THE UNIVERSITY of EDINBURGH
informatics

UDRC Summer School: Machine Learning: Deep Neural Networks III

Dr Timothy Hospedales

-

Machine Intelligence Group

http://mig.inf.ed.ac.uk

University of Edinburgh

-

UDRC Co-Investigator

-

Samsung AI Centre, Cambridge

Dr. Timothy Hospedales

- Background
  - BA Computer Science, Cambridge 2002
  - Lecturer/Senior Lecturer, QMUL, 2012-16
  - Reader (Assoc. Prof), Edinburgh, 2016-
    - Head of Machine Intelligence Group
  - Funded by EPSRC, DSTL, EU Horizon 2020
  - Alan Turing Institute Fellow, 2018-
  - Principal Scientist, Samsung AI Research Center, 2019-
- Research Area:
  - Deep Learning. Meta-Learning. Vision. Vision & Language.
- Track record:
  - Over 50 papers in Tier 1 venues of AI, ML, Vision.
  - => CVPR, ICCV, ECCV, ICLR, AAAI, IJCAI, ICML. T-PAMI, IJCV.
  - Four best paper prizes.
  - Three patents

Outline

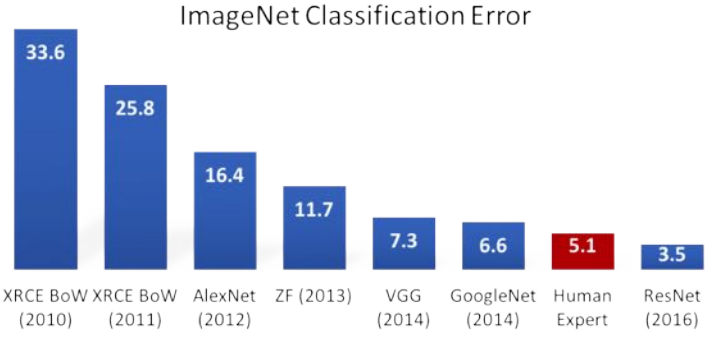
- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Outline

- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Part I: Sparse Data Deep Learning

Deep Learning Success



DeepMind

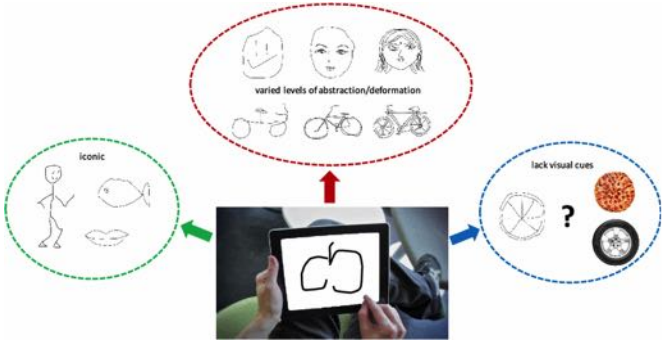
"Human-level control through deep reinforcement learning"

letter

Deep Q-Learning

5

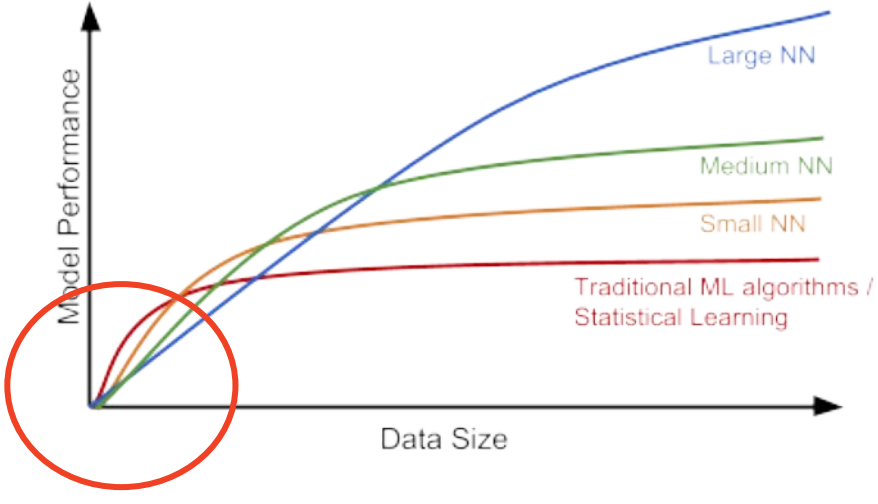
Sketch-a-Net that beats humans



Superhuman Pictionary

Mechanism of Deep Learning Era Success?

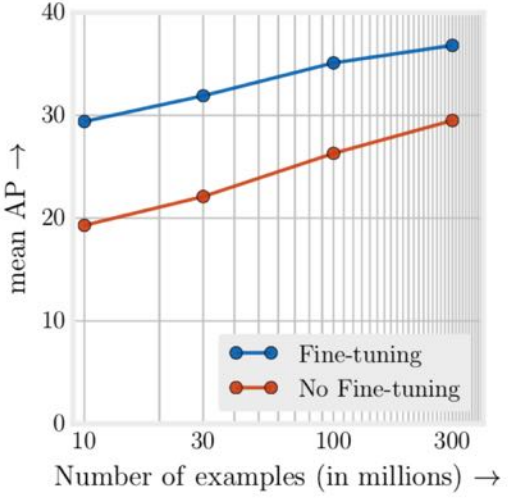
- Gather and annotate bigger datasets.
- Train bigger models.



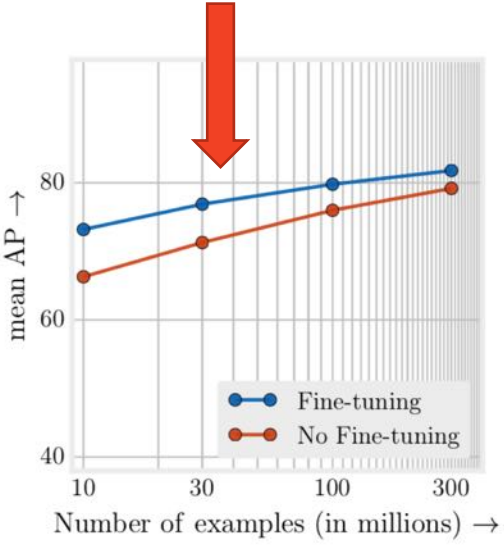
Success Story of Deep Learning Era

- Gather and annotate bigger datasets.
- Train bigger models.

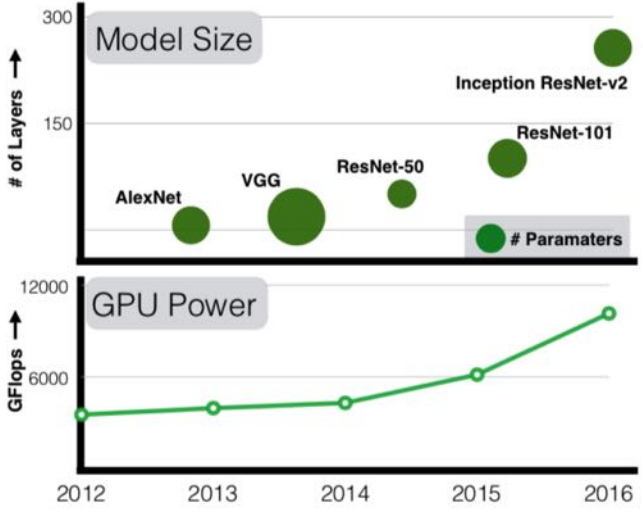
No saturation so far!
Performance grows with log of train data.



JFT-300M => MS-COCO

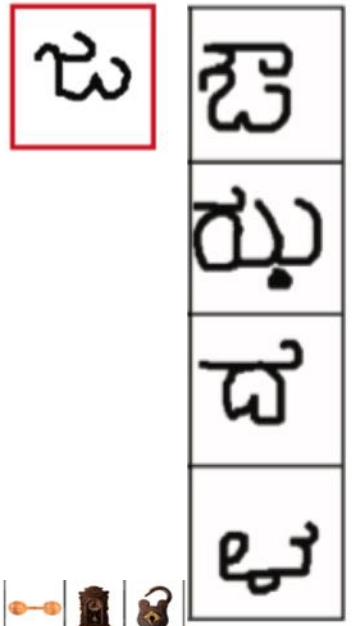


JFT-300M => PASCAL



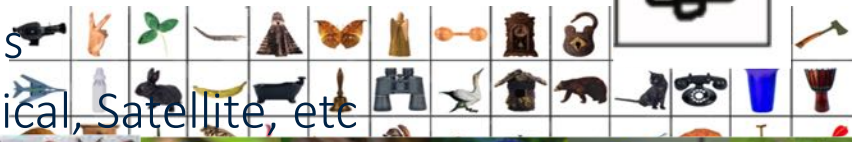
[Sun et al, Revisiting Unreasonable Effectiveness of Data in Deep Learning Era, ICCV'17]

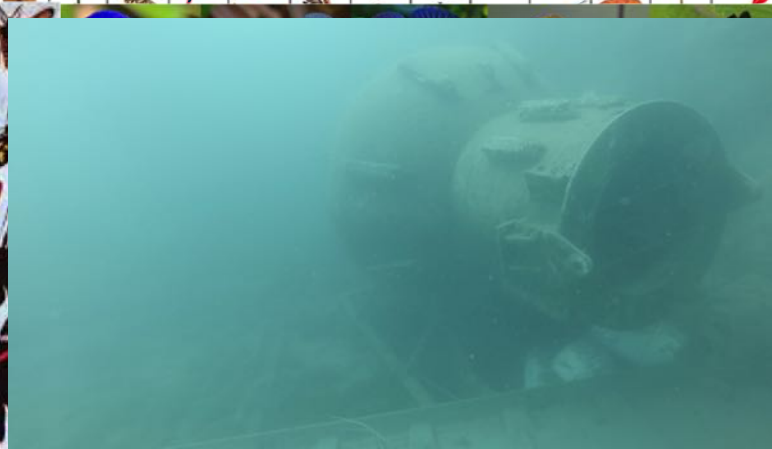
Do we need another paradigm?



Do we need another paradigm?

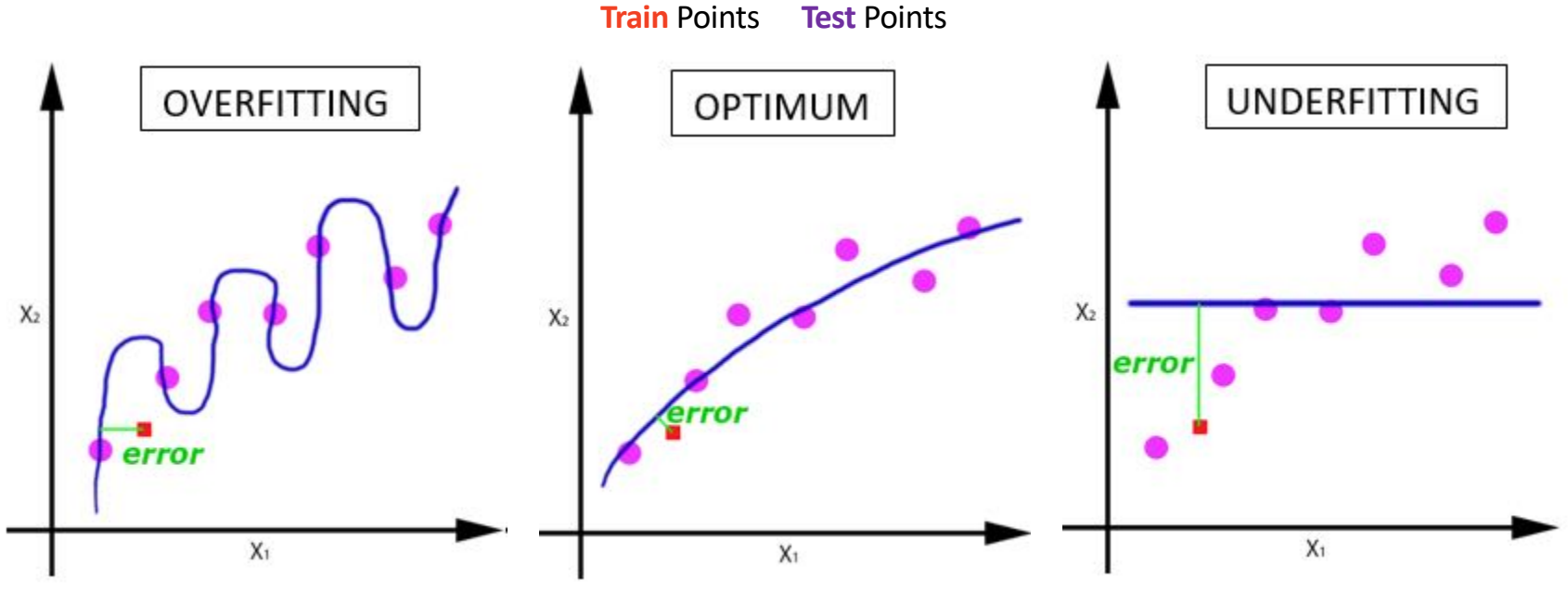
- Humans have one shot learning
  - Learn 5 objects per day for first 18 years.
- Annotating the long tail of object categories?
  - Emerging categories
- Annotating data in the long tail of domains
  - Underwater, Radar, Sonar, LIDAR, Medical, Satellite, etc
- Defense applications training data, or rare





Why is Few-Shot Learning Hard? Overfitting

- Underfitting vs Overfitting. Linear regression example.



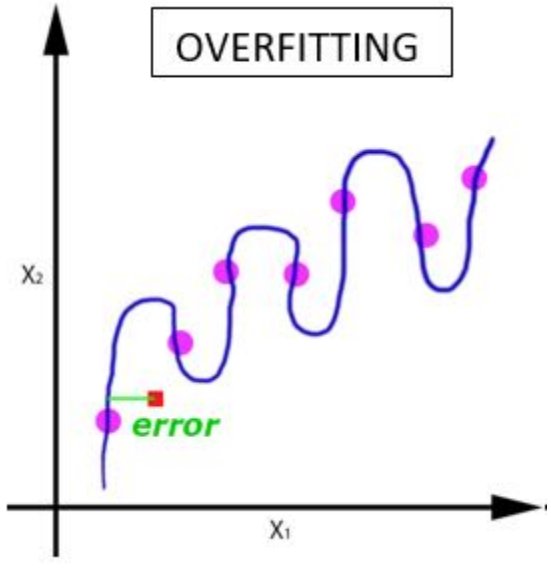
EG: 20th order polynomial.

EG: 2nd order polynomial.

EG: 0th order polynomial.

Why is Few-Shot Learning Hard? Overfitting

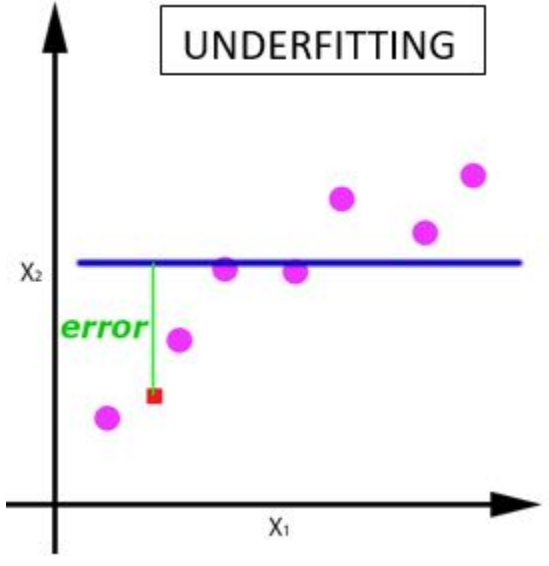
- Underfitting vs Overfitting. Linear regression example.



Question: How to diagnose over- vs under-fitting?

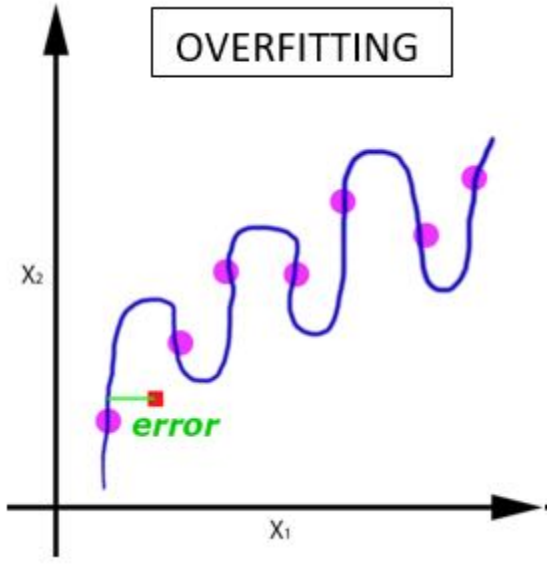
Overfitting?
- Low train error
- High test error

Underfitting?
- High train error
- High test error



Why is Few-Shot Learning Hard? Overfitting

- Underfitting vs Overfitting. Linear regression example.



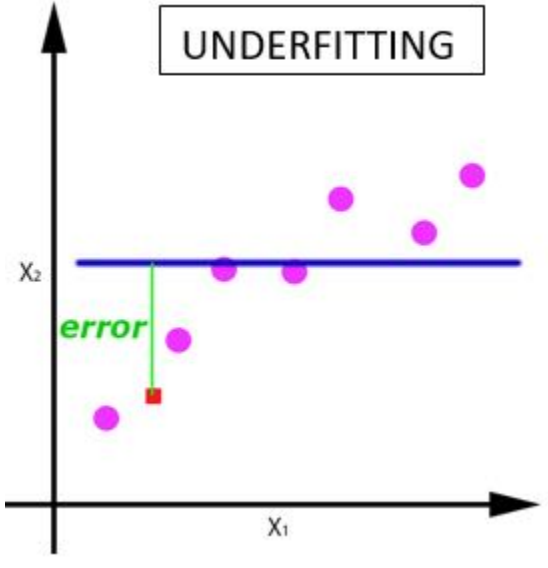
EG: 20th order polynomial.

Classic Solution?

- Try several model complexities
- Evaluate validation set performance of each
- Pick the model with best validation performance

Issue for deep learning?

- Too many complexity parameters in DL (depth, width, non-linearity, etc)
- Few-Shot: Would pick a simple model that doesn't provide deep learning level performance.



EG: 0th order polynomial.

Common Techniques for Sparse Data Applications

Reducing Data Dependence

Common techniques for overfitting reduction: Regularisation

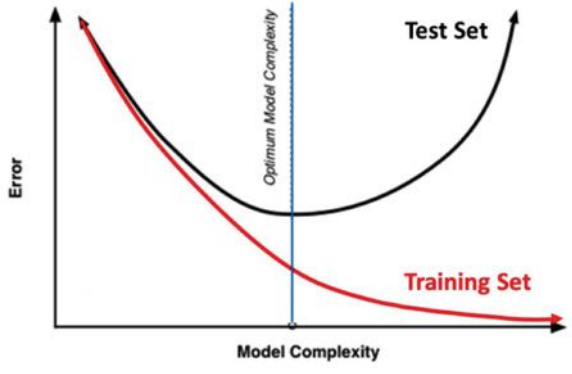
- Weight decay / L2 regularization
- Early Stopping
- Dropout
- Transfer Learning

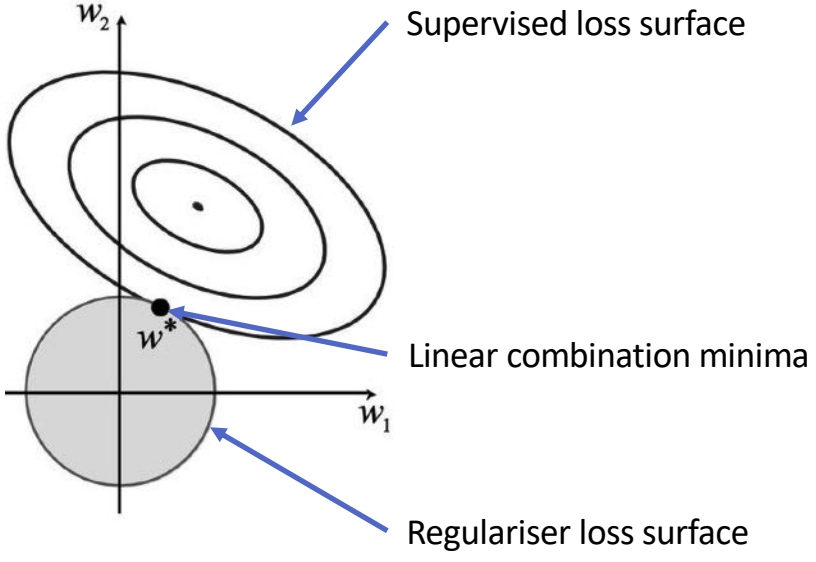
Reducing Data Dependence

Weight decay. L2 regularisation.

