Lecture 24: Recent Progress in Deep Learning

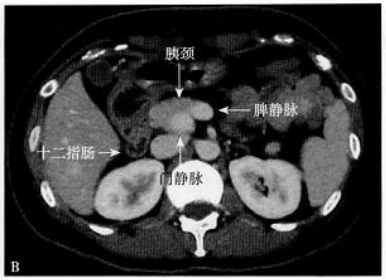
Xuming He
SIST, ShanghaiTech
Fall, 2019

Outline

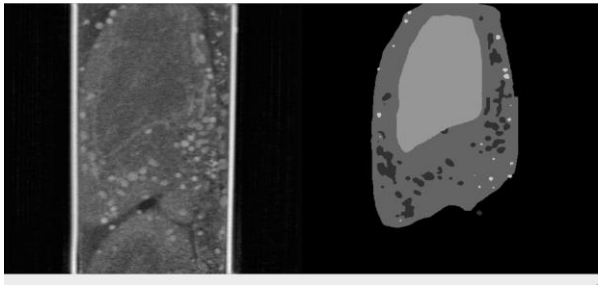
- The limitations of standard deep learning
  - Data efficiency
    - Few-shot learning
  - Irregular structured data
    - Graph neural network
  - Bridging unsupervised and supervised learning
    - Self-supervised representation learning

Real-world scenarios

- Data annotation is costly
  - Many specific domain and cross modality tasks



Medical image understanding
(image credit: 廖飞. 胰腺影像学. 2015.)

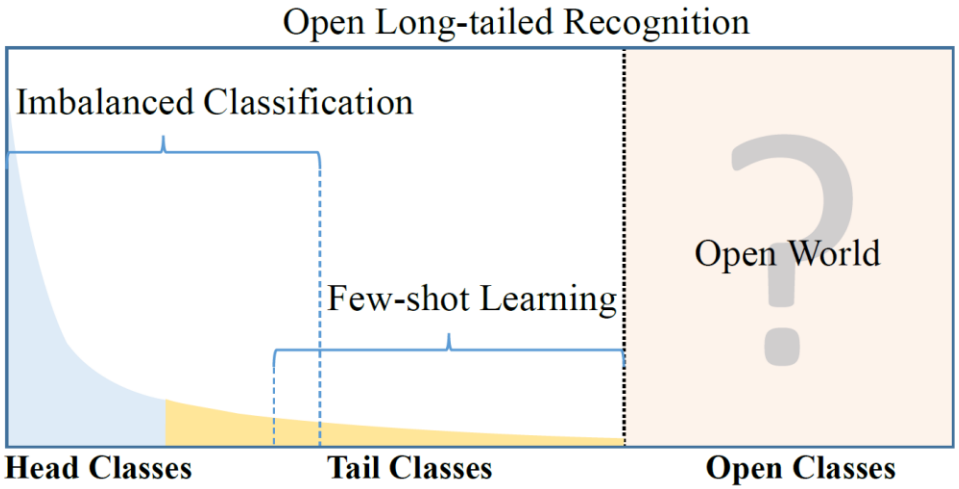


Biological image analysis
(Zhang and He, 2019)

A house cat laying on a couch beside a remote.

Vision & Language (MSCOCO)

- Visual concept learning in wild



(Liu et al CVPR 2019)

Challenges

- Limitation in naïve transfer learning
  - Insufficient instance variations of novel classes
  - Fine-tuning usually fails given a few examples per class

Image Credit: Ravi & Larochelle et al 2017

- Human (child) performance is much better
  - How do we achieve such data efficiency?
  - What representations are used?
  - What are the underlying learning algorithms?

Main intuitions in few-shot learning

- Prior knowledge in different vision tasks
  - Similarity between visual categories
    - Feature representations, etc.
  - Similarity between visual recognition tasks
    - Learning a classifier, etc.

- Focusing on generic aspects of similar tasks
  - Generic visual representations
    - Not category-specific
  - Transferrable learning strategies
    - Very data-efficient

Few-shot learning problem

- Learning from (very) limited annotated data
- Typical setting:
  - Classification using a few training examples per visual category
  - Formally, given a small dataset $D_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$
    - N categories $y_i \in \mathcal{Y}, |\mathcal{Y}| = N,$
    - K shot: each class has K examples, or $L = N \times K$
  - The goal is to learn a model *F* parametrized by θ to minimize

$$E_{D_{test}} \left[\text{loss}(y_i, F_\theta(x_i))\right]$$

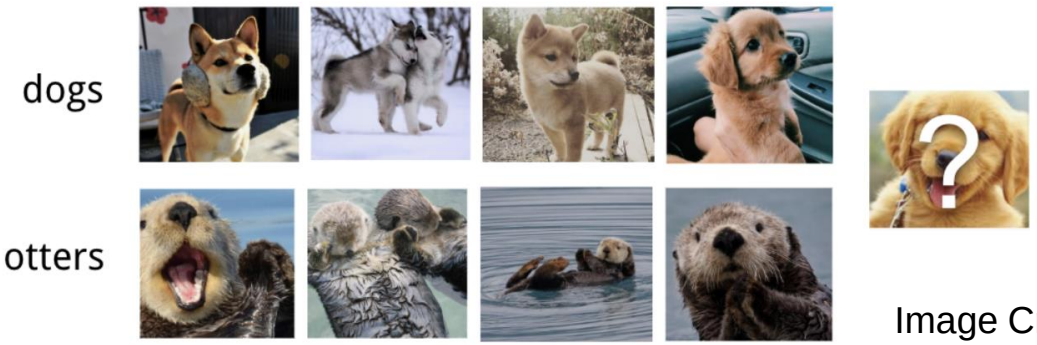


Image Credit: Weng, Lil-log, 2018

Few-shot learning problem

- For a single isolated task, this is difficult
  - But *if* we have access to many *similar* few-shot learning tasks, we can exploit such prior knowledge.
- Main idea is to consider task-level learning
  - Learn a representation shared by all those tasks
  - Learn an efficient classifier learning algorithm that can be applied to all the tasks

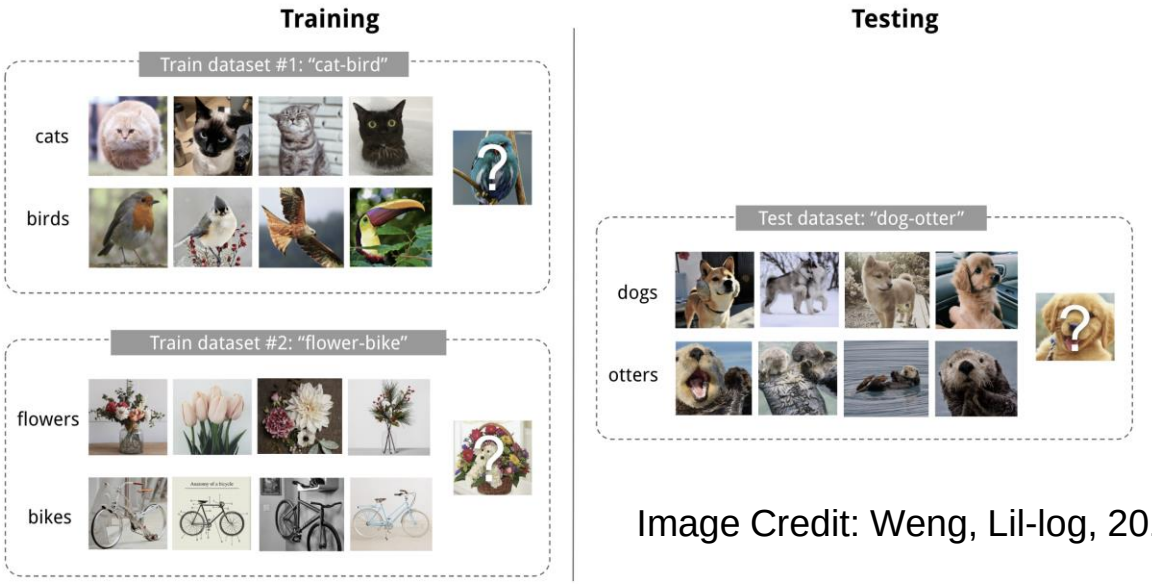


Image Credit: Weng, Lil-log, 2018

Meta-learning framework

- Problem formulation
  - Each few-shot classification problem as a task

Each Task: $T \in \mathcal{T}$ $\quad T \sim P(T)$

  - Each task (or an *episode*) consists of

$$T = (D_{train}, D_{test}, \mathcal{Y}_T)$$

  - Task-train (*support*) set

$$D_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{L} \quad \forall y_i \in \mathcal{Y}_T$$

  - Task-test set (query) D_{test}
  - For each task, we adopt an learning algorithm A_ϕ
    - to learn its own classifier F_θ via $F_\theta = A_\phi(D_{train})$
    - to perform well on the task-test set D_{test}

Meta-learning formulation

- Key assumptions:
  - The learning algorithm A_ϕ is shared across tasks
  - We can sample many tasks to learn a good A_ϕ

- A meta-learning strategy
  - Input: meta-training set $\mathcal{D}_{meta-train} = \{(D_{train}^{(n)}, D_{test}^{(n)})\}_{n=1}^{N}$
  - Output: algorithm parameter ϕ^*
  - Objective: good performance on meta-test set

$$\mathcal{D}_{meta-test} = \{(D_{train}^{\prime(n)}, D_{test}^{\prime(n)})\}_{n=1}^{N'}$$

  - Minimizing the empirical loss on the meta-training set

$$\min_{\phi} E_{\mathcal{D}_{meta-train}} \left[\text{loss}(F_\theta^{(n)}, D_{test}^{(n)}) \right]$$

    - Each meta-train task $F_\theta^{(n)} = A_\phi(D_{train}^{(n)})$

Meta-learning formulation

- Analogy to standard supervised learning

Supervised-Learning	Train	Test	One data point
Meta-learning	Meta-training	Meta-testing	One task

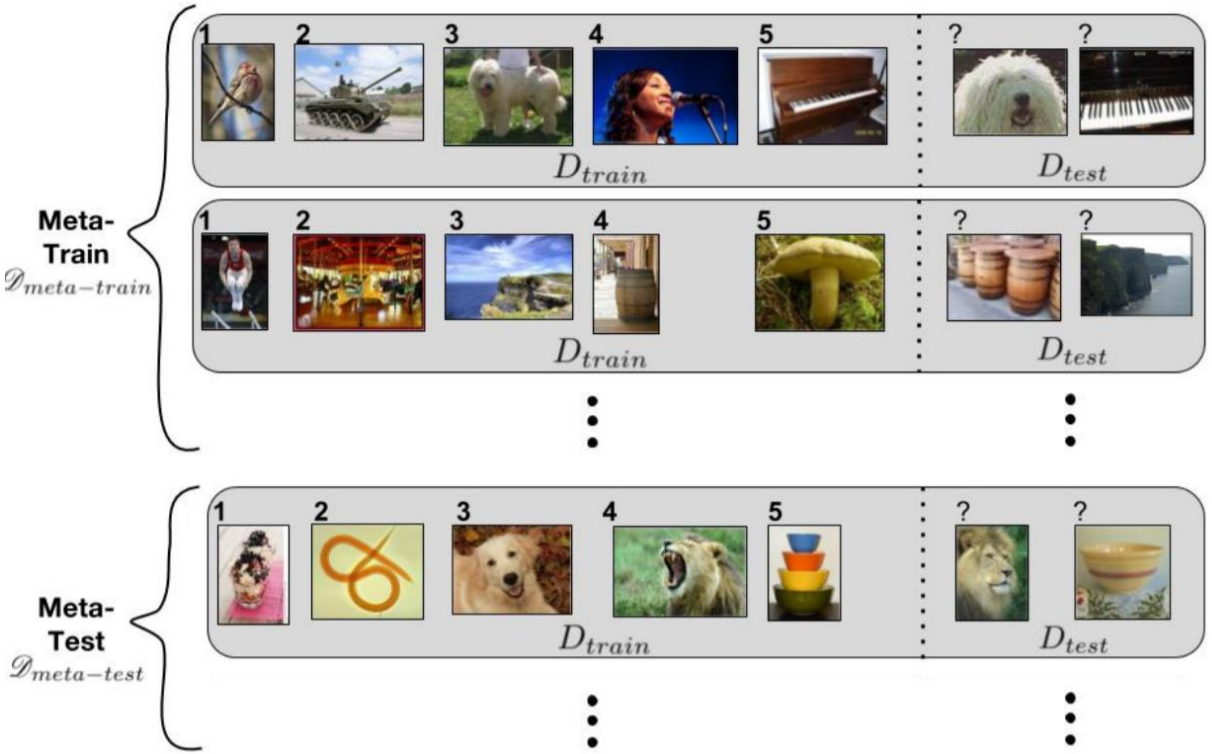
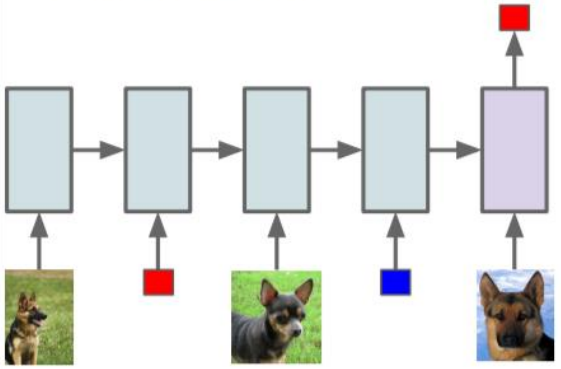


