CS 2770: Computer Vision

Self-Supervised and Embodied Learning

Prof. Adriana Kovashka
University of Pittsburgh
April 13, 2021

Motivation

- What's the data we've learned from thus far?
- Labeled static datasets
  - Expensive to obtain
  - Doesn't match how humans learn
- Alternatives
  - Unsupervised learning (no labels)
  - Self-supervised learning ("fake"/emergent labels)
  - Embodied/active learning (agents in environments)

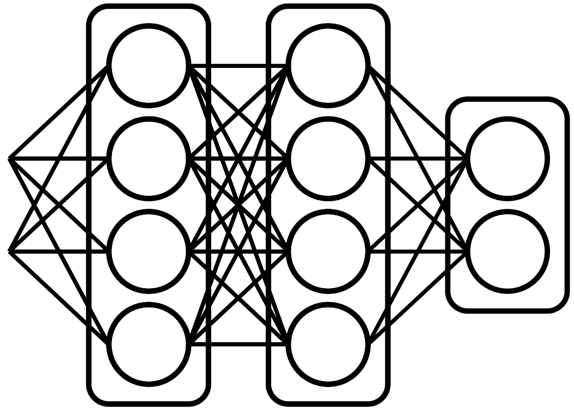
Self-supervised learning

Unsupervised Visual Representation Learning by Context Prediction

Carl Doersch, Alexei Efros and Abhinav Gupta

ICCV 2015

ImageNet + Deep Learning

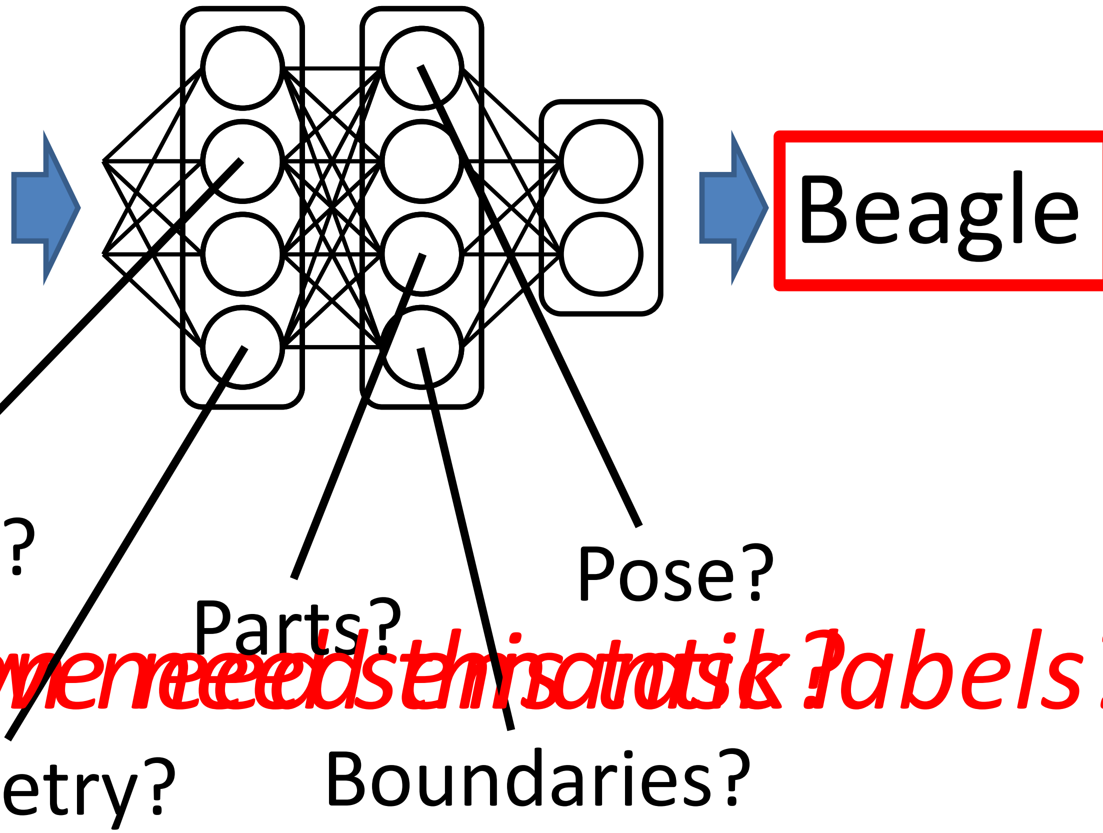


Beagle

- Image Retrieval
- Detection (RCNN)
- Segmentation (FCN)
- Depth Estimation
- ...

Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

ImageNet + Deep Learning



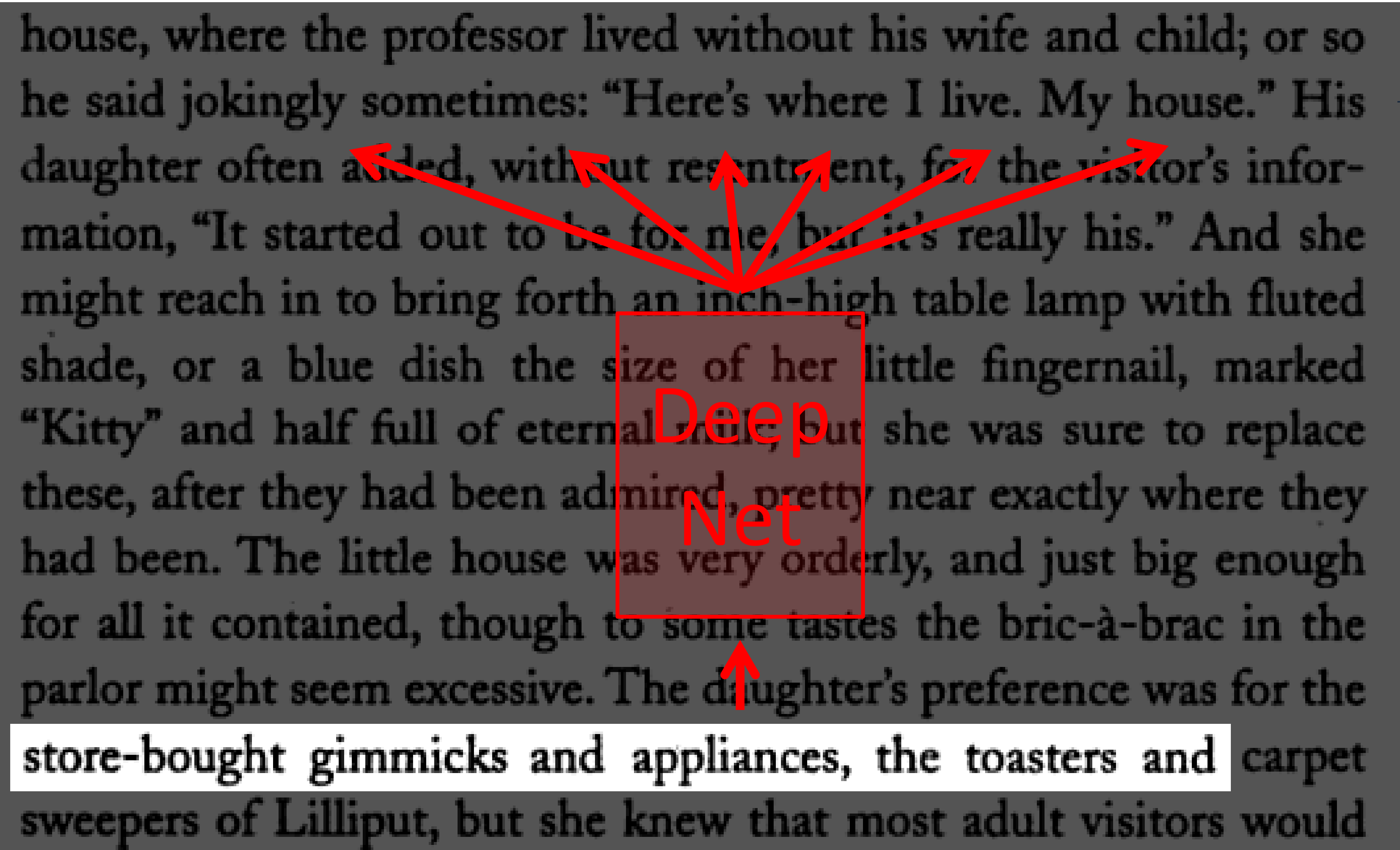
Do we need semantic labels?

Do we even need this task?

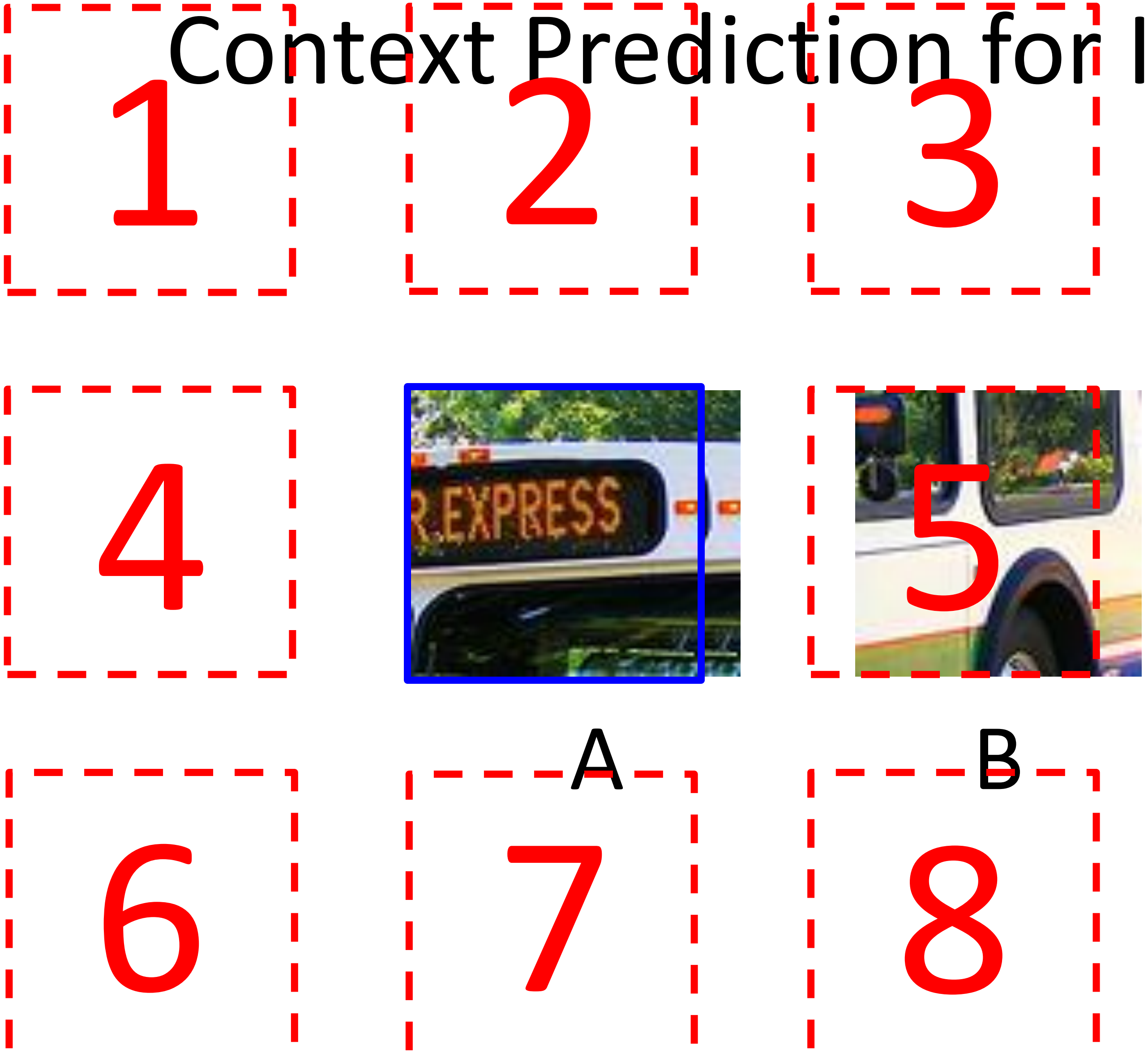
Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015

Context as Supervision

[Collobert & Weston 2008; Mikolov et al. 2013]