- Other things being equal, prefer weights near zero.

$$\min_{\boldsymbol{w}} \mathcal{L}(D; \boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2$$

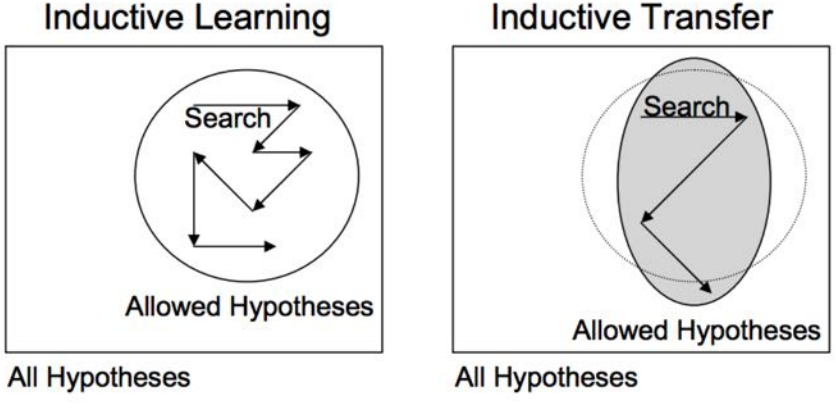




Reducing Data Dependence

Transfer Learning

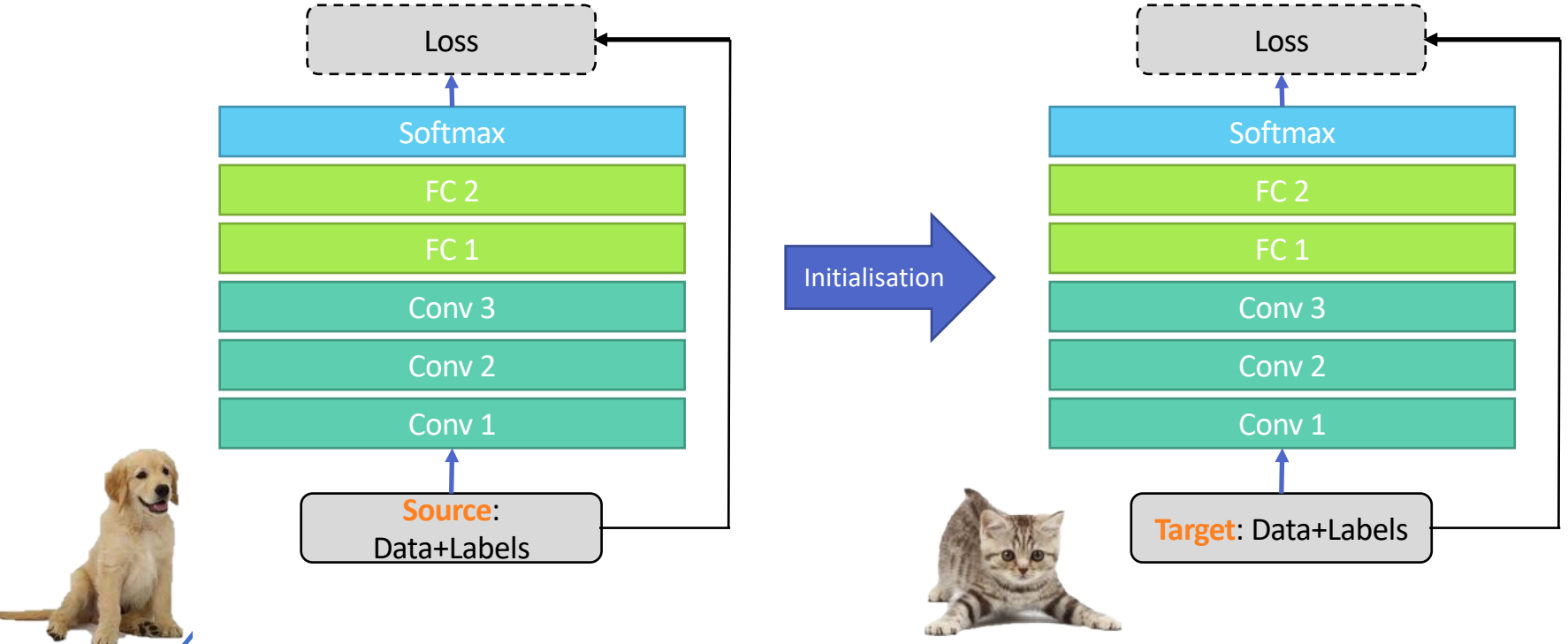
- Transfer Learning: "*The application of skills, knowledge, and/or attitudes that were learned in one situation to another learning situation*" (Perkins, 1992).
- Note:
  - Transfer Learning ≠ "Fine-Tuning"
  - Fine Tuning⊂ Transfer Learning



Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

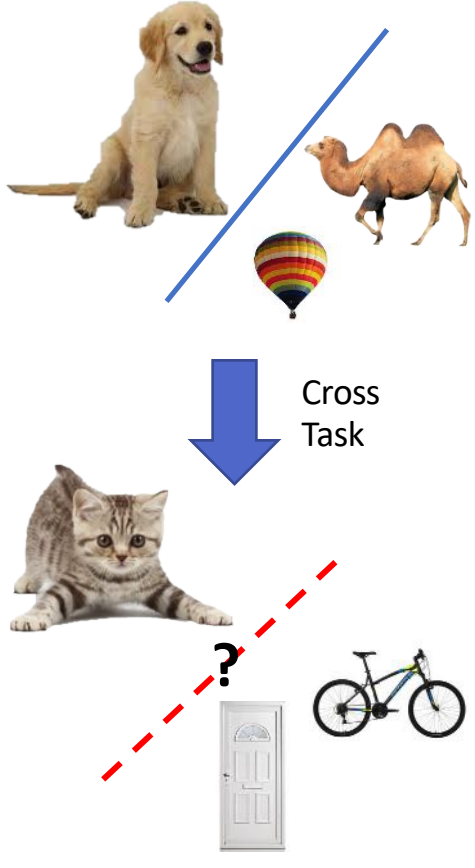
- Initialise target parameters using source, and continue training



Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Application:
  - Across Task: $p(Y_s|X_s) \neq p(Y_t|X_t)$
    - Change of label-space. $Y_s \neq Y_t$
  - Across Domain
    - Change of data statistics. $p(X_s) \neq p(X_t)$

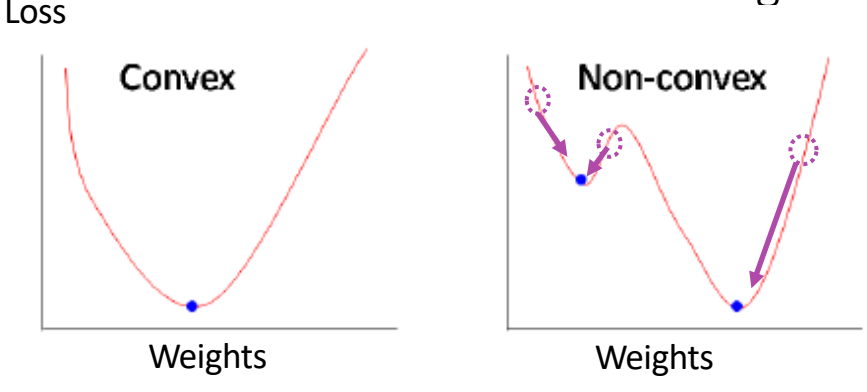


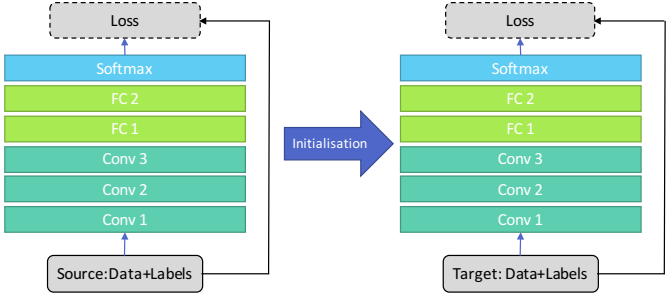
RGB => IR

Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Why does fine-tuning work?
  - Neural network optimisation is non-convex.
  - With small learning rate, target task parameters do not change much.
  - Transfer initialization effectively regularizes target weights towards source weights, rather than towards zero.
    - Assume source task is relevant to target task.
      - => Better chance of good minima.





Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

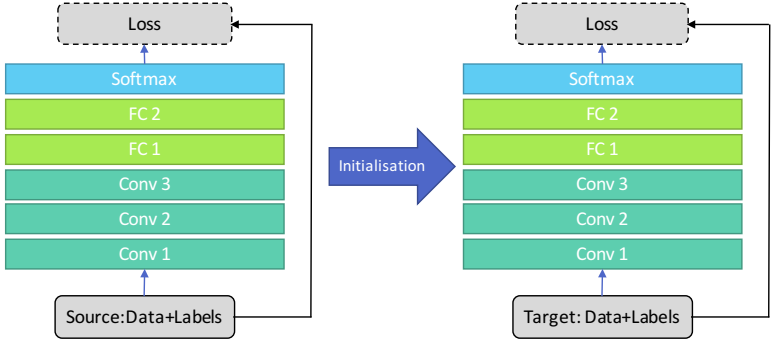
- Very similar to explicit source→target regularisation.
  - Used in many classic learning methods(*).

$$\min_{\boldsymbol{w}_t} \mathcal{L}(D_t; \boldsymbol{w}_t) + \lambda \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_2^2$$

Compare:

$$\min_{\boldsymbol{w}} \bar{\mathcal{L}} = \mathcal{L}(D; \boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2$$

$$\min_{\boldsymbol{w}} \bar{\mathcal{L}} = \mathcal{L}(D; \boldsymbol{w}) + \lambda \|\boldsymbol{w} - \boldsymbol{0}\|_2^2$$



[Improving SVM Accuracy by Training on Auxiliary Data Sources, ICML'04]
[Cross-domain video concept detection using adaptive svms, ACM MM'07]

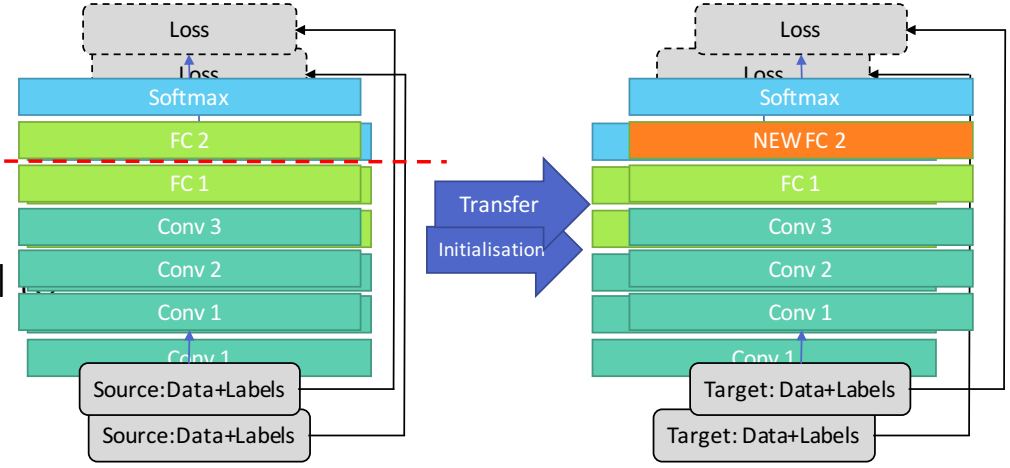
Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Assumption:
  - Source task relevant to target.
- Practical Considerations. Questions:
  - How to control relevance degree?
  - Which layers to transfer?
  - What learning rate to use?
- Typical:
  - Relearn top while freeze bottom.
  - Then LR tune all w/ LR proportional depth

$$\min_{\boldsymbol{w}} \mathcal{L}(D_t; \boldsymbol{w}_t) + \lambda \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_2^2$$

Question:
Always good practice?



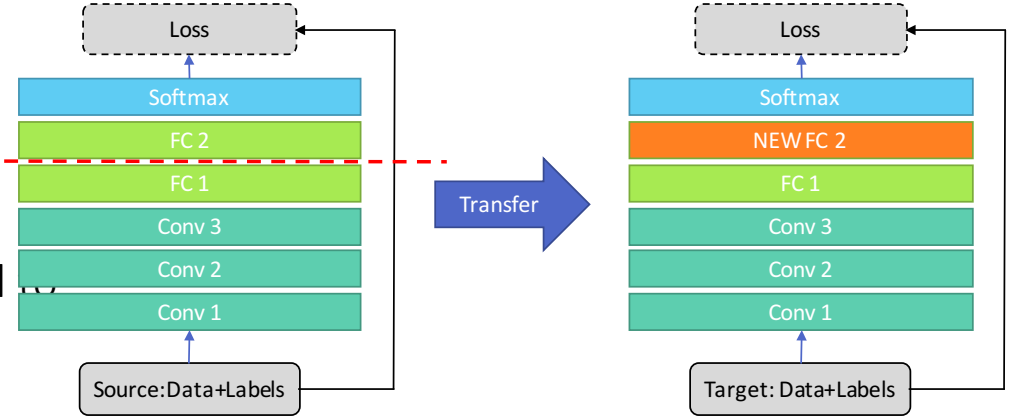
[How transferable are features in deep neural networks?, NIPS'14]

Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Assumption:
  - Source task relevant to target.
- Practical Considerations. Questions:
  - How to control relevance degree?
  - Which layers to transfer?
  - What learning rate to use?
- Typical:
  - Relearn top while freeze bottom.
  - Then LR tune all w/ LR proportional to depth

$$\min_{\boldsymbol{w}} \mathcal{L}(D_t; \boldsymbol{w}_t) + \lambda \|\boldsymbol{w}_t - \boldsymbol{w}_s\|_2^2$$



Only for cross-task.

Assumption:
- Relevance proportional to depth.

[How transferable are features in deep neural networks?, NIPS'14]

Reducing Data Dependence: Transfer

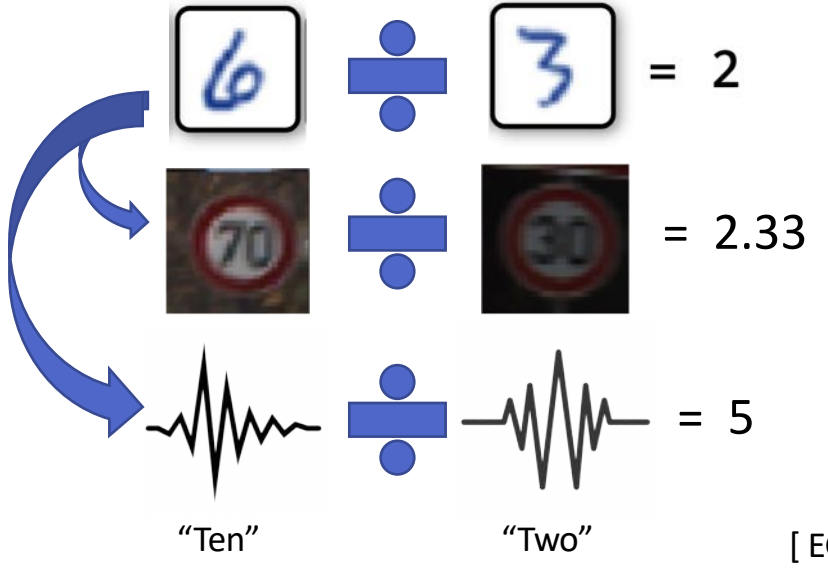
Transfer Learning: Fine-Tuning

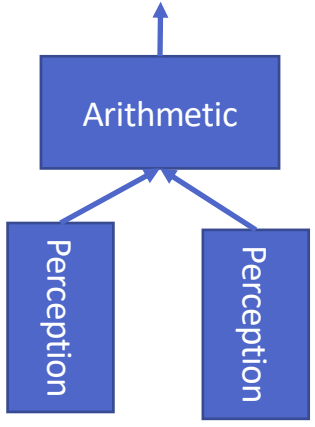
- Challenges:
  - What if the inputs are heterogeneous
    - (EG: Task 1: RGB, Task 2: Depth)?
  - How to know if a given source is relevant?
    - How to prevent negative transfer if source is irrelevant?
  - How to select the relevant source among many?

Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Issues:
  - What if the inputs are heterogeneous?
    - EG: Perceptual arithmetic network.





[EG: HOUDINI: Lifelong Learning as Program Synthesis, NIPS'18]

Reducing Data Dependence: Transfer

Transfer Learning: Fine-Tuning

- Issues:
  - How to select the relevant source among many?
- For linear models….
  - Optimising target model $\boldsymbol{w}$, assuming a set of potentially relevant sources $\{\boldsymbol{w}_k\}$:

$$\min_{\boldsymbol{w}}[\|\boldsymbol{y} - \boldsymbol{w}X\| + \min_k \|\boldsymbol{w} - \boldsymbol{w}_k\|]$$

  - …. not so straightforward with deep models

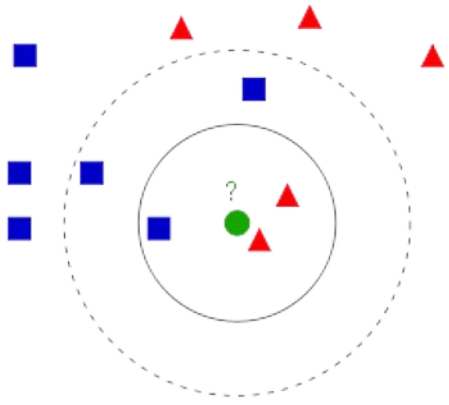
E.g., Evgeniou et al, JMLR, 2005
E.g., Kang et al, ICML, 2011

Outline

- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Metric Learning

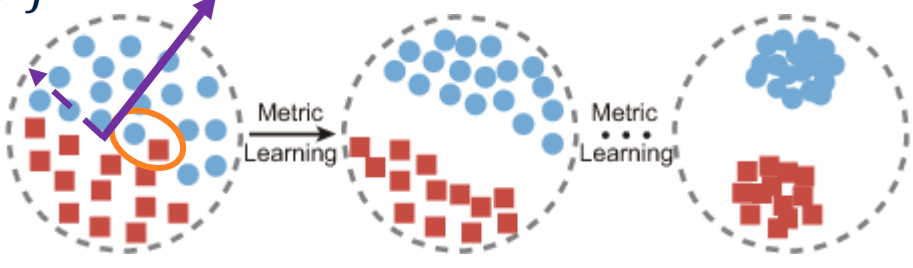
Nearest Neighbor Classifier



- Classic Method in ML
  - Given training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$
- 1-NN:
  - Classify a new point $\boldsymbol{x}^*$. Euclidean nearest neighbor:
    - $y^* = y_{i^*}, i^* = \text{argmin}\|\boldsymbol{x}^* - \boldsymbol{x}_i\|$
- K-NN: For 1-hot $\boldsymbol{y}$.
  - Classify a new point $\boldsymbol{x}^*$ as:
    - $p(\boldsymbol{y}^*|\boldsymbol{x}^*) = \frac{1}{K}\sum_{\boldsymbol{x}_j \in NN(\boldsymbol{x}^*)} \boldsymbol{y}_j$

Can we measure similarity by a better distance than Euclidean?

Find a similarity measure where different categories are far, and same categories are near.



Classic Metric Learning

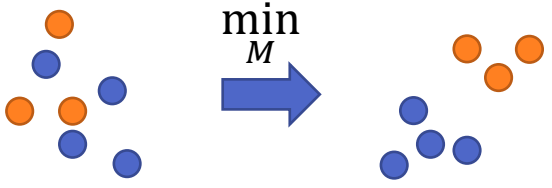
Factorize M=L'L to view as linear transform

- Define a Mahalanobis Metric for NN classification:

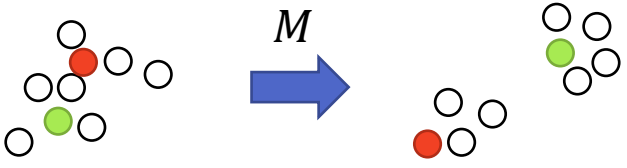
$$d_M(x_i, x_j) = (x_i - x_j)^T M (x_i - x_j) = (x_i - x_j)^T L^T L (x_i - x_j) = (Lx_i - Lx_j)^T (Lx_i - Lx_j)$$

- Learn the Metric that separates the training data:

$$\min_M L(M) = \sum_{i,j \in Pos} d_M(x_i, x_j) - \sum_{i,j \in Neg} d_M(x_i, x_j)$$

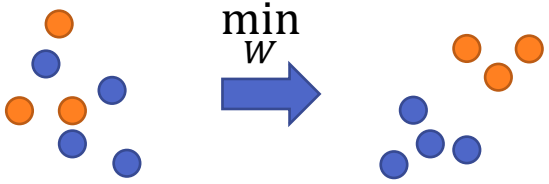


- Reuse the metric to solve 1-shot learning by NN matching:



[EG: Hirzer, ECCV'12; Mignon, CVPR'12]

Deep Metric Learning



- Linear => Non-linear feature transform

$$d_M(x_i, x_j) = (x_i - x_j)^T M (x_i - x_j) = (Lx_i - Lx_j)^T \ (Lx_i - Lx_j)$$

$$\Rightarrow \quad d_W(x_i, x_j) = (f_W(x_i) - f_W(x_j))^T \ (f_W(x_i) - f_W(x_j))$$

- Common Losses:

$$\min_W L(W) = \sum_{i,j \in Pos} d_W(x_i, x_j) - \sum_{i,j \in Neg} d_W(x_i, x_j)$$

Minimize average positive distance,
Maximize average negative distance

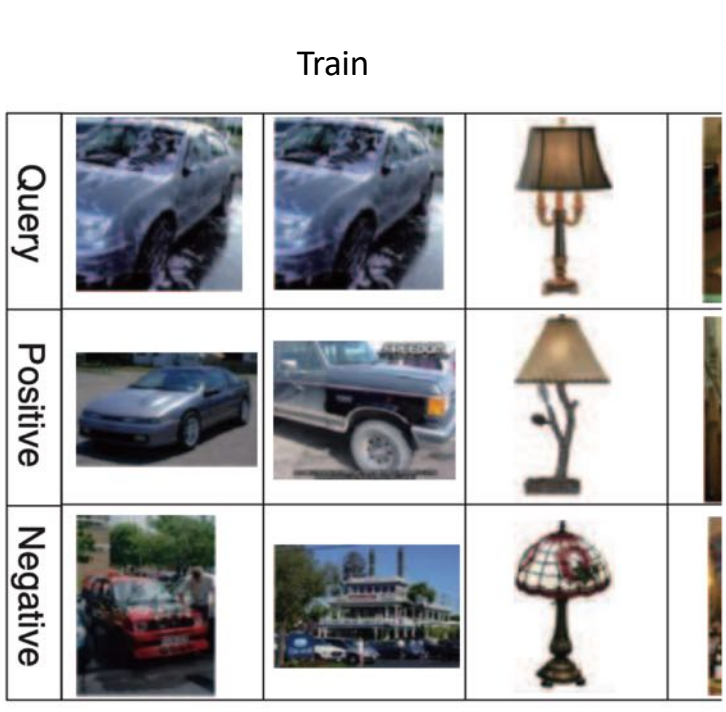
$$\min_W L(W) = \sum_{i \in D} \sum_{j \in Pos(i), k \in Neg(i),} \left| d_W^{Pos}(x_i, x_j) + \Delta - d_W^{Neg}(x_i, x_k) \right|^+$$

Every negative distance more than every positive distance

$$\min_W L(W) = \sum_{i,j \in Pos} \left\| d_W(x_i, x_j) - 0 \right\| + \sum_{i,j \in Neg} \left\| d_W(x_i, x_j) - 1 \right\|$$

All positive pair distances = 0
All negative pair distances = 1

Deep Metric Learning: Example: Triplet Ranking for Similarity Search

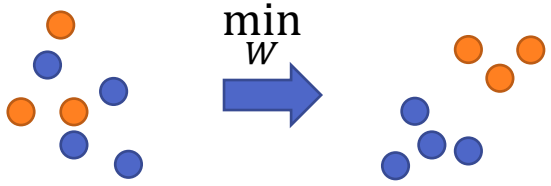


$$\min_{W} L(W) = \sum_{i \in D} \sum_{j \in Pos(i), k \in Neg(i}$$

[CVPR'14, Learning Fine-grained Image Similarity with Deep Ranking]

Deep Metric Learning: Challenges?

$$d_W(x_i, x_j) = (f_W(x_i) - f_W(x_j))^T \ (f_W(x_i) - f_W(x_j))$$

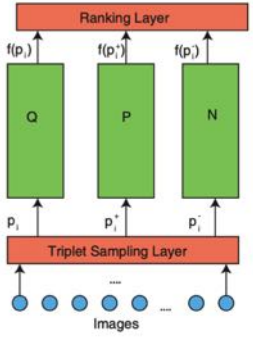


- Challenges? Training: Scalibility.
  - Num triplets: O(N^3). $L(W) = \sum_{i \in D, j \in Pos(i), k \in Neg(i)} \left| d_W^+(x_i, x_j) + \Delta - d_W^-(x_i, x_k) \right|^+$
    - Sampling useful ones
      - => Hard negative mining, learning sampling policy.
    - Annotating them.
- Challenges? Testing: Scalability.
  - Binary embedding for fast lookup/hashing

$$f_W(x) = \text{sign}(Wh + b)$$

  - => Gradient descent on discrete gradient operation

[Wang, PAMI'18, A Survey on Learning to Hash]



Metric Learning for Few-Shot

Few-Shot Learning: Setup

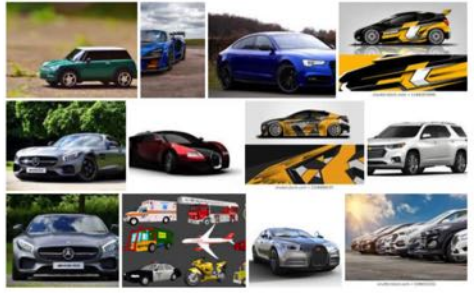
- Meta-train on Auxiliary Set. $D^{aux} = \{X^{aux}, Y^{aux}\} = \{x_i, y_i\}_{i=1}^{N^{aux}}$
  - EG: Vehicles.