Image Credit: Ravi & Larochelle et al 2017

Overview of existing methods

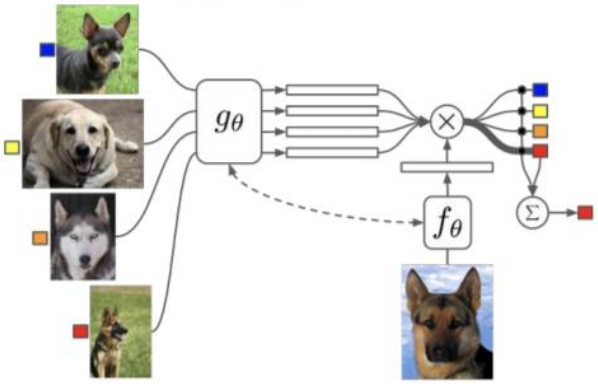
- Depending on the meta-learners used in few-shot tasks

Model Based



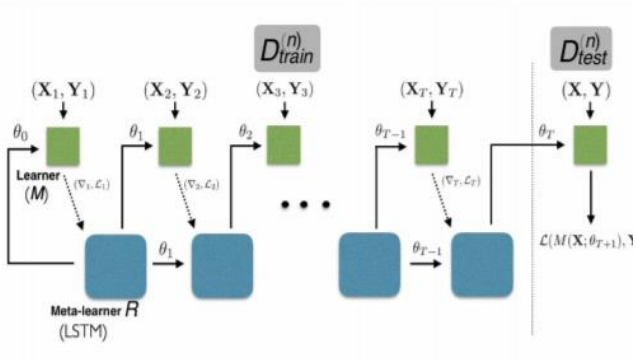
- Santoro et al. ’16
- Duan et al. ’17
- Wang et al. ‘17
- Munkhdalai & Yu ‘17
- Mishra et al. ‘17

Metric Based



- Koch ’15
- Vinyals et al. ‘16
- Snell et al. ‘17
- Shyam et al. ‘17
- Sung et al. ‘17

Optimization Based



- Schmidhuber ’87, ’92
- Bengio et al. ’90, ‘92
- Hochreiter et al. ’01
- Li & Malik ‘16
- Andrychowicz et al. ’16
- Ravi & Larochelle ‘17
- Finn et al. ‘17

Slide Credit: Vinyals, NIPS 2017

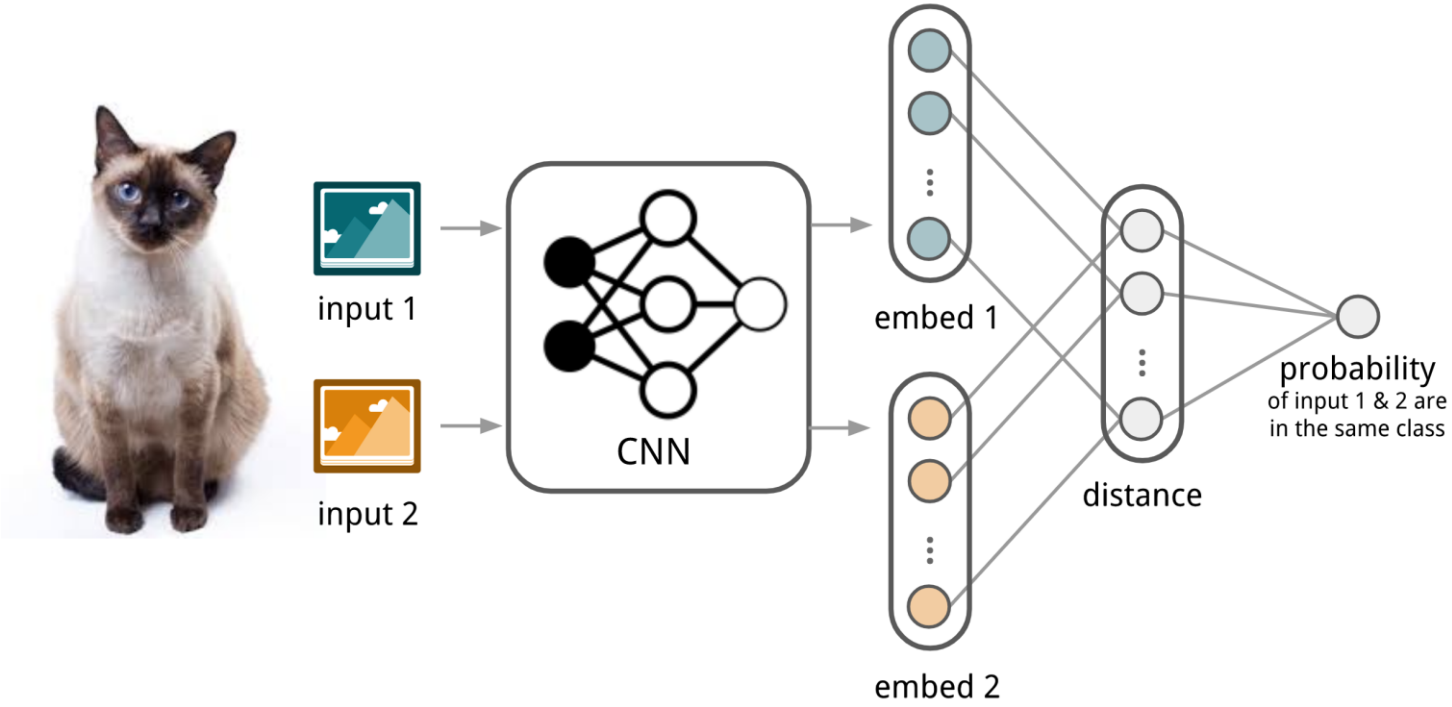
Metric-based methods

- Basic idea: Learn a generic distance metric

$$P_\theta(y|\mathbf{x}, D_{train}) = \sum_{(\mathbf{x}_i, y_i) \in D_{train}} k_\theta(\mathbf{x}, \mathbf{x}_i) y_i$$

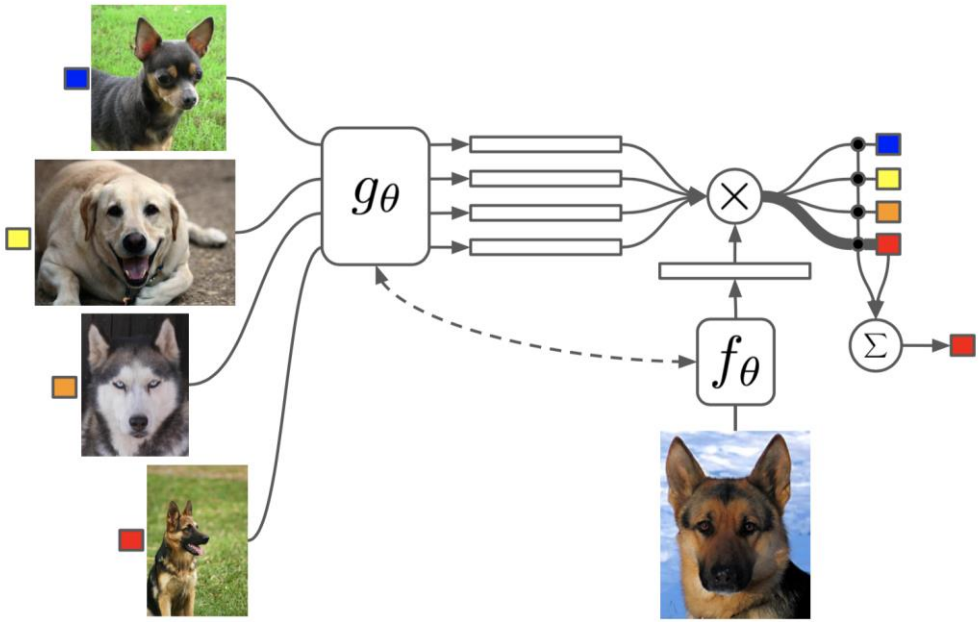
- Typical methods
  - Siamese network (Koch, Zemel & Salakhutdinov, 2015)
  - Matching network (Vinyals et al, 2016)
  - Relation network (Sung et al. 2018)
  - Prototypical network (Snell, Swersky & Zemel, 2017)

Siamese Neural Network



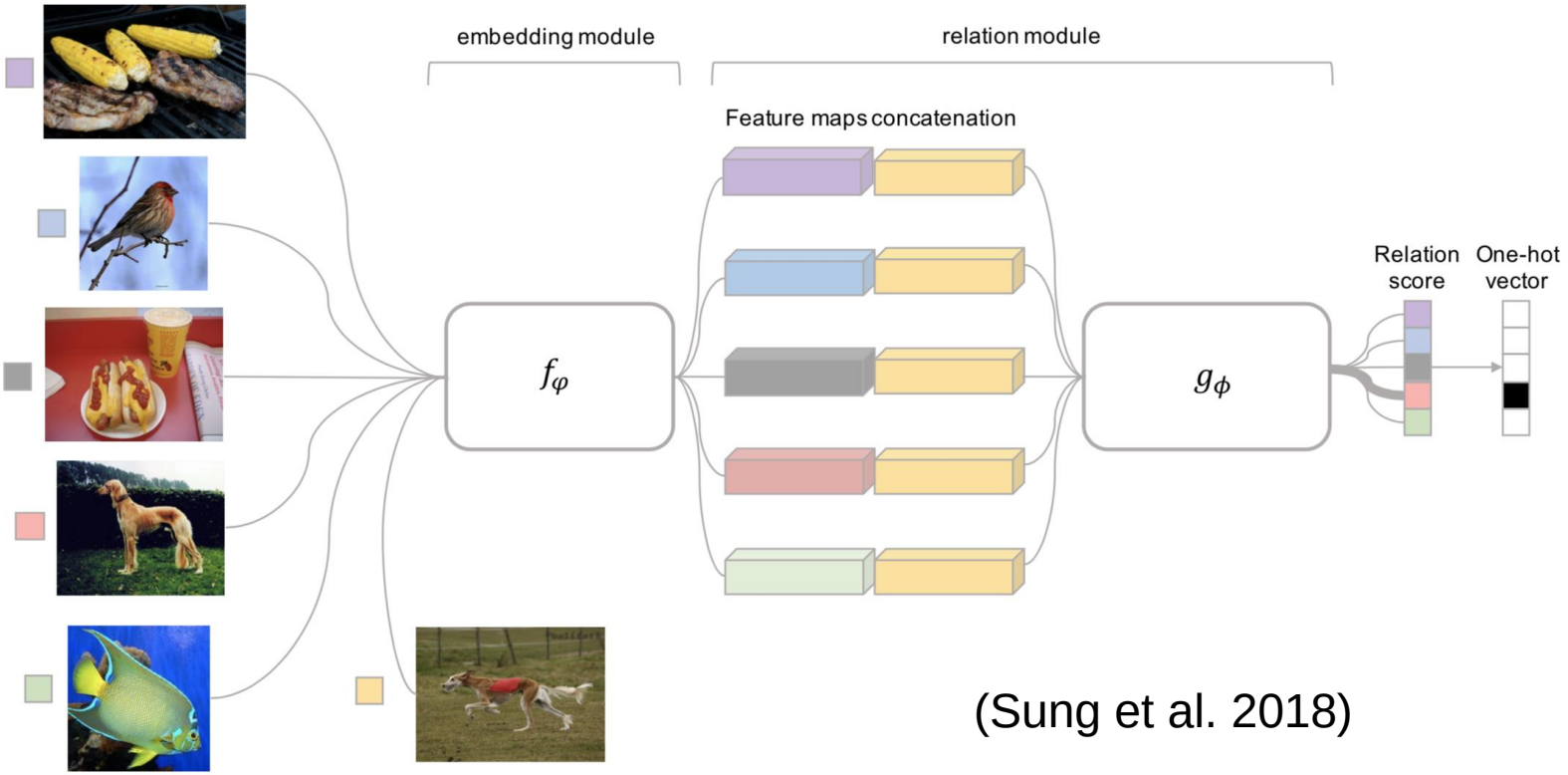
- The learned embedding can be generalized to unknown categories (Koch, Zemel & Salakhutdinov, 2015)

Matching Networks



- Full Contextual Embedding (Vinyals et al, 2016)
  - Encoding input in the context of the entire support set
  - The learned embedding can be adjusted based on the relationship with other support samples.