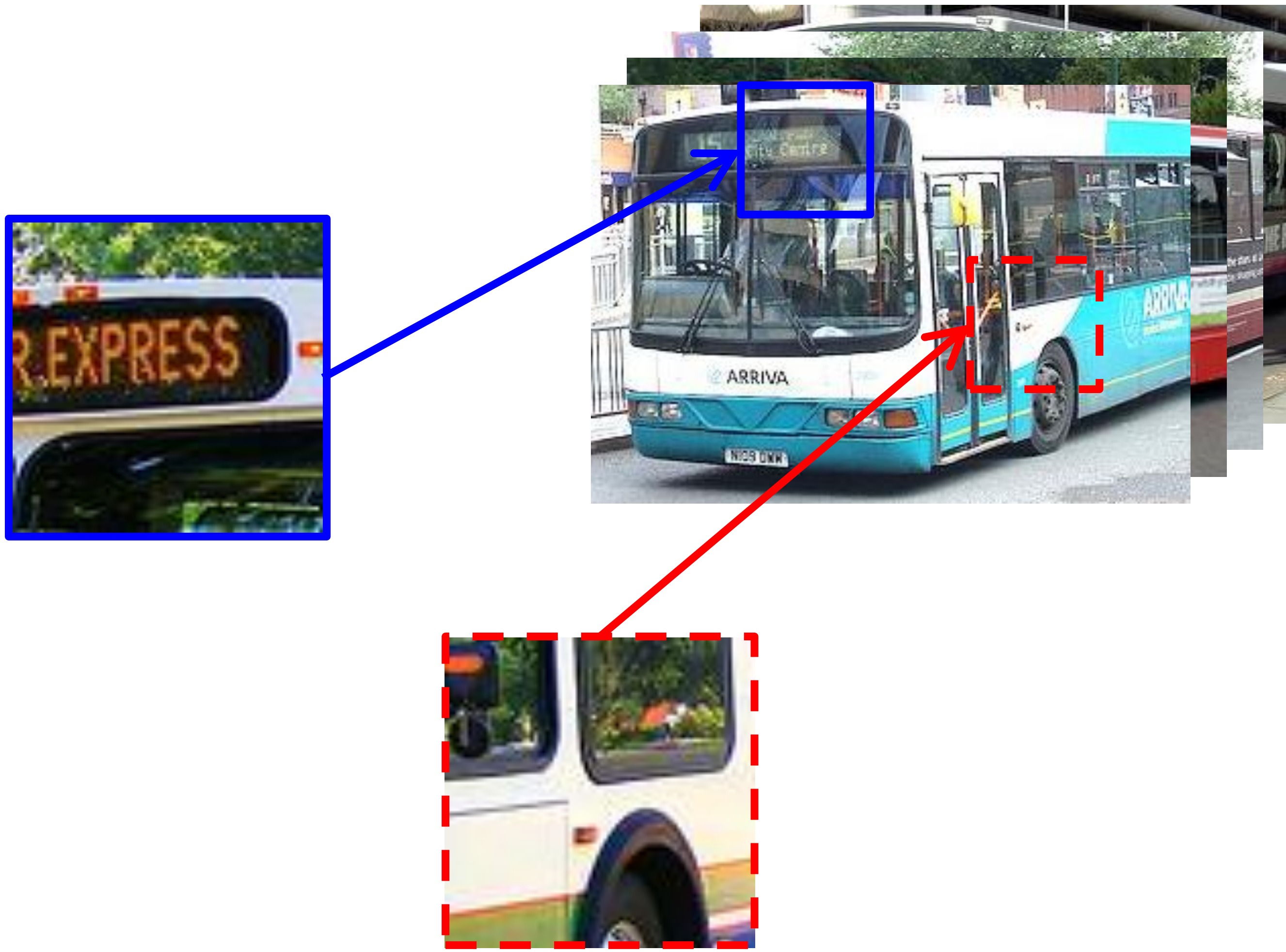


Context Prediction for Images



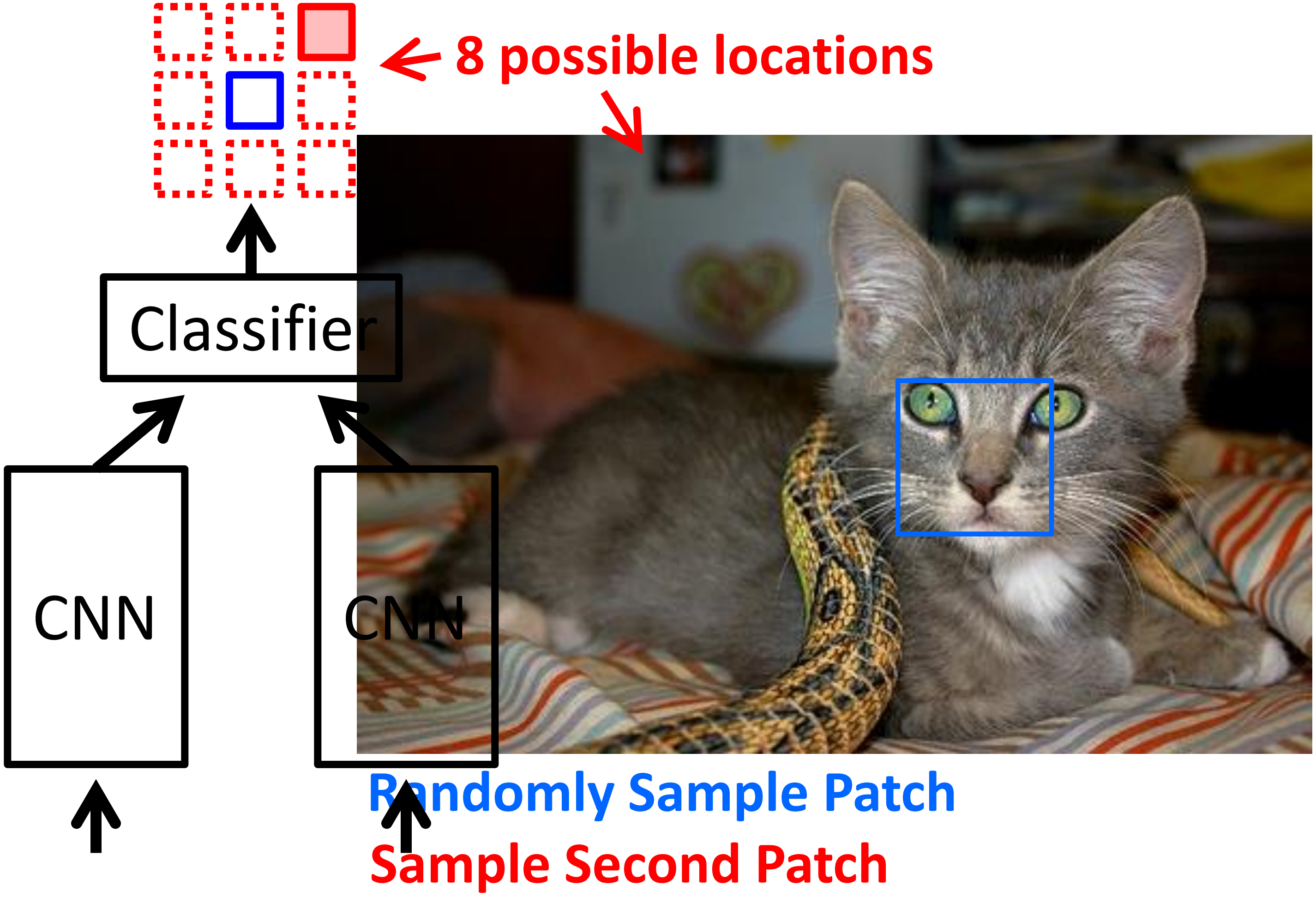
Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Semantics from a non-semantic task

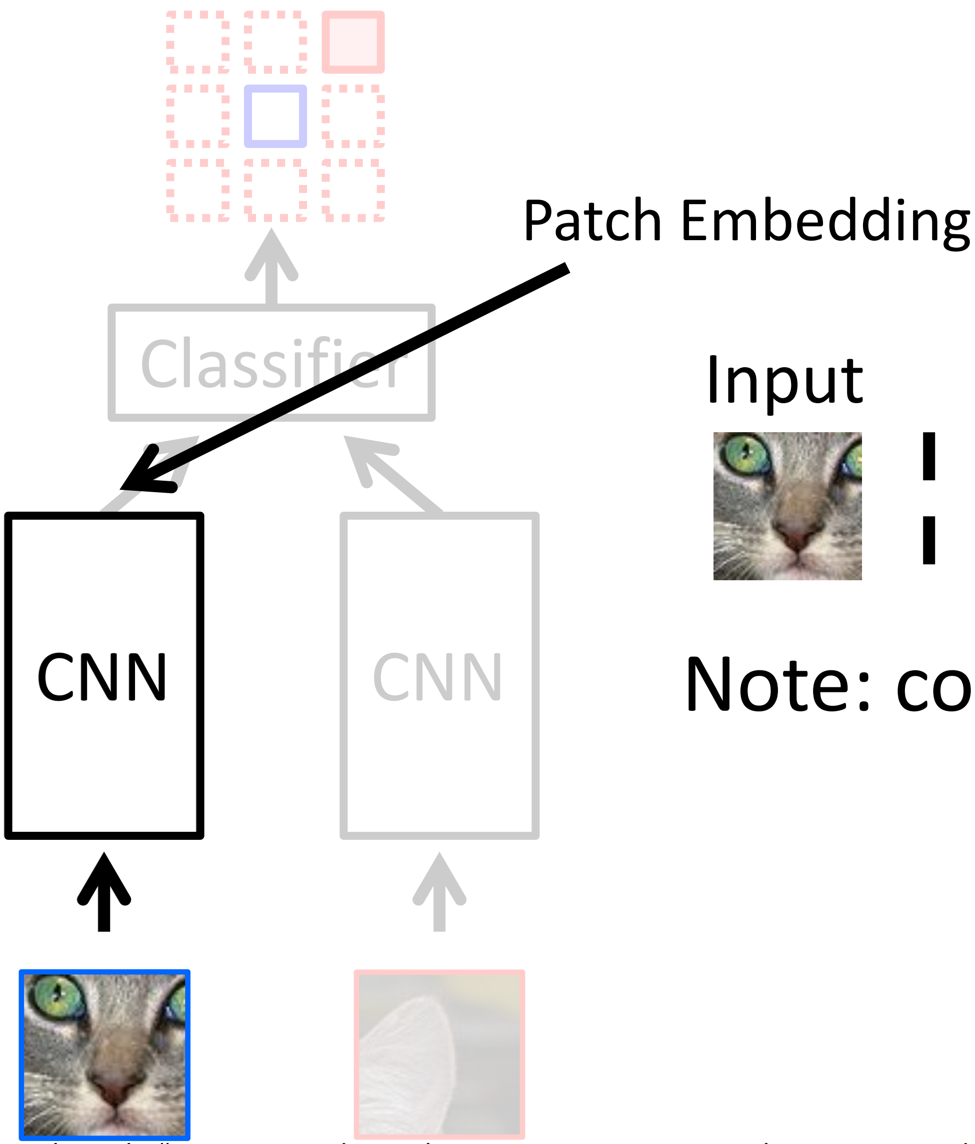


Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Relative Position Task



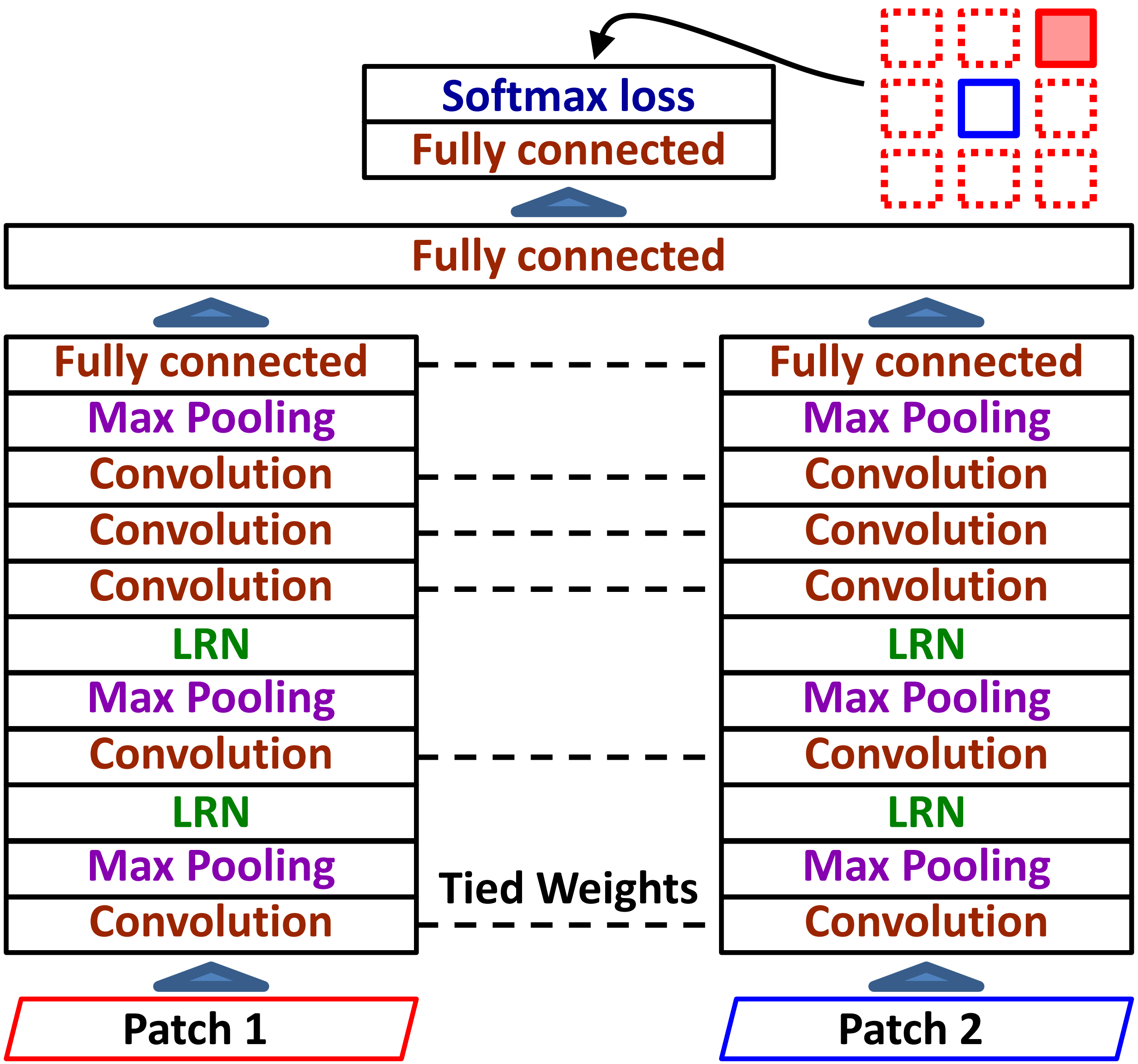
Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015



Note: connects ***across*** instances!

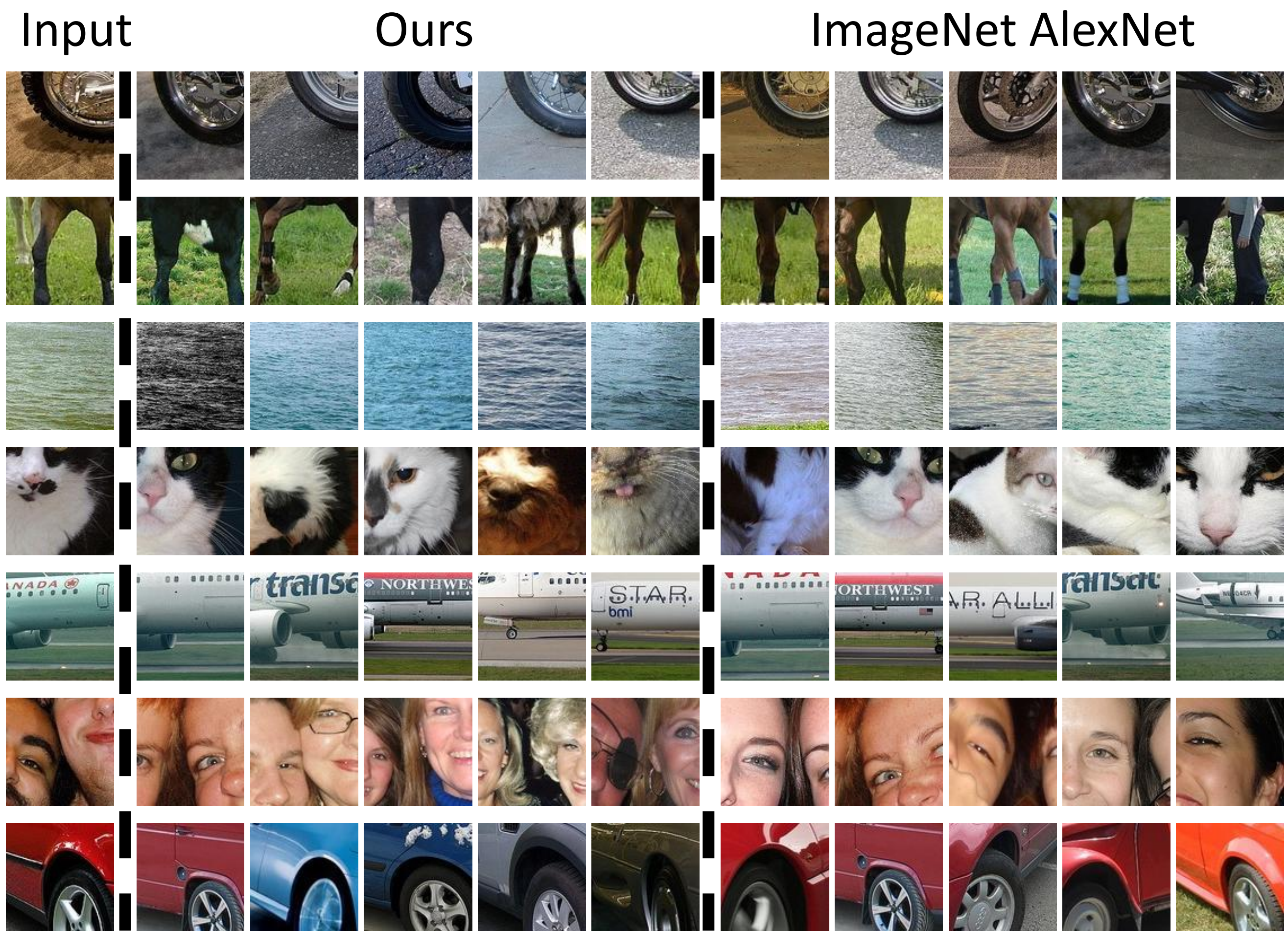
Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Architecture



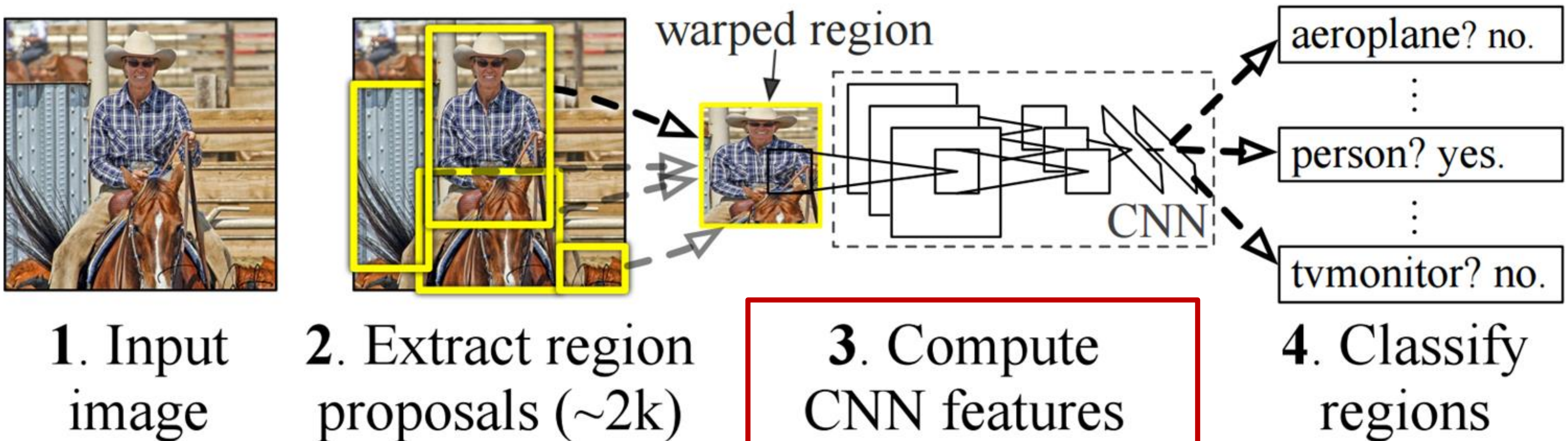
Doersch et al., "Unsupervised Visual Representation Learning by Context Prediction", ICCV 2015

What is learned?



Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Pre-Training for R-CNN



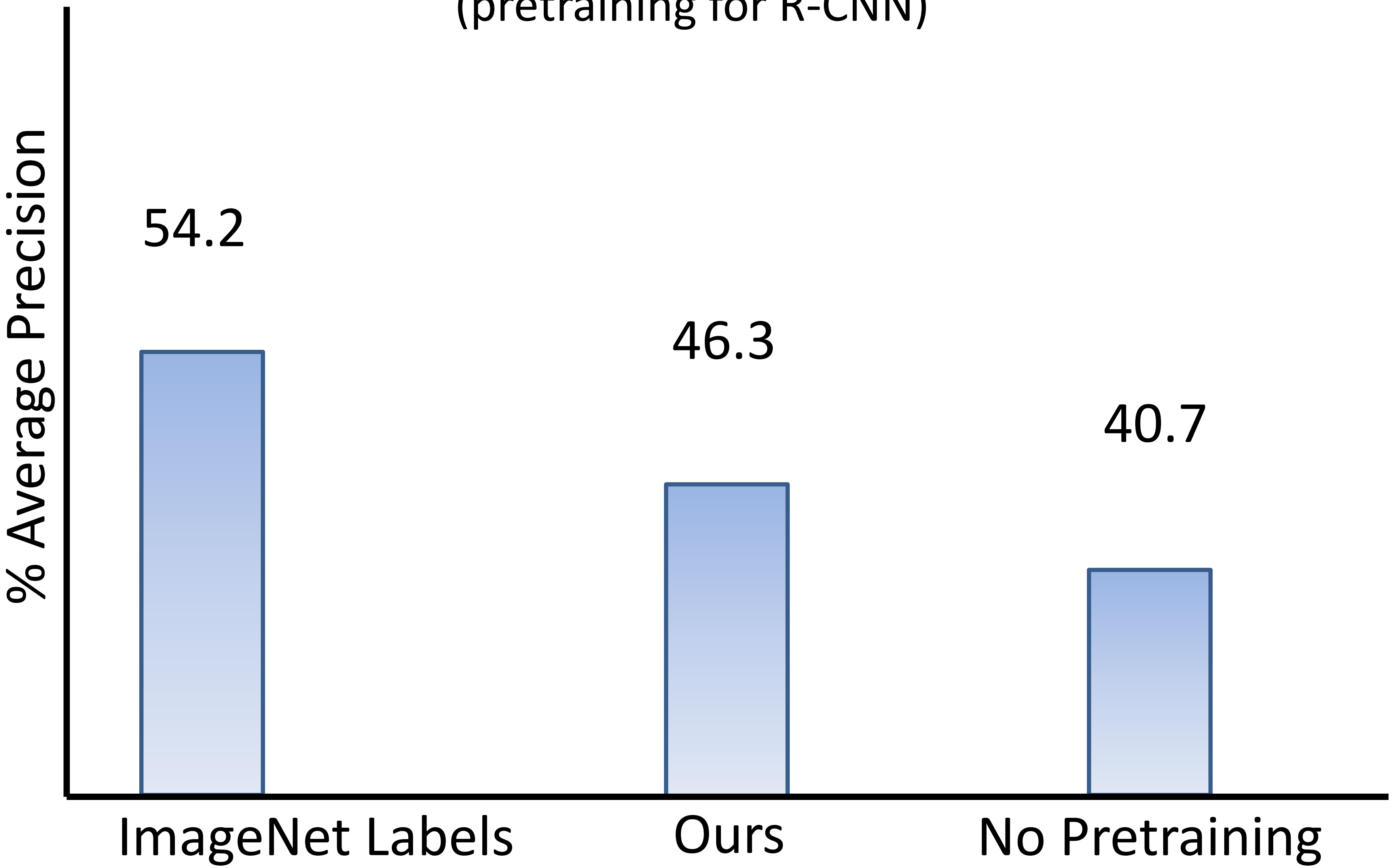
Pre-train on relative-position task, w/o labels

Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

[Girshick et al. 2014]

VOC 2007 Performance

(pretraining for R-CNN)



Doersch et al., “Unsupervised Visual Representation Learning by Context Prediction”, ICCV 2015

Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

Ishan Misra, C. Lawrence Zitnick, and Martial Hebert

ECCV 2016

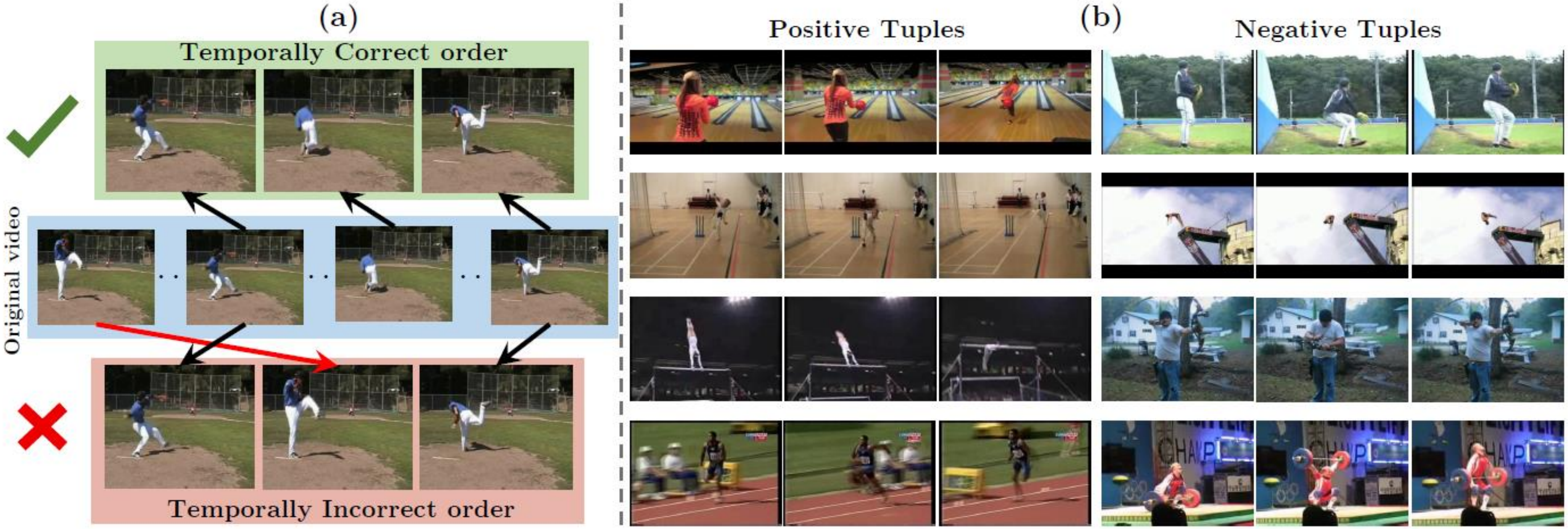


Fig. 1: **(a)** A video imposes a natural temporal structure for visual data. In many cases, one can easily verify whether frames are in the correct temporal order (shuffled or not). Such a simple sequential verification task captures important spatiotemporal signals in videos. We use this task for unsupervised pre-training of a Convolutional Neural Network (CNN). **(b)** Some examples of the automatically extracted positive and negative tuples used to formulate a classification task for a CNN.

Misra et al., "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification", ECCV 2016

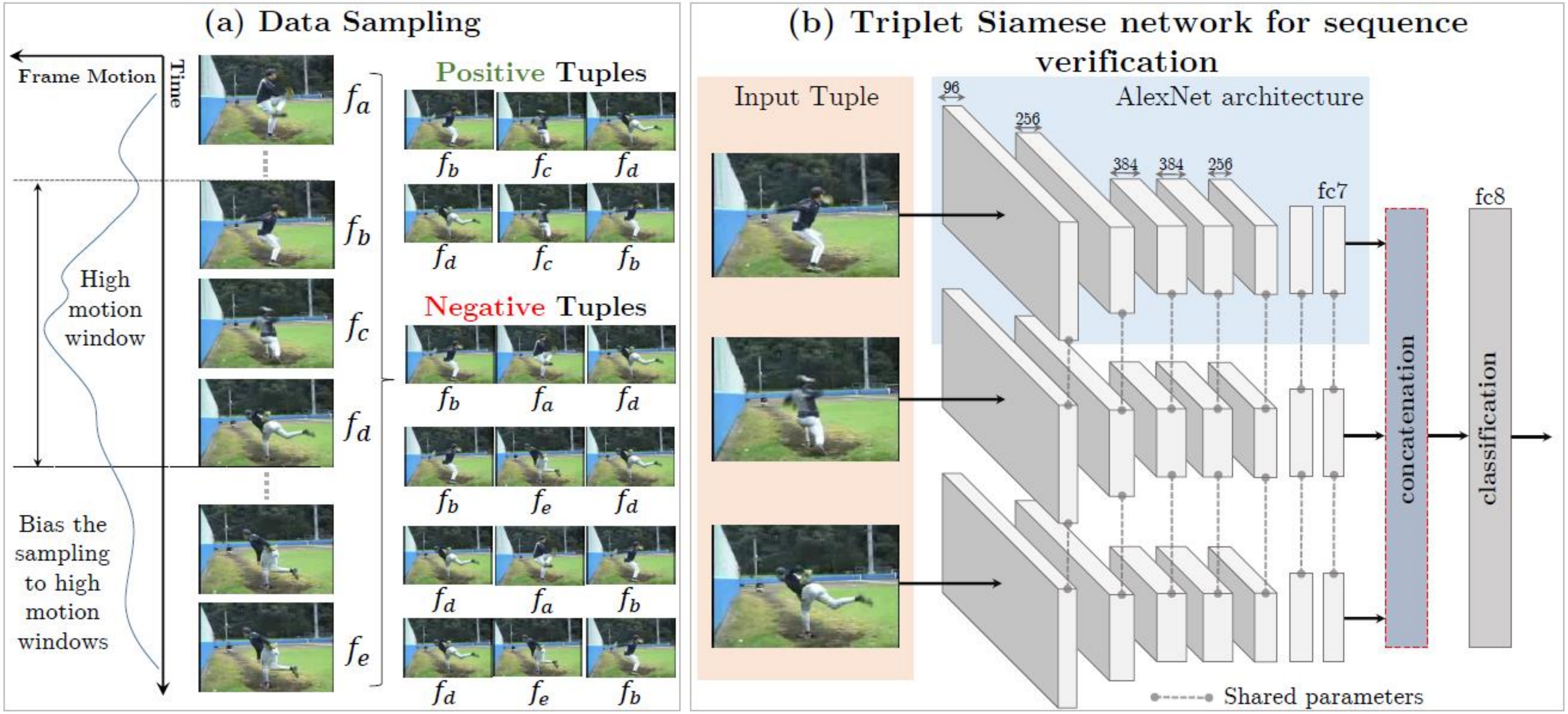


Fig. 2: **(a)** We sample tuples of frames from high motion windows in a video. We form positive and negative tuples based on whether the three input frames are in the correct temporal order. **(b)** Our triplet Siamese network architecture has three parallel network stacks with shared weights upto the `fc7` layer. Each stack takes a frame as input, and produces a representation at the `fc7` layer. The concatenated `fc7` representations are used to predict whether the input tuple is in the correct temporal order.

Misra et al., "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification", ECCV 2016

Table 2: Mean classification accuracies over the 3 splits of UCF101 and HMDB51 datasets. We compare different initializations and finetune them for action recognition.

Dataset	Initialization	Mean Accuracy
UCF101	Random	38.6
	(Ours) Tuple verification	**50.2**
HMDB51	Random	13.3
	UCF Supervised	15.2
	(Ours) Tuple verification	**18.1**

Misra et al., "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification", ECCV 2016

Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, Ross Girshick

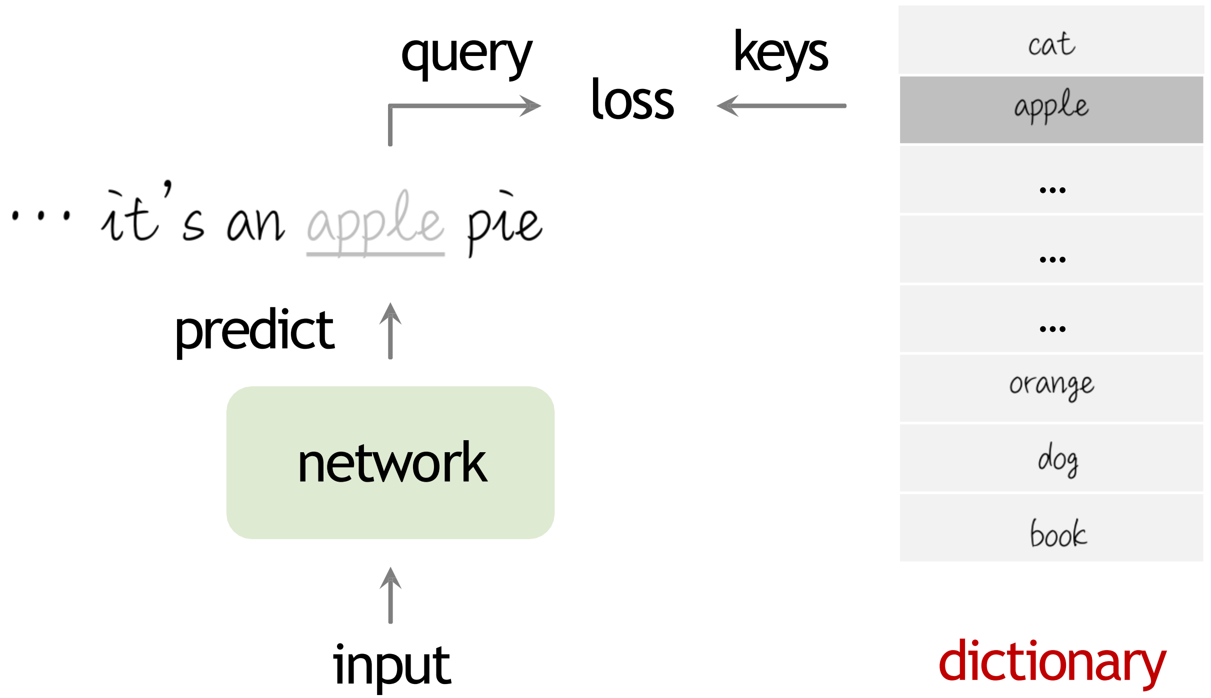
CVPR 2020

Highlights

- Unsupervised pre-training: surpass supervised counterparts
- … in 7 vision tasks on detection, segmentation
- … by big margins in some tasks
- … scaled out to 1 billion images

He et al., “Momentum Contrast for Unsupervised Visual Representation Learning”, CVPR 2020

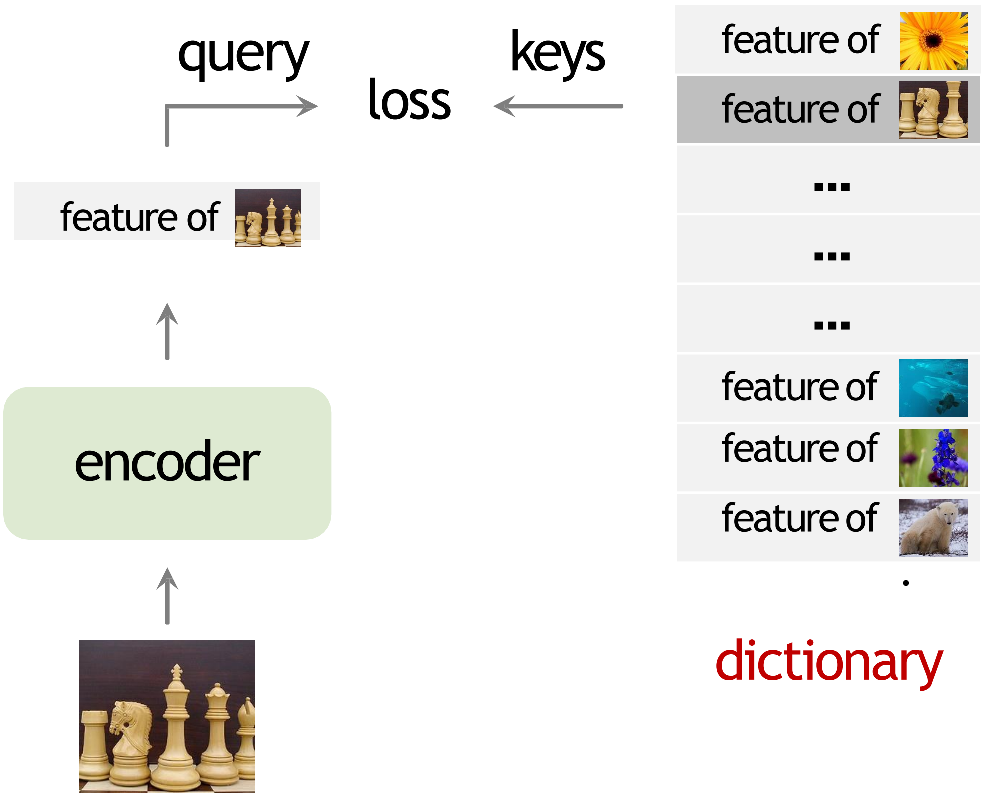
Unsupervised learning in NLP: BERT



Devlin *et al*. NAACL2019

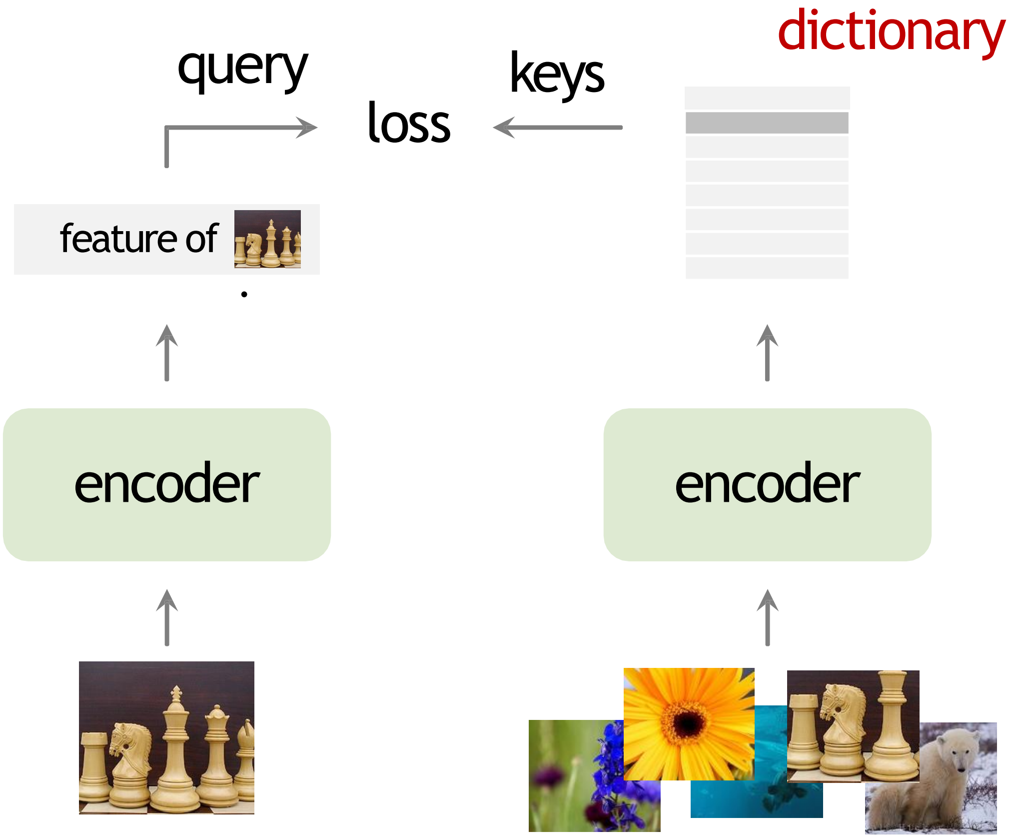
He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Analogy in Computer Vision