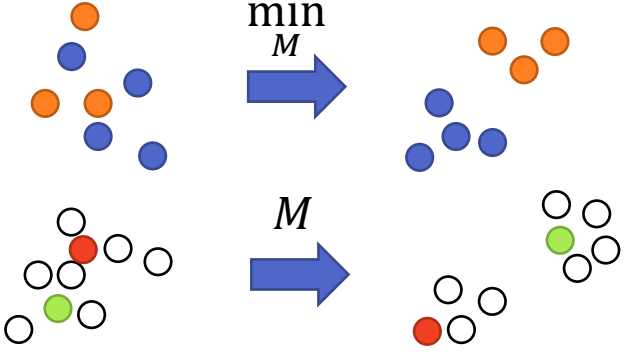
Disjoint labels: $y^{aux} \neq y^{sup}$

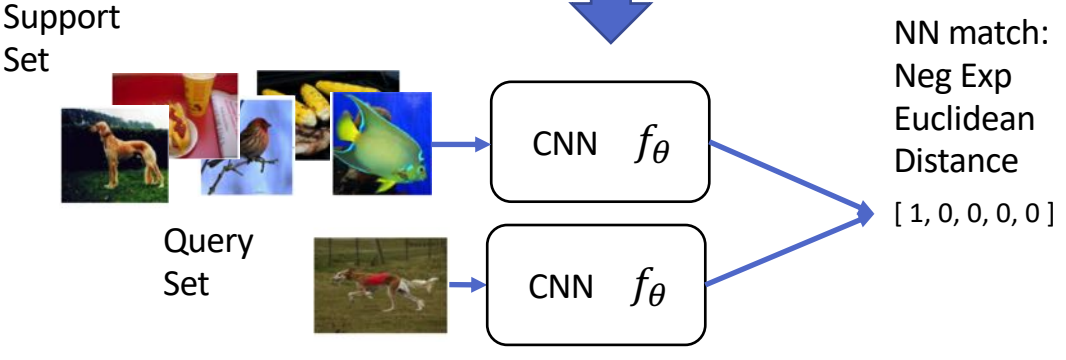
- Train on Support Set $D^{sup} = \{x_i, y_i\}_{i=1}^{1}$
  - EG: 1-example per animal

$y^{sup} = y^{tst}$

- Test on Query Set. $D^{tst} = \{x_i, y_i\}_{i=1}^{N}$
  - EG: New animal images to classify.







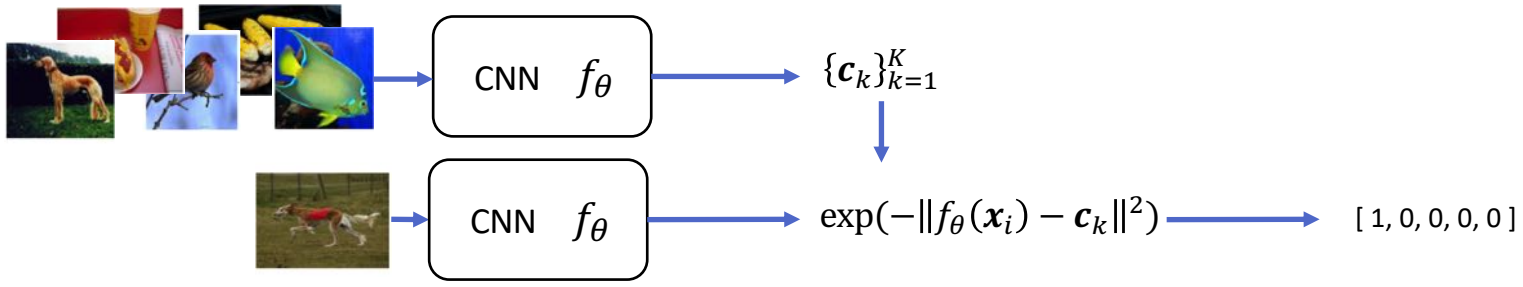
Few-Shot Learning: Prototypical Nets (NIPS'17)

- Calculate a "Prototype" per class:

$$\boldsymbol{c}_k = \frac{1}{|S_k|} \sum_{(\boldsymbol{x}_i, y_i) \in S_k} f_\theta(\boldsymbol{x}_i)$$

- Classify with

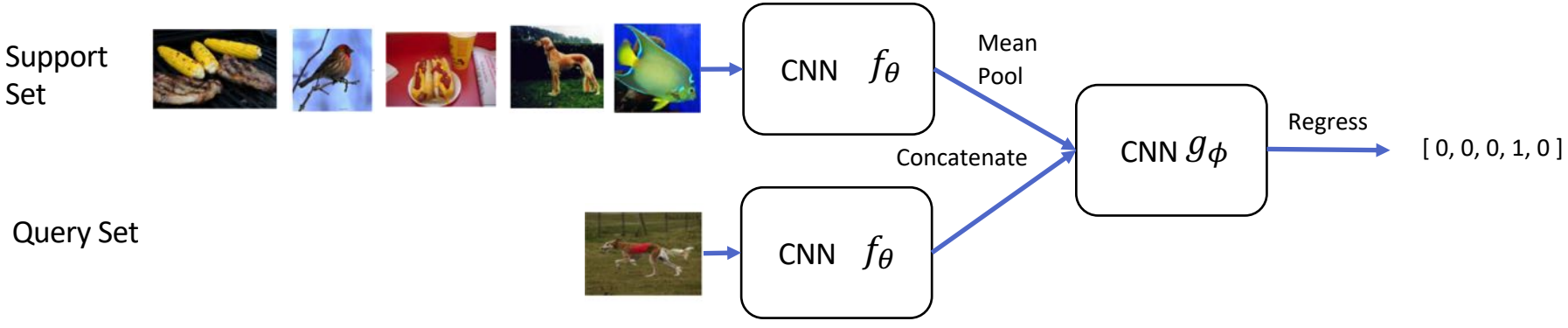
$$p(y = k|x) \propto \exp(-\|f_\theta(\boldsymbol{x}_i) - \boldsymbol{c}_k\|^2)$$



[Snell et al, NIPS'17, Prototypical Networks for Few Shot Learning]

Few-Shot Deep Learning: Relation Net

- Relation Net: Similar to Prototypical, but:
  - Architecture: Concatenate branch feature maps and convolve more.
  - Loss: Regress to 0/1.



[Sung, CVPR'18, "Learning to Compare: Relation Networks for Few Shot Learning"]

Few-Shot Deep Learning: Architectures

Deep Metric Learning

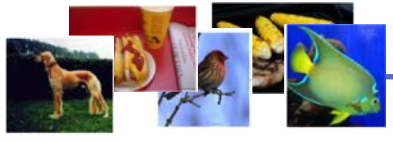
CNN f_θ

CNN f_θ

Neg Exp Euclidean Distance to NN

[1, 0, 0, 0, 0]

Prototypical Nets



CNN f_θ

$\{\boldsymbol{c}_k\}_{k=1}^{K}$

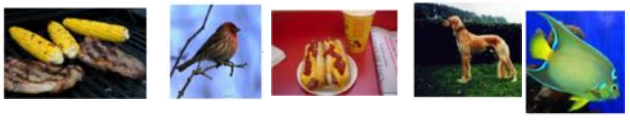
CNN f_θ

$\exp(-\|f_\theta(\boldsymbol{x}_i) - \boldsymbol{c}_k\|^2)$

[1, 0, 0, 0, 0]

Neg Exp Euclidean Distance to Prototype

Relation Net



CNN f_θ

CNN f_θ

Concatenate

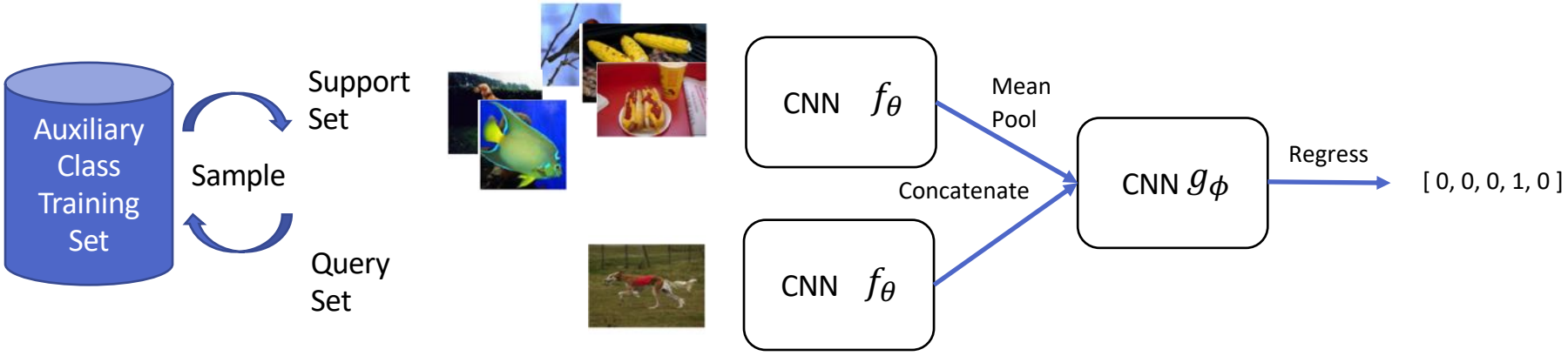
CNN g_ϕ

Regress

[0, 0, 0, 1, 0]

Few-Shot Deep Learning: Relation Net

- Relation Net: Episodic Training



Objective

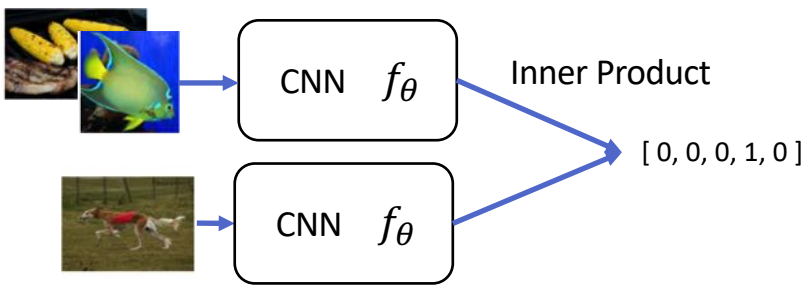
$$\text{argmin}_{\phi,\theta} \mathrm{E}_{(Q,S)\in D^{aux}} \sum_{x_q \in Q, S_k \subseteq S} \left\| g_\phi \left([f_\theta(x_q), \frac{1}{|S_k|} \sum_{x_i \in S_k} f_\theta(x_i)] \right) - I(y_q = k) \right\|$$

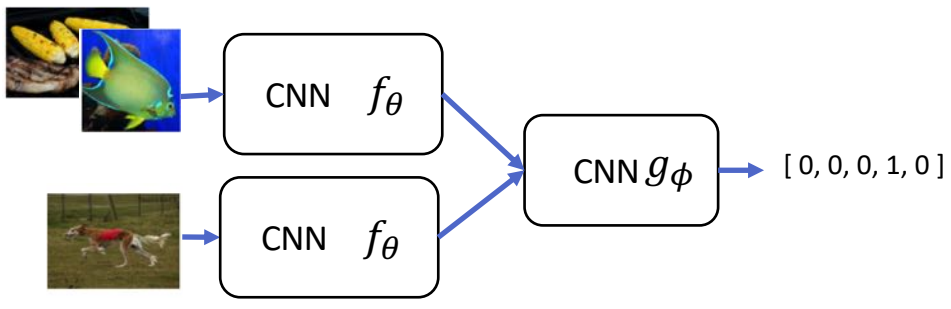
Episodic Training. Each mini-batch is a K-way N-shot "episode"
=> Trains features to support few-shot recognition

[Sung, CVPR'18, "Learning to Compare: Relation Networks for Few Shot Learning"]

Relation Net: Analysis

- PrototypeNet/RelationNet: Pros & Cons?
  - Pros:
    - Train+test cardinality don't have to match.
    - Training is super fast if K-way/N-shot is small.
  - Cons:
    - Testing is slow if K-way is large.
    - (RelationNet) Can't cache features for fast binary/approx NN lookup
  - Neutral
    - Can do old+new classes together. (But cost + accuracy….)



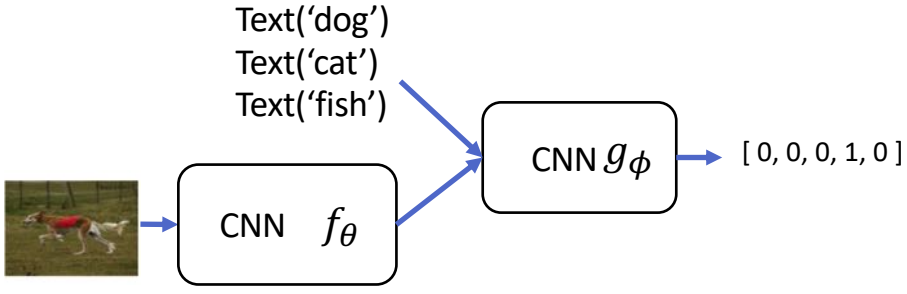


[Sung, CVPR'18, "Learning to Compare: Relation Networks for Few Shot Learning"]

Relation Net

- Summary:
  - State of the art on miniImageNet and tieredImageNet
  - Simple to implement and fast
  - Also supports zero-shot learning

Model (SENet)	5-way 1-shot Acc
MAML	55.9%
Prototypical	51.6%
RelationNet	57.4%
RelationNet 2.0	62.8%



[Sung, CVPR'18, "Learning to Compare: Relation Networks for Few Shot Learning"]
[Zhang, arXiv'18, "Relation Columns for Few Shot Learning"]

Outline

- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Tensor-Based Models

Multi-task Learning

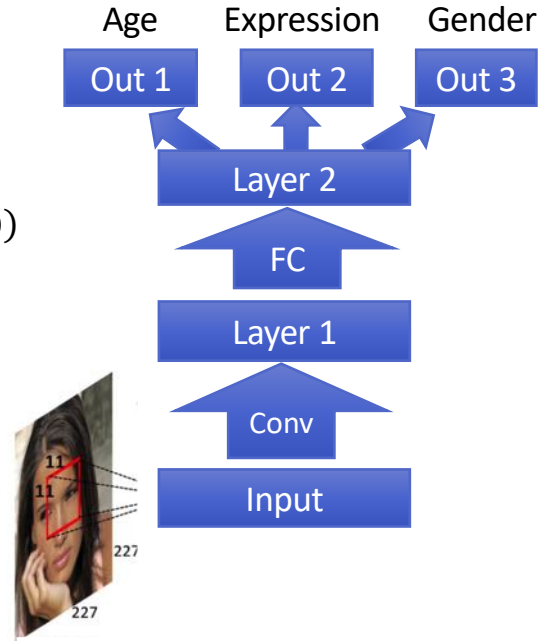
Deep Multi-Task Learning

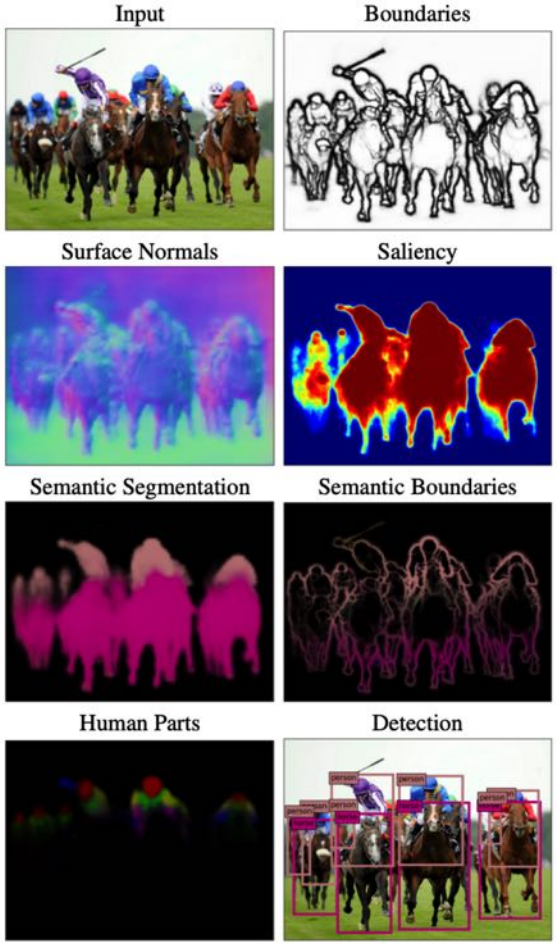
- Typically:
  - Share feature extraction layers.
  - Multiple predictions & losses.

Given Data: $\{\boldsymbol{x}_i, y_i^1, .., y_i^K\}$

Solve:

$$\min_{\theta,\phi_k} \sum_i \sum_k \mathcal{L}(y_i^k, f_{\phi_k}(g_\theta(x_i)))$$





[TPAMI 2017, HyperFace: A Deep Multi-task Learning Framework for Face Detection… ;
CVPR'17, UberNet: Training a `Universal' Convolutional Neural Network]

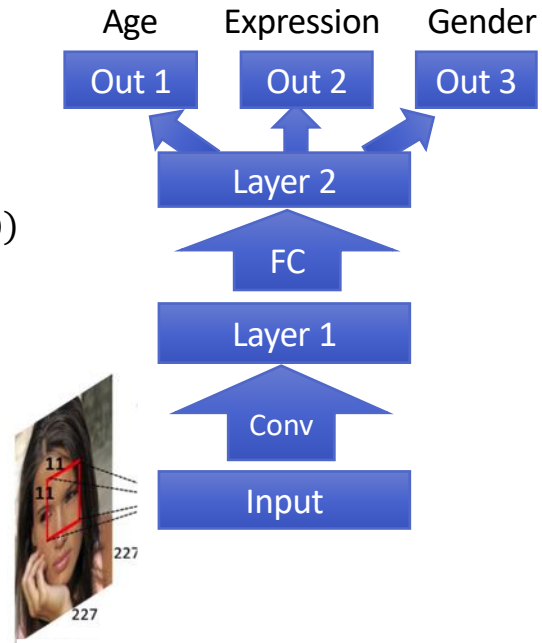
Deep Multi-Task Learning

- Typically:
  - Share feature extraction layers.
  - Multiple predictions & losses.

Given Data: $\{\boldsymbol{x}_i, y_i^1, .., y_i^K\}$

Solve:

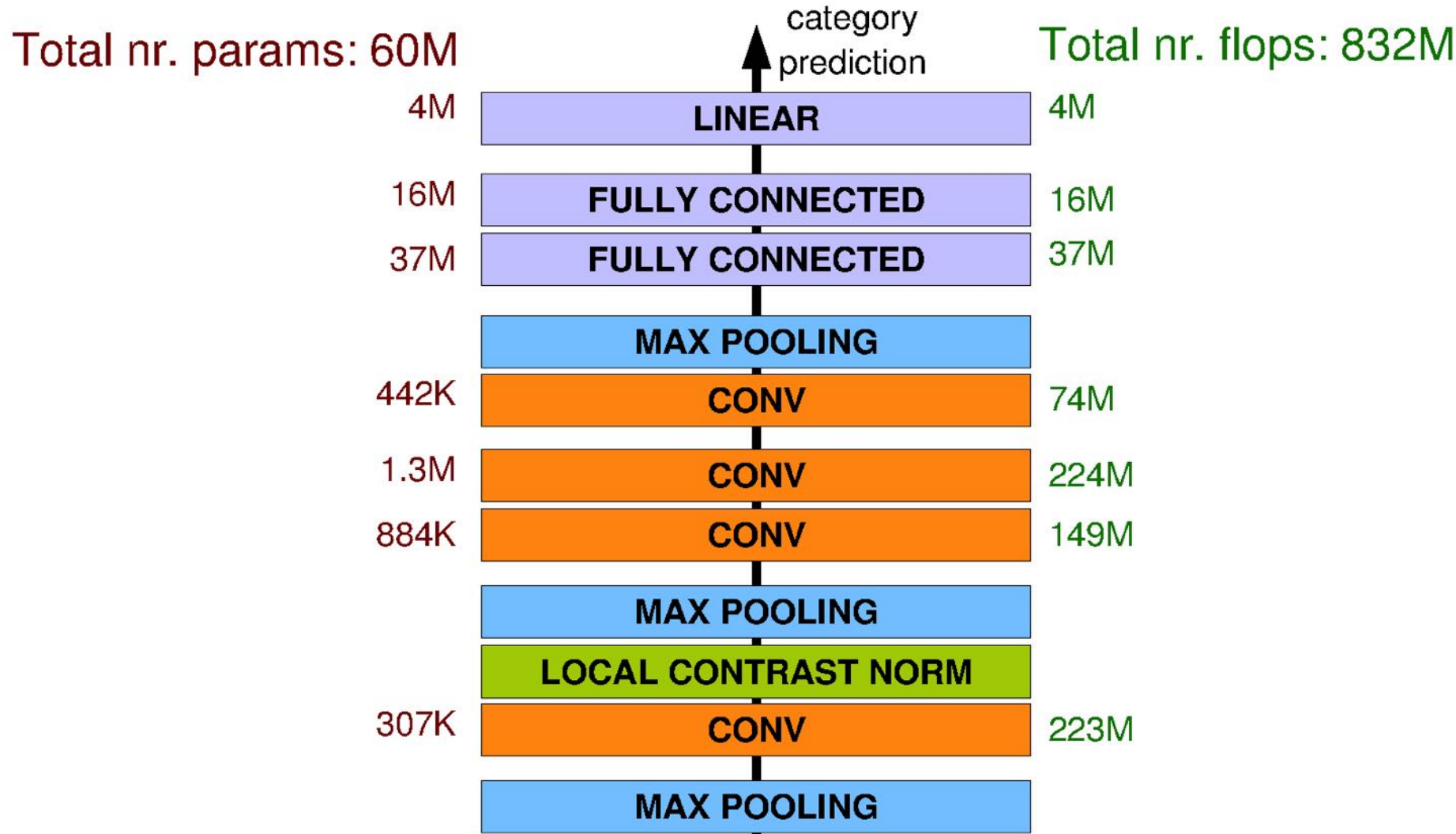
$$\min_{\theta,\phi_k} \sum_i \sum_k \mathcal{L}(y_i^k, f_{\phi_k}(g_\theta(x_i)))$$



- Why/When to use MTL?
  - Care about performance on all tasks (else use TL)
  - Share compute & memory of feature extraction.