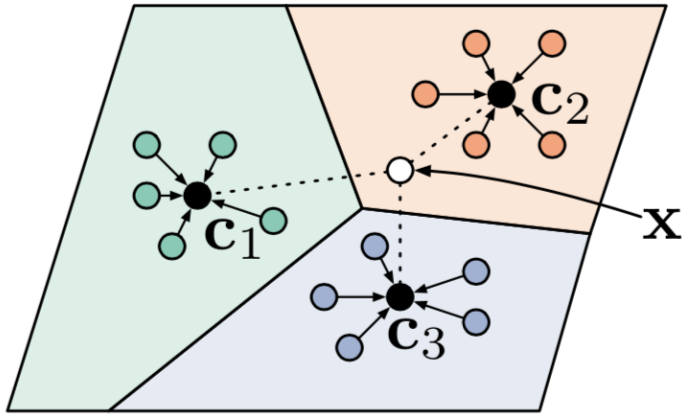
Relation Network



(Sung et al. 2018)

- Similar to Siamese network
- More complex metric learning $r_{ij} = g_\phi([\mathbf{x}_i, \mathbf{x}_j])$

Prototypical Networks



(a) Few-shot

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

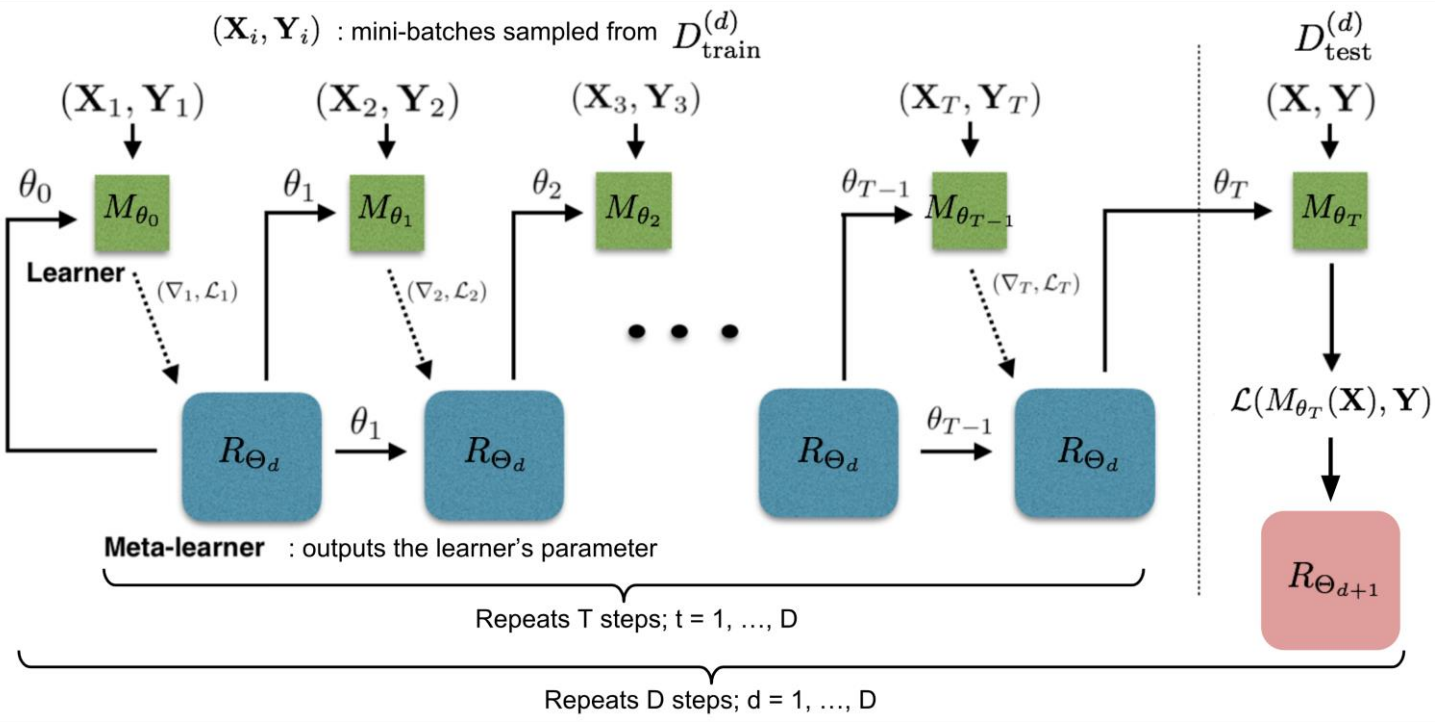
- Prototype vectors (Snell, Swersky & Zemel, 2017)

$$P(y = c|\mathbf{x}) = \text{softmax}(-d_\varphi(f_\theta(\mathbf{x}), \mathbf{v}_c))$$

Optimization-based methods

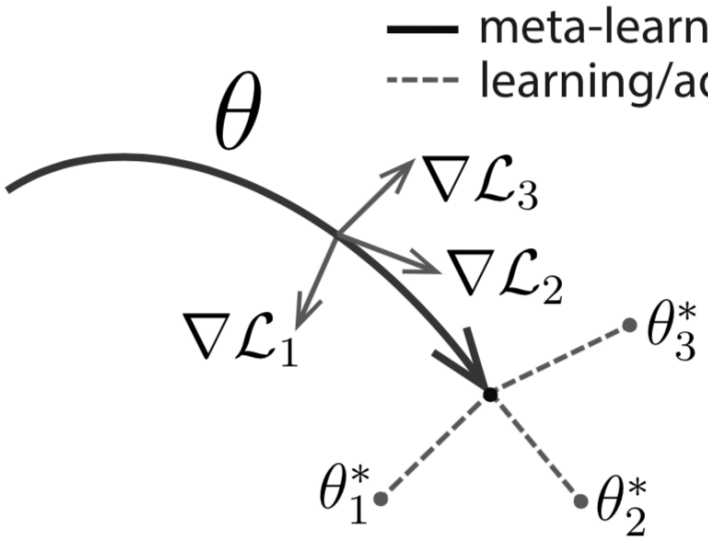
- Basic idea: Adjust the optimization in model learning so that the model can effectively learn from a few examples
- Typical methods
  - LSTM meta-learner (Ravi & Larochelle, 2017)
  - MAML (Finn, et al. 2017)
  - Reptile (Nichol, Achiam & Schulman, 2018)

LSTM meta-learner



- The optimization algorithm is explicitly modeled based on an LSTM meta-learner (Ravi & Larochelle, 2017)

MAML



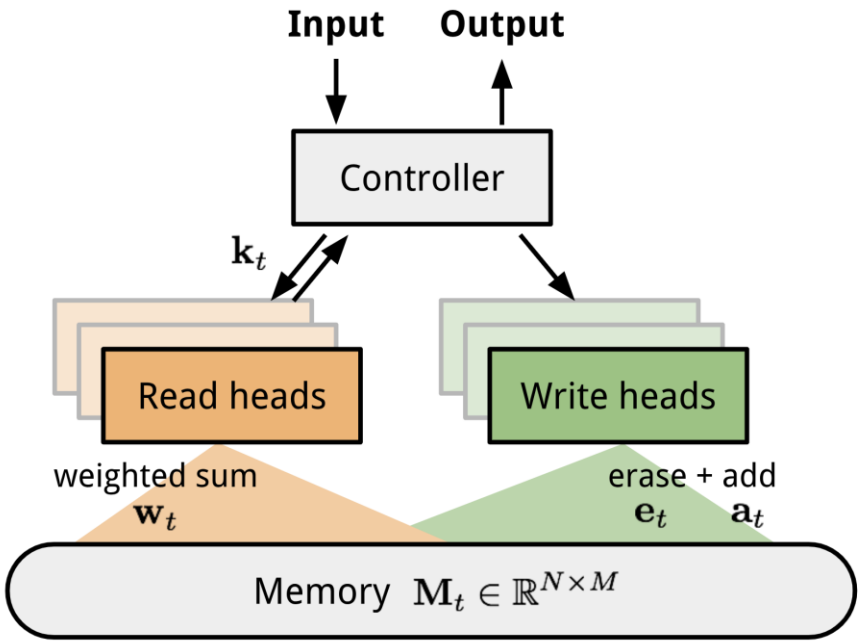
$$\theta^* = \arg\min_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{\tau_i}^{(1)}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}^{(0)}(f_\theta)})$$

- Model-Agnostic Meta-Learning (Finn, et al. 2017) aims to generate a fast gradient based learner

Model-based methods

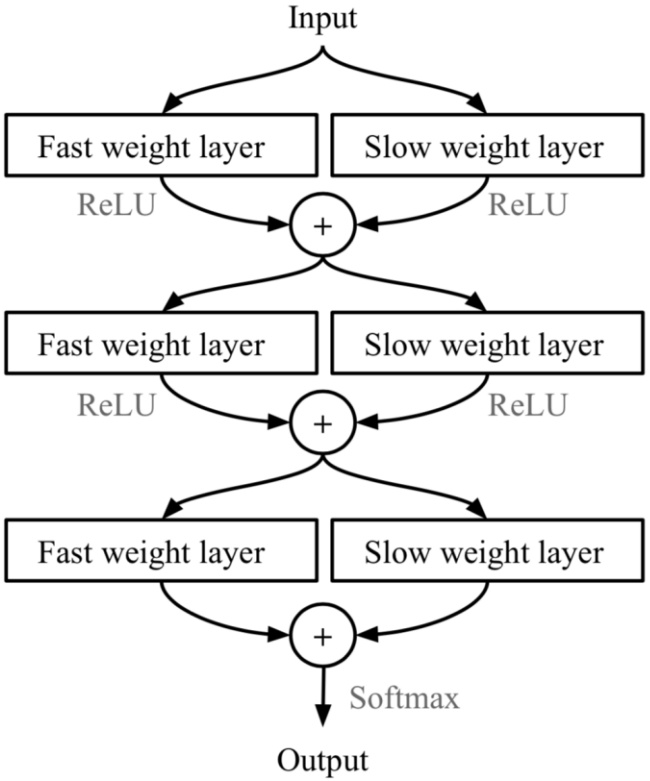
- Basic idea: Using a black-box neural network designed specifically for fast learning
- Typical methods
  - Memory-augmented network (Santoro et al., 2016)
  - Meta networks (Munkhdalai & Yu, 2017)
  - SNAIL (Mishra et al., 2018)

Memory-augmented network



- With an explicit storage buffer, it is easier for the network to rapidly incorporate new information.
- (Santoro et al., 2016) train it in a way that the memory can encode and capture information of new tasks fast and is easily and stably accessible.

Meta networks



- The MetaNet relies on “fast weights” to achieve rapid generalization across tasks (Munkhdalai & Yu, 2017)

Outline

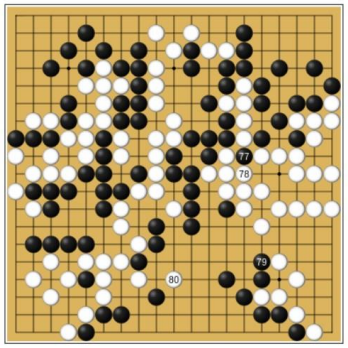
- The limitations of standard deep learning
  - Data efficiency
    - Few-shot learning
  - Irregular structured data
    - Graph neural network
  - Bridging unsupervised and supervised learning
    - Self-supervised representation learning

Traditional **Neural Networks**

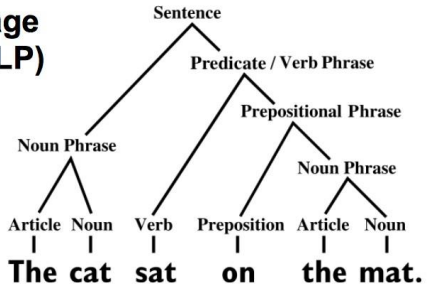
IMAGENET

Speech data

Grid games



Natural language processing (NLP)



...