He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Contrastive Learning



Hadsell *et al*. CVPR 2006,
Wu *et al*. CVPR 2018, ...

He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Our Method: Momentum Contrast (MoCo)

- Contrastive learning as dictionary look-up
- **Large** dictionary
- **Consistent** dictionary

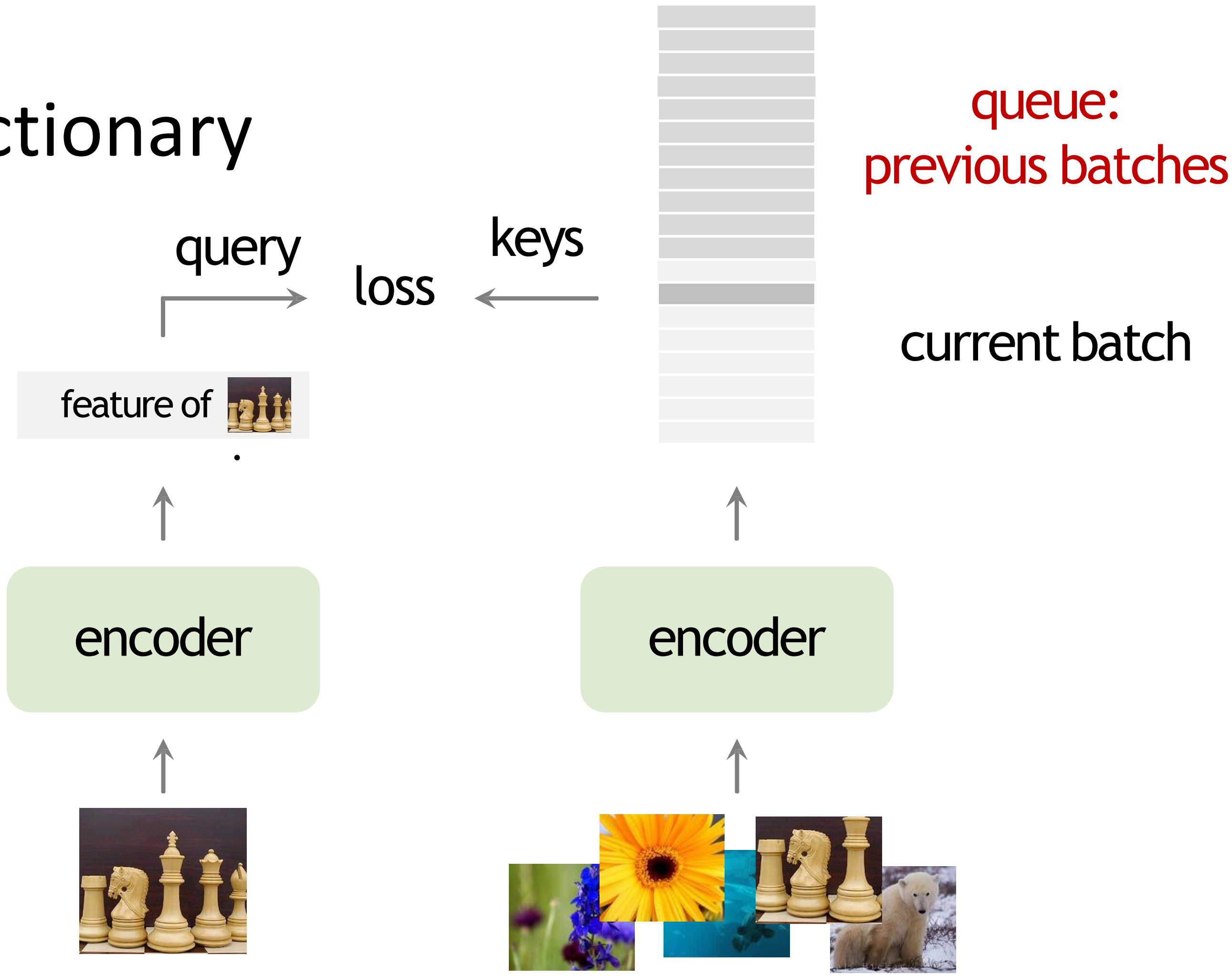
Consider an encoded query q and a set of encoded samples $\{k_0, k_1, k_2, ...\}$ that are the keys of a dictionary. Assume that there is a single key (denoted as k_+) in the dictionary that q matches. A contrastive loss [29] is a function whose value is low when q is similar to its positive key k_+ and dissimilar to all other keys (considered negative keys for q). With similarity measured by dot product, a form of a contrastive loss function, called InfoNCE [46], is considered in this paper:

$$\mathcal{L}_q = -\log \frac{\exp(q\cdot k_+/\tau)}{\sum_{i=0}^{K} \exp(q\cdot k_i/\tau)} \tag{1}$$

where τ is a temperature hyper-parameter per [61]. The sum is over one positive and K negative samples. Intuitively, this loss is the log loss of a $(K+1)$-way softmax-based classifier that tries to classify q as k_+. Contrastive loss functions can also be based on other forms [29, 59, 61, 36], such as margin-based losses and variants of NCE losses.

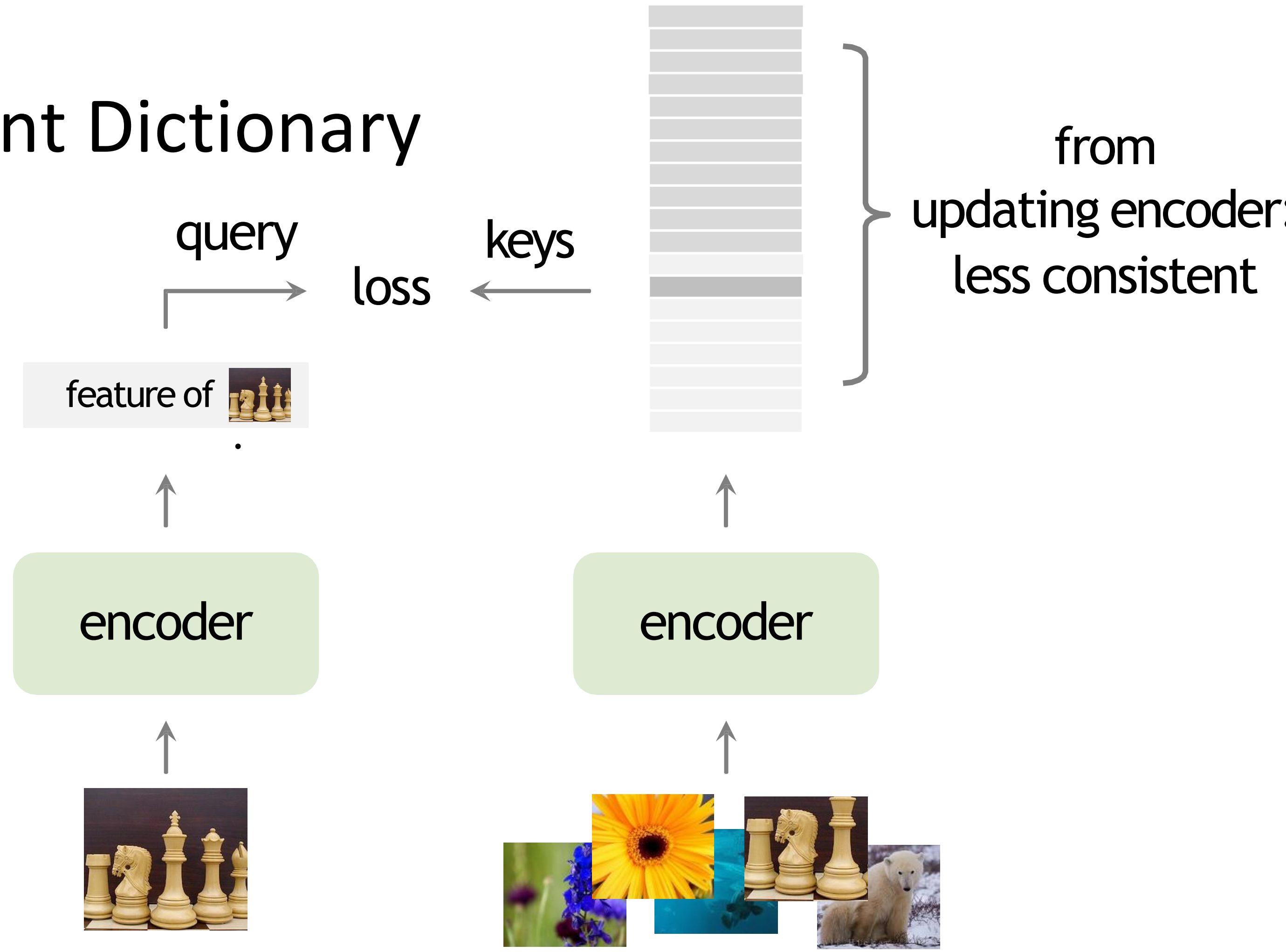
He et al., “Momentum Contrast for Unsupervised Visual Representation Learning”, CVPR 2020

Large Dictionary



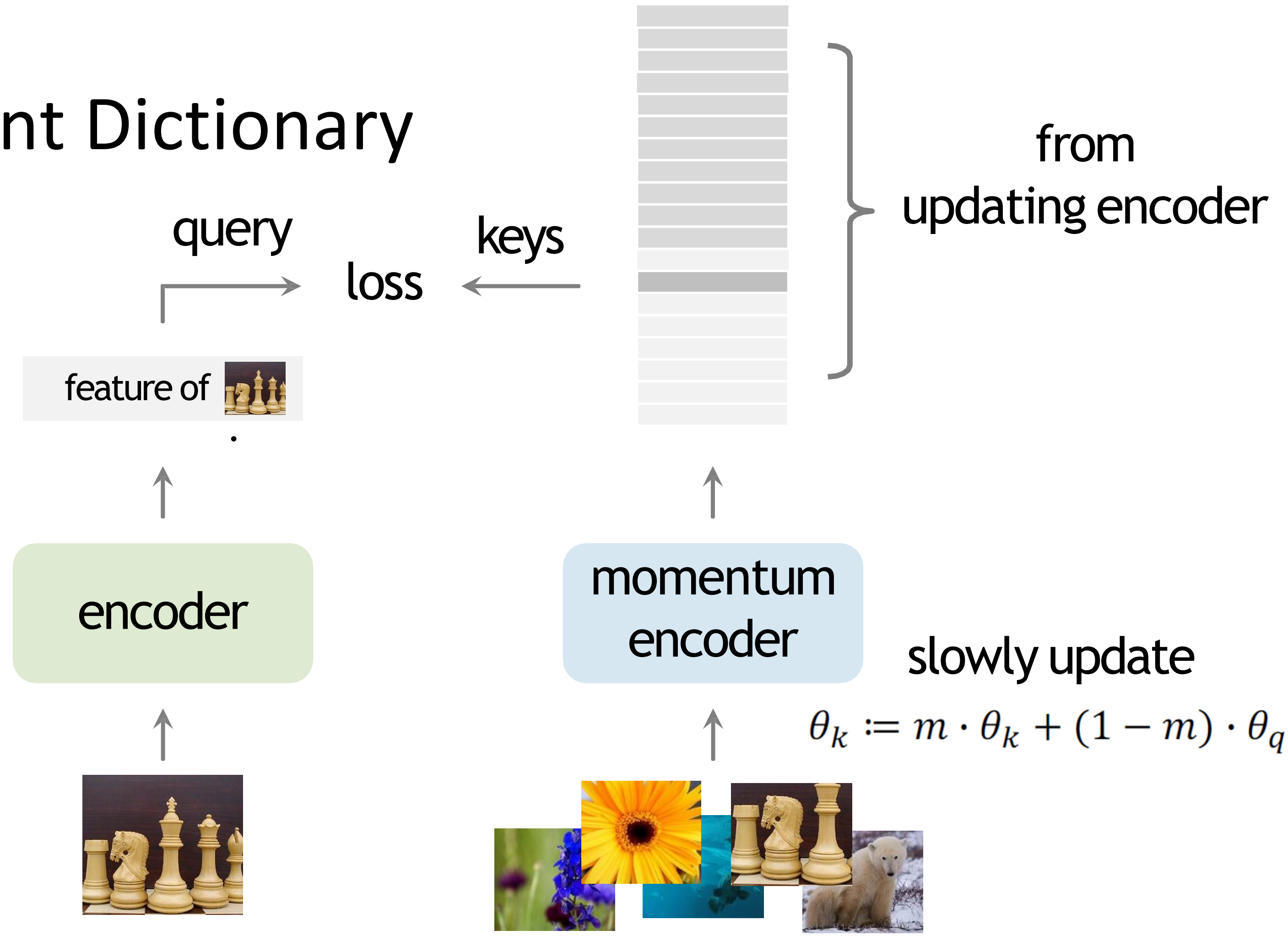
He et al., “Momentum Contrast for Unsupervised Visual Representation Learning”, CVPR 2020