[TPAMI 2017, HyperFace: A Deep Multi-task Learning Framework for Face Detection… ;
CVPR'17, UberNet: Training a `Universal' Convolutional Neural Network]

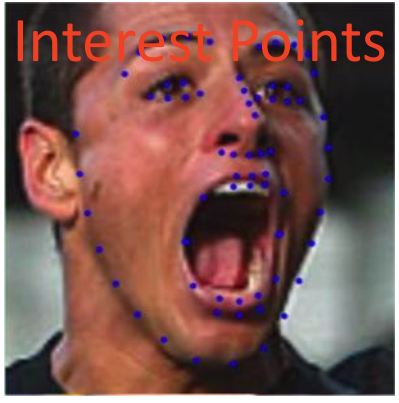
Architecture for Classification

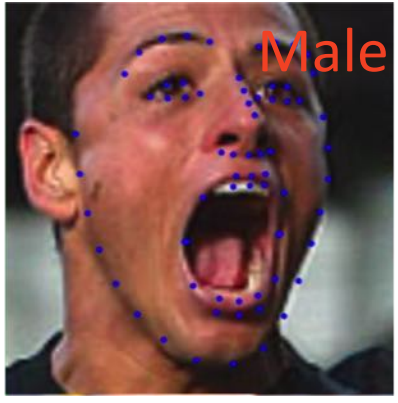


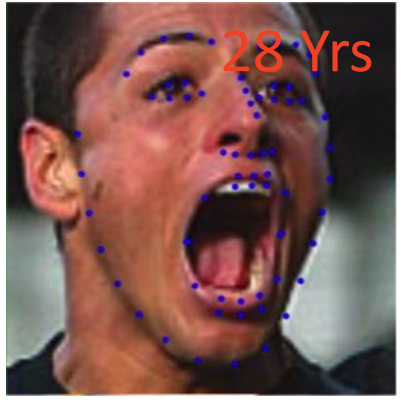
Tensor-Based Models

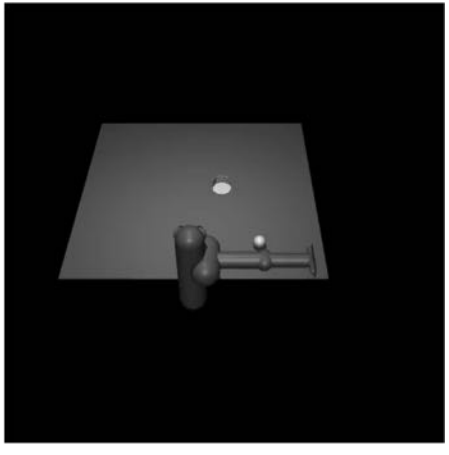
Transfer Learning

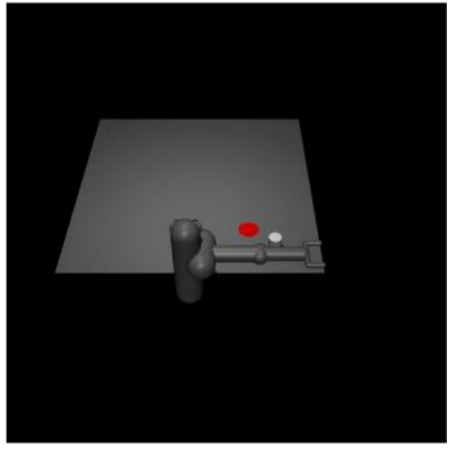
Multi-Source Knowledge Transfer Scenarios

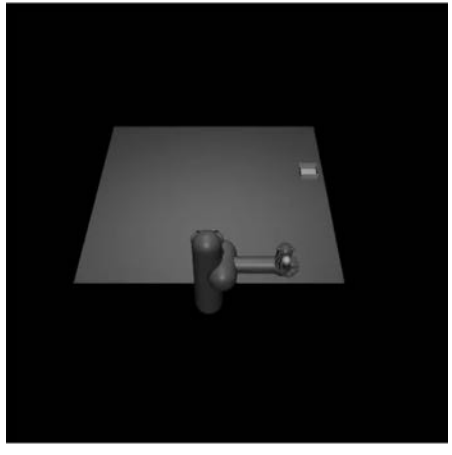












Hit

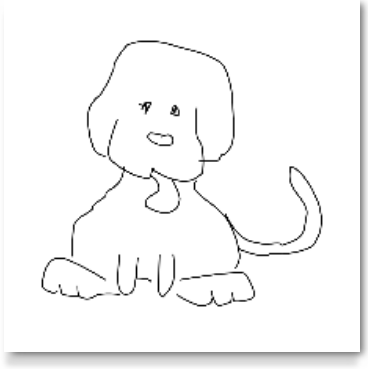
Push

Throw

Multi-Source Knowledge Transfer Scenarios

Source

Target

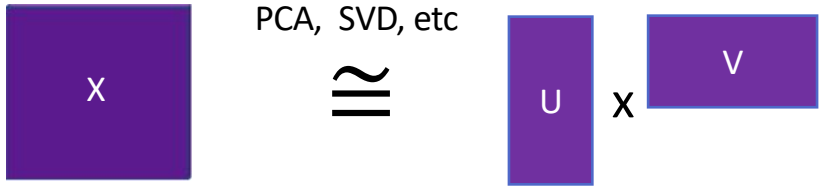


Tensor Composition vs Decomposition

- You may have heard of matrix factorisation.
  - We also have tensor factorisation .

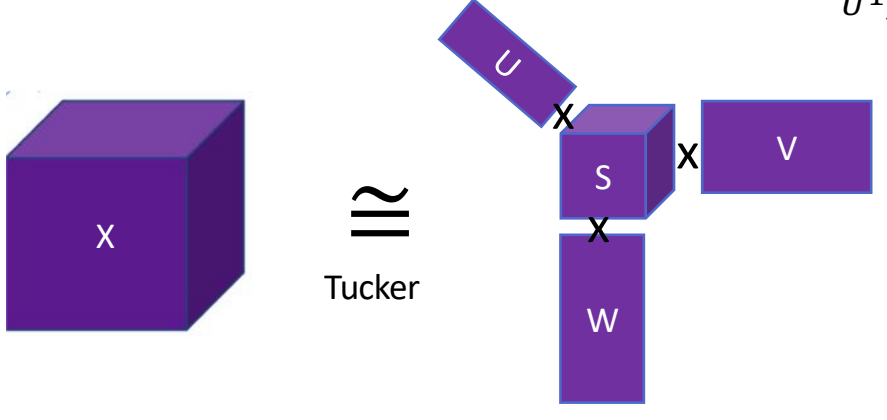
$$\min_{U,V} \|X - UV\|^2$$

$$\text{Dim}(X) = N \times D$$
$$\text{Dim}(U) = N \times R$$
$$\text{Dim}(V) = R \times D$$



$$\min_{U^1,U^2,U^3} \left\| X_{i,j,k} - S \times_1 U^1 \times_2 U^2 \times_3 U^3 \right\|^2$$

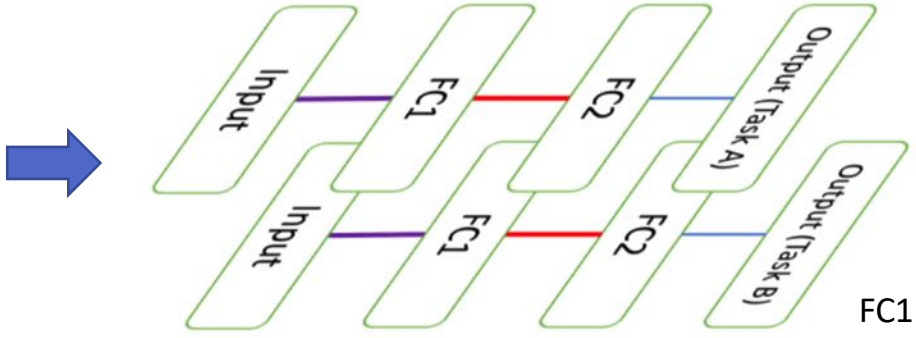
$$\text{Dim}(X) = L \times M \times N$$
$$\text{Dim}(S) = R_1 \times R_2 \times R_3$$
$$\text{Dim}(U^1) = L \times R_1$$
$$\text{Dim}(U^2) = M \times R_2$$
$$\text{Dim}(U^3) = N \times R_3$$

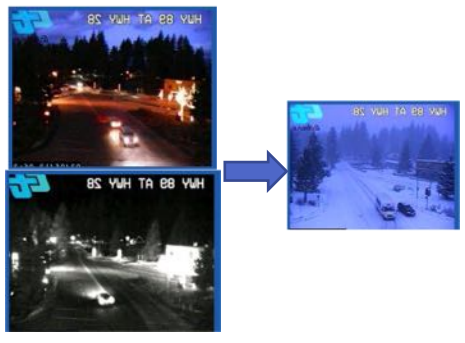


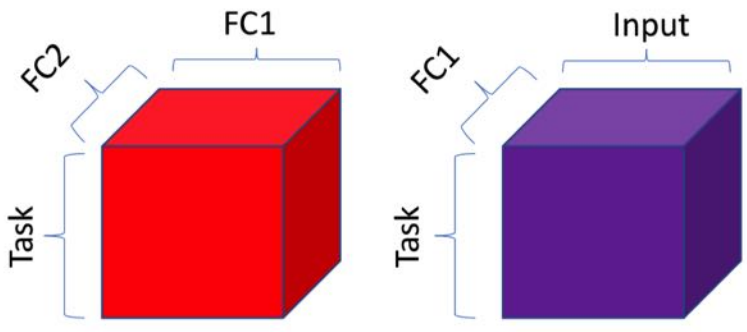
Tensor-based Transfer: Background

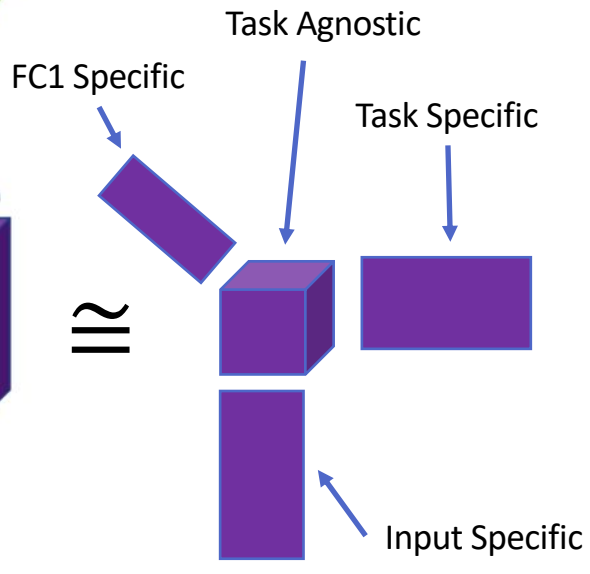
1. Find the low-rank subspace spanned by the source tasks.

Source Tasks
Day/Night/Summer
Expression/Gender
Push/Hit







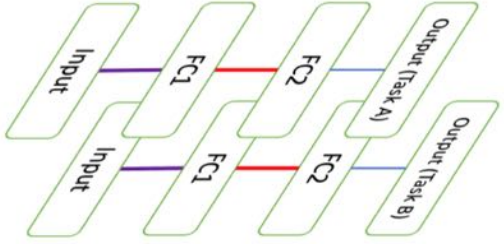


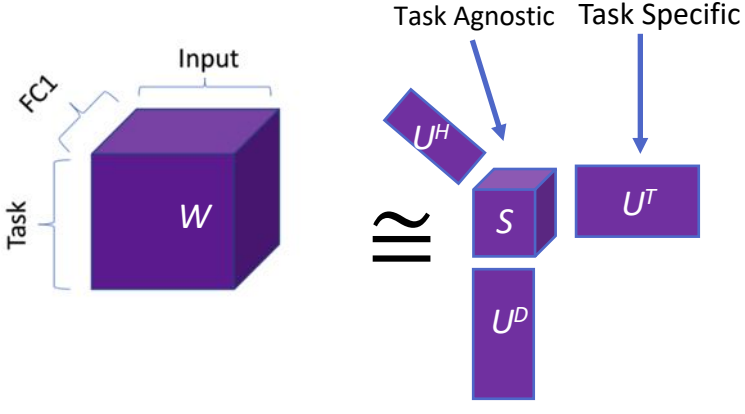
Tucker Decomposition

Tensor-based Transfer: Background

1. Find the low-rank subspace spanned by the source tasks.

Source Tasks
Day/Night/Summer
Expression/Gender
Push/Hit





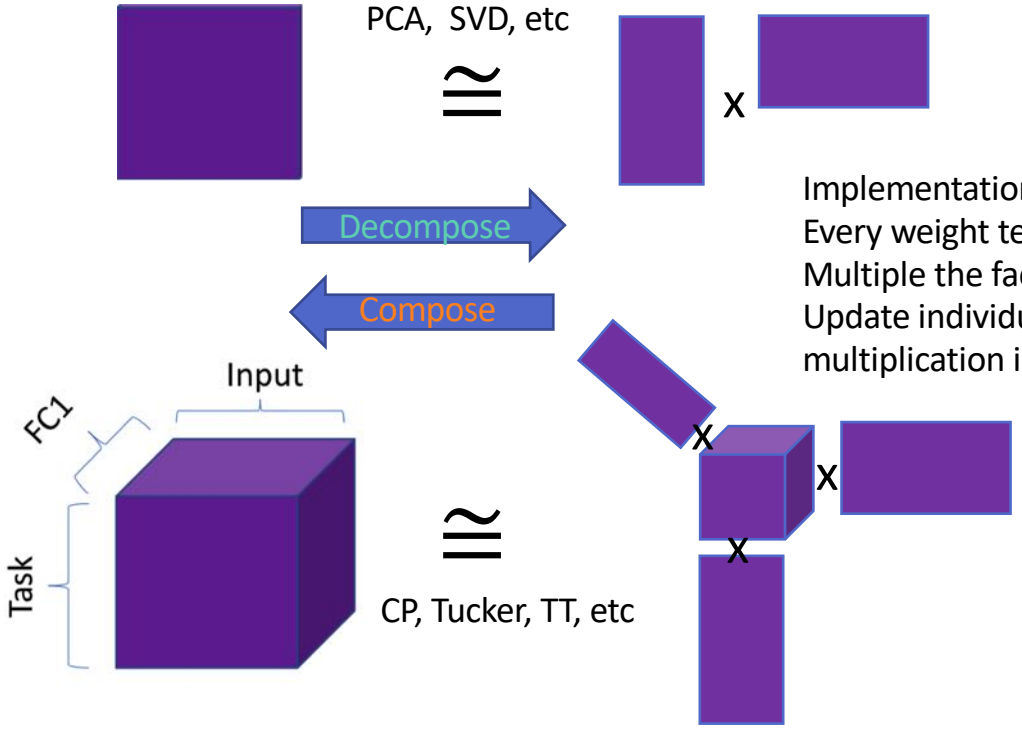
Original Tensor W is size: $T \times D \times H$.
Approximate it as product of low-rank factors:
$W \approx S \times_1 U^T \times_2 U^D \times_3 U^H$

$\dim(S)$: $R \times R \times R$ (Agnostic)
$\dim(U^T)$: $R \times T$ (Task specific)
$\dim(U^D)$: $R \times D$ (Input specific)
$\dim(U^H)$: $R \times H$ (Hidden unit specific)

Compress params from TDH to $R^3 + RT + RD + RH$

Note: Tensor Composition vs Decomposition

- You may have heard of matrix/tensor decomposition.
  - Note that we are doing matrix/tensor composition.

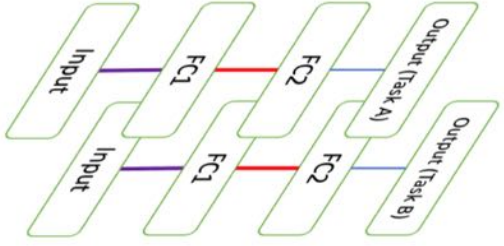


Implementation:
Every weight tensor is replaced with a set of factors.
Multiple the factors out into the tensor for forward pass.
Update individual factors in back pass (no problem, multiplication is differentiable)

Tensor-based Transfer: Multi-Task Source Training

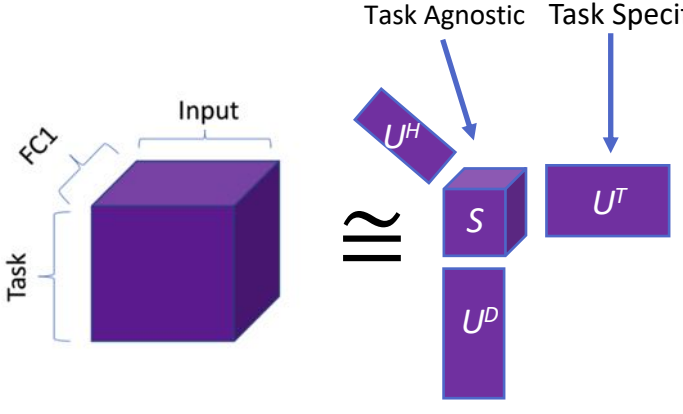
1. Find the low-rank subspace spanned by the source tasks.

Source Tasks
Day/Night/Summer
Expression/Gender
Push/Hit



$$\min_{..W..} \sum_i \sum_k \mathcal{L}(y_i^k, f_{\phi_k}(g_W(x_i))$$

$$\min_{..S,U^T,U^D,U^H,..} \sum_i \sum_k \mathcal{L}(y_i^k, f_{\phi_k}(g_W(x_i))$$



Original Tensor W is size: $T \times D \times H$.
Approximate it as product of low-rank factors:
$W \approx S \times_1 U^T \times_2 U^D \times_3 U^H$

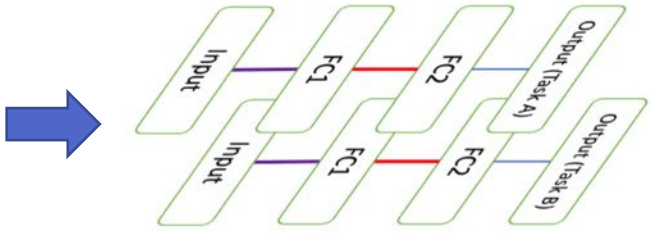
Where: $R < T, D, H$
$\dim(S)$: $R \times R \times R$ (Agnostic)
$\dim(U^T)$: $R \times T$ (Task specific)
$\dim(U^D)$: $R \times D$ (Input specific)
$\dim(U^H)$: $R \times H$ (Hidden unit specific)

Compress params from TDH to $R^3 + RT + RD + RH$

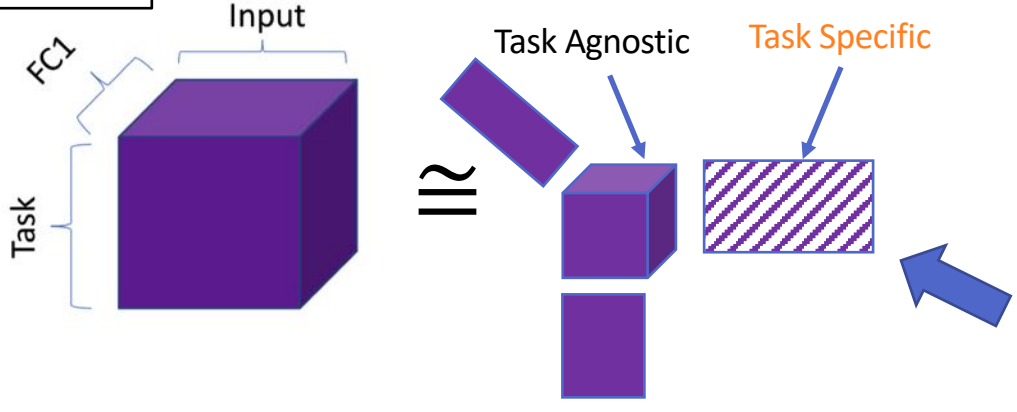
Tensor-based Transfer: Idea

1. Find the low-rank subspace spanned by the source tasks.
2. Train only the task-specific parameters for new task.

Source Tasks
Day/Night/Summer
Expression/Gender
Push/Hit

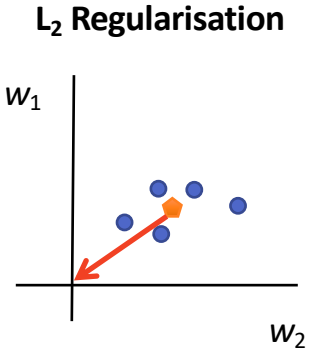


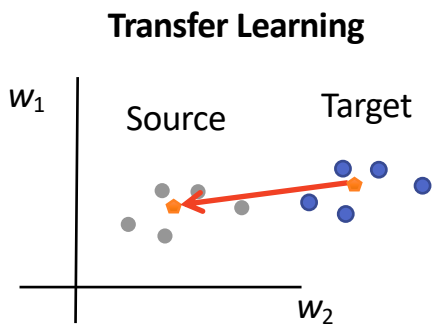
- Few parameters to train.
  - => Fast and overfitting resistant

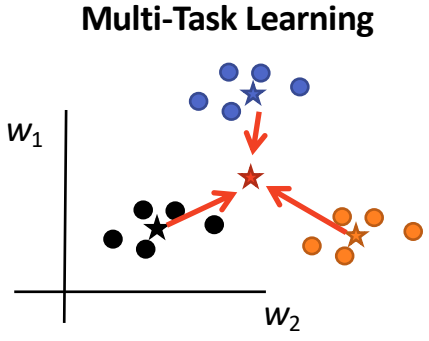


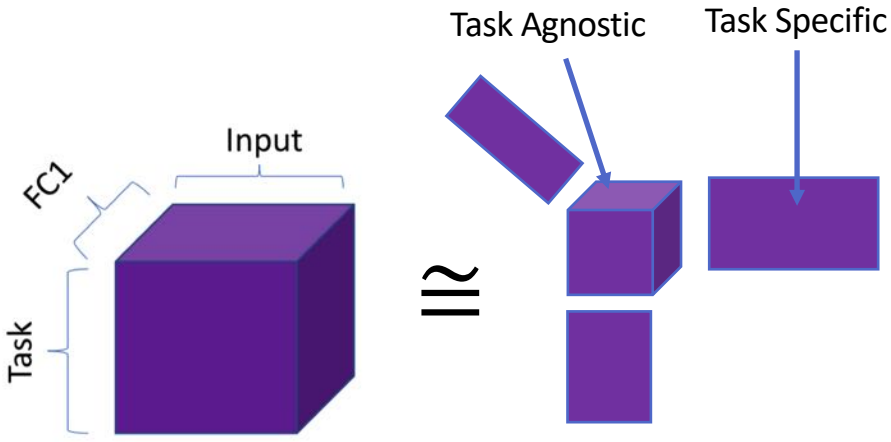
Target Tasks
Winter
Age
Throw

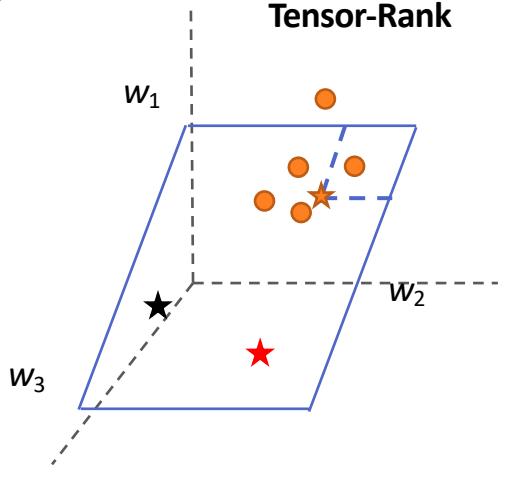
Tensor-based Transfer: Context







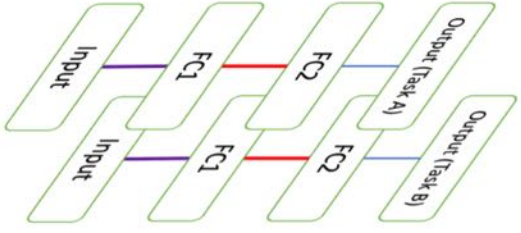


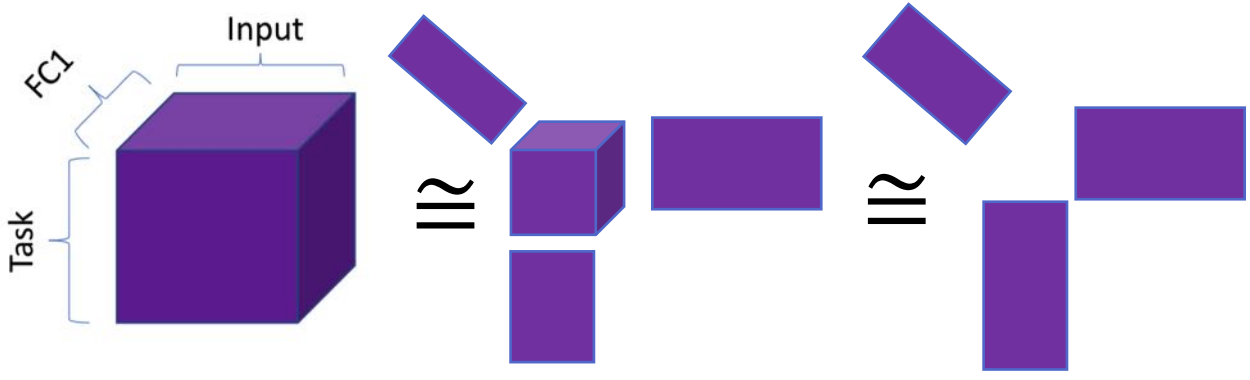


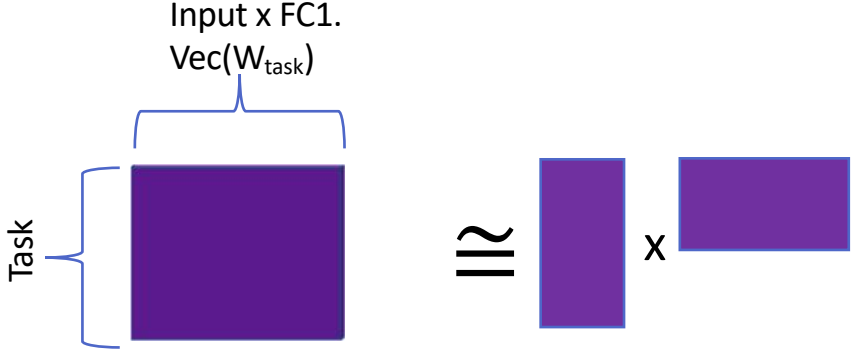
[Yang & Hospedales, ICLR'15; Yang & Hospedales ICLR'17]

Tensor-based Transfer:

Question: What is the difference between these two strategies?