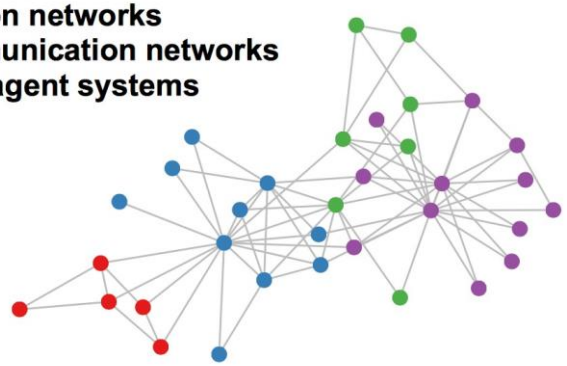
Deep neural nets that exploit:

- translation equivariance (weight sharing)
- hierarchical compositionality

Graph-structured Data

A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

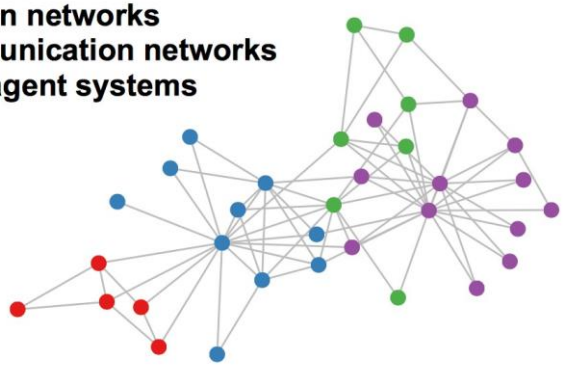


* slide from Thomas Kipf,
University of Amsterdam

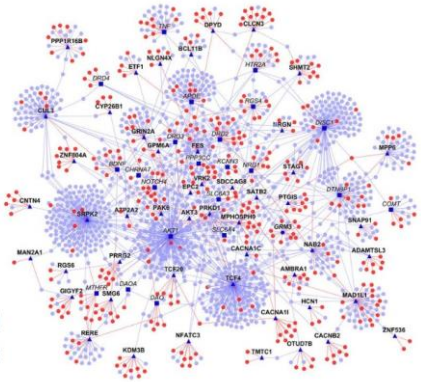
Graph-structured Data

A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems



Protein interaction networks

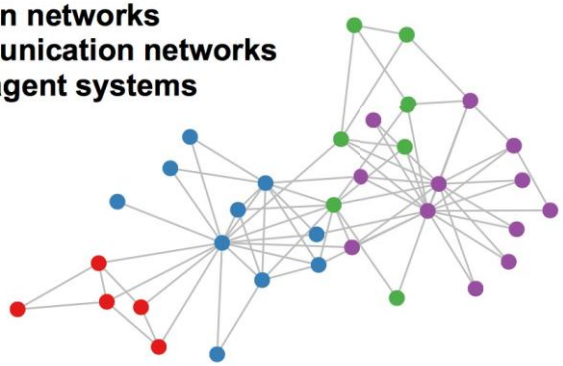


* slide from Thomas Kipf,
University of Amsterdam

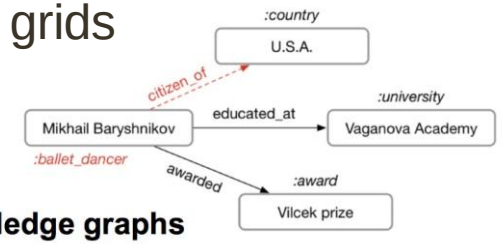
Graph-structured Data

A lot of real-world data does not “live” on grids

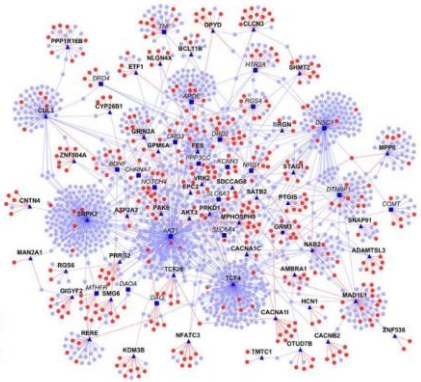
Social networks
Citation networks
Communication networks
Multi-agent systems



ds



Knowledge graphs



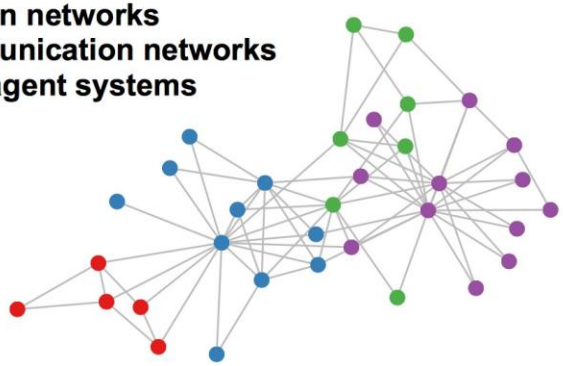
Protein interaction networks

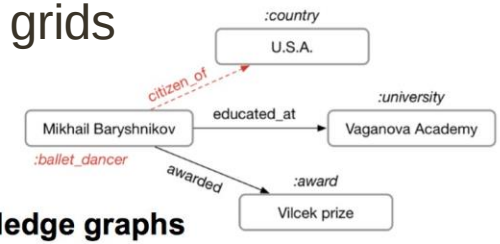
* slide from Thomas Kipf,
University of Amsterdam

Graph-structured Data

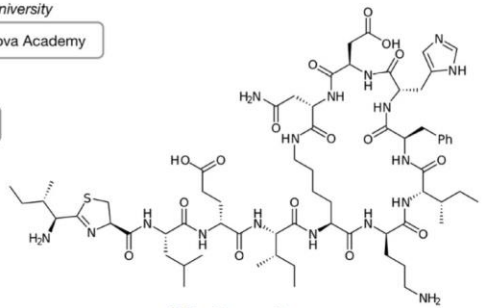
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

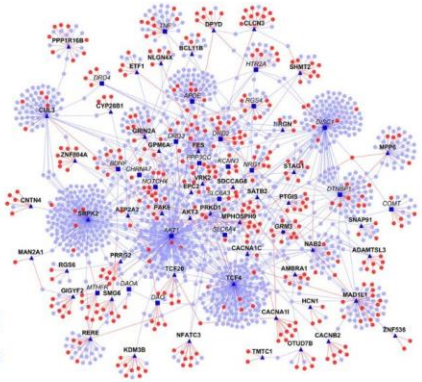




Knowledge graphs



Molecules



Protein interaction networks

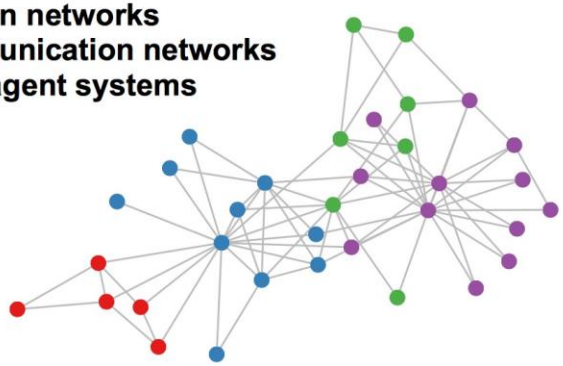
Road maps

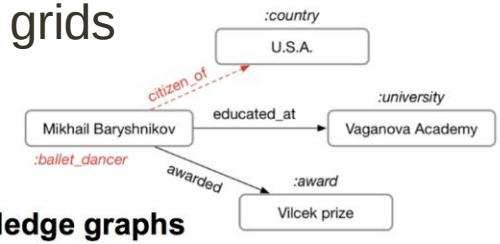
* slide from Thomas Kipf,
University of Amsterdam

Graph-structured Data

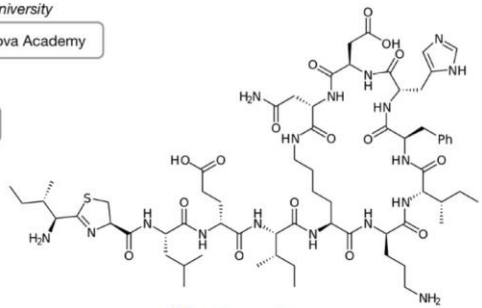
A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

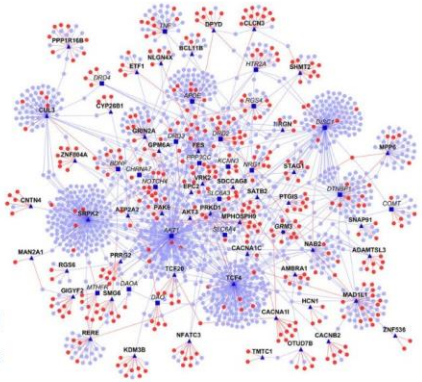




Knowledge graphs



Molecules



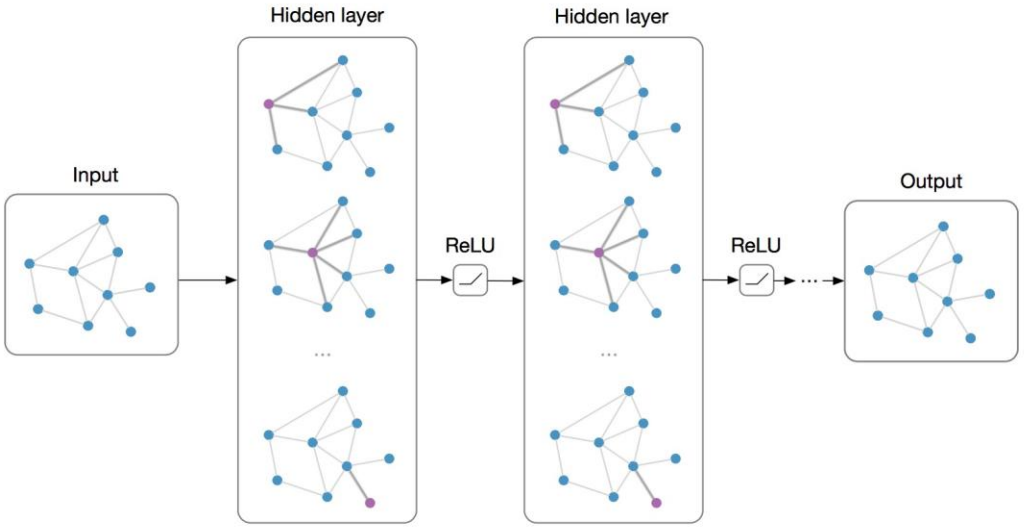
Protein interaction networks

Road maps

Standard **CNN** and **RNN** architectures don’t work on this data

* slide from Thomas Kipf,
University of Amsterdam

Graph Neural Networks (GNNs)

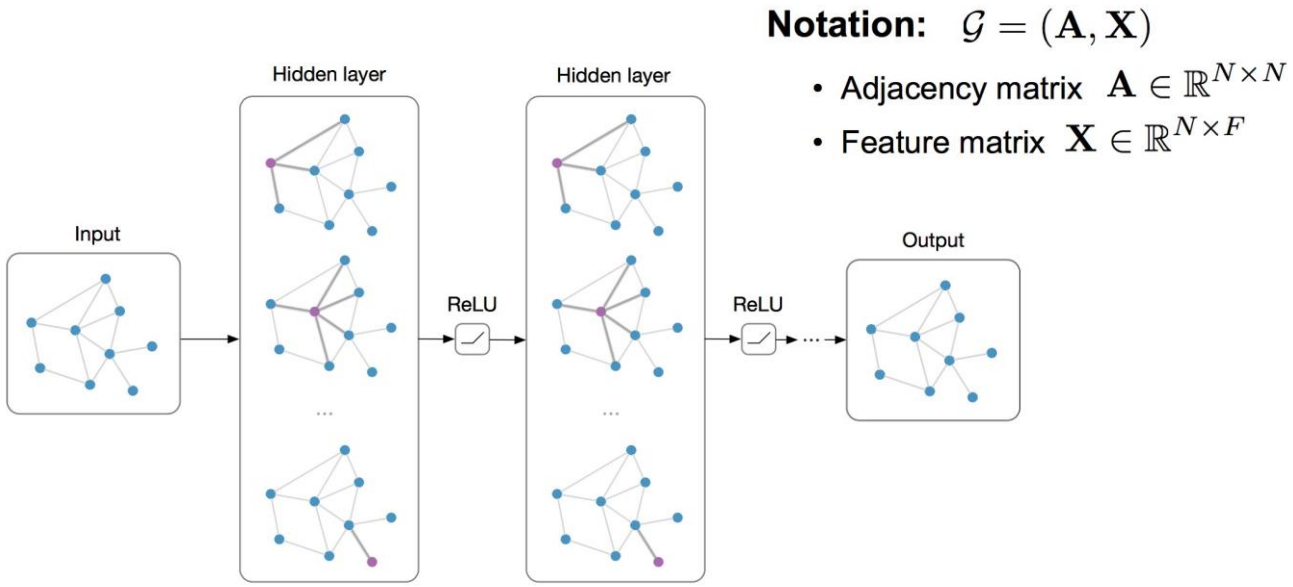


Main Idea: Pass massages between pairs of nodes and agglomerate

Alternative Interpretation: Pass massages between nodes to refine node (and possibly edge) representations