Consistent Dictionary



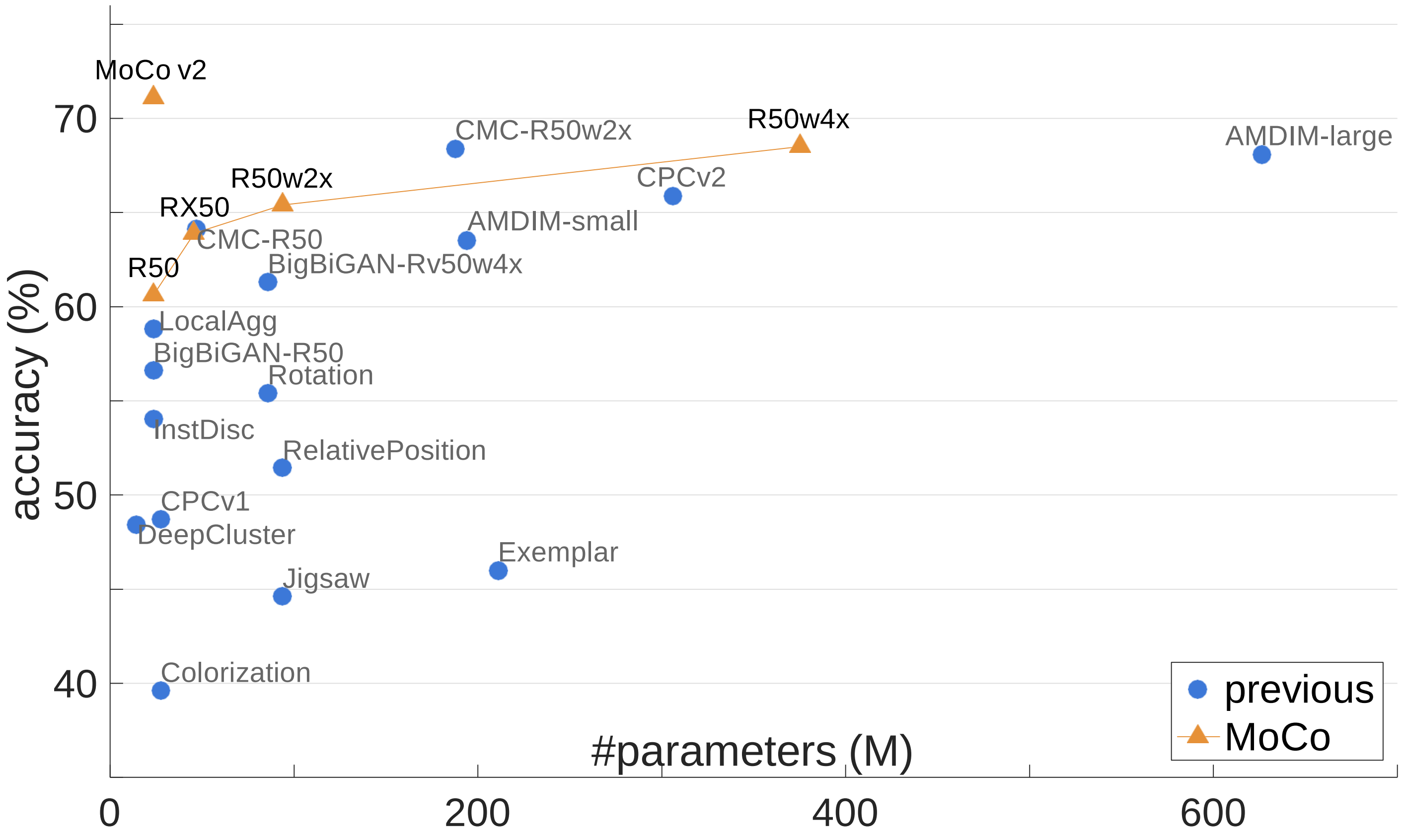
He et al., “Momentum Contrast for Unsupervised Visual Representation Learning”, CVPR 2020

Consistent Dictionary



He et al., “Momentum Contrast for Unsupervised Visual Representation Learning”, CVPR 2020

Results: ImageNet Linear Classifiers



He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Results: Transferring Features

VOC 2007 Detection, Faster R-CNN, ResNet-50

pre-train	AP_{50} RelPos, by [10]	Multi-task [10]	Jigsaw, by [22]	LocalAgg [60]
super. IN-1M	74.2	74.2	70.5	74.6
unsup. IN-1M	66.8 (−7.4)	70.5 (−3.7)	61.4 (−9.1)	69.1 (−5.5)

Previous:
behind supervised pre-training

He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Results: Transferring Features

VOC 2007 Detection, Faster R-CNN, ResNet-50

pre-train	AP$_{50}$ RelPos, by [10]	Multi-task [10]	Jigsaw, by [22]	LocalAgg [60]	**MoCo**
super. IN-1M	74.2	74.2	70.5	74.6	74.4
unsup. IN-1M	66.8 (−7.4)	70.5 (−3.7)	61.4 (−9.1)	69.1 (−5.5)	74.9 (+**0.5**)

MoCo:
surpass supervised pre-training

He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Results: Transferring Features

VOC 2007 Detection, Faster R-CNN, ResNet-50

pre-train	AP_{50} RelPos, by [10]	Multi-task [10]	Jigsaw, by [22]	LocalAgg [60]	**MoCo**
super. IN-1M	74.2	74.2	70.5	74.6	74.4
unsup. IN-1M	66.8 (−7.4)	70.5 (−3.7)	61.4 (−9.1)	69.1 (−5.5)	74.9 (+**0.5**)
unsup. IN-14M	-	-	69.2 (−1.3)	-	75.2 (+**0.8**)
unsup. IG-1B	-	-	-	-	75.6 (+**1.2**)

MoCo:
benefit from 1 billion images

He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Results: Transferring Features

VOC 2007 Detection, Faster R-CNN, ResNet-50

		AP_{50}	AP	AP_{75}
pre-train		**MoCo**	**MoCo**	**MoCo**
super.	IN-1M	74.4	42.4	42.7
unsup.	IN-1M	74.9 (+**0.5**)	46.6 (+**4.2**)	50.1 (+**7.4**)
unsup.	IN-14M	75.2 (+**0.8**)	46.9 (+**4.5**)	50.2 (+**7.5**)
unsup.	IG-1B	75.6 (+**1.2**)	47.6 (+**5.2**)	51.7 (+**9.0**)

MoCo:
big gains in stringent metrics
+9.0 AP_{75}

He et al., “Momentum Contrast for Unsupervised Visual Representation Learning”, CVPR 2020

Results: Transferring Features

pre-train	AP_{50}	AP	AP_{75}
random init.	64.4	37.9	38.6
super. IN-1M	81.4	54.0	59.1
MoCo IN-1M	81.1 (−0.3)	54.6 (+0.6)	59.9 (+0.8)
MoCo IG-1B	81.6 (+0.2)	55.5 (+1.5)	61.2 (+2.1)

(a) Faster R-CNN, R50-**dilated-C5**

pre-train	AP_{50}	AP	AP_{75}
random init.	60.2	33.8	33.1
super. IN-1M	81.3	53.5	58.8
MoCo IN-1M	81.5 (+0.2)	55.9 (+2.4)	62.6 (+3.8)
MoCo IG-1B	82.2 (+0.9)	57.2 (+3.7)	63.7 (+4.9)

(b) Faster R-CNN, R50-**C4**

AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
36.7	56.7	40.0	33.7	53.8	35.9
40.6	61.3	44.4	36.8	58.1	39.5
40.8 (+0.2)	61.6 (+0.3)	44.7 (+0.3)	36.9 (+0.1)	58.4 (+0.3)	39.7 (+0.2)
41.1 (+0.5)	61.8 (+0.5)	45.1 (+0.7)	37.4 (+0.6)	59.1 (+1.0)	40.2 (+0.7)

(b) Mask R-CNN, R50-**FPN**, **2**× schedule

AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
35.6	54.6	38.2	31.4	51.5	33.5
40.0	59.9	43.1	34.7	56.5	36.9
40.7 (+0.7)	60.5 (+0.6)	44.1 (+1.0)	35.4 (+0.7)	57.3 (+0.8)	37.6 (+0.7)
41.1 (+1.1)	60.7 (+0.8)	44.8 (+1.7)	35.6 (+0.9)	57.4 (+0.9)	38.1 (+1.2)

(d) Mask R-CNN, R50-**C4**, **2**× schedule

VOC 07+12
Detection
surpass, +4.9 AP_{75}

COCO Detection
COCO Instance seg.
surpass

He et al., “Momentum Contrast for Unsupervised Visual Representation Learning”, CVPR 2020

Results: Transferring Features

pre-train	COCO keypoint detection AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
random init.	65.9	86.5	71.7
super. IN-1M	65.8	86.9	71.9
MoCo IN-1M	66.8 (+**1.0**)	87.4 (+**0.5**)	72.5 (+**0.6**)
MoCo IG-1B	66.9 (+**1.1**)	87.8 (+**0.9**)	73.0 (+**1.1**)

COCO Keypoint
surpass

pre-train	COCO dense pose estimation AP^{dp}	AP^{dp}_{50}	AP^{dp}_{75}
random init.	39.4	78.5	35.1
super. IN-1M	48.3	85.6	50.6
MoCo IN-1M	50.1 (+**1.8**)	86.8 (+**1.2**)	53.9 (+**3.3**)
MoCo IG-1B	50.6 (+**2.3**)	87.0 (+**1.4**)	54.3 (+**3.7**)

COCO Dense pose
surpass, +3.7 AP_{75}

pre-train	LVIS instance segmentation AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	22.5	34.8	23.8
super. IN-1M†	24.4	37.8	25.8
MoCo IN-1M	24.1 (−0.3)	37.4 (−0.4)	25.5 (−0.3)
MoCo IG-1B	24.9 (+**0.5**)	38.2 (+0.4)	26.4 (+**0.6**)

LVIS
Instance seg.
surpass

pre-train	Cityscapes instance seg. AP^{mk}	AP^{mk}_{50}	Semantic Cityscapes
random init.	25.4	51.1	65.3
super. IN-1M	32.9	59.6	74.6
MoCo IN-1M	32.3 (−0.6)	59.3 (−0.3)	75.3 (+**0.7**)
MoCo IG-1B	32.9 (0.0)	60.3 (+**0.7**)	75.5 (+**0.9**)

Cityscapes
Semantic seg.
surpass

pre-train	Semantic VOC
random init.	39.5
super. IN-1M	74.4
MoCo IN-1M	72.5 (−1.9)
MoCo IG-1B	73.6 (−0.8)

VOC
Semantic seg.
-0.8 point

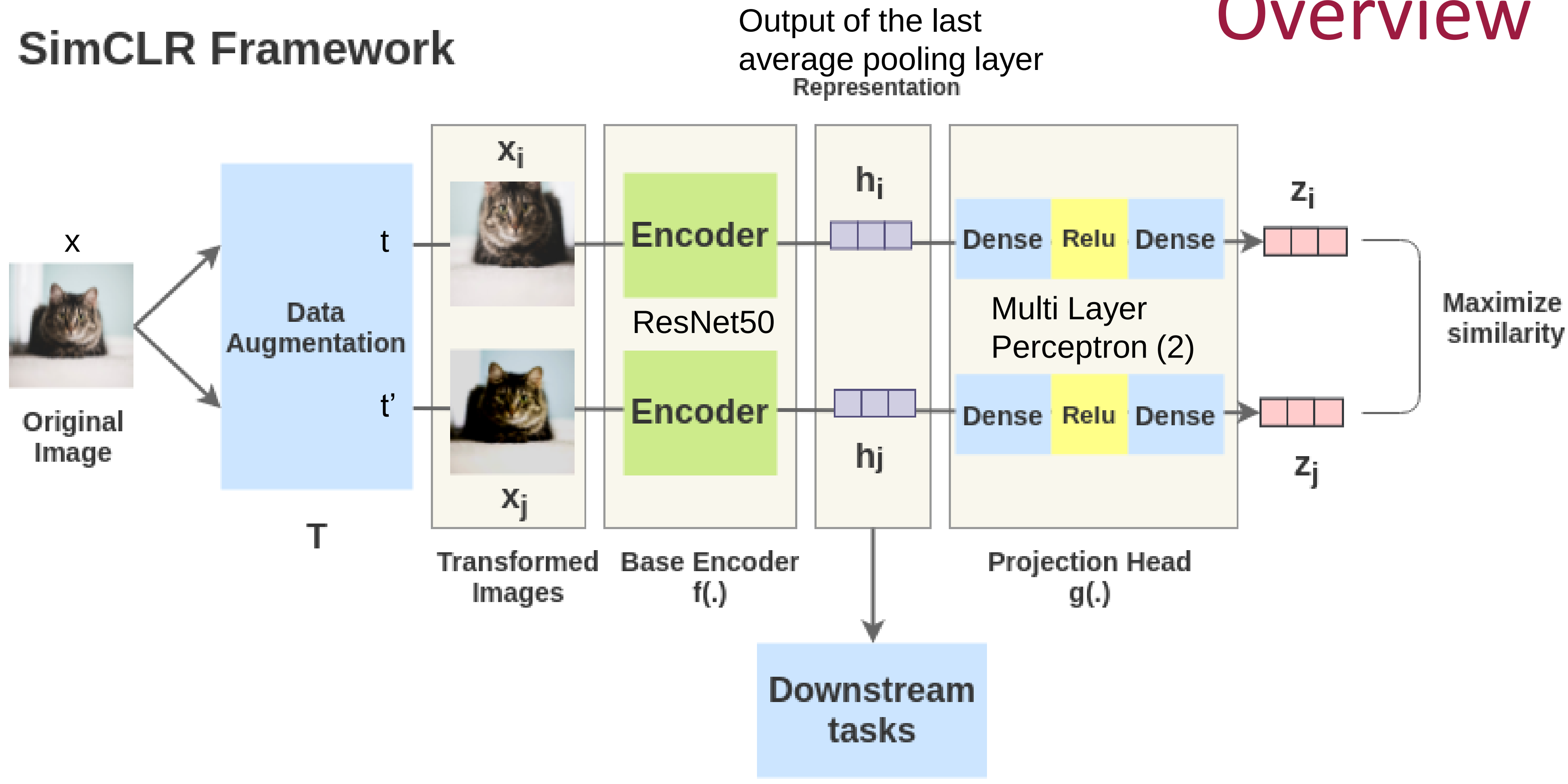
He et al., "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton