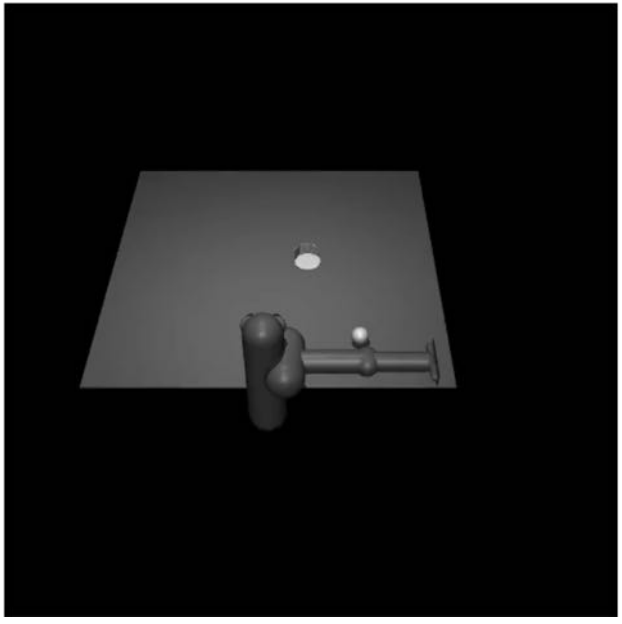


Tensor-Based Transfer Learning: Some Examples

Robot Control

Robot Control Task

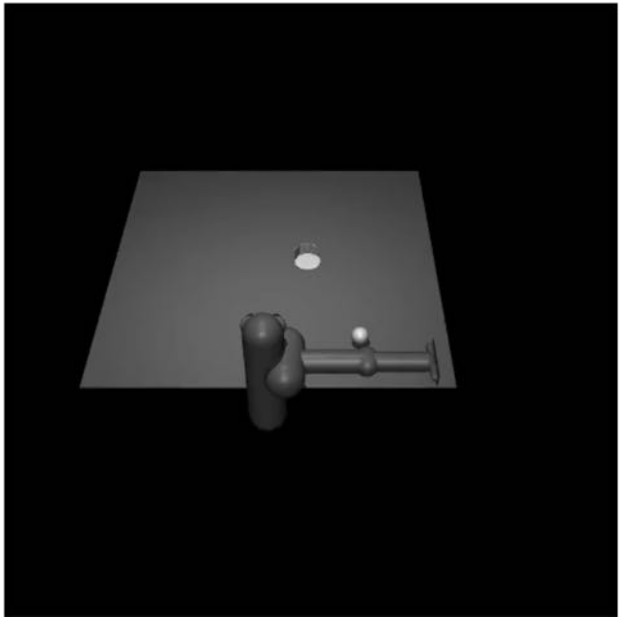
- Consider robot with an arm.
  - Model: $\boldsymbol{a}_t = f_{\boldsymbol{w}}(\boldsymbol{x}_t)$
    - $\boldsymbol{x}_t$: Angle of each joint, time, (ball position, target position)
    - $\boldsymbol{a}_t$: Force to apply at each joint at time t.
    - $f_{\boldsymbol{w}}$: Neural network controller
  - Train with reinforcement learning
    - $R(\boldsymbol{x}_t) = \begin{cases} +1 \text{ if ball hits target} \\ -0.1 \text{ otherwise} \end{cases}$
    - Train: $\boldsymbol{w} = \text{argmax}_{\boldsymbol{w}} \sum_t R(\boldsymbol{x}_t)$



Robot Control Task

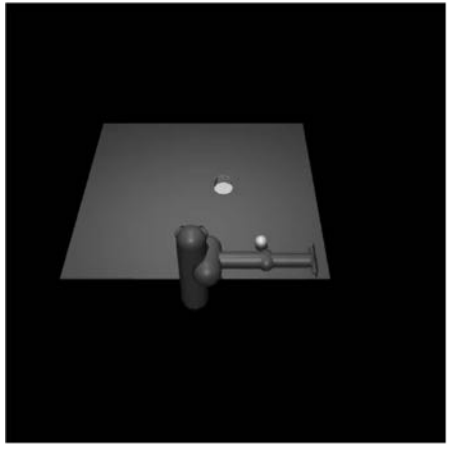
- Consider robot with an arm.
  - Model: $\boldsymbol{a_t} = f_{\boldsymbol{w}}(\boldsymbol{x_t})$
    - $\boldsymbol{x_t}$: Angle of each joint, time, (ball position, target position)
    - $\boldsymbol{a_t}$: Force to apply at each joint at time t.
    - $f_{\boldsymbol{w}}$: Neural network controller
  - Train with reinforcement learning
    - $R(\boldsymbol{x_t}) = \begin{cases} +1 \text{ if ball hits target} \\ -0.1 \text{ otherwise} \end{cases}$
    - Train: $\boldsymbol{w} = \text{argmax}_{\boldsymbol{w}} \sum_t R(\boldsymbol{x_t})$

- Issue:
  - Requires lots of data: $\{\{a_t, x_t, R(x_t)\}_{t=1}^{T}\}_{n=1}^{N}$
  - Each is trial slow and expensive.
  - Redo for each new task.

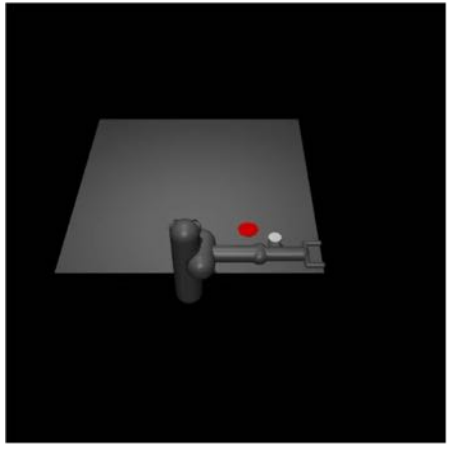


Robot Control Task

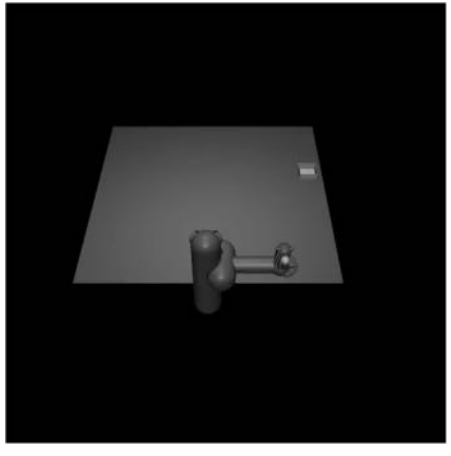
- Given some known tasks, use knowledge transfer to learn new task quicker.



Hit



Push



Throw

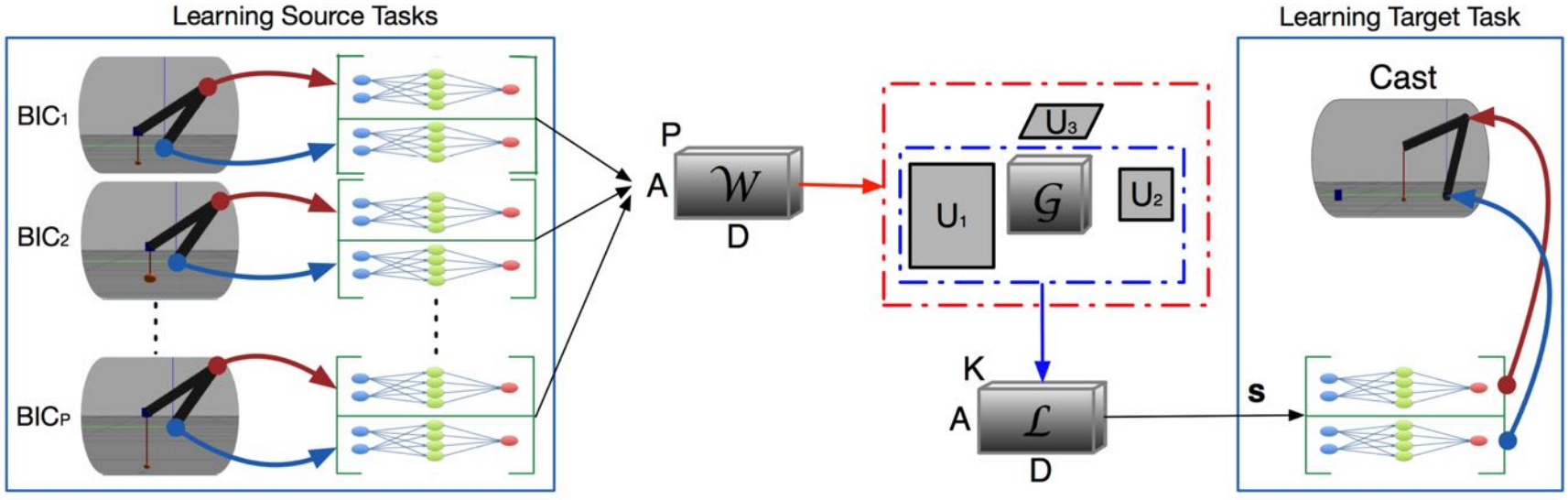
Robot Control Task

- Given some known tasks, use knowledge transfer to learn new task quicker.

Throw Cast Ball-In-Cup

Tensor Transfer for Robot Control

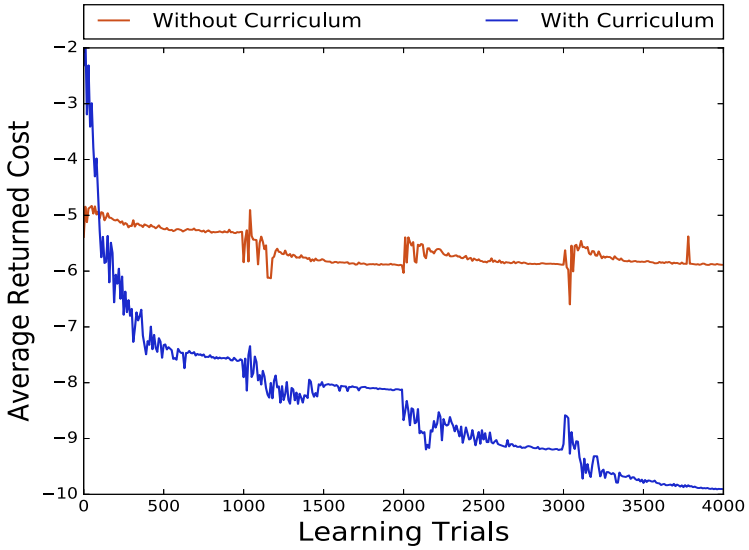
- In this case vectorize entire neural network….

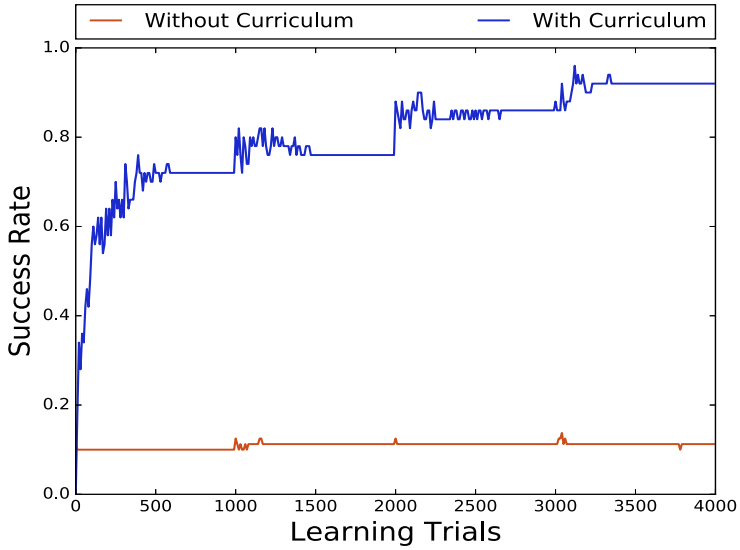


[Yang et al, ICLR'17. Zhao et al, Tensor Based Knowledge Transfer Across Skill Categories for Robot Control, IJCAI, 2017]

Tensor Transfer for Robot Control

- Dramatic Improvement in Learning Efficiency for Target Task:
  - Now possible to learn BIC with RL.





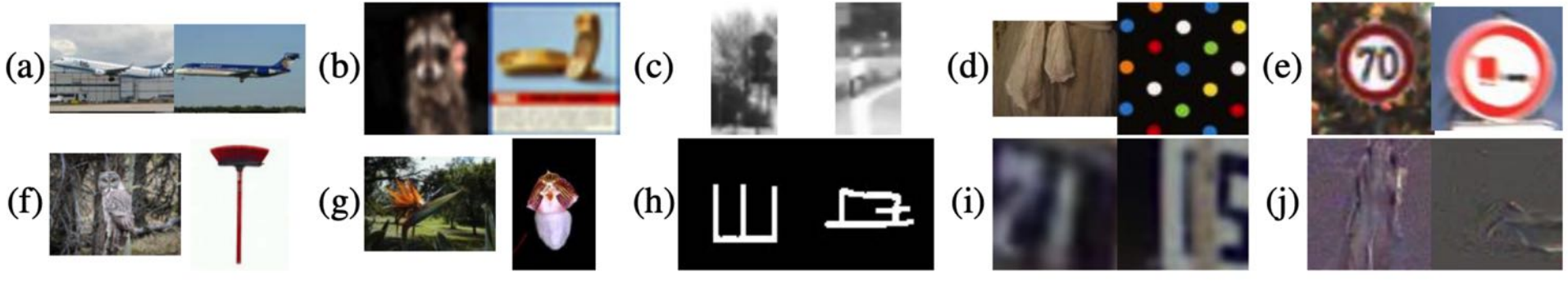
[Yang et al, ICLR'17. Zhao et al, Tensor Based Knowledge Transfer Across Skill Categories for Robot Control, IJCAI, 2017]

Tensor-Based Transfer Learning: Some Examples

Multiple Visual Domains

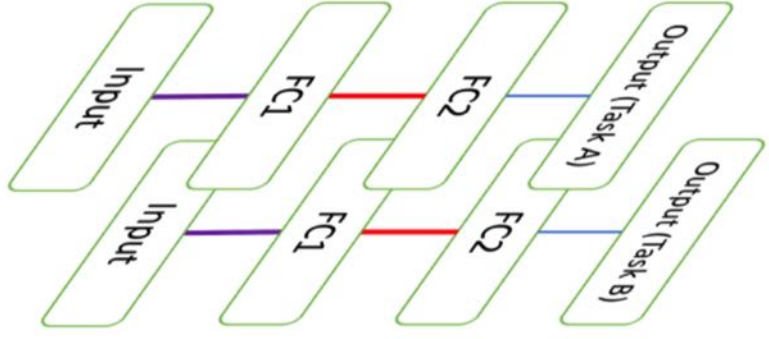
Visual Decathlon Challenge

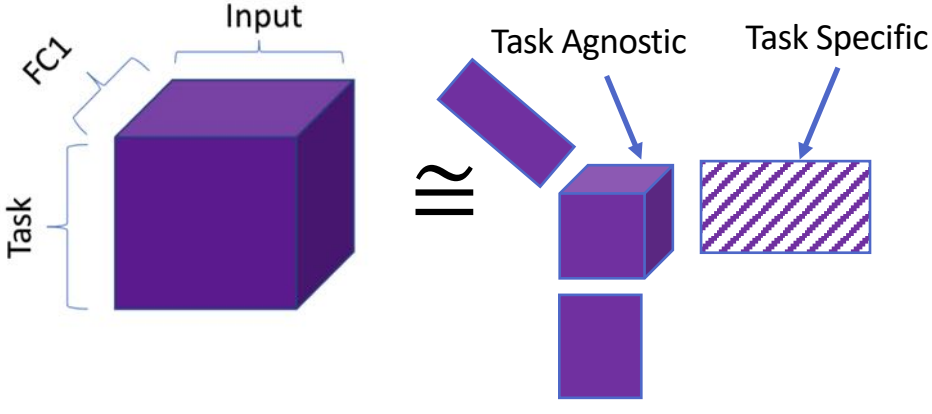
- A small sub-challenge of Artificial General Intelligence (AGI):
  - A general visual feature extractor.
    - Current best results: train one deep feature per dataset.
    - But for general visual intelligence, prefer a single deep network.
- Visual Decathlon Challenge [Rebuffi, NIPS'17, Learning multiple visual domains with residual adapters]
  - 10 datasets: Aircraft, Handwriting, Flowers, Pedestrians, Traffic Signs…
  - Goal is to learn them incrementally, without forgetting, and with minimal parameter growth.



Visual Decathlon: Tensor Solution

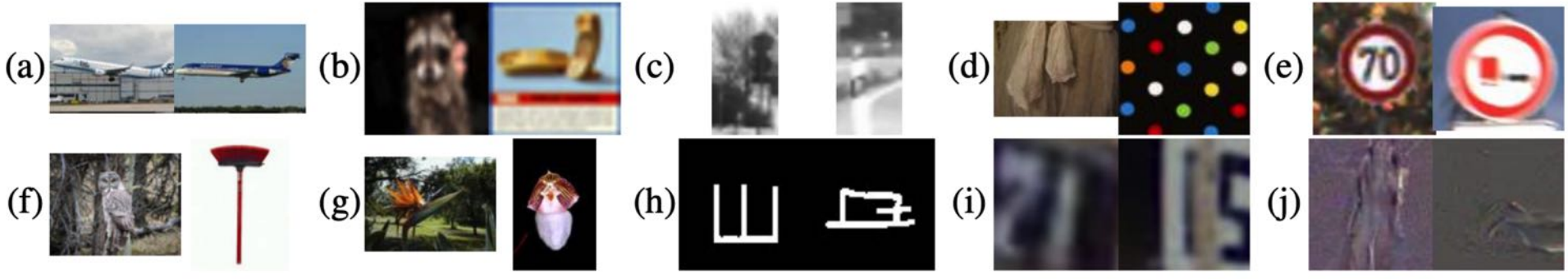
- Train Deep networks for the first K domains
- Extract and fix the core tensor
  - (no forgetting)
- For each new domain, train only task specific factor
  - (small growth pert task)





Tensor-based Transfer: Results

- Visual Decathlon [Bulat et al, arXiv'19]



Model	#param	Dataset										mean	Score
		ImNet	Airc.	C100	DPed	DTD	GTSR	Flwr	OGlt	SVHN	UCF		
#images	-	1.3M	7k	50k	30k	4k	40k	2k	26k	70k	9k	-	-
Rebuffi et al. [29]	2×	59.23	63.73	81.31	93.30	57.02	97.47	83.43	**89.82**	96.17	50.28	77.17	2643
Rosenfeld et al. [32]	2×	57.74	64.11	80.07	91.29	56.54	98.46	86.05	89.67	96.77	49.38	77.01	2851
Mallaya et al. [22]	1.28×	57.69	65.29	79.87	**96.99**	57.45	97.27	79.09	87.63	97.24	47.48	76.60	2838
Series Adap. [30]	2×	60.32	61.87	81.22	93.88	57.13	99.27	81.67	89.62	96.57	50.12	77.17	3159
Parallel Adap. [30]	2×	60.32	64.21	**81.91**	94.73	58.83	99.38	84.68	89.21	96.54	50.94	78.07	3412
Parallel SVD [29]	1.5×	60.32	66.04	81.86	94.23	57.82	99.24	85.74	89.25	96.62	**52.50**	78.36	3398
Ours	1.35×	**61.48**	**67.36**	80.84	93.22	**59.10**	**99.64**	**88.99**	88.91	**96.95**	47.90	**78.43**	**3585**

Tensor-Based Transfer Learning: Some Examples

Fact Induction

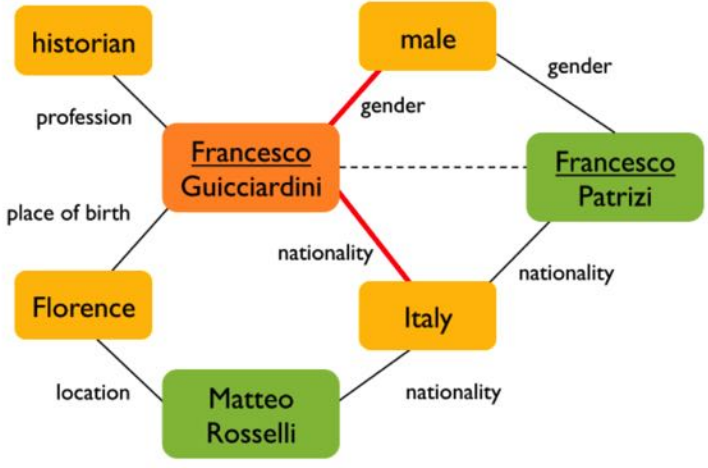
Knowledgebase Completion

- Induce Missing Facts:
  - Completion model induces missing tuples:
    - <Cambridge, locatedIn, England>
  - Task := Relation
    - → Independent tasks.