* slide from Thomas Kipf,
University of Amsterdam

Graph Neural Networks (GNNs)



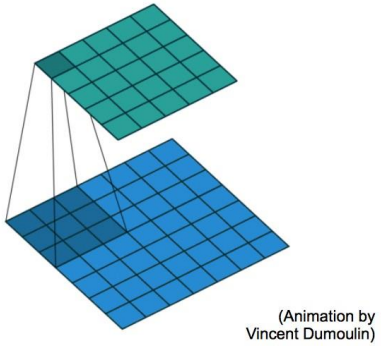
Main Idea: Pass massages between pairs of nodes and agglomerate

Alternative Interpretation: Pass massages between nodes to refine node (and possibly edge) representations

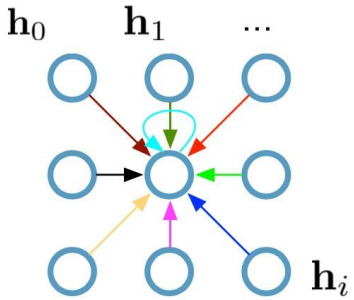
* slide from Thomas Kipf,
University of Amsterdam

Recap: Convolutional Neural Networks (CNNs) on Grids

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)

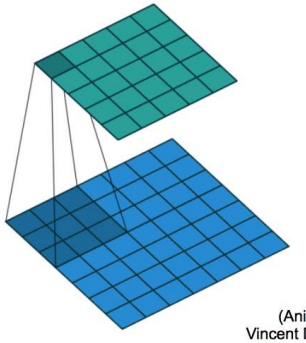


$\mathbf{h}_i \in \mathbb{R}^F$ are (hidd

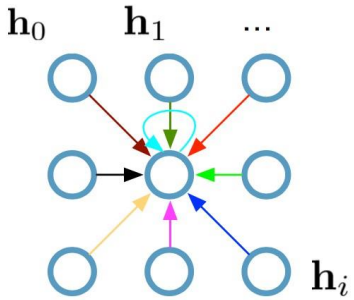
* slide from Thomas Kipf,
University of Amsterdam

Recap: Convolutional Neural Networks (CNNs) on Grids

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

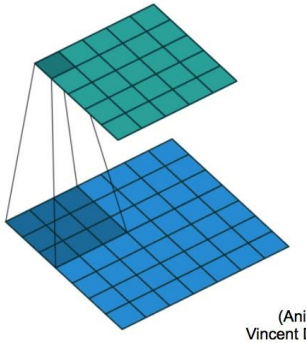
- Transform messages individually $\mathbf{W}_i\mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i\mathbf{h}_i$

$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

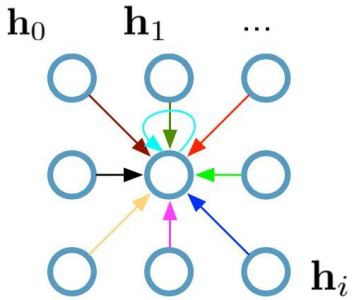
* slide from Thomas Kipf, **University of Amsterdam**

Recap: Convolutional Neural Networks (CNNs) on Grids

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

- Transform messages individually $\mathbf{W}_i\mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i\mathbf{h}_i$

$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:

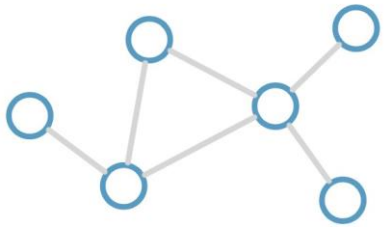
$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

* slide from Thomas Kipf,
University of Amsterdam

Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:

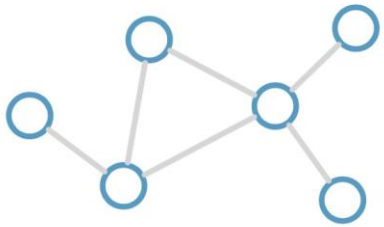


* slide from Thomas Kipf, **University of Amsterdam**

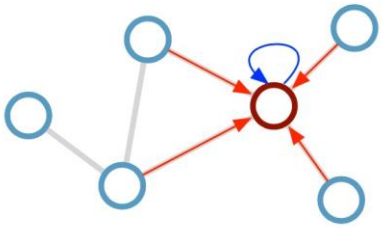
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:

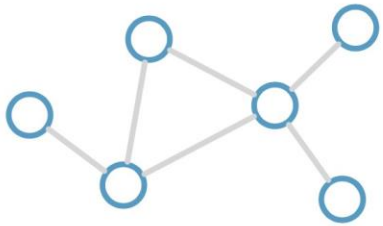


* slide from Thomas Kipf, **University of Amsterdam**

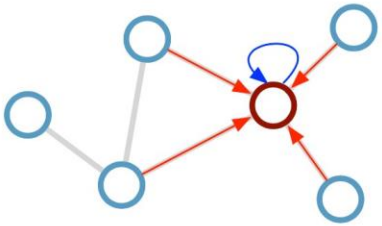
Graph Convolutional Networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



Desirable properties:

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity O(E)
- Applicable both in transductive and inductive settings

Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

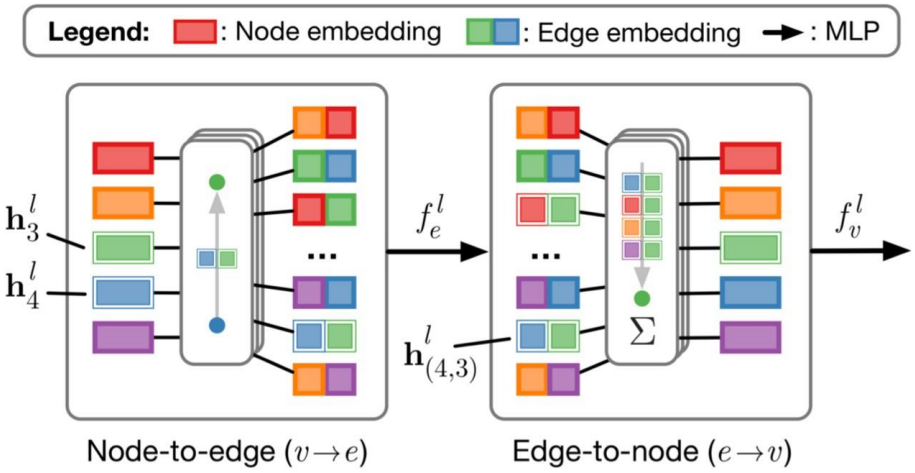
$\mathcal{N}_i$: neighbor indices

c_{ij} : norm. constant (fixed/trainable)

* slide from Thomas Kipf, **University of Amsterdam**

GNNs with **Edge** Embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



Formally:

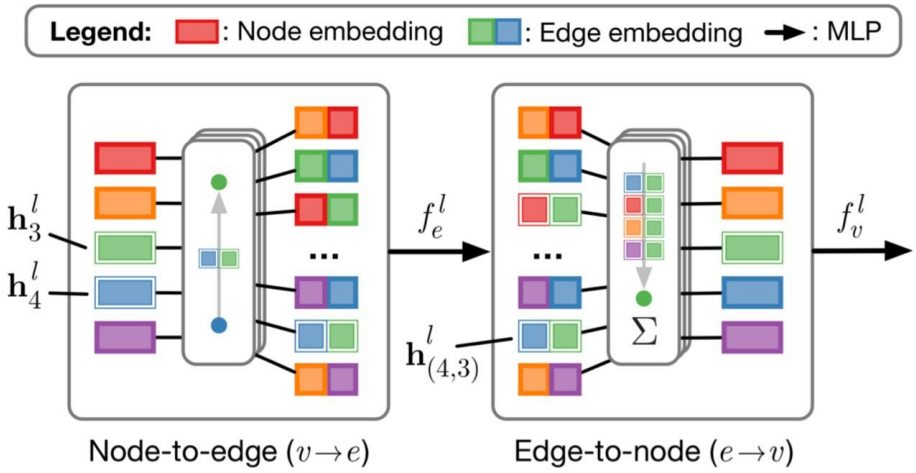
$$v \rightarrow e: \quad \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$$

$$e \rightarrow v: \quad \mathbf{h}_j^{l+1} = f_v^l([\textstyle\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

* slide from Thomas Kipf,
University of Amsterdam

GNNs with **Edge** Embeddings

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)



Formally:

$$v \to e: \quad \mathbf{h}_{(i,j)}^l = f_e^l([\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}])$$

$$e \to v: \quad \mathbf{h}_j^{l+1} = f_v^l([\textstyle\sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j])$$

Pros:

- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

Cons:

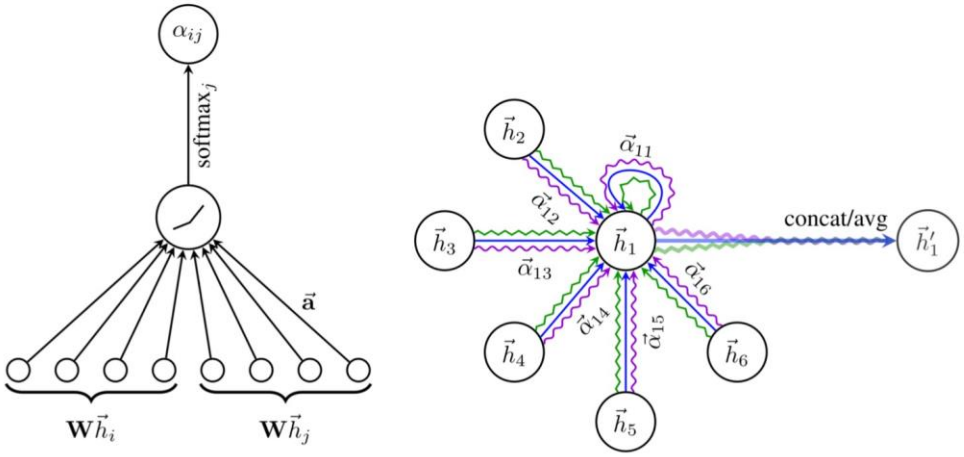
- Need to store intermediate edge-based activations
- Difficult to implement with subsampling

➡ In practice limited to small graphs

* slide from Thomas Kipf, **University of Amsterdam**

Graph Neural Networks (GNNs) with **Attention**

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)



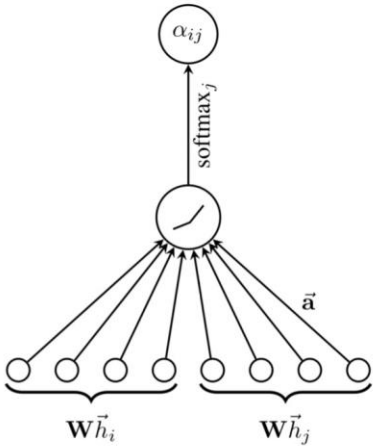
[Figure from Veličković et al. (ICLR 2018)]

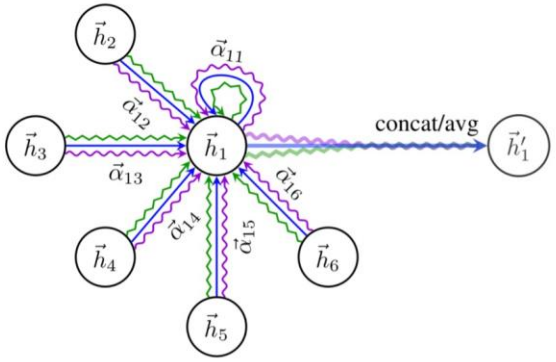
$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

* slide from Thomas Kipf,
University of Amsterdam

Graph Neural Networks (GNNs) with **Attention**

Monti et al. (CVPR 2017), Hoshen (NIPS 2017), Veličković et al. (ICLR 2018)





[Figure from Veličković et al. (ICLR 2018)]

Pros:

- No need to store intermediate edge-based activation vectors (when using dot-product attn.)
- Slower than GCNs but faster than GNNs with edge embeddings