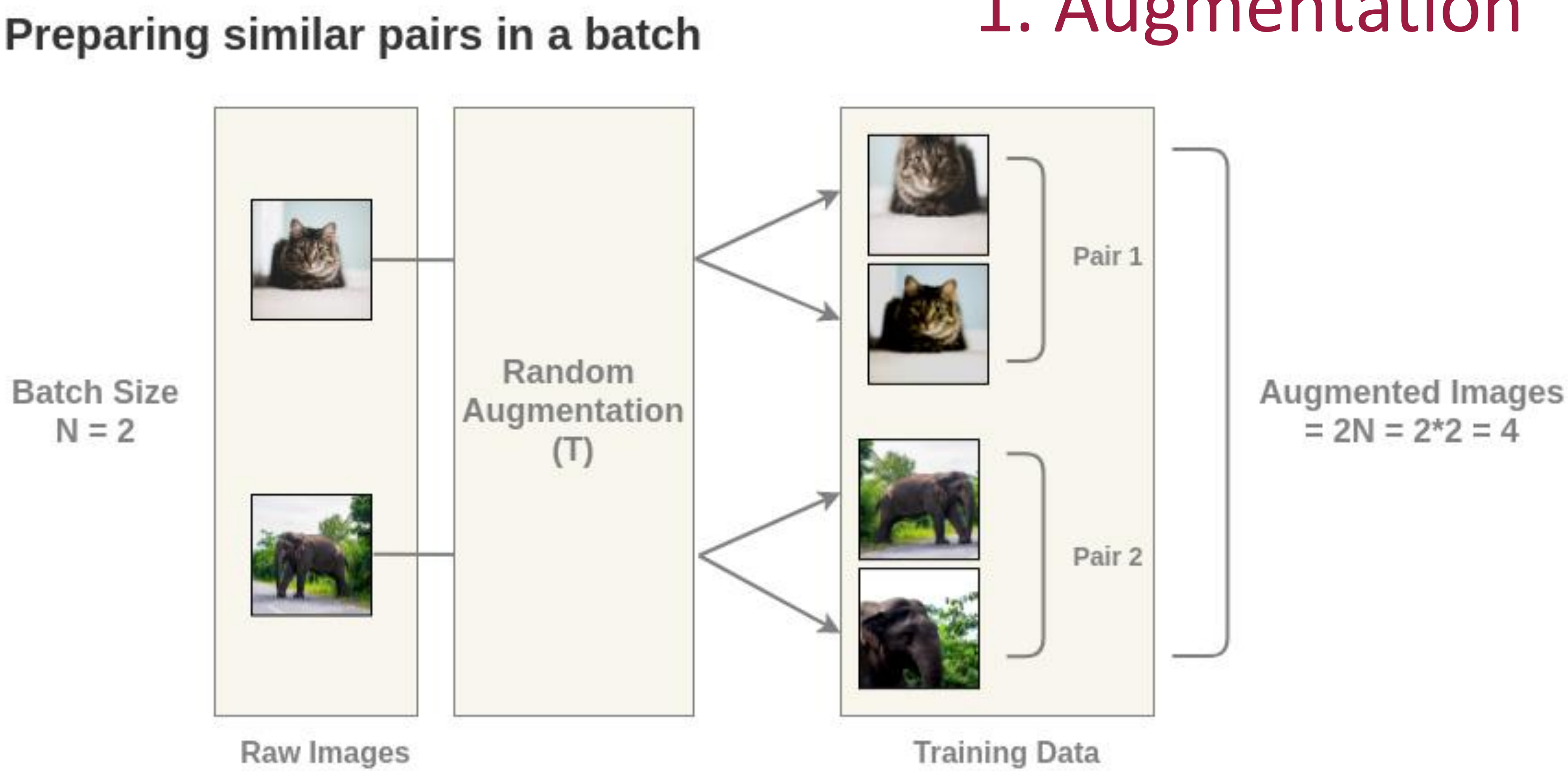
ICML 2020

Overview



https://amitness.com/2020/03/illustrated-simclr/

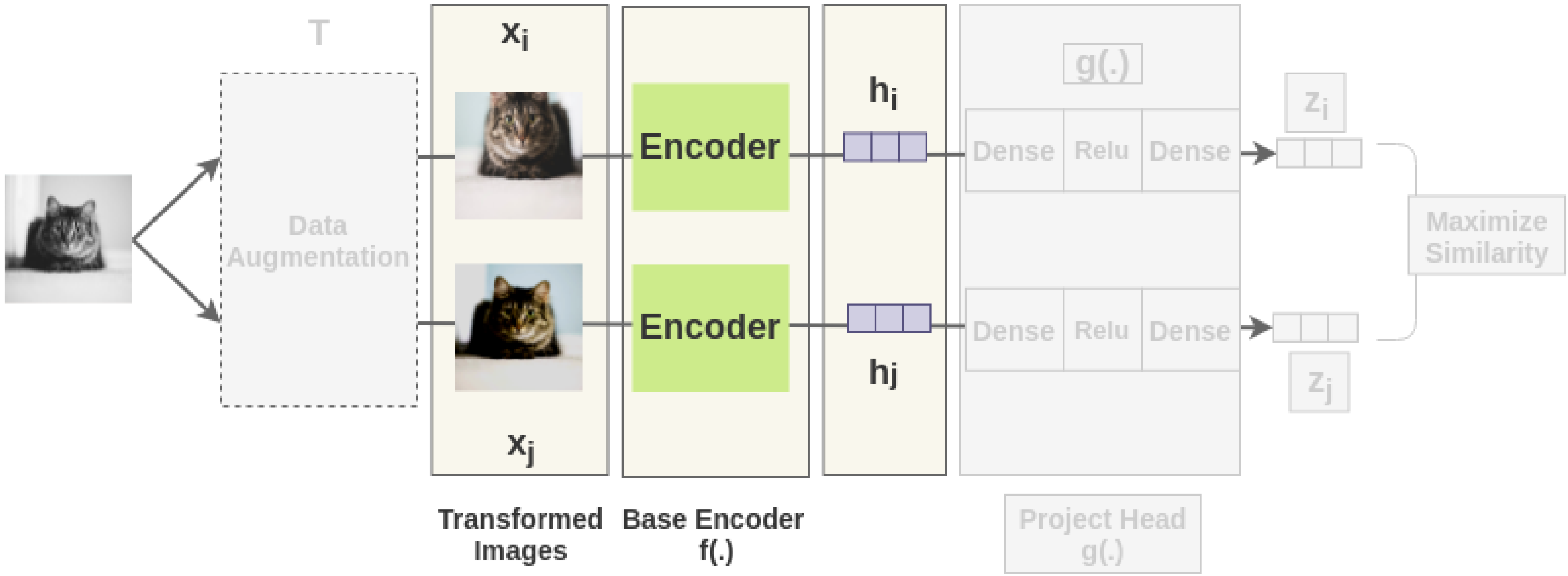
1. Augmentation



https://amitness.com/2020/03/illustrated-simclr/

2. Representation

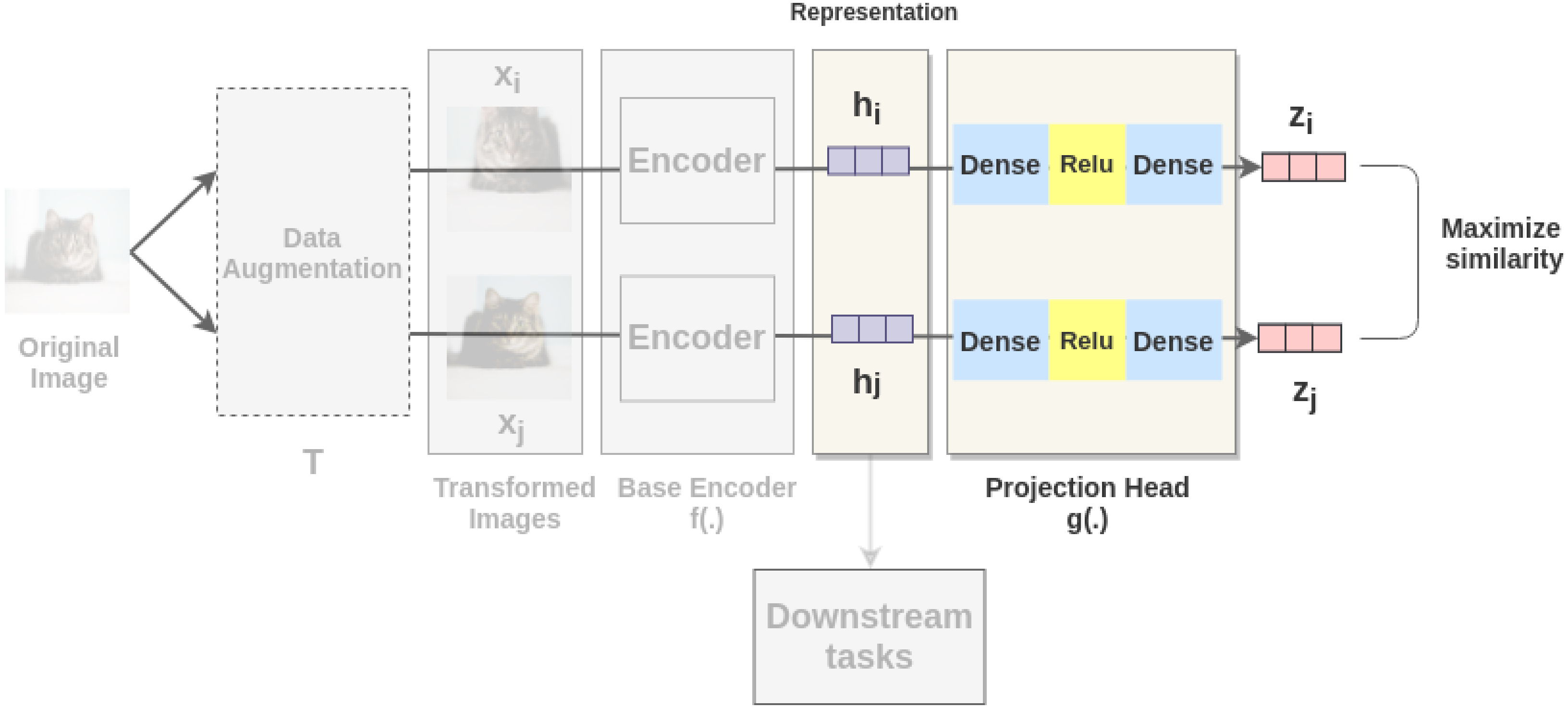
Encoder Component of Framework



https://amitness.com/2020/03/illustrated-simclr/

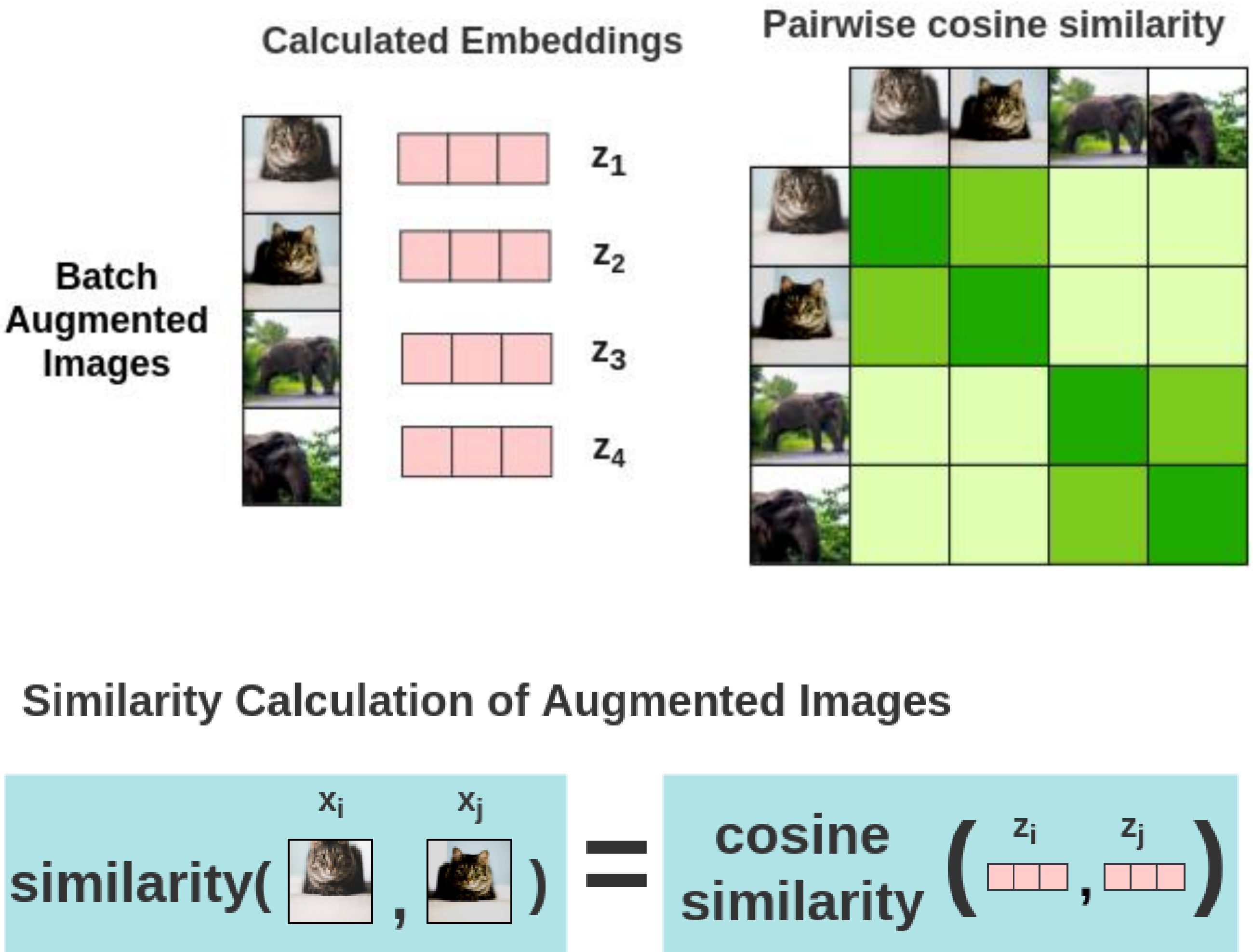
3. Projection

Projection Head Component



https://amitness.com/2020/03/illustrated-simclr/

4. Learning



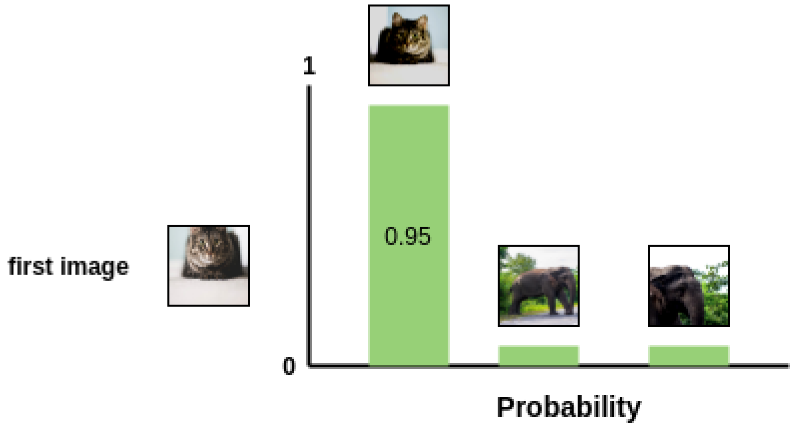
https://amitness.com/2020/03/illustrated-simclr/

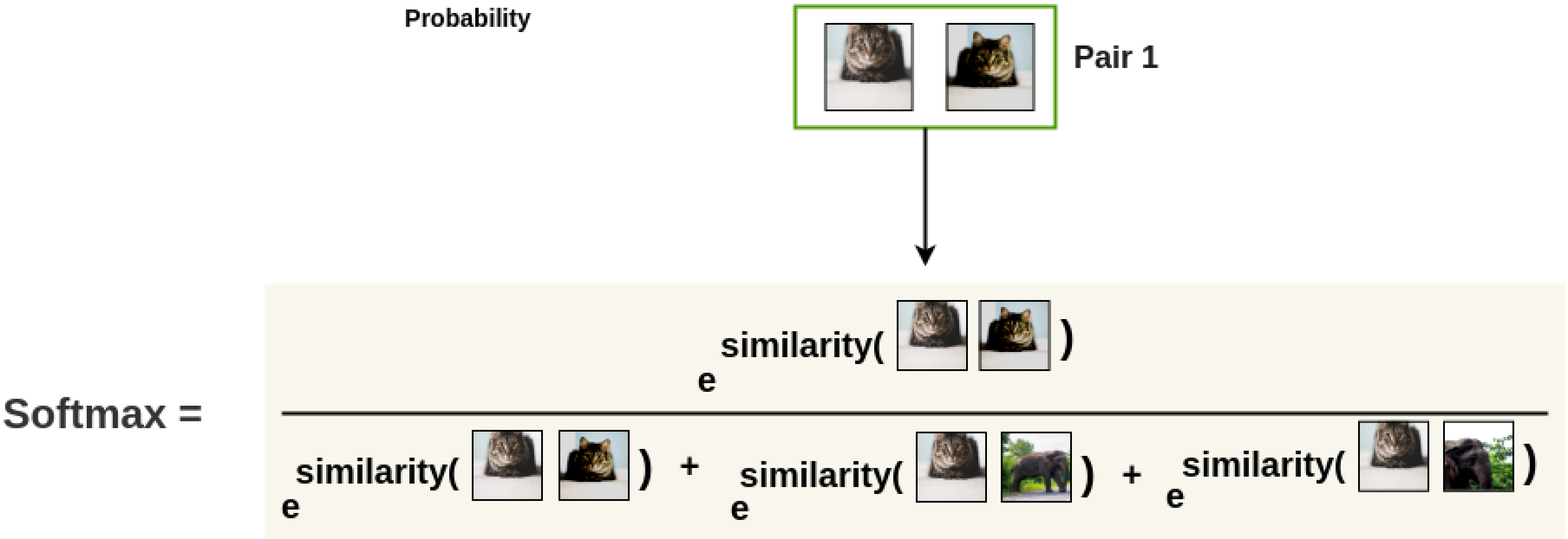
$$s_{i,j} = \frac{z_i^T z_j}{(\tau ||z_i|| ||z_j||)}$$

τ = temperature hyperparameter. It can scale the input and widen the range [-1, 1] of cosine similarity

||z|| = vector norm

4. Learning





https://amitness.com/2020/03/illustrated-simclr/

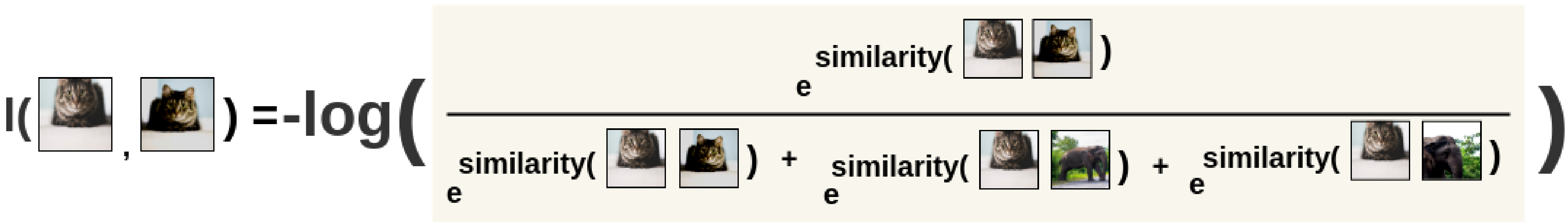
4. Learning

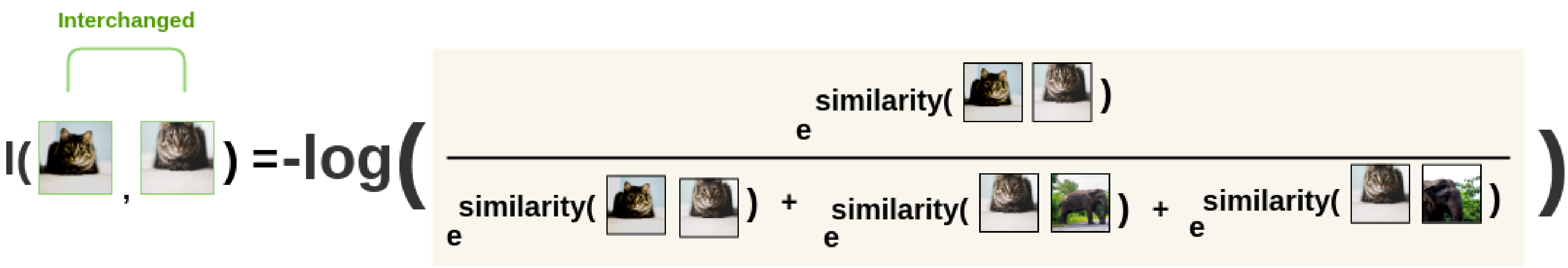
NCE (Noise Contrastive Estimator)

NT-Xent (Normalized Temperature-Scaled Cross-Entropy Loss).

