blocks each of them composed by 1) a module that computes the adjacency matrix and 2) a graph convolutional layer. A more detailed description of each block can be found at Figure 3.



# Mini-Imagenet

**Dataset:** Mini-Imagenet is a more challenging dataset for one-shot learning proposed by Vinyals et al. (2016) derived from the original ILSVRC-12 dataset Krizhevsky et al. (2012). It consists of  $84 \times 84$  RGB images from 100 different classes with 600 samples per class. It was created with the purpose of increasing the complexity for one-shot tasks while keeping the simplicity of a light size dataset, that makes it suitable for fast prototyping. We used the splits proposed by Ravi & Larochelle (2016) of 64 classes for training, 16 for validation and 20 for testing. Using 64 classes for training, and the 16 validation classes only for early stopping and parameter tuning.

**Architecture:** The embedding architecture used for Mini-Imagenet is formed by 4 convolutional layers followed by a fully-connected layer resulting in a 128 dimensional embedding. This light architecture is useful for fast prototyping:

$1 \times \{3 \times 3\text{-conv. layer (64 filters), batch normalization, max pool}(2, 2), \text{leaky relu}\},$   
 $1 \times \{3 \times 3\text{-conv. layer (96 filters), batch normalization, max pool}(2, 2), \text{leaky relu}\},$   
 $1 \times \{3 \times 3\text{-conv. layer (128 filters), batch normalization, max pool}(2, 2), \text{leaky relu, dropout}(0.5)\},$   
 $1 \times \{3 \times 3\text{-conv. layer (256 filters), batch normalization, max pool}(2, 2), \text{leaky relu, dropout}(0.5)\},$   
 $1 \times \{ \text{fc-layer (128 filters), batch normalization} \}.$

The two dropout layers are useful to avoid overfitting the GNN in Mini-Imagenet dataset. The GNN architecture is similar than for Omniglot, it is formed by 3 blocks, each block is described at Figure

3