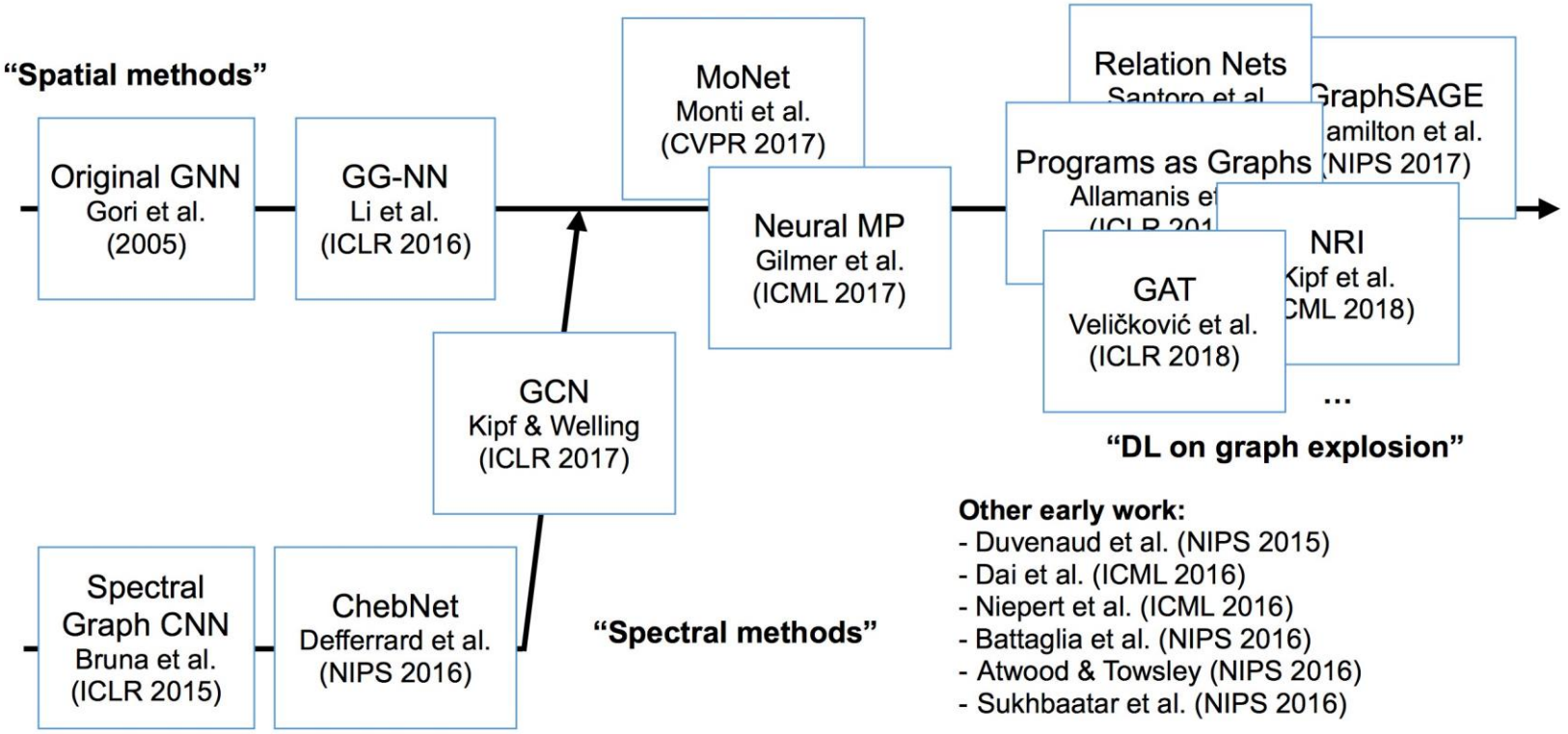
Cons:

- (Most likely) less expressive than GNNs with edge embeddings
- Can be more difficult to optimize

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right) \qquad \alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_k]\right)\right)}$$

* slide from Thomas Kipf,
University of Amsterdam

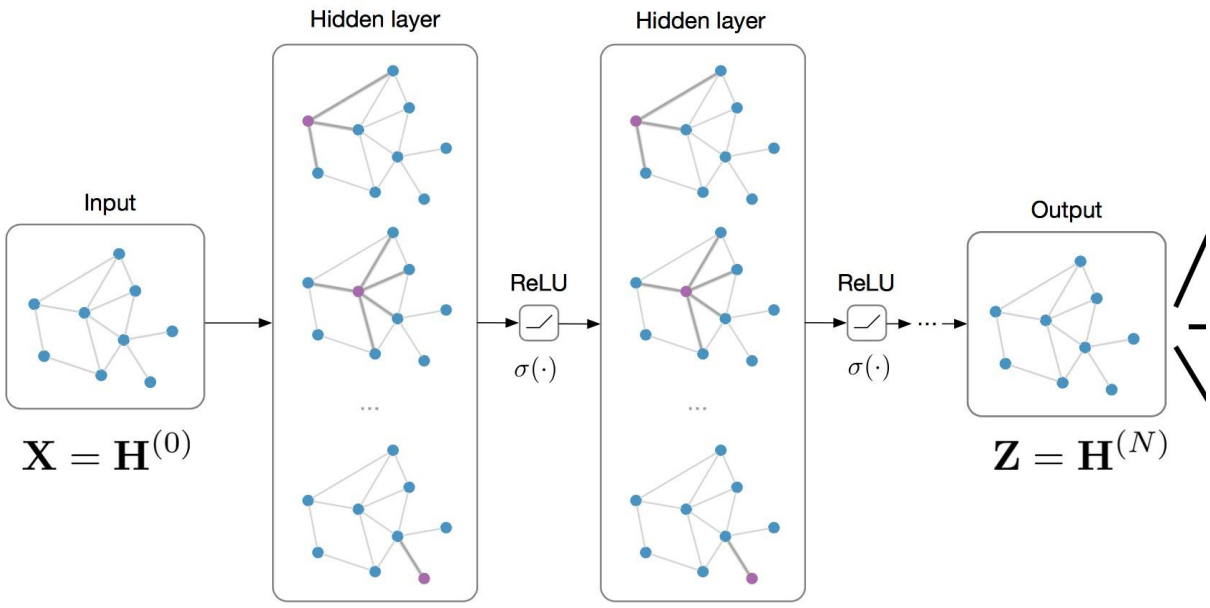
A **Brief History** of Graph Neural Nets



* slide from Thomas Kipf,
University of Amsterdam

Classification and **Link Prediction** with GNNs / GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$

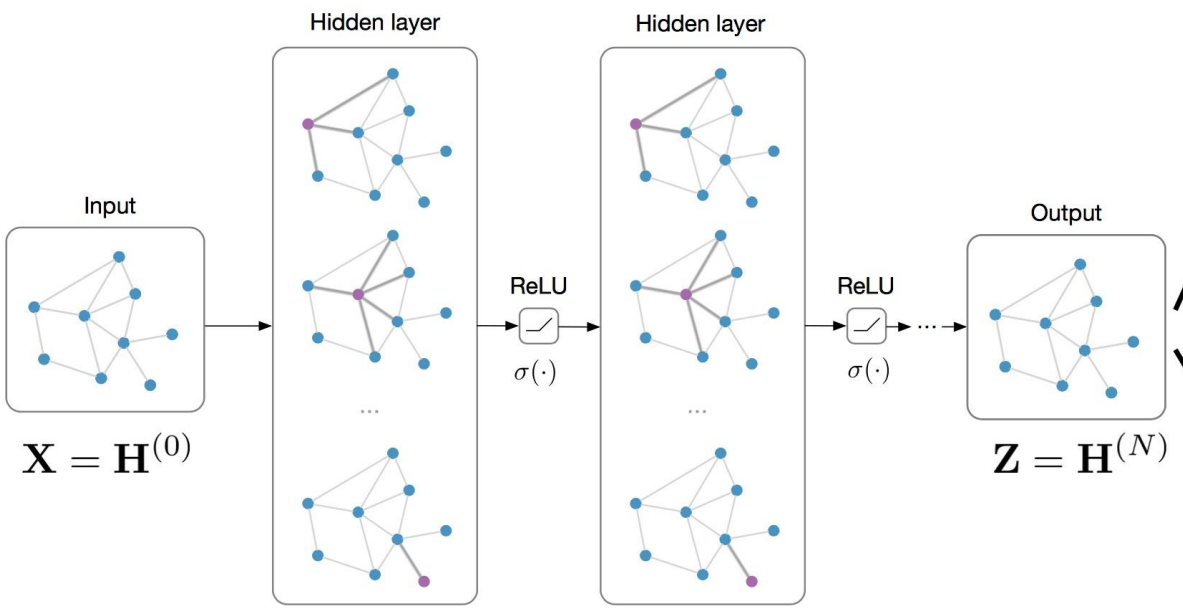


$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

* slide from Thomas Kipf,
University of Amsterdam

Classification and **Link Prediction** with GNNs / GCNs

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Node classification:

$\mathrm{softmax}(\mathbf{z_n})$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$\mathrm{softmax}(\sum_n \mathbf{z_n})$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$

Kipf & Welling (NIPS BDL 2016)

"Graph Auto-Encoders"

* slide from Thomas Kipf,
University of Amsterdam

Outline

- The limitations of standard deep learning
  - Data efficiency
    - Few-shot learning
  - Irregular structured data
    - Graph neural network
  - Bridging unsupervised and supervised learning
    - Self-supervised representation learning

Why Self-Supervision?

1. Expense of producing a new dataset for each new task

2. Some areas are supervision-starved, e.g. medical data, where it is hard to obtain annotation

3. Untapped/availability of vast numbers of unlabelled images/videos
   - Facebook: one billion images uploaded per day
   - 300 hours of video are uploaded to YouTube every minute

4. How infants may learn …

* slide from Andrew Zisserman, **University of Oxford**

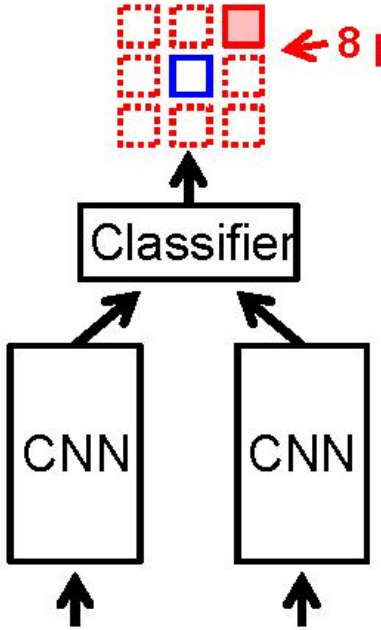
What is Self-Supervision?

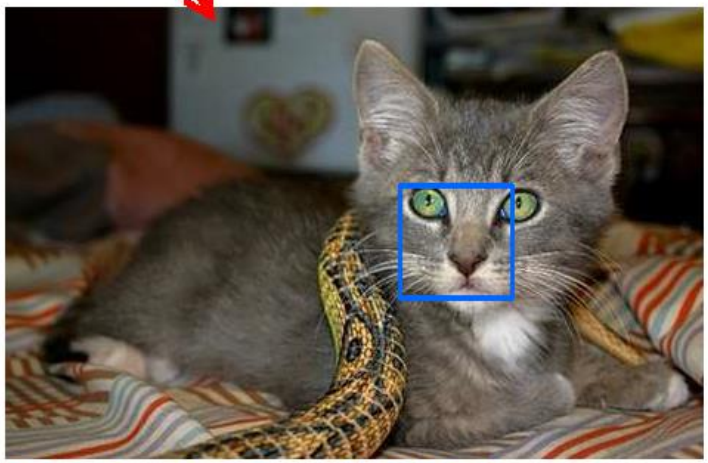
- A form of unsupervised learning where the data provides the **supervision**
- In general, withhold some part of the data, and task the network with predicting it
- The task defines a proxy loss, and the network is forced to learn what we really care about, e.g. a semantic representation, in order to solve it

* slide from Andrew Zisserman, **University of Oxford**

Example: relative positioning

Train network to predict relative position of two regions in the same image





Randomly Sample Patch
Sample Second Patch

Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

* slide from Andrew Zisserman, **University of Oxford**

Semantics from a non-semantic task

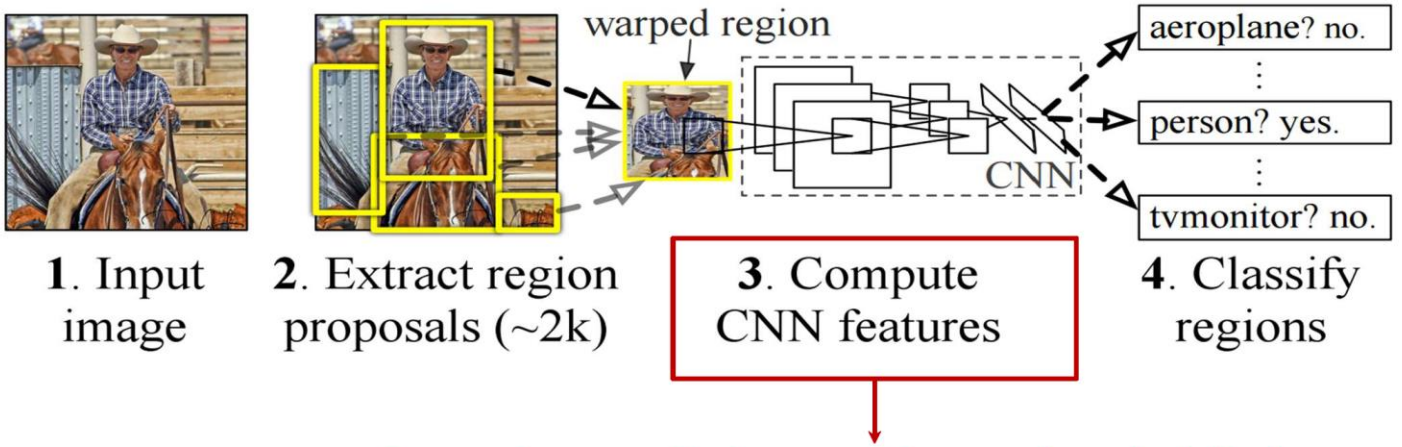
Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