$$l(i,j) = -\log \frac{\exp(s_{i,j})}{\sum_{k=1}^{2N} 1_{[k!=i]} \exp(s_{i,k})}$$

1[k!=i] = 1 iff k!=i

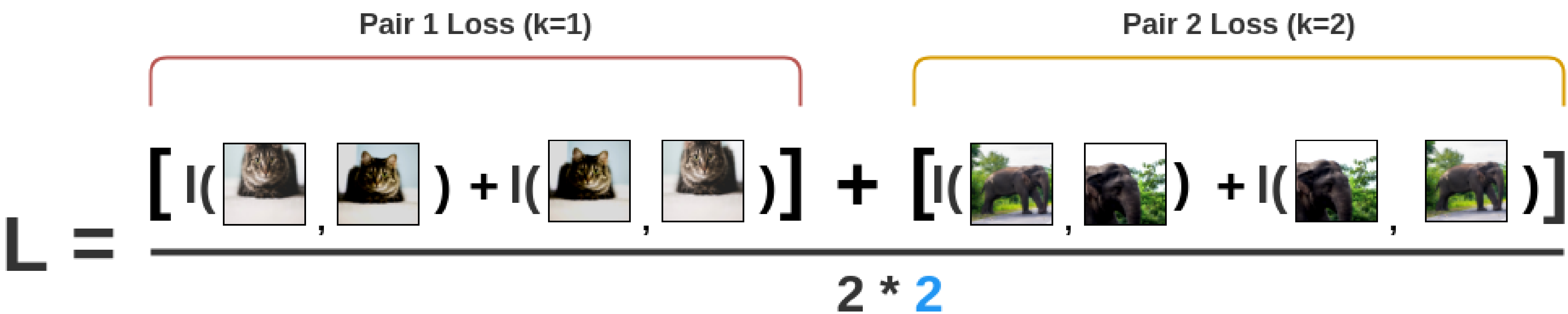




https://amitness.com/2020/03/illustrated-simclr/

4. Learning

$$L = \frac{1}{2N} \sum_{k=1}^{N} [l(2k-1, 2k) + l(2k, 2k-1)]$$

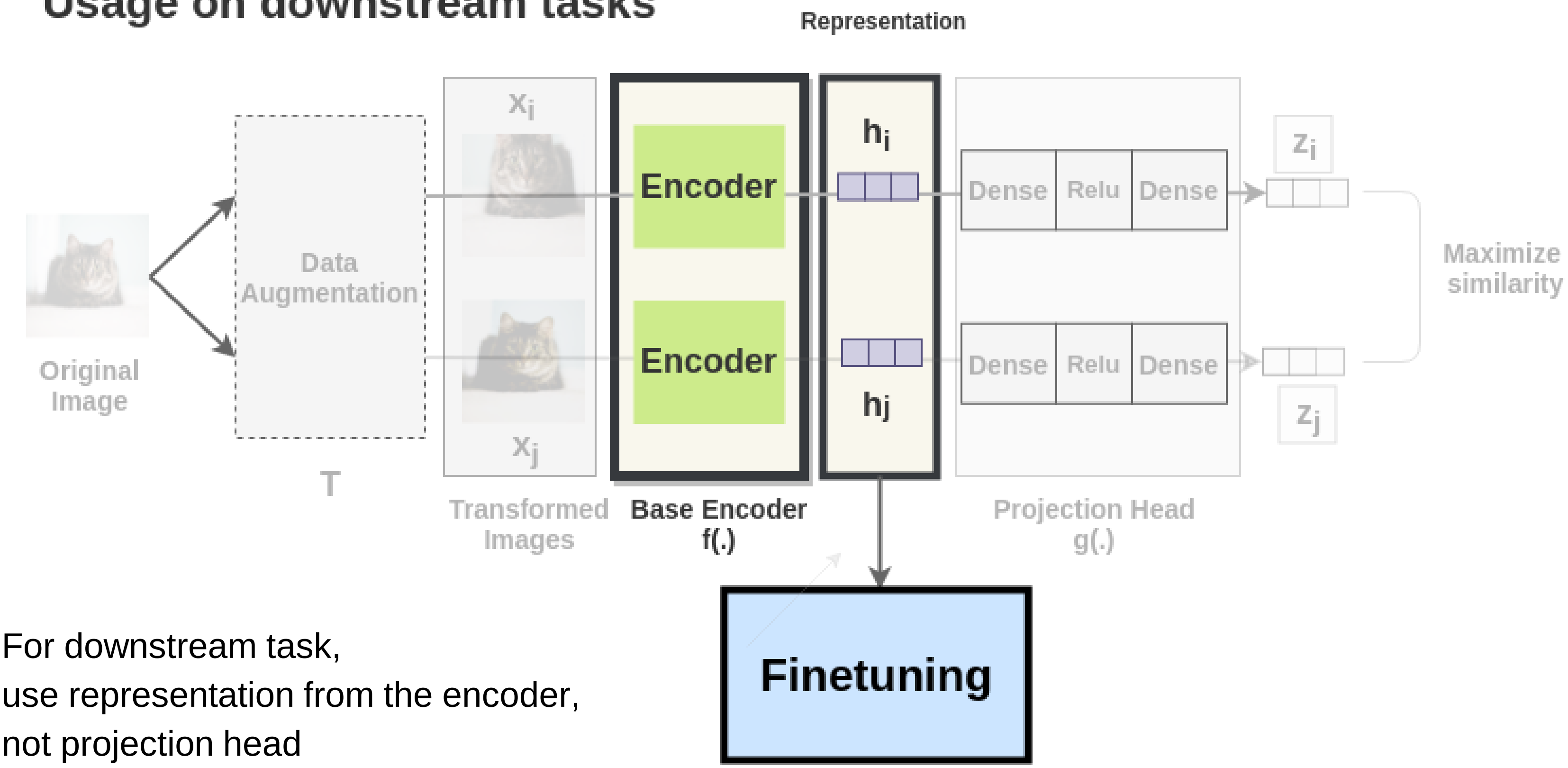


Update encoder f(.) and projection head g(.) to minimize this loss

https://amitness.com/2020/03/illustrated-simclr/

5. Adaptation

Usage on downstream tasks



For downstream task,
use representation from the encoder,
not projection head

classification, detection, ...

https://amitness.com/2020/03/illustrated-simclr/

4 main findings aside the algorithm

1 Augmentation

Composition of multiple augmentations
Unsupervised: stronger augmentation than supervised

2 Projection head

Better result: projection head in training, but not downstream task

3 Contrastive loss

Cross entropy works well, but requires l2 normalized embeddings and proper temperature hyperparameter

4 Batch size, epochs, network

Bigger batch size, longer training = better (> supervised)
Deeper and wider network = better (= supervised)

1. Augmentation

Spatial / Geometric

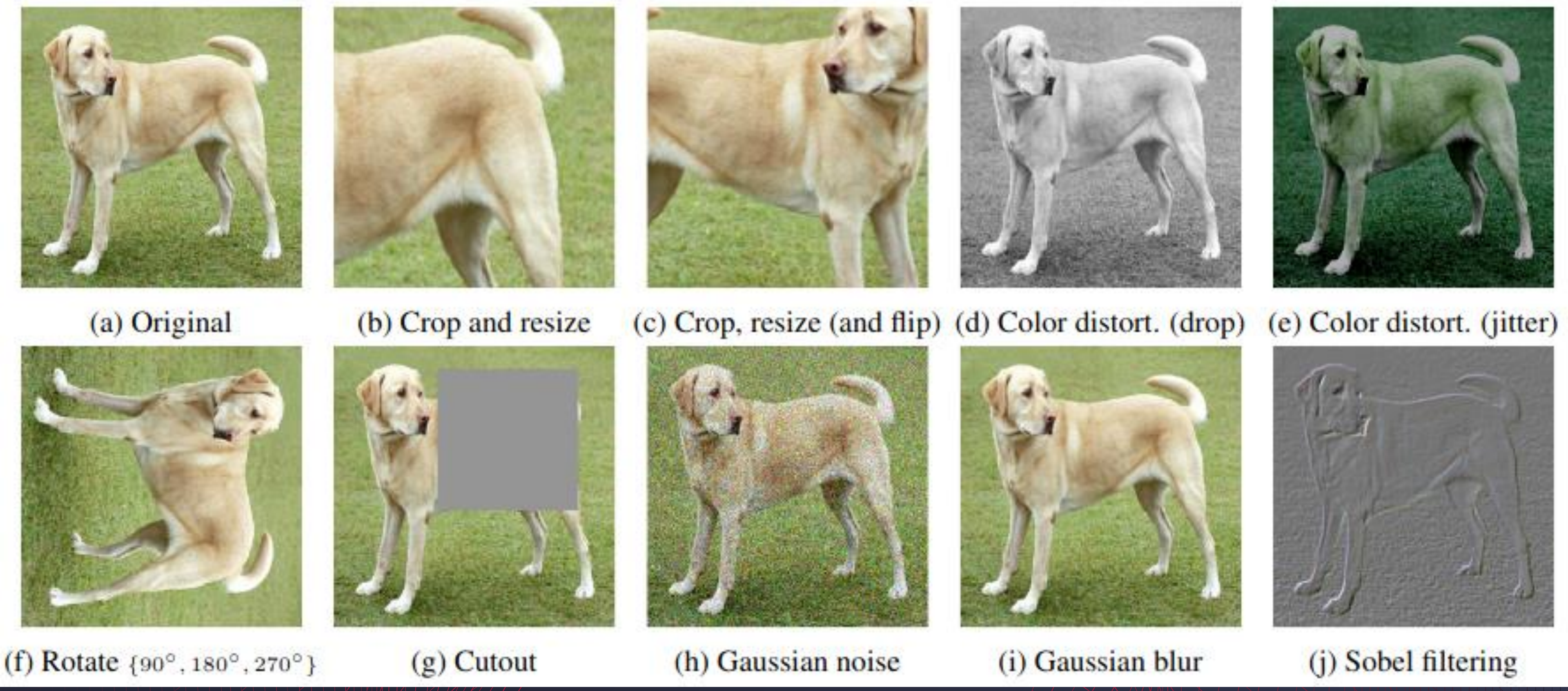
- Crop
- Resize
- Flip
- Rotate
- Cutout

Appearance

- Color distortion: color dropping, brightness, contrast, saturation, hue
- Gaussian blur
- Sobel filtering

Train: random crop (with flip and resize), color distortion, and Gaussian blur

1. Augmentation



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate {90°, 180°, 270°} (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

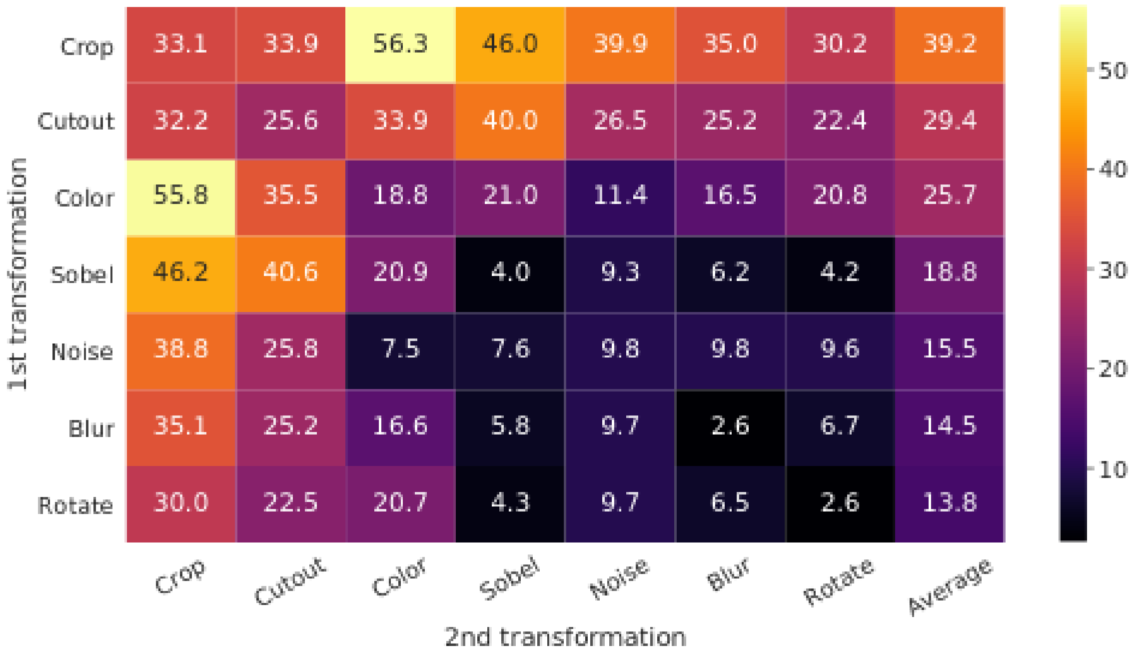


Figure 5. Linear evaluation (ImageNet top-1 accuracy) under individual or composition of data augmentations, applied only to one branch. For all columns but the last, diagonal entries correspond to single transformation, and off-diagonals correspond to composition of two transformations (applied sequentially). The last column reflects the average over the row.

Augmentation

Composite augmentation
-> harder contrastive prediction task
-> better representation

Crop + color distortion = best performance

Loss function

Only cross entropy weighs the negatives by their relative hardness

NT-Xent requires l2 norm and proper temperature hyperparameter

Cross entropy, logistic, margin triplet

Margin	NT-Logi.	Margin (sh)	NT-Logi.(sh)	NT-Xent
50.9	51.6	57.5	57.9	63.9

Table 4. Linear evaluation (top-1) for models trained with different loss functions. "sh" means using semi-hard negative mining.

ℓ_2 norm?	τ	Entropy	Contrastive acc.	Top 1
Yes	0.05	1.0	90.5	59.7
	0.1	4.5	87.8	64.4
	0.5	8.2	68.2	60.7
	1	8.3	59.1	58.0
No	10	0.5	91.7	57.2
	100	0.5	92.1	57.0

Table 5. Linear evaluation for models trained with different choices of ℓ_2 norm and temperature τ for NT-Xent loss. The contrastive distribution is over 4096 examples.

Benchmark

Linear classifier on top of frozen base network

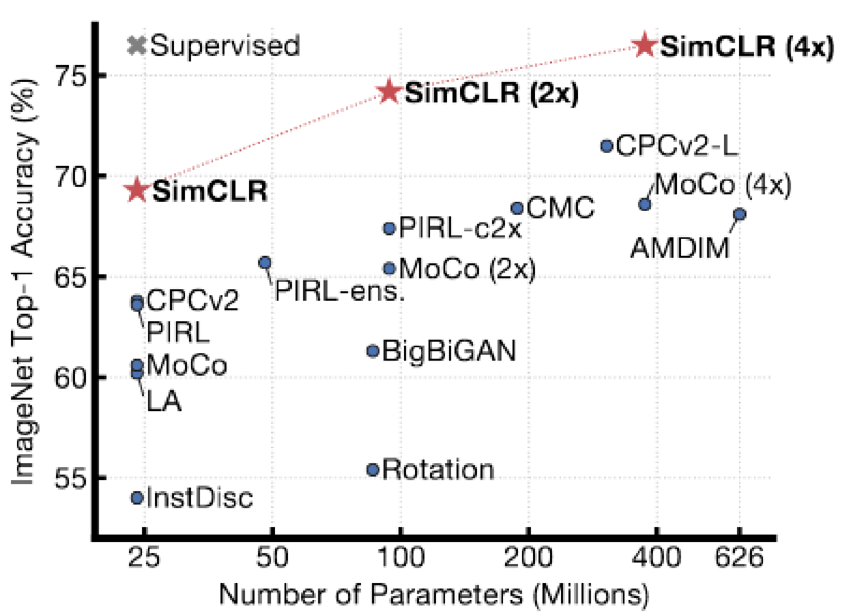


Figure 1. ImageNet top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold.

Fine tune whole base network

Method	Architecture	Label fraction 1%	10%
		Top 5	
Methods using other label-propagation:			
Pseudo-label	ResNet50	51.6	82.4
VAT+Entropy Min.	ResNet50	47.0	83.4
UDA (w. RandAug)	ResNet50	-	88.5
FixMatch (w. RandAug)	ResNet50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet50 (4×)	-	91.2
Methods using representation learning only:			
InstDisc	ResNet50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(∗)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	**85.8**	**92.6**

Table 7. ImageNet accuracy of models trained with few labels.