Typical Solution: Train Embeddings θ:

- vec_θ("Cambs.")+vec_θ("locatedIn")≅vec_θ("England")
- vec_θ("Cambs.")*$matrix_\theta$("locatedIn")≅vec_θ("England")
- vec_θ("Cambs.")*$matrix_\theta$("locatedIn")*vec_θ("England") ≅ 1

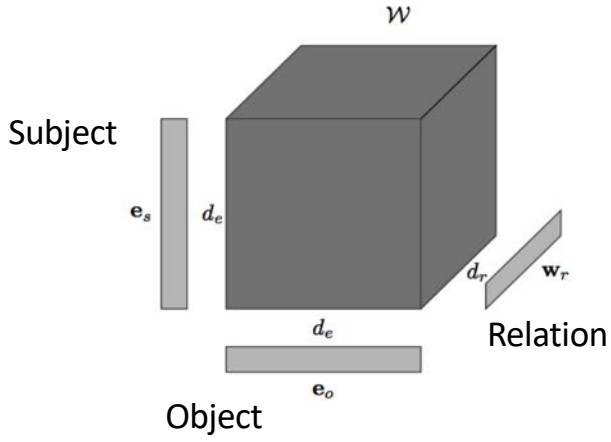
Regression Problem. One per relation.



[Balazevic et al, arXiv'18; arXiv'19]

Knowledgebase Completion

Share across regressions (and entities)



- Induce Missing Facts:

Typical Solution: Train Embeddings θ:

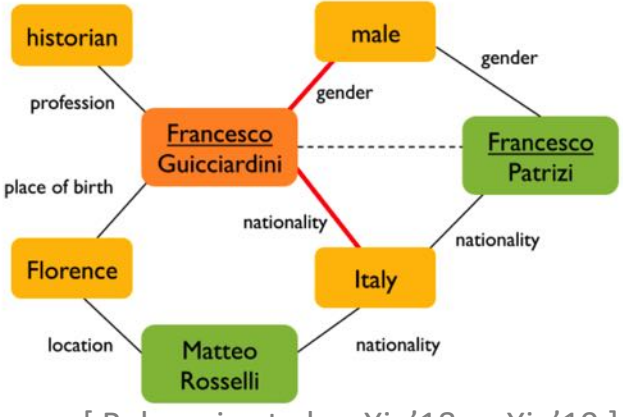
- $\text{vec}_\theta(\text{“Cambs.”}) * \text{matrix}_\theta(\text{“locatedIn”}) \cong \text{vec}_\theta(\text{“England”})$
- $\text{vec}_\theta(\text{“Cambs.”}) * \text{matrix}_\theta(\text{“locatedIn”}) * \text{vec}_\theta(\text{“England”}) \cong 1$

- CoreTensor x_1 $\text{vec}_\theta(\text{“locatedIn”})$ x_2 $\text{vec}_\theta(\text{“Cambs.”})$ x_3 $\text{vec}_\theta(\text{“England”}) \cong 1$
- CoreTensor x_1 $\text{vec}_\theta(\text{“locatedIn”})$ x_2 $\text{vec}_\theta(\text{“Cambs.”}) \cong \text{vec}_\theta(\text{“England”})$

Generates a DxD matrix

⇒ View as a weight generation net.

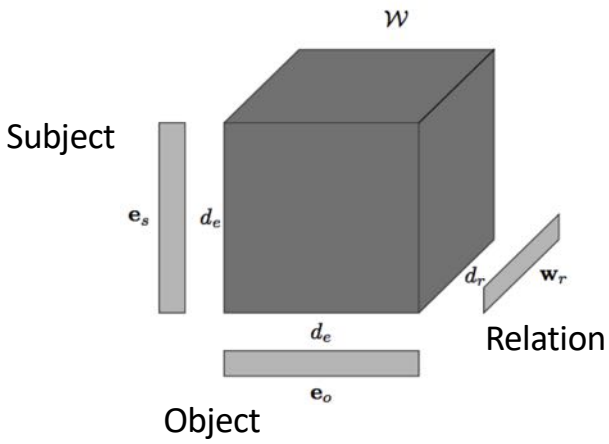
To learn one relation:
Before: D^2 parameters.
Now: R<D parameters.



[Balazevic et al, arXiv'18; arXiv'19]

Knowledgebase Completion

Share across regressions (and entities)

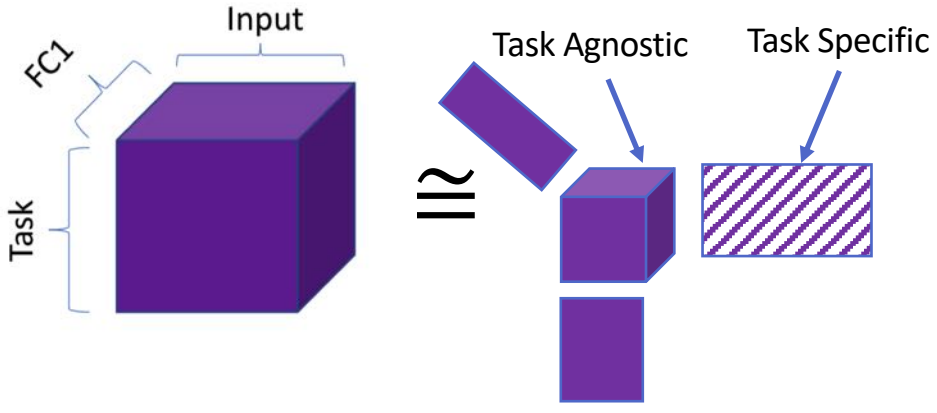


- Induce Missing Facts:
  - Completion model induces missing tuples:
    - <Cambridge, locatedIn, England>
  - Task := Relation
    - → Independent tasks.

		WN18RR				FB15k-237			
	Linear	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
DistMult (Yang et al., 2015)	yes	.430	.490	.440	.390	.241	.419	.263	.155
ComplEx (Trouillon et al., 2016)	yes	.440	.510	.460	.410	.247	.428	.275	.158
Neural LP (Yang et al., 2017)	no	–	–	–	–	.250	.408	–	–
R-GCN (Schlichtkrull et al., 2018)	no	–	–	–	–	.248	.417	.264	.151
MINERVA (Das et al., 2018)	no	–	–	–	–	–	.456	–	–
ConvE (Dettmers et al., 2018)	no	.430	.520	.440	.400	.325	.501	.356	.237
HypER (Balažević et al., 2018)	no	.465	.522	.477	.436	.341	.520	.376	.252
M-Walk (Shen et al., 2018)	no	.437	–	.445	.414	–	–	–	–
TuckER (ours)	yes	**.470**	**.526**	**.482**	**.443**	**.358**	**.544**	**.394**	**.266**

Tensor Knowledge Transfer

Summary:



- Pros:
  - Fast
  - Sample-efficient
  - Parameter-efficient
  - Relatedness is not crucial if enough ranks.

- Cons:
  - Need many source tasks (cf: Few-Shot vs ”Fine-tuning”)
  - Tasks need same “shape”

Tensor vs Metric Based Few-Shot

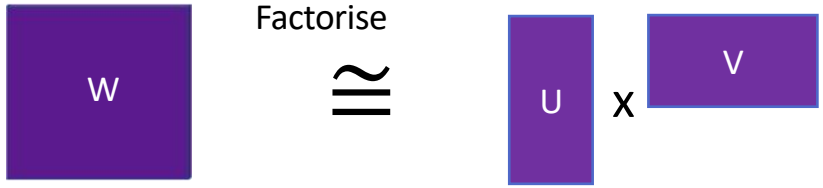
- Tensor-based
  - Suitable for any task (classification, regression, detection)
  - Weak assumption on known source task relevance (vs fine-tune).
  - Requires (-) and exploits (+) multiple sources-.
- Metric-based
  - Suitable for classification.
  - Suitable to extend an existing label-space.
  - No requirement (+), and limited exploitation (-) of multi-source.

Tensors: Bonus Topics

Tensor-Based Network Compression

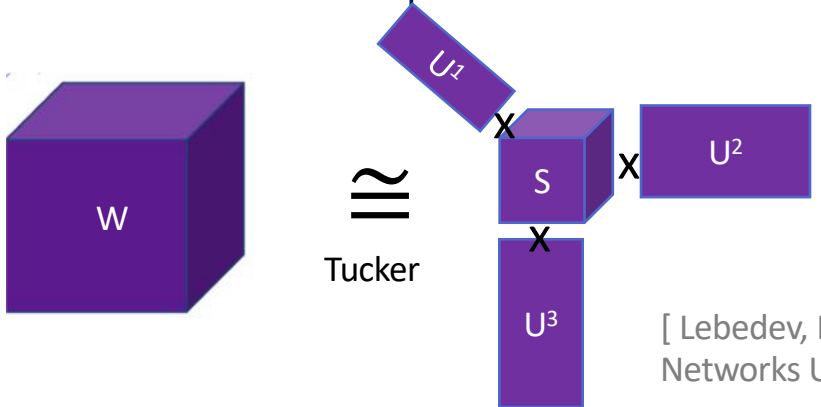
Tensor-Factorisation for Model Compression

- Aside: These tensor methods also widely used for model compression & acceleration.
  - Fully Connected Compression: $D^1 \times D^2 \rightarrow D^1 \times r + D^2 \times r$



Example Pipeline:
1. Train full network.
2. Decompose
3. Fine-tune

  - Convolution Compression:



$$\text{Dim}(W) = W \times H \times C_{in} \times C_{out}$$
$$\text{Dim}(S) = R_1 \times R_2 \times R_3 \times R_4$$
$$\text{Dim}(U^1) = W \times R_1$$
$$\text{Dim}(U^2) = H \times R_2$$
$$\text{Dim}(U^3) = C_{in} \times R_3$$
$$\text{Dim}(U^4) = C_{out} \times R_3$$

[Lebedev, ICLR'15, Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition]

Tensors: Bonus Topics

Tensor-Based Modality Fusion

How to fuse multi-modal data?

- Often: One prediction, Two signals: Audio, video. Still, Motion.
- Typical answer: $y = \boldsymbol{w}^T[\boldsymbol{x}_a, \boldsymbol{x}_b] = \boldsymbol{w}_a^T\boldsymbol{x}_a + \boldsymbol{w}_b^T\boldsymbol{x}_b$
  - Shallow model: concatenate (implies linear combination)
  - Deep model: concatenate and pass through MLP. $\boldsymbol{y} = \boldsymbol{V}\sigma(\boldsymbol{W}[\boldsymbol{x}_a, \boldsymbol{x}_b])$
- Also: Tensor-based data fusion
  - Tensor fusion layer: $y = \boldsymbol{W}\times_1\boldsymbol{x}_a\times_2\boldsymbol{x}_b$
    - Question: What's the problem with this?
    - Too many $O(N^3)$ parameters:
  - Low-rank tensor fusion layer: $y = \boldsymbol{S}\times_1(U\boldsymbol{x}_a)\times_2(V\boldsymbol{x}_b)$

[Hu et al, Neural Tensor Fusion Networks, ICCV'17]

Outline

- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Part II: Learning with Multiple Domains

What is the difference between transfer learning and domain adaptation?

- Confusion caused by vagueness of term ”transfer learning”
  - If “TL” means <u>any</u> knowledge transfer: DA is a subset of TL.
  - IF “TL” means “supervised” or more specifically ”fine-tune”:
    - (Unsupervised) Domain adaptation is disjoint to TL.
- Problem: Domain-Shift vs Task Shift
- Data:

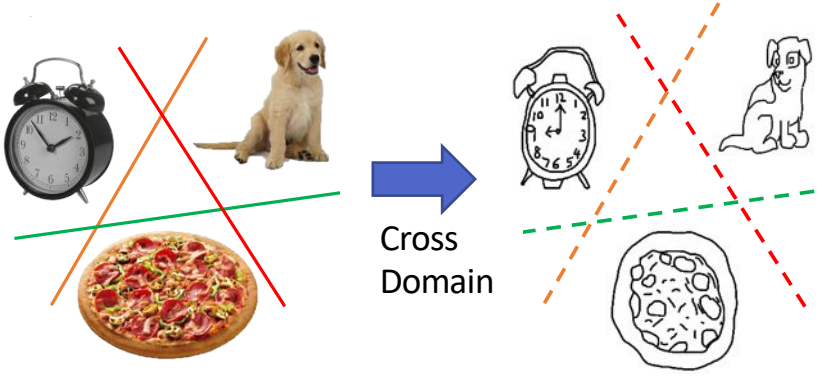
	$p(X_s) \neq p(X_t)$	$p(Y_s\|X_s) \neq p(Y_t\|X_t)$
$\{X_s, Y_s\} \rightarrow \{X_t, Y_t\}$	Supervised Domain Adaptation. (Incl: Fine-tuning, Tensor).	Supervised Task Adaptation. (Incl Fine-tuning. Tensor).
$\{X_s, Y_s\} \rightarrow \{X_t\}$	Unsupervised DA.	Impossible. Unless ZSL.
$\{X_s, Y_s\} \rightarrow \{\}$	Domain Generalization	Impossible. Unless ZSL.

Confusing:
Some methods, e.g., FT, Tensor, can apply to both.

Learning with Multiple Domains

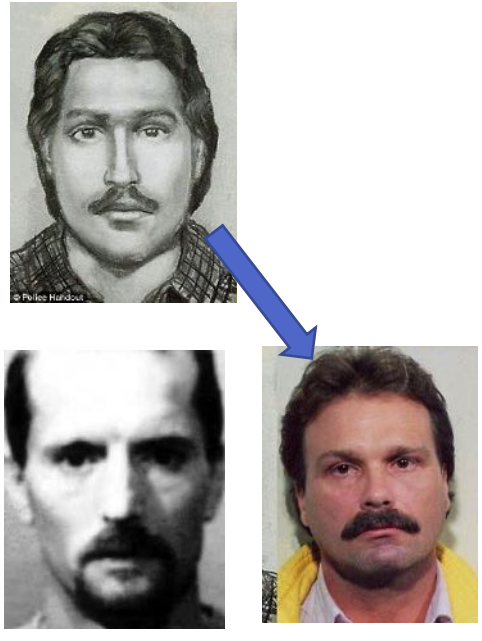
Solving Domain-Shift:

- Model still works, even when data statistics change:
  - Different domains are a nuisance variable:



Cross Domain Matching:

- Your goal is to match across domains.



Cross-Domain Matching

Cross Domain Matching: Examples

- Text->Image Retrieval

a white plate topped with a cut in half sandwich.

a city street filled with lots of traffic.

a train traveling down tracks next to a forest.

Cross Domain Matching: Examples

- Image->Caption Retrieval

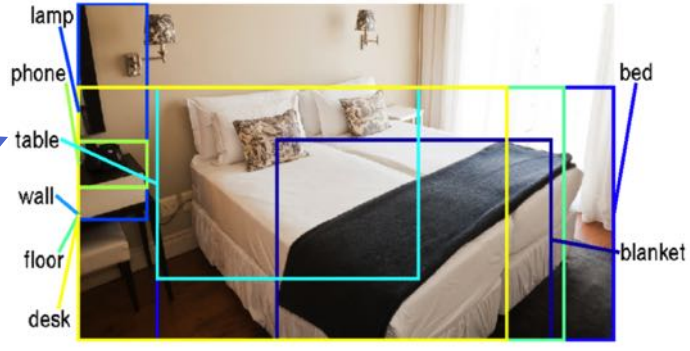
a white plate topped with a cut in half sandwich.

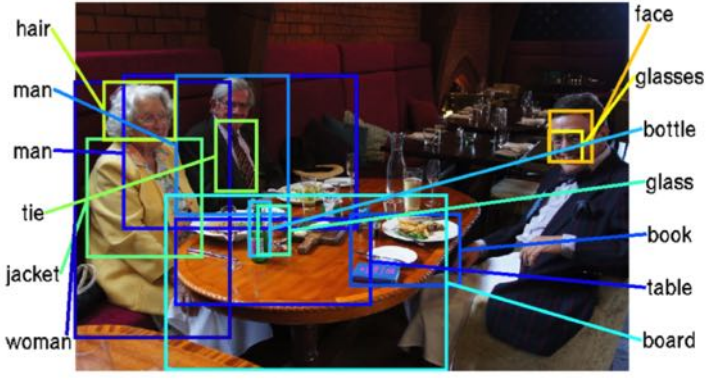
a city street filled with lots of traffic.

a train traveling down tracks next to a forest.

Cross Domain Matching: Examples

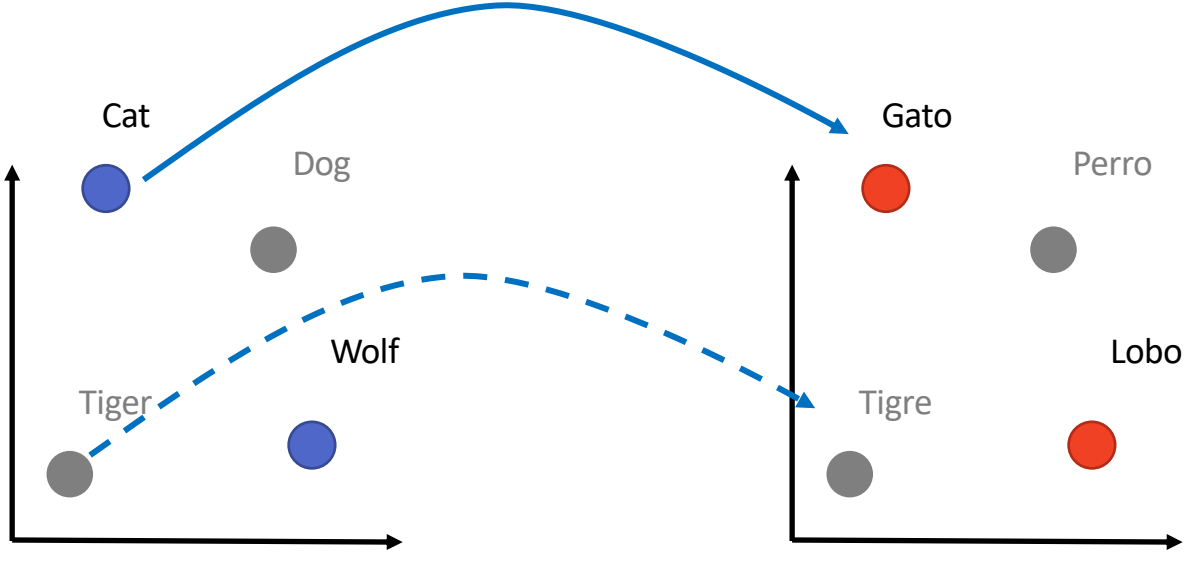
- Graph -> Image Retrieval





Cross Domain Matching: Examples

- Bilingual Dictionary Induction



[Mikolov, arXiv'13, Exploiting similarities among languages for machine translation]

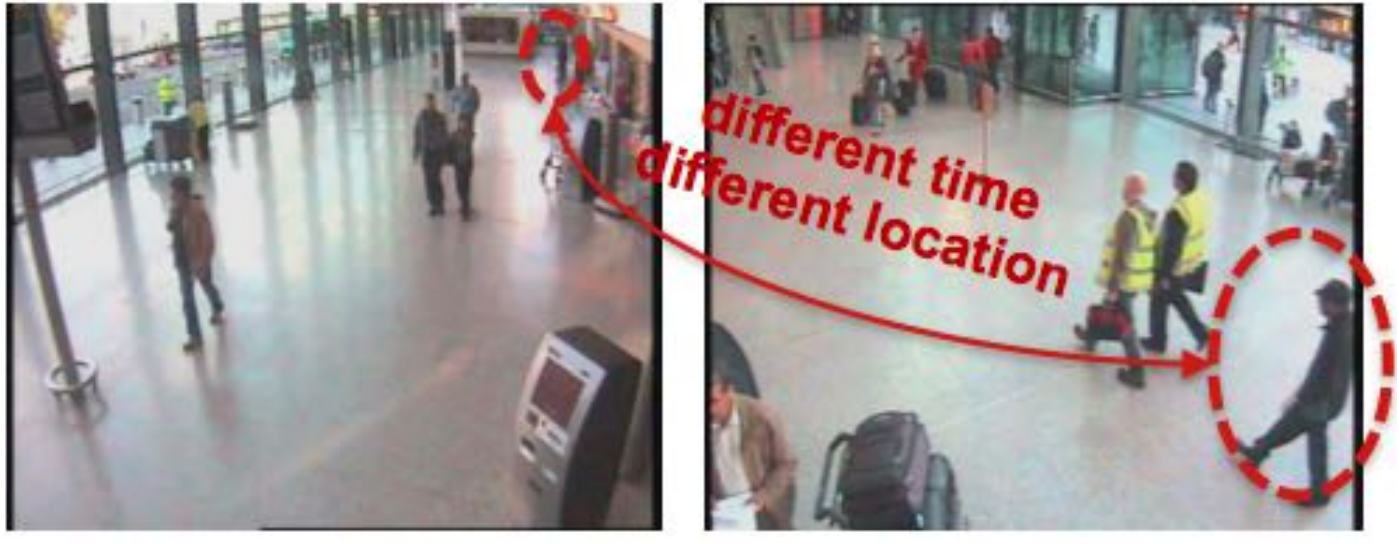
Cross Domain Matching: Examples

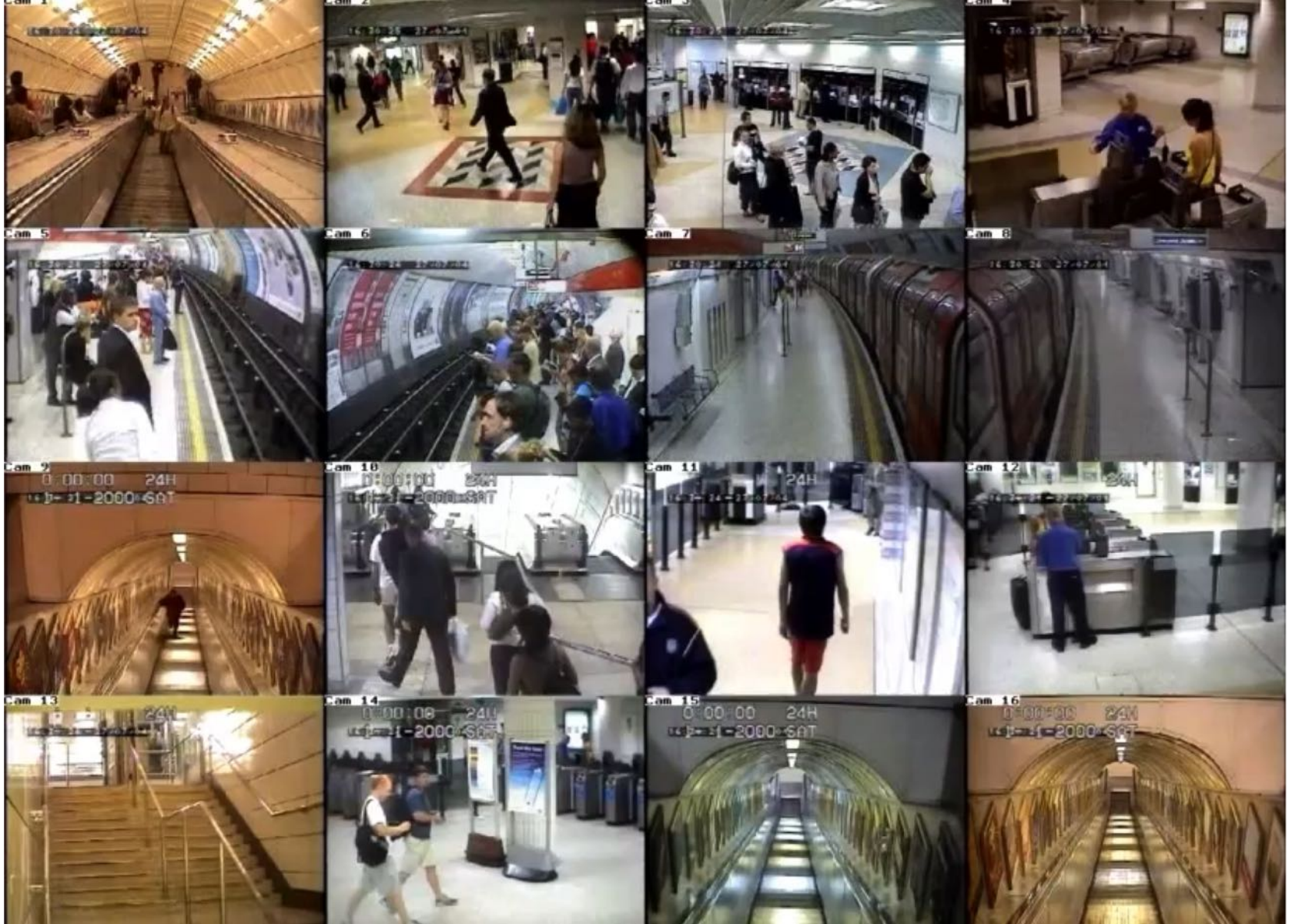
- Sketch-based Image Retrieval: Online Shopping