* slide from Andrew Zisserman, **University of Oxford**

Evaluation: PASCAL VOC Detection

- Pre-train CNN using self-supervision (no labels)
- Train CNN for detection in R-CNN object category detection pipeline

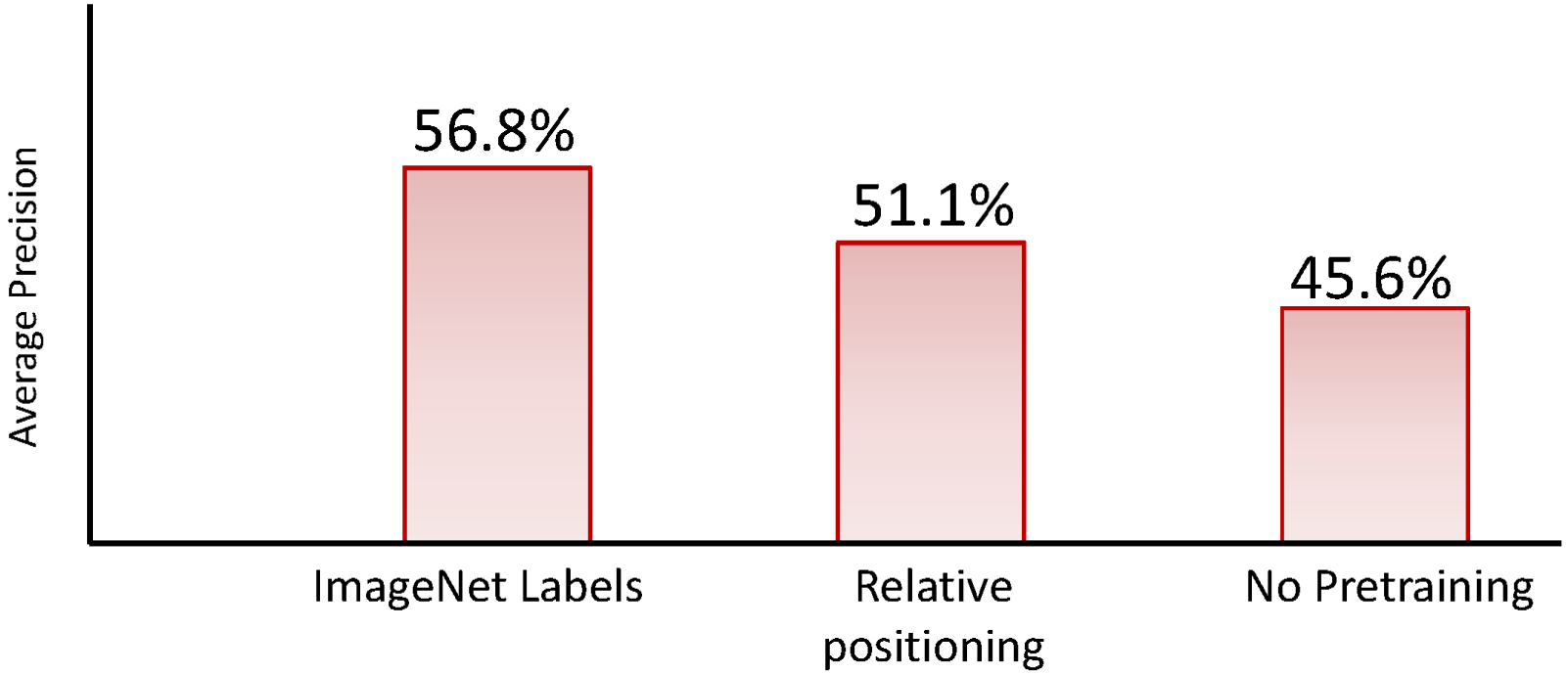
R-CNN



[Girshick et al. 2014]

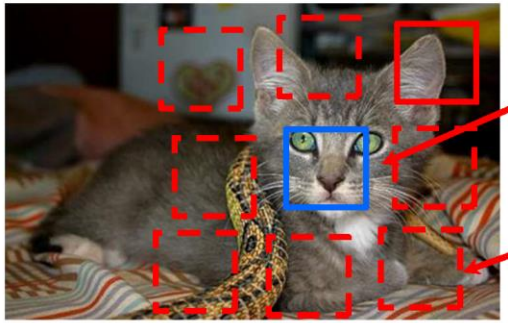
* slide from Andrew Zisserman, **University of Oxford**

Evaluation: PASCAL VOC Detection



* slide from Andrew Zisserman, **University of Oxford**

Avoiding Trivial Shortcuts



Include a gap

Jitter the patch locations

* slide from Andrew Zisserman, **University of Oxford**

Image example II: colourization

Train network to predict pixel colour from a monochrome input

Colorful Image Colorization, Zhang *et al.*, ECCV 2016

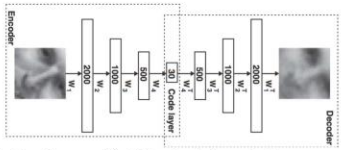
* slide from Andrew Zisserman, **University of Oxford**

Image example III: exemplar networks

- Exemplar Networks (Dosovitskiy *et al.*, 2014)
- Perturb/distort image patches, e.g. by cropping and affine transformations
- Train to classify these exemplars as same class

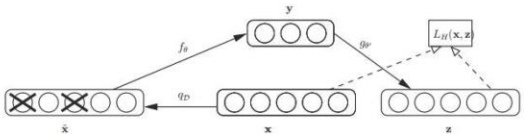
* slide from Andrew Zisserman, **University of Oxford**

Autoencoders



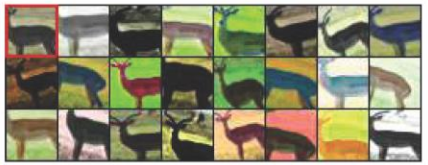
Hinton & Salakhutdinov. Science 2006.

Denoising Autoencoders



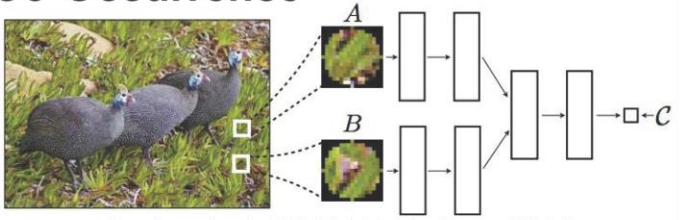
Vincent *et al.* ICML 2008.

Exemplar networks



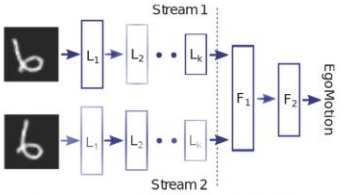
Dosovitskiy *et al.*, NIPS 2014

Co-Occurrence



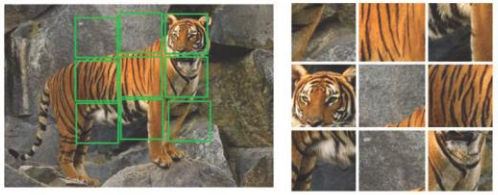
Isola *et al.* ICLR Workshop 2016.

Egomotion



Agrawal *et al.* ICCV 2015 Jayaraman *et al.* ICCV 2015

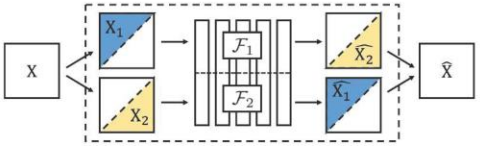
Context



Noroozi et al 2016

Pathak *et al.* CVPR 2016

Split-brain auto-encoders



Zhang *et al.* CVPR 2017

* slide from Andrew Zisserman, **University of Oxford**

Multi-Task Self-Supervised Learning

Procedure:

- ImageNet-frozen: self-supervised training, network fixed, classifier trained on features
- PASCAL: self-supervised pre-training, then train Faster-RCNN
- ImageNet labels: strong supervision

NB: all methods re-implemented on same backbone network (ResNet-101)

Self-supervision task	ImageNet Classification top-5 accuracy	PASCAL VOC Detection mAP
Rel. Pos	59.21	66.75
Colour	62.48	65.47
Exemplar	53.08	60.94
Rel. Pos + colour	66.64	68.75
Rel. Pos + Exemplar	65.24	69.44
Rel. Pos + colour + Exemplar	68.65	69.48
ImageNet labels	85.10	74.17

Multi-task self-supervised visual learning, C Doersch, A Zisserman, ICCV 2017

* slide from Andrew Zisserman, **University of Oxford**

Multi-Task Self-Supervised Learning

Findings:

- Deeper network improves performance (ResNet vs AlexNet)
- Colour and Rel-Pos superior to Exemplar
- Gap between self-supervision and strong supervision closing

Self-supervision task	ImageNet Classification top-5 accuracy	PASCAL VOC Detection mAP
Rel. Pos	59.21	66.75
Colour	62.48	65.47
Exemplar	53.08	60.94
Rel. Pos + colour	66.64	68.75
Rel. Pos + Exemplar	65.24	69.44
Rel. Pos + colour + Exemplar	68.65	69.48
ImageNet labels	85.10	74.17

Procedure:

- ImageNet-frozen: self-supervised training, network fixed, classifier trained on features
- PASCAL: self-supervised pre-training, then train Faster-RCNN
- ImageNet labels: strong supervision

Multi-task self-supervised visual learning, C Doersch, A Zisserman, ICCV 2017

* slide from Andrew Zisserman, **University of Oxford**

Video

A temporal sequence of frames

What can we use to define a proxy loss?

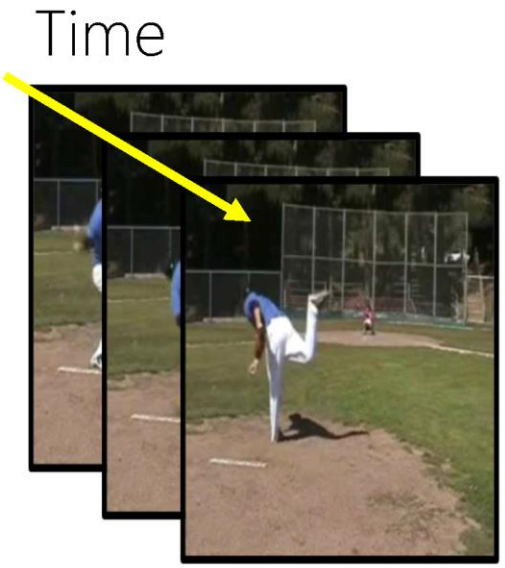
- Nearby (in time) frames are strongly correlated, further away may not be
- Temporal order of the frames
- Motion of objects (via optical flow)
- …

* slide from Andrew Zisserman, **University of Oxford**

Temporal structure in videos

Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

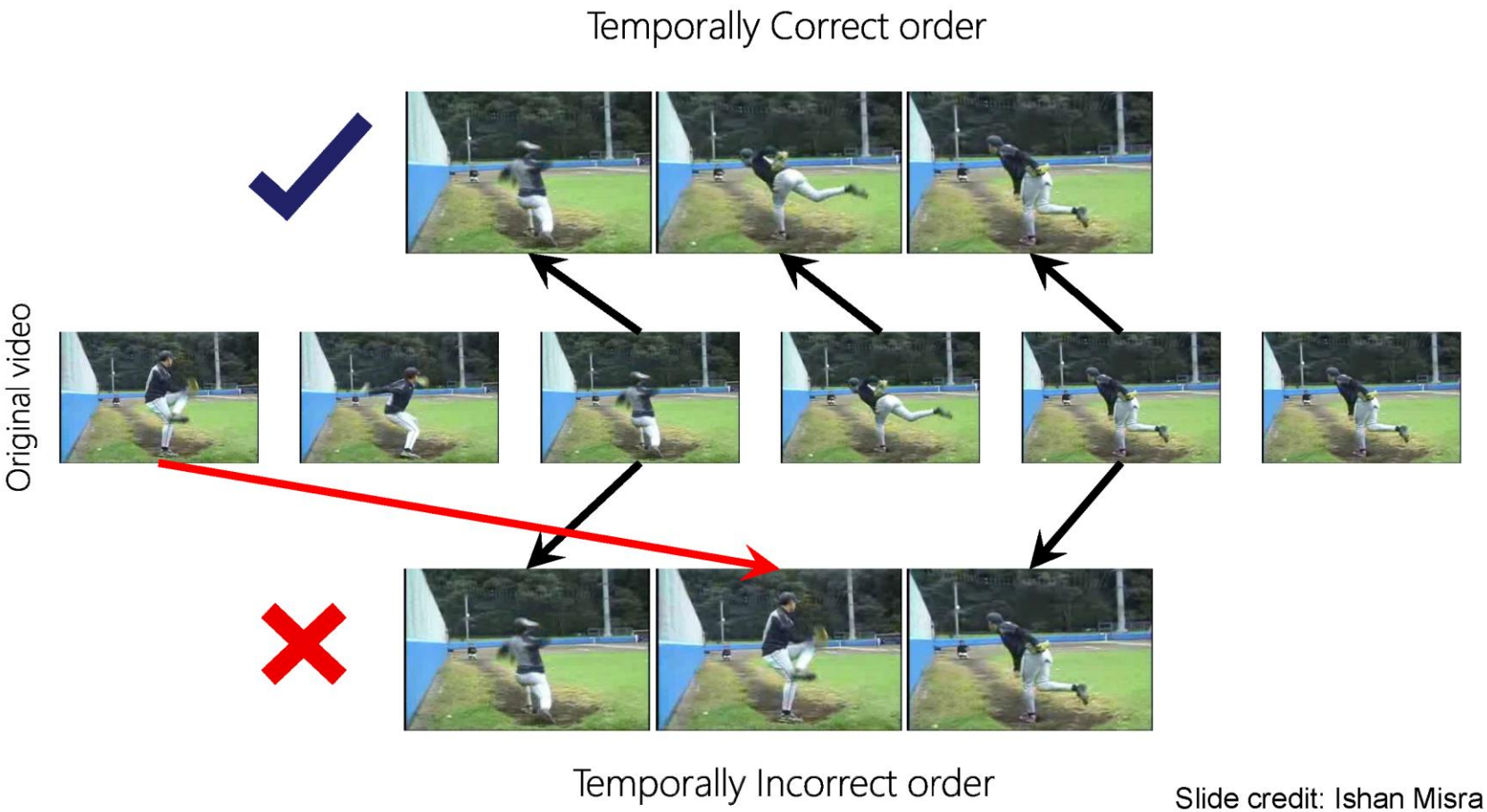
Ishan Misra, C. Lawrence Zitnick and Martial Hebert
ECCV 2016



"Sequence" of data

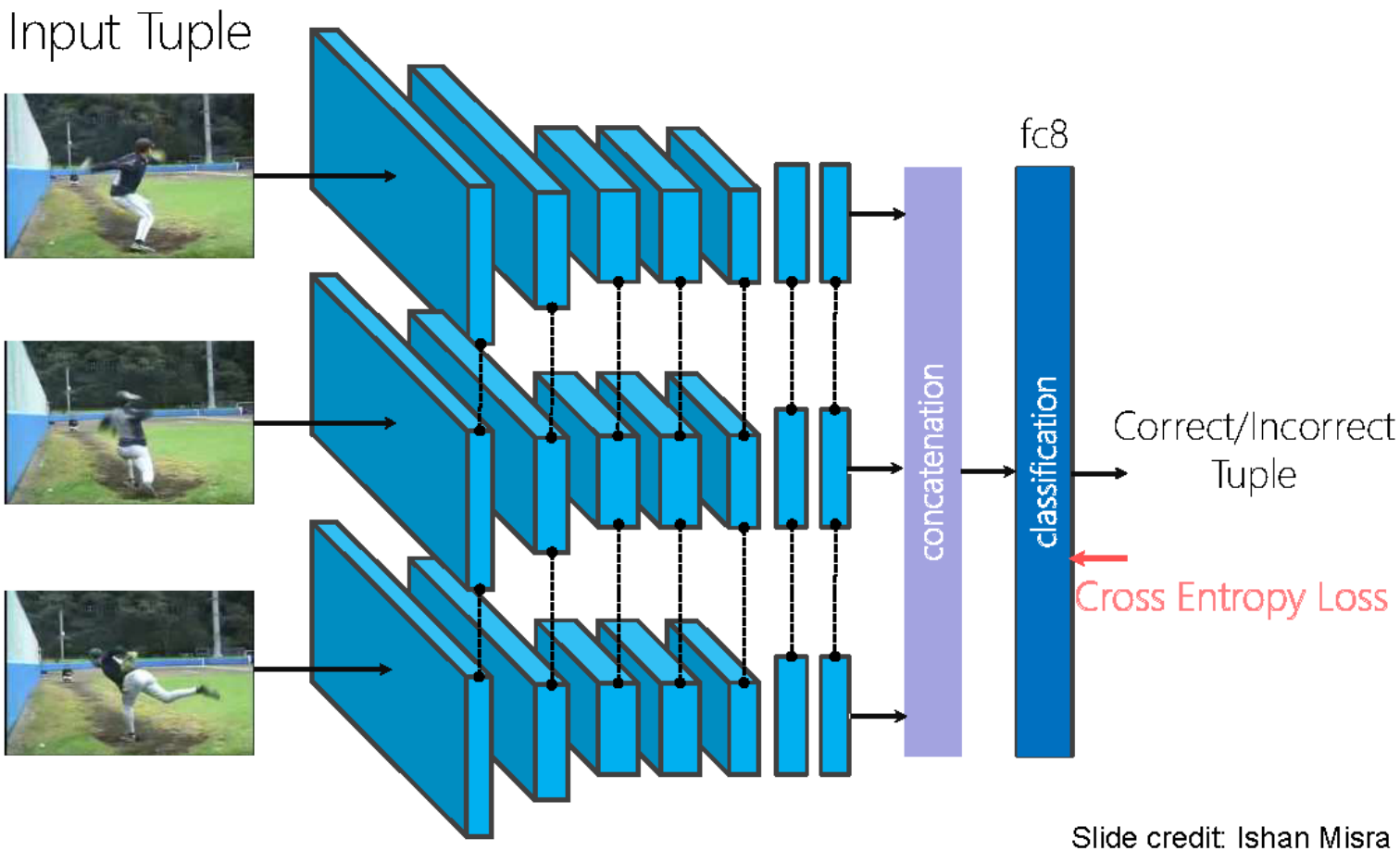
Slide credit: Ishan Misra

* slide from Andrew Zisserman, **University of Oxford**



Slide credit: Ishan Misra

* slide from Andrew Zisserman, **University of Oxford**

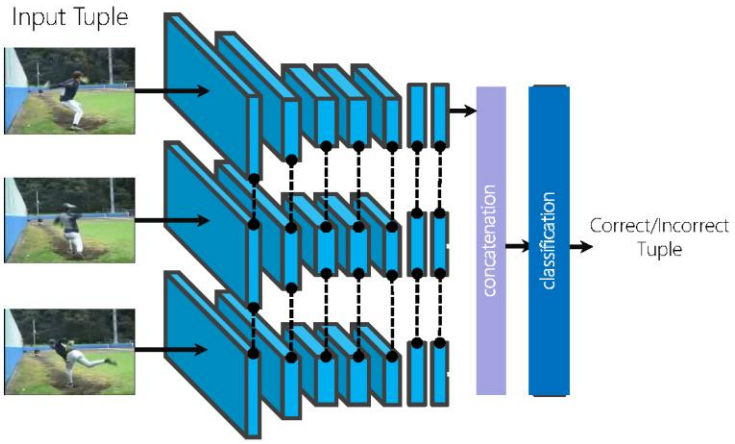


Slide credit: Ishan Misra

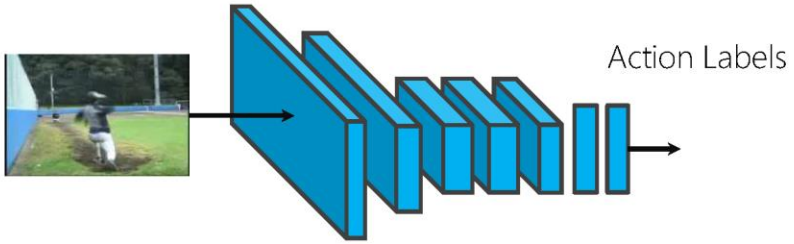
* slide from Andrew Zisserman, **University of Oxford**

Finetuning setup

Self-supervised Pre-train



Test -> Finetune



Slide credit: Ishan Misra

* slide from Andrew Zisserman, **University of Oxford**

Results: Finetune on Action Recognition

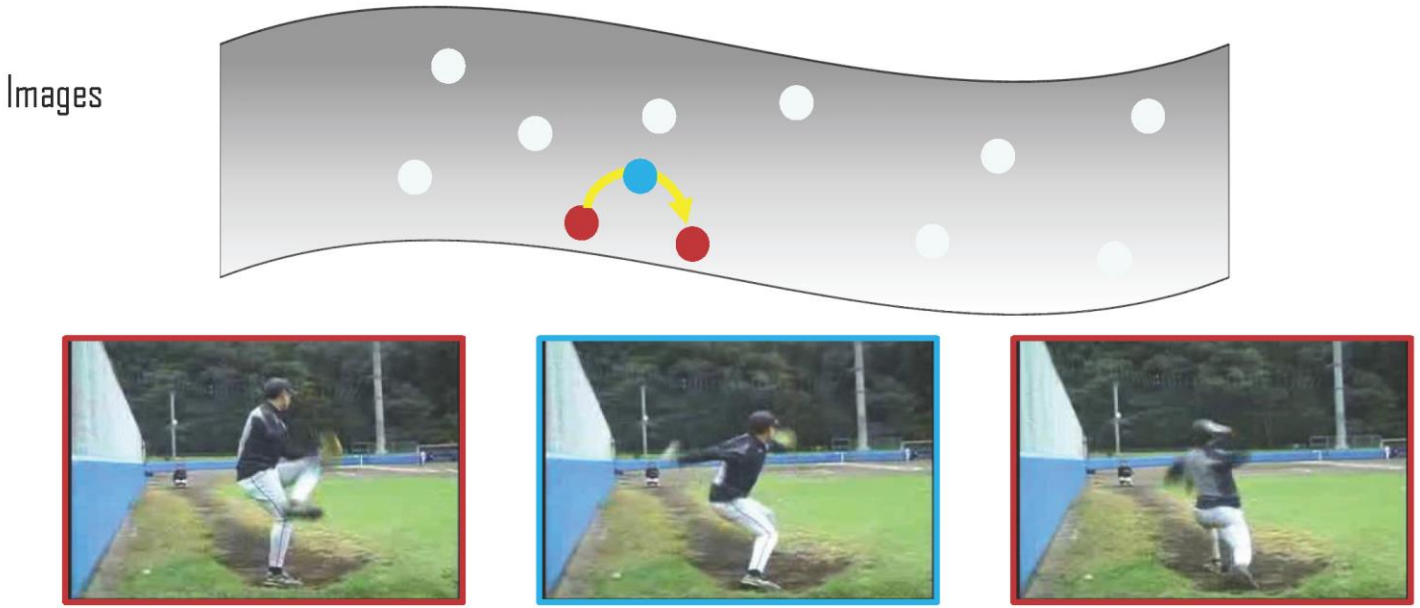
Dataset	Initialization	Mean Classification Accuracy
UCF101	Random	38.6
	Shuffle & Learn	50.2
	ImageNet pre-trained	**67.1**

Setup from - Simonyan & Zisserman, 2014

Slide credit: Ishan Misra

* slide from Andrew Zisserman, **University of Oxford**

What does the network learn?



Given a start and an end, can this point lie in between?

Shuffle and Learn – I. Misra, L. Zitnick, M. Hebert – ECCV 2016

Slide credit: Ishan Misra

* slide from Andrew Zisserman, **University of Oxford**

Summary Point

- Self-Supervision:
  - A form of unsupervised learning where the data provides the **supervision**
  - In general, withhold some information about the data, and task the network with predicting it
  - The task defines a proxy loss, and the network is forced to learn what we really care about, e.g. a semantic representation, in order to solve it
- Many self-supervised tasks for images
- Often complementary, and combining improves performance
- Closing gap with strong supervision from ImageNet label training
  - ImageNet image classification, PASCAL VOC detection
- Deeper networks improve performance

* slide from Andrew Zisserman, **University of Oxford**