[Yu et al, CVPR 2016, Sketch me that shoe]

Cross Domain Matching: Examples

- Person Re-Identification



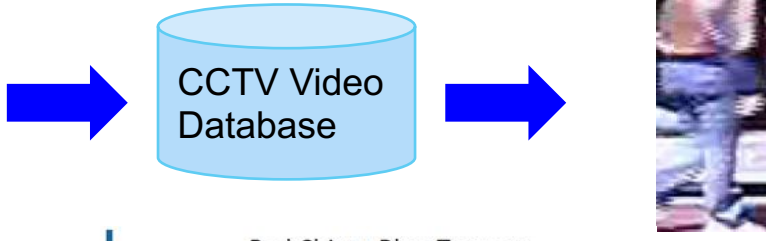

Cam 1
Cam 2
Cam 3
Cam 4
Cam 5
Cam 6
Cam 7
Cam 8
Cam 9
0:00:00 24H
Cam 10
Cam 11
24H
Cam 12
Cam 13
24H
Cam 14
Cam 15
0:00:00 24H
Cam 16

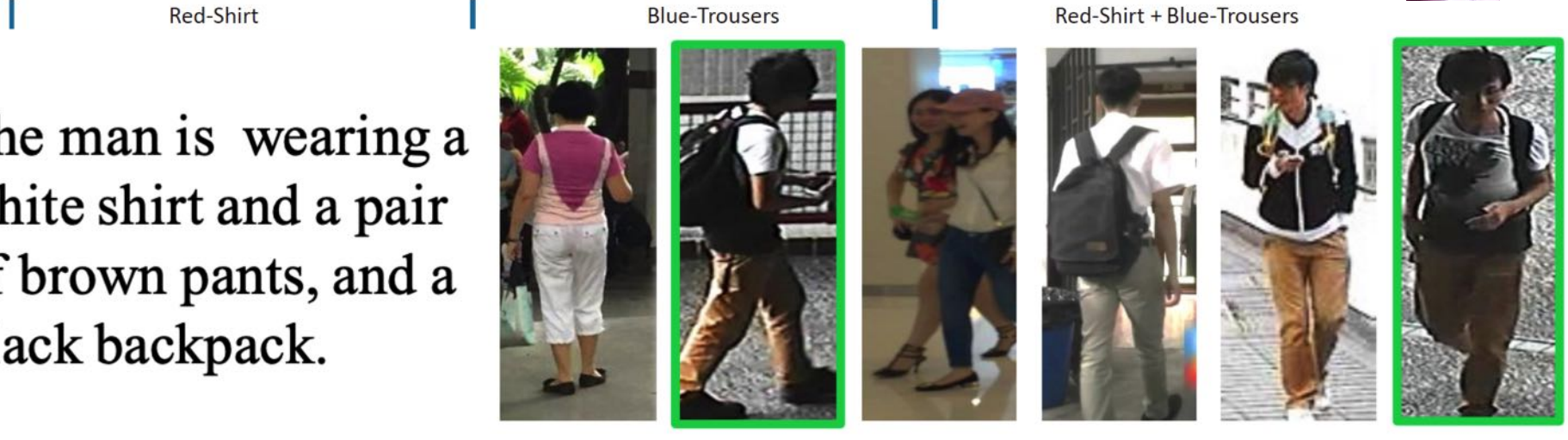
Cross Domain Matching: Examples

- Description-Based Person Search

Description-based person search:

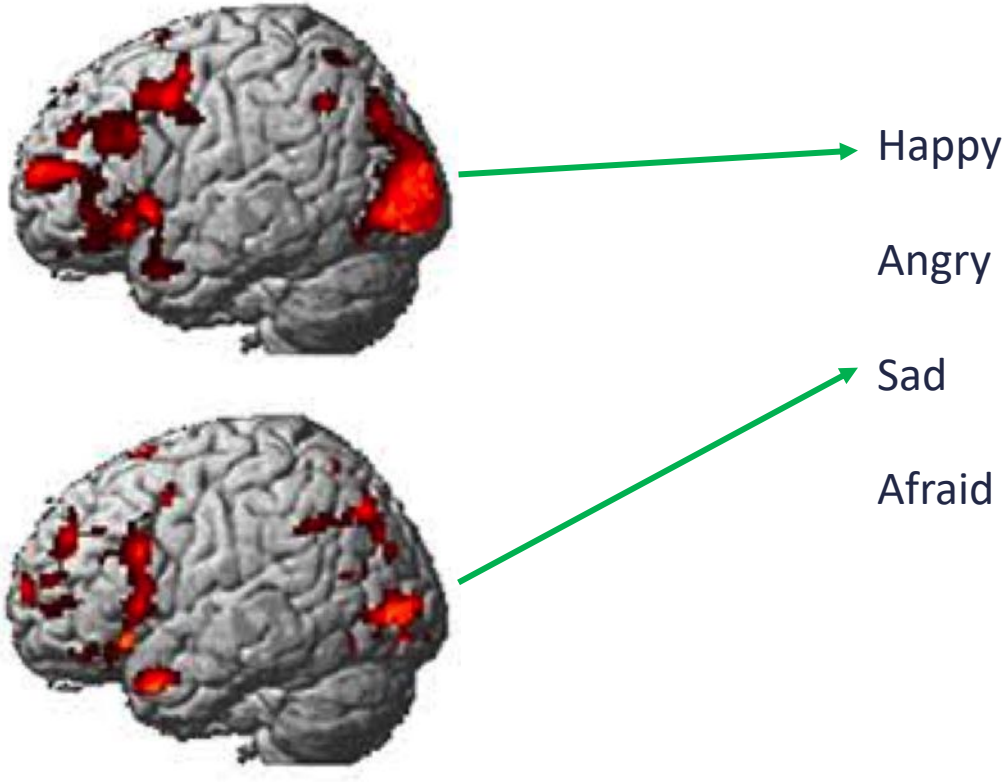
- E.g., “Suspect had blue jeans, brown jacket, and blonde hair”





Cross Domain Matching: Examples

- Mind Reading



[Mitchell, Science'08, Predicting Human Brain Activity Associated with the Meanings of Nouns]

Outline

- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Regression and Energy Function Methods

Matching Approaches: Regression

Linear

- Train:

$$\min_{\boldsymbol{W}} \sum_i \|\boldsymbol{y}_i - \boldsymbol{W}\boldsymbol{x}_i\|$$

- Test: Nearest Neighbour

$$\text{argmin}_{\boldsymbol{y}_j \in Gallery} \|\boldsymbol{y}_j - \boldsymbol{W}\boldsymbol{x}^*\|$$

Deep

- Train:

$$\min_{\boldsymbol{W}} \sum_i \|\boldsymbol{y}_i - f_{\boldsymbol{W}}(\boldsymbol{x}_i)\|$$

- Test: Nearest Neighbour

$$\text{argmin}_{\boldsymbol{y}_j \in Gallery} \|\boldsymbol{y}_j - f_{\boldsymbol{W}}(\boldsymbol{x}^*)\|$$

Matching Approaches: Energy Functions

- Bilinear Train:

$$\min_{\boldsymbol{W}} \sum_{(i,j)\in Pos,(i,k)\in Neg} \left|-\boldsymbol{x}_i^T \boldsymbol{W} \boldsymbol{y}_i - \Delta + \boldsymbol{x}_i^T \boldsymbol{W} \boldsymbol{y}_k\right|^+$$

- Bilinear Test:

$$\text{argmax}_{\boldsymbol{y}_j \in Gallery} \left\| \boldsymbol{y}_j^T \boldsymbol{W} \boldsymbol{x}^* \right\|$$

- Low-Rank

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{(i,j)\in Pos,(i,k)\in Neg} \left|-\boldsymbol{x}_i \boldsymbol{U}\boldsymbol{V} \boldsymbol{y}_i - \Delta + \boldsymbol{x}_i \boldsymbol{U}\boldsymbol{V} \boldsymbol{y}_k\right|^+$$

Matching Approaches: Energy Functions

- Deep Train:

$$\min_{U,V} \sum_{(i,j)\in Pos,(i,k)\in Neg} \left| f_U(x_i)^T f_V(y_i) - f_U(x_i)^T f_V(y_j) - \Delta \right|^+$$

- Deep Test

$$\max_{\boldsymbol{y}_j \in Gal} f_U(x^*)^T f_V(y_j)$$

Connection to Metric Learning:

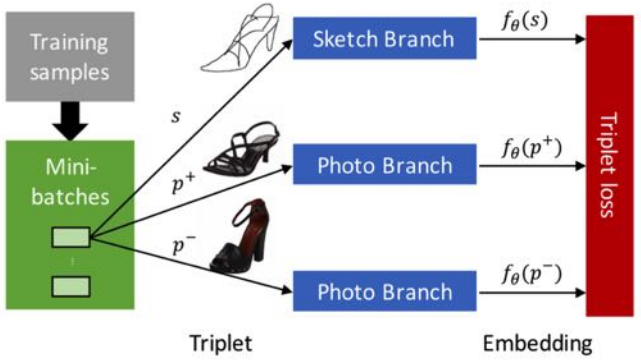
- "Siamese"
- Max energy/min distance

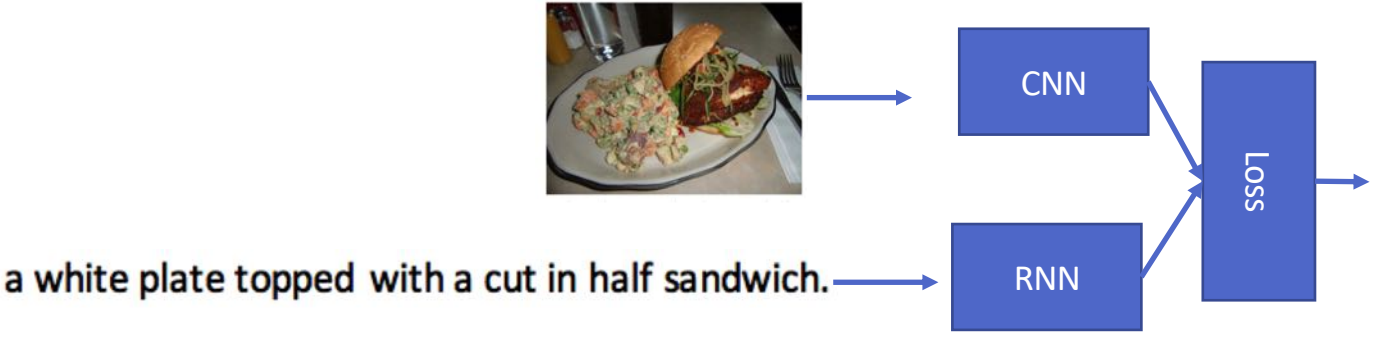
$$\min_{U,V} \sum_{\substack{(i,j)\in Pos \\ (i,k)\in Neg}} \left| d^- \left(f_U(x_i), f_V(x_k) \right) - d^+ \left(f_U(x_i), f_V(y_j) \right) - \Delta \right|^+$$

$$\min_{W} \sum_{\substack{(i,j)\in Pos \\ (i,k)\in Neg}} \left| d^+ \left(f_W(x_i), f_W(y_j) \right) - d^- \left(f_W(x_i), f_W(x_k) \right) + \Delta \right|^+$$

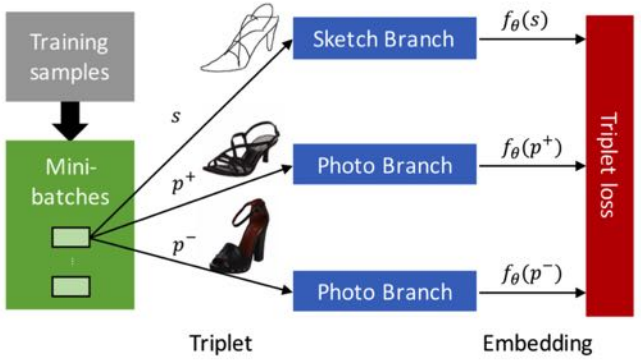
Discussion

- Multiple losses
  - Sometimes good to use classification, matching, and ranking.
- Siamese. Homo. vs Hetero. Branches.
  - Depends on modality heterogeneity.
- Non-CNN encoders:

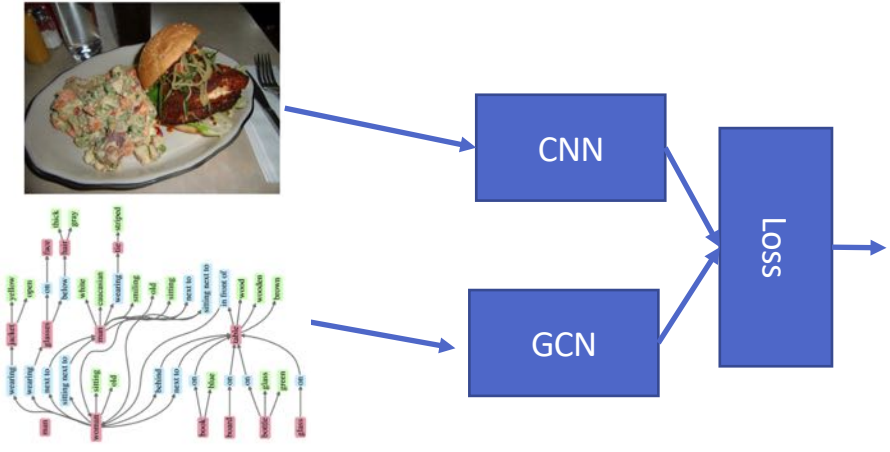




Discussion



- Multiple losses
  - Sometimes good to use classification, matching, and ranking.
- Siamese. Homo. vs Hetero. Branches.
  - Depends on modality heterogeneity.
- Non-CNN encoders:



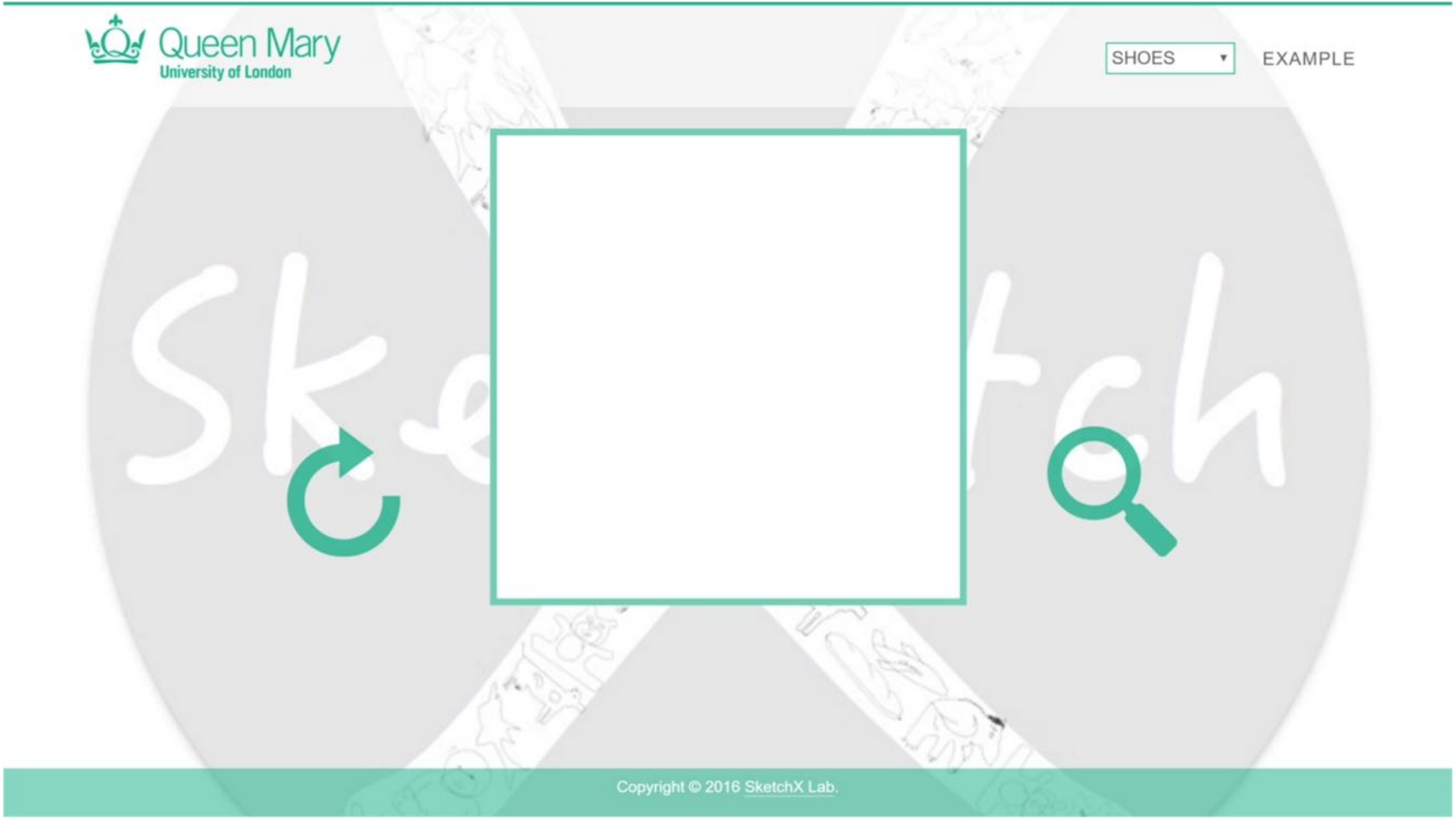
Outline

- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Cross-Domain Regression

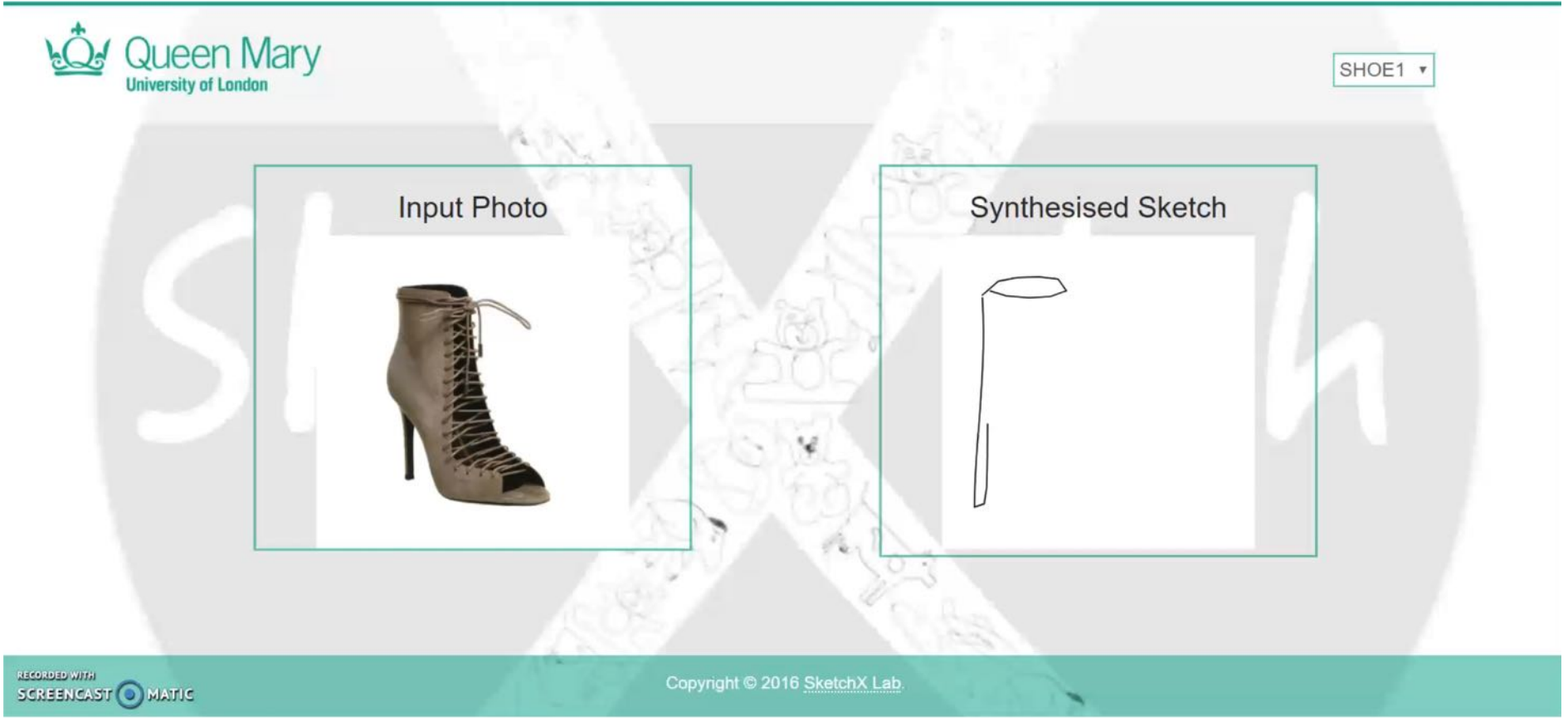
Examples

Sketch-to-Image: Sketch-based Image Retrieval



[CVPR'16 – Sketch me that shoe, ICCV'17, CVPR'18]

Image-to-Sketch: Learning to Draw



[Song, CVPR'18, Learning to Sketch with Shortcut Cycle Consistency]

Zero-Shot Learning

What is Zero-Shot Learning?

Live Demonstration!

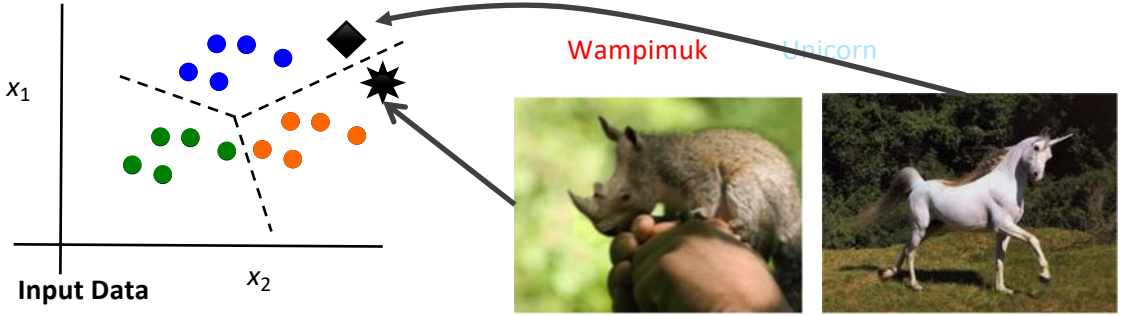
- Audience Task: Recognise the Wampimuk.
  - Impossible?
- Solution: Semantic Transfer
  - Domain Ontology:
    - Wampimuk := small, horns, furry, cute
  - Wikipedia Page:

Zero-Shot:

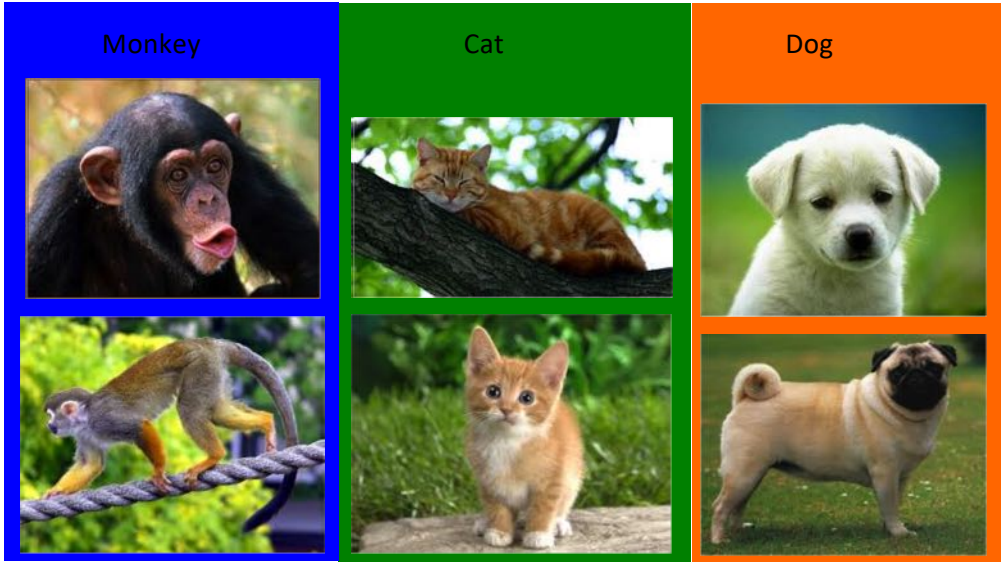
1. Pattern recognition with no training examples
2. Solved by semantic transfer

From Supervised to Zero-Shot Pattern Recognition

Output Labels:
Dog
Cat
Monkey



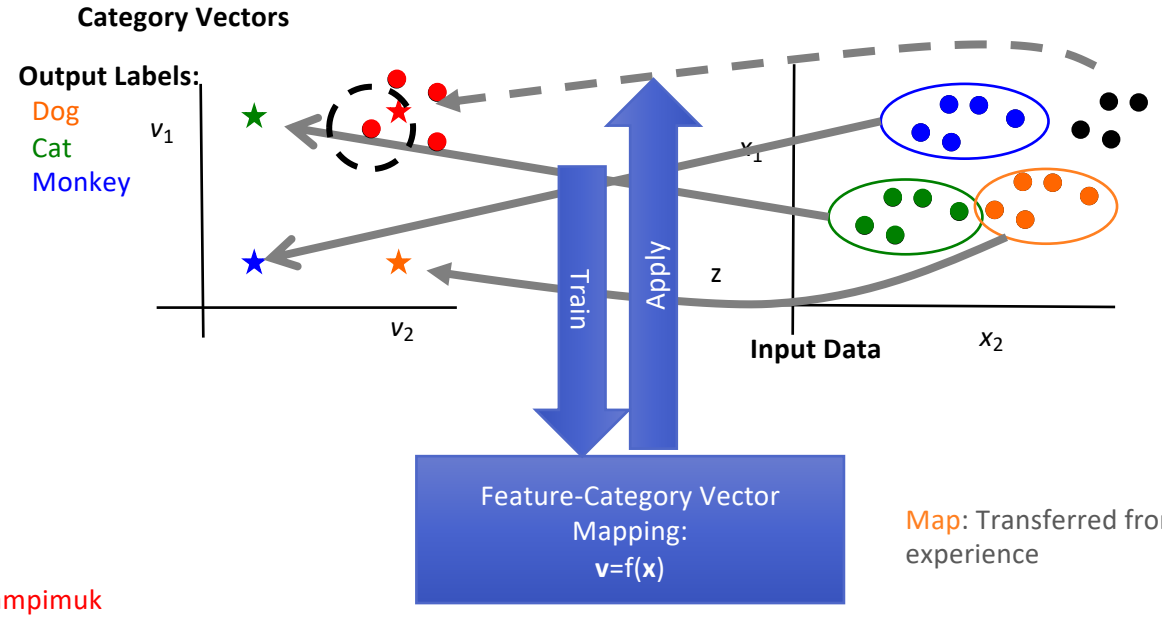
???



From Supervised to Zero-Shot Pattern Recognition

Key is to embed categories as vectors

Task: Transferred from external source



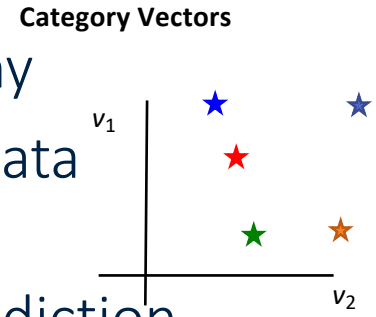
Map: Transferred from past experience

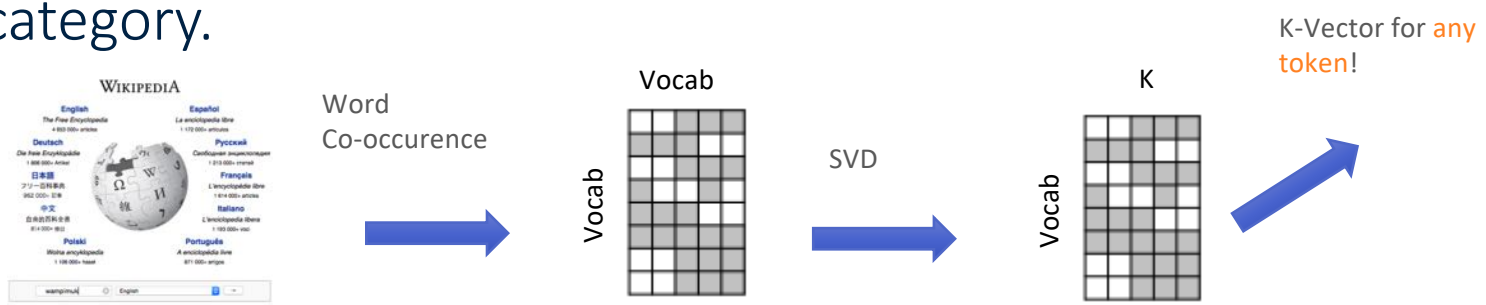
Wampimuk

Data-category vector map can generalize to new categories.

Where to get Category Vectors?

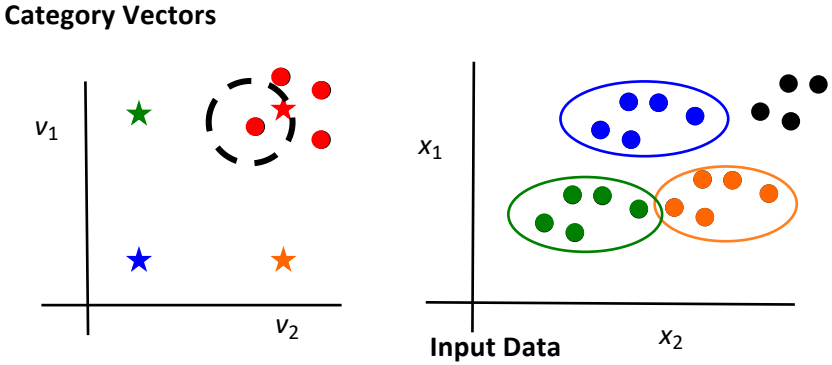
- "Supervised" sources:
  - Manual annotation of class properties
  - Vector encoding of a taxonomic class hierarchy
- "Unsupervised" sources: Existing unstructured data
  - Word (token) co-occurrence.
    - OR: word2vec: Representation of word prediction neural net.
  - => Automatic+Free word vector for any "nameable" category.





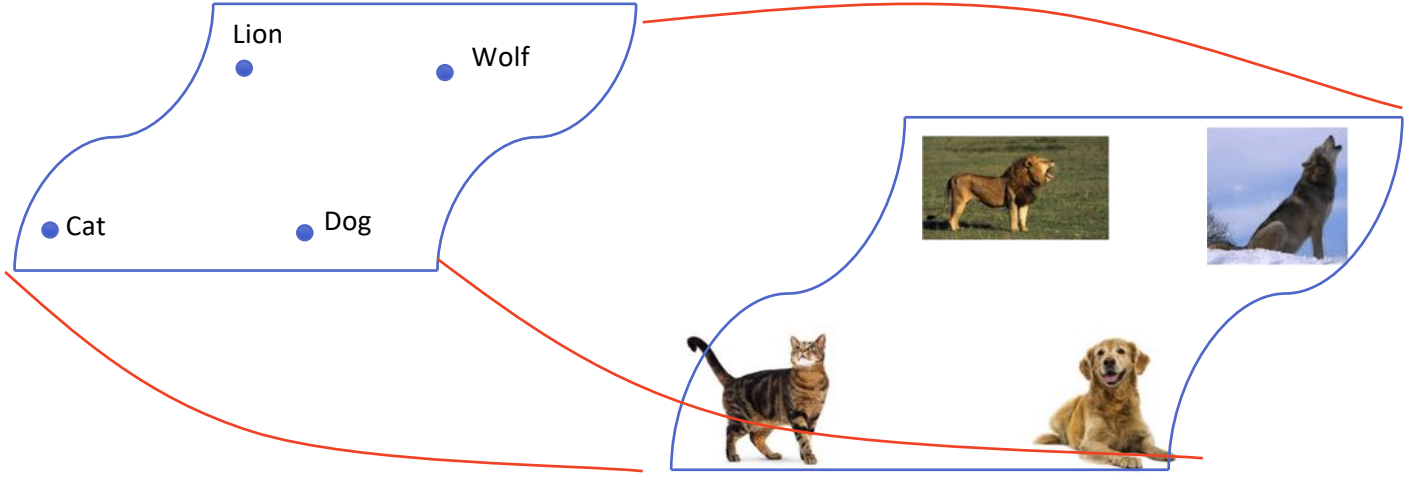
Regression/Classification Approach

- A simple category vector: class-level attribute description.
  - Wampimuk := small, cute, furry, horns.
- Train:
  - Given some known class-category vectors v and data x:
  - Learn image-attribute classifiers/regressors v=f(x).
    - E.g., SVM/SVR. Deep Neural Nets.
- Test:
  - Specify vec v* for new class
  - Map new data f(x*)
  - NN matching of v* vs f(x*)
- Pros:
  - Easy and fast!
- Cons
  - Category separability



Why Does Zero-Shot Recognition Work?

- Very little theory.
- Intuition:
  - IF training category vectors and image vectors lie in the same relative positions on their respective manifolds….
  - Then a few examples can establish correspondence between the two spaces
  - And any new category can also be recognized.



ZSL Summary

- Family of ways to recognize (or regress, etc) with no training examples.
- Pros:
  - Avoid data bottleneck.
  - ZSL performance can be similar to K (1-5) shot performance
- Cons:
  - Need category embeddings
  - Performance is worse than many-shot learning.

From Supervised to Unsupervised Cross-Domain Matching

A Dictionary Induction Example

[Mukherjee & Hospedales, Learning Unsupervised Word Translations Without Adversaries, EMNLP 2018]

Intro: Bilingual Dictionary Induction

Goal: Inferring a bilingual lexicon from data

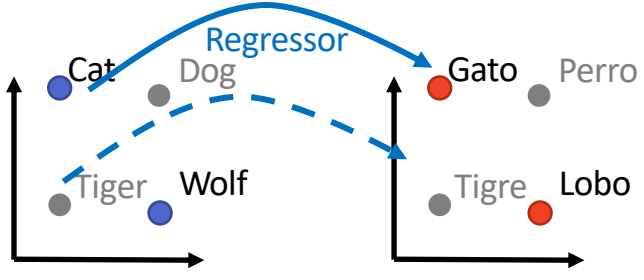
- Bilingual Lexicons:
  - Important for multi- and cross-lingual tasks: Multilingual word embedding, cross-lingual transfer learning, etc.
  - Not all pairs of languages have them.
  - Can we infer them from data?

English	Spanish
Horse	Caballo
Pig	Cerdo
Cat	Gato

Intro: Bilingual Dictionary Induction

Supervised Lexicon Induction

- Given some matching word pairs, deduce the full bilingual lexicon.
- Famously studied by Mikolov et al, 2013.
- Approach:
  - Monolingual word embeddings
  - Train a regressor on known pairs
  - Apply regressor to match unknown pairs

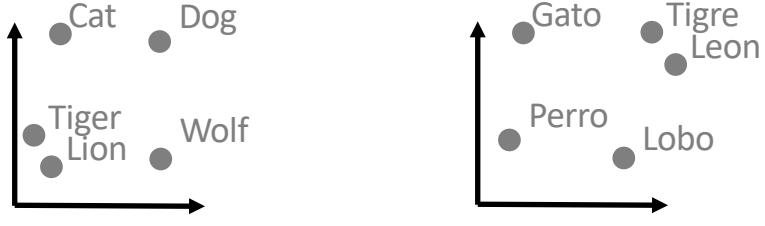


	English	Spanish
Train	Cat	Gato
	Wolf	Lobo
	...	
Test	Dog	?
	Tiger	?

Intro: Bilingual Dictionary Induction

Unsupervised Lexicon Induction

- Related to statistical decipherment (Knight, Coling'06)
- How to deduce the full bilingual lexicon starting with no pairs?



Deep Distribution Matching: Matching

Our Approach: Statistical dependency:

- EG: Hilbert Schmid Independency Criteria (HSIC). Squared loss mutual information (SMI). Search for the representation and the pairing, that maximises statistical dependency.

1. Update: Association to maximize statistical dependency (given representation)

Roughly: Search for the sorting that makes the two within domain kernel matrices “look” similar.

$$\Pi$$

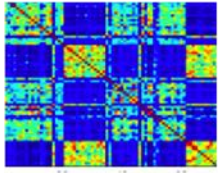
English, p(*x*)

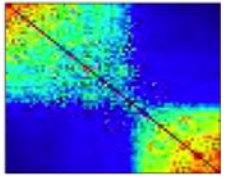
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

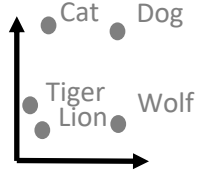
Spanish, p(*y*)

$$\mathrm{tr}(\boldsymbol{K}\boldsymbol{L}) \Rightarrow \mathrm{tr}(\boldsymbol{K}\boldsymbol{\Pi}^{\top}\boldsymbol{L}\boldsymbol{\Pi})$$

$\boldsymbol{K}$ $\boldsymbol{L}$







Method: Deep Distribution Matching

Our Approach: Statistical dependency:

- EG: Hilbert Schmid Independency Criteria (HSIC). Squared loss mutual information (SMI). Search for the representation and the pairing, that maximises statistical dependency.

1. Update: Association
➡️ Linear Assignment Problem

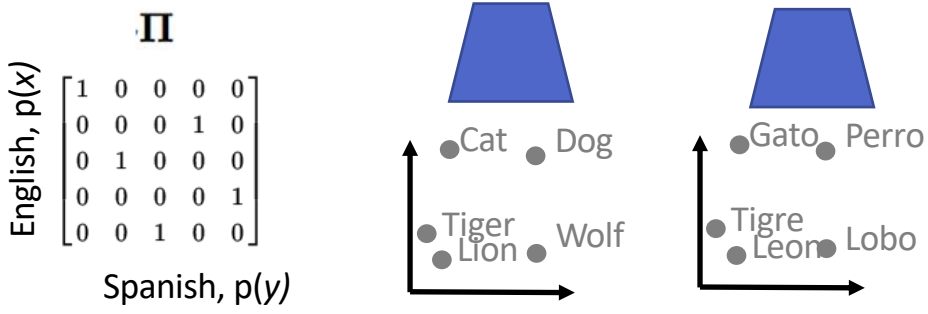
2. Update: Representation
➡️ Normal back-prop.

$$\min_{\boldsymbol{\Theta}_x, \boldsymbol{\Theta}_y, \boldsymbol{\Pi}} \underbrace{\Omega(\mathcal{D}; \boldsymbol{\Theta}_x, \boldsymbol{\Theta}_y)}_{\textit{Regularizer}} - \underbrace{\lambda D_{\boldsymbol{\Pi}}(\mathcal{D}; \boldsymbol{\Theta}_x, \boldsymbol{\Theta}_y)}_{\textit{Dependency}},$$

$$D_{\boldsymbol{\Pi}}(\mathcal{D}; \boldsymbol{\Theta}_x, \boldsymbol{\Theta}_y) = D_{\boldsymbol{\Pi}}(\{\boldsymbol{g}_x(\boldsymbol{x}_i), \boldsymbol{g}_y(\boldsymbol{y}_{\pi(i)})\}_{i=1}^n),$$

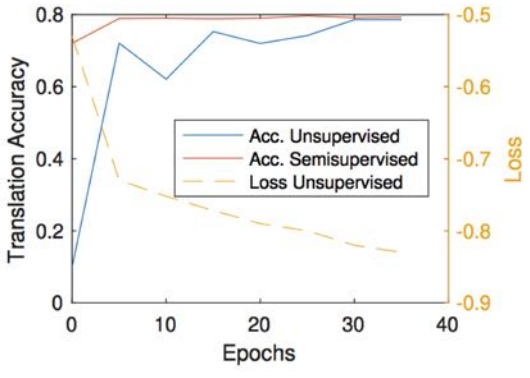
$$\Omega(\mathcal{D}; \boldsymbol{\Theta}_x, \boldsymbol{\Theta}_y) = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{f}_x(\boldsymbol{g}_x(\boldsymbol{x}_i))\|_2^2$$

- Good:
  - Every update decreases objective
  - No adversarial min-max.



Results

- Results Summary:
  - Promising results compared to recent adversarial methods.
  - Both much better than prior non-adversarial methods.
  - Simple convergence compared to adversarial methods.



	Methods	MUSE Dataset es-en	en-es	it-en	en-it	zh-en	en-zh
Previous Heuristics	TM (Mikolov et al., 2013)	5.6	4.8	5.2	4.8	2.6	1.8
	CCA (Faruqui and Dyer, 2014)	6.1	5.6	5.8	5.2	3.1	2.3
	MCCA (Haghighi et al., 2008)	5.7	5.1	5.4	4.8	3.0	2.2
	KS (Quadrianto et al., 2009)	8.3	7.4	6.3	5.7	4.8	3.2
	Self-Training (Artetxe et al., 2017)	12.4	12.2	10.7	10.2	5.8	5.6
Adversarial	EMDOT (Zhang et al., 2017b)	72.4	71.8	72.8	72.6	32.8	31.7
	W-GAN (Zhang et al., 2017b)	78.2	77.4	75.3	74.8	38.6	37.5
	GAN-NN (Conneau et al., 2018)	69.8	71.3	72.1	71.5	41.3	40.2
Ours	Deep-SMI (Ours)	75.9	80.6	75.7	75.2	38.5	38.1
	Deep-SMI-CSLS	79.2	84.5	78.8	78.5	43.7	42.8

Outline

- Part I: Sparse Data Deep Learning
  - Intro & common techniques.
  - Metric learning
  - Tensor-based
- Part II: Learning with multiple domains
  - Regression and Energy Functions for matching
  - Vision and Language Examples
- Links to Practical Examples & Exercises

Practical Exercises

- Examples and exercises on metric-based few-shot recognition.
  - Python/pytorch on google colab platform.
- Link to walkthrough (google docs):
  - http://tiny.cc/ye2s8y

References

- [Sun et al, Revisiting Unreasonable Effectiveness of Data in Deep Learning Era, ICCV'17]
- [Improving SVM Accuracy by Training on Auxiliary Data Sources, ICML'04]
- [Cross-domain video concept detection using adaptive svms, ACM MM'07]
- [How transferable are features in deep neural networks?, NIPS'14]
- [Taskonomy: Disentangling Task Transfer Learning, CVPR'18]
- [HOUDINI: Lifelong Learning as Program Synthesis, NIPS'18]
- [Evgeniou, Learning Multiple Tasks with Kernel Methods, JMLR'05]
- [Kang, Learning with Whom to Share in Multi-task Feature Learning, ICML'15]
- [Yang & Hospedales, ICLR'15, A unified perspective on multi-task and multi-domain learning]
- [Yang & Hospedales, ICLR'17, Deep Multi-task Representation Learning: A Tensor Factorisation Approach]
- [Rebuffi, NIPS'17, Learning multiple visual domains with residual adapters]
- [Bulat, arXiv'19, Incremental multi-domain learning with network latent tensor factorization]
- [Balažević , arXiv'19, TuckER: Tensor Factorization for Knowledge Graph Completion]
- [Lebedev, ICLR'15, Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition]
- [Kossaifi, CVPR'19, T-Net: Parametrizing Fully Convolutional Nets with a Single High-Order Tensor]
- [Hu et al, Neural Tensor Fusion Networks, ICCV'17]
- [EG: Hirzer, ECCV'12; Mignon, CVPR'12]
- [Learning Fine-grained Image Similarity with Deep Ranking, CVPR'14]
- [Wang, PAMI'18, A Survey on Learning to Hash]
- [Snell et al, NIPS'17, Prototypical Networks for Few Shot Learning]
- [Fort, arXiv'17, Gaussian Prototypical Networks for Few-Shot Learning]
- [Sung, CVPR'18, "Learning to Compare: Relation Networks for Few Shot Learning"]
- [Zhang, arXiv'18, "Relation Columns for Few Shot Learning"]
- [Alemi, ICLR'17, Deep Variational Information Bottleneck]
- [Finn, ICML'17, Model Agnostic Meta Learning]
- [Nichol, arXiv'18, On First-Order Meta-Learning Algorithms]
- [Wang, PAMI'18, A Survey on Learning to Hash]
- [Mukherjee, EMNLP'16, Gaussian Visual-Linguistic Embedding for Zero-Shot Recognition]