Self-supervised Pretraining of Visual Features in the Wild

Priya Goyal[1] Mathilde Caron[1,2] Benjamin Lefaudeux[1] Min Xu[1] Pengchao Wang[1] Vivek Pai[1]
Mannat Singh[1] Vitaliy Liptchinsky[1] Ishan Misra[1] Armand Joulin[1] Piotr Bojanowski[1]

[1] Facebook AI Research [2] Inria*

Code: https://github.com/facebookresearch/vissl

Abstract

Recently, self-supervised learning methods like MoCo [22], SimCLR [8], BYOL [20] and SwAV [7] have reduced the gap with supervised methods. These results have been achieved in a control environment, that is the highly curated ImageNet dataset. However, the premise of self-supervised learning is that it can learn from any random image and from any unbounded dataset. In this work, we explore if self-supervision lives to its expectation by training large models on random, uncurated images with no supervision. Our final SElf-supERvised (SEER) model, a RegNetY with 1.3B parameters trained on 1B random images with 512 GPUs achieves 84.2% top-1 accuracy, surpassing the best self-supervised pretrained model by 1% and confirming that self-supervised learning works in a real world setting. Interestingly, we also observe that self-supervised models are good few-shot learners achieving 77.9% top-1 with access to only 10% of ImageNet.

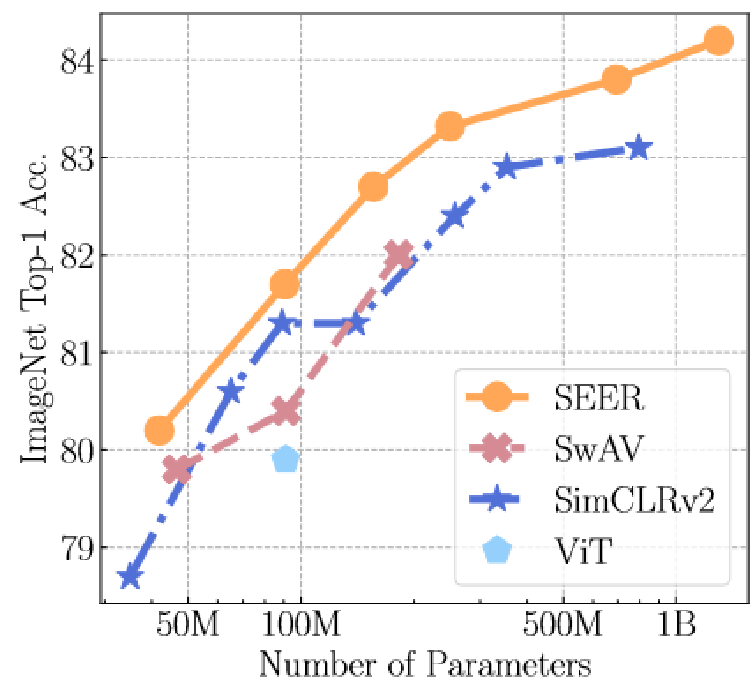


Figure 1: Performance of large pretrained models on ImageNet. We pretrain our SEER models n an uncurated and random images. They are RegNet architectures [40] trained with the SwAV self-supervised method [7] We compare with the original models trained in Caron et al. [7] as well as the pretraining on curated data from SimCLRv2 [9] and ViT [14]. The network architectures are different. We report the top-1 accuracy after finetuning on ImageNet.

1. Introduction

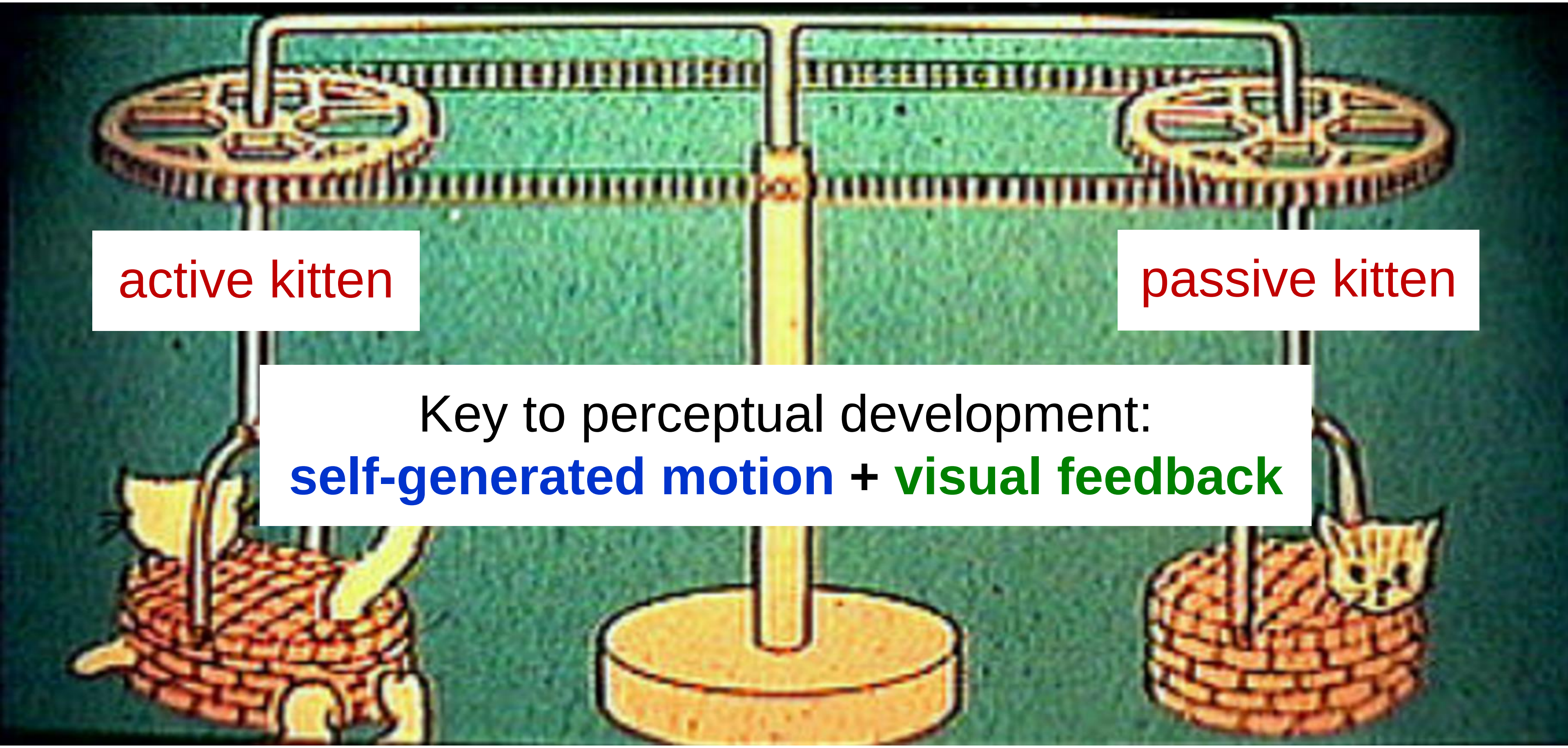
A recent trend shows that well-tailored model pre-training approaches (weakly-supervised, semi-supervised,

Learning image representations tied to ego-motion

Dinesh Jayaraman and Kristen Grauman

ICCV 2015

The kitten carousel experiment
[Held & Hein, 1963]



Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Problem with today's visual learning

Status quo: Learn from "disembodied" bag of labeled snapshots.

Our goal: Learn in the context of acting and moving in the world.

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Our idea: Ego-motion ↔ vision

Goal: Teach computer vision system the connection:
"how I move" ↔ "how my visual surroundings change"

+

Ego-motion motor signals

Unlabeled video

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Ego-motion ↔ vision: view prediction

After moving:

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Ego-motion ↔ vision for recognition

Learning this connection requires:

- Depth, 3D geometry
- Semantics
- Context

Also key to recognition!

Can be learned without manual labels!

Our approach: unsupervised feature learning using egocentric video + motor signals

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Approach idea: Ego-motion equivariance

Invariant features: unresponsive to some classes of transformations

$$\mathbf{z}(g\mathbf{x}) \approx \mathbf{z}(\mathbf{x})$$

Equivariant features : *predictably* responsive to some classes of transformations, through simple mappings (e.g., linear)

"equivariance map"

$$\mathbf{z}(g\mathbf{x}) \approx M_g\mathbf{z}(\mathbf{x})$$

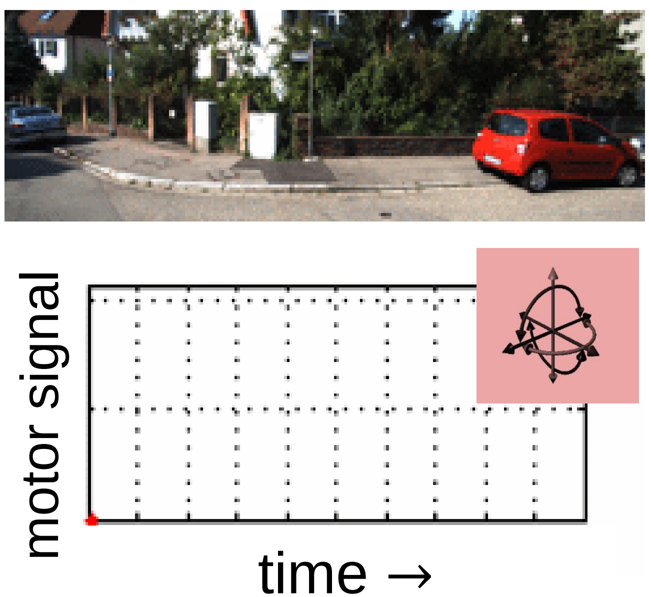
Invariance *discards* information; equivariance *organizes* it.

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Approach idea: Ego-motion equivariance

Training data

Unlabeled video + motor signals



Learn

Equivariant embedding

organized by ego-motions

Pairs of frames related by similar ego-motion should be related by same feature transformation

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Approach overview

Our approach: unsupervised feature learning using egocentric video + motor signals

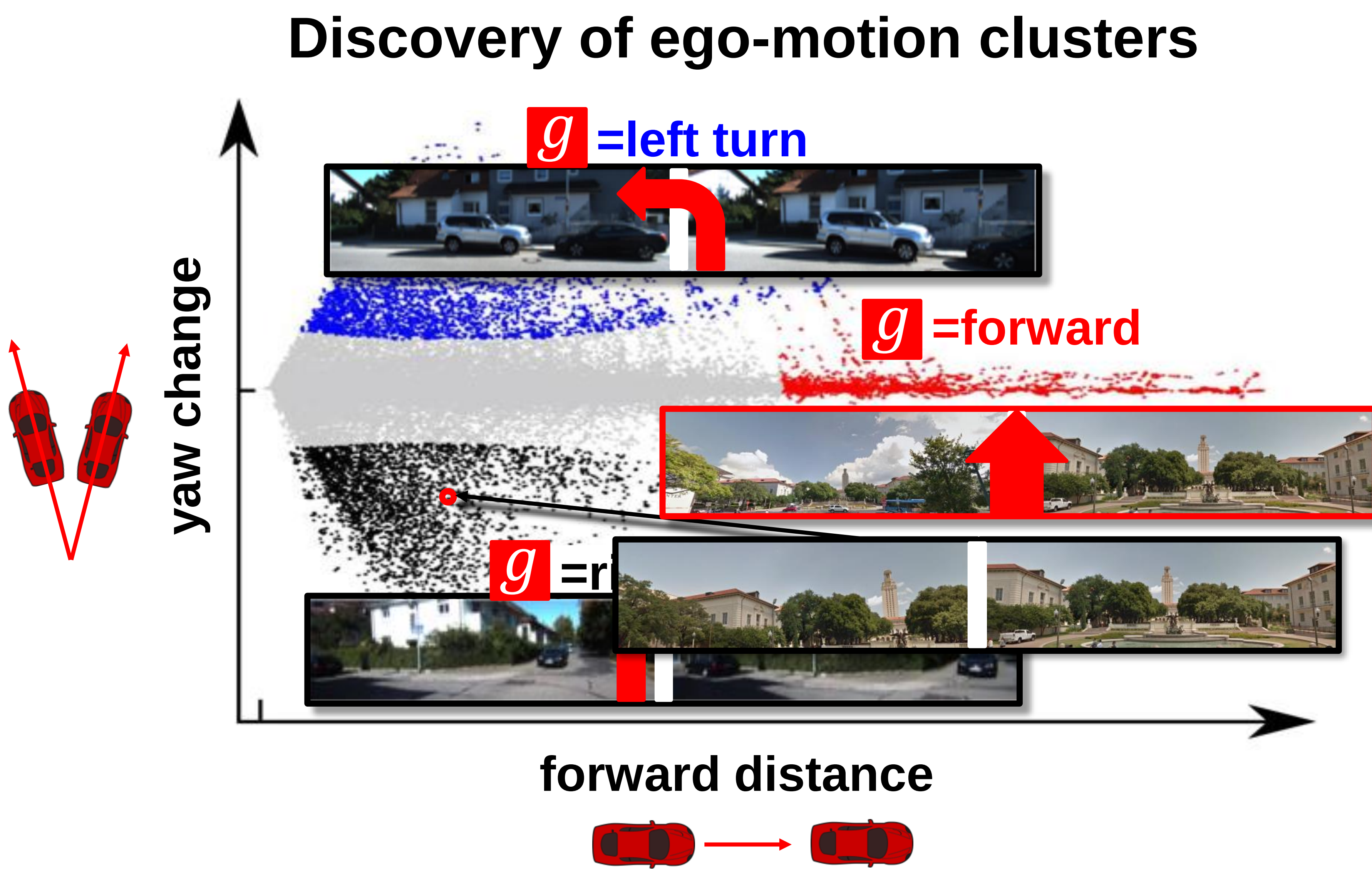
1. Extract training frame pairs from video

2. Learn ego-motion-equivariant image features

3. Train on target recognition task in parallel

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Training frame pair mining



Jayaraman and Grauman, “Learning image representations tied to ego-motion”, ICCV 2015

Ego-motion equivariant feature learning

Given:

$\mathbf{x}_i$

g

$g\mathbf{x}_i$

$\mathbf{x}_k$

class y_k

Desired: for all motions g and all images $\mathbf{x}$,

$$\mathbf{z}_\theta(g\mathbf{x}) \approx M_g \mathbf{z}_\theta(\mathbf{x})$$

Unsupervised training

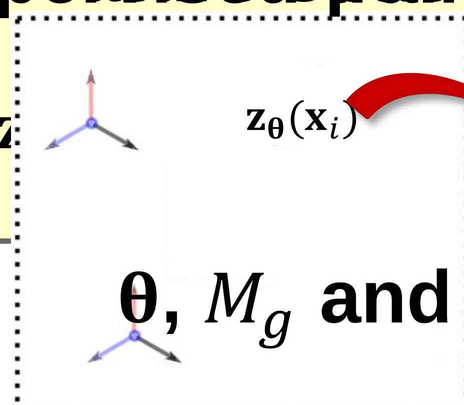
θ → $\mathbf{z}_\theta(\mathbf{x}_i)$ → M_g →

θ → $\mathbf{z}_\theta(g\mathbf{x}_i)$ →

$\| M_g \mathbf{z}_\theta(\mathbf{x}_i) - \mathbf{z}_\theta(g\mathbf{x}_i) \|_2$

Supervised training

Feature space



θ → z

max loss $L_C(\mathbf{x}_k, y_k)$

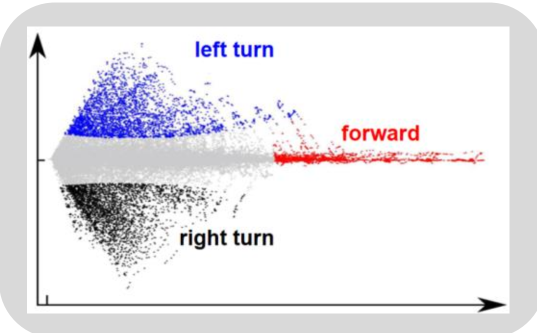
θ, M_g and W jointly trained

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

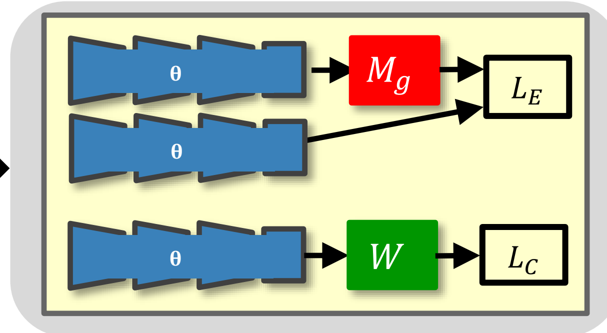
Summary

APPROACH

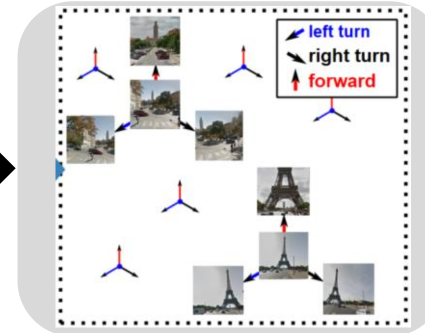
Ego-motion training pairs



Neural network training



Equivariant embedding



RESULTS

Scene and object recognition

Football field?
Pagoda?
Airport?
Cathedral?
Army base?

Next-best view selection

cup

frying pan

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Results: Recognition

Learn from **unlabeled car video** (KITTI)

Geiger et al, IJRR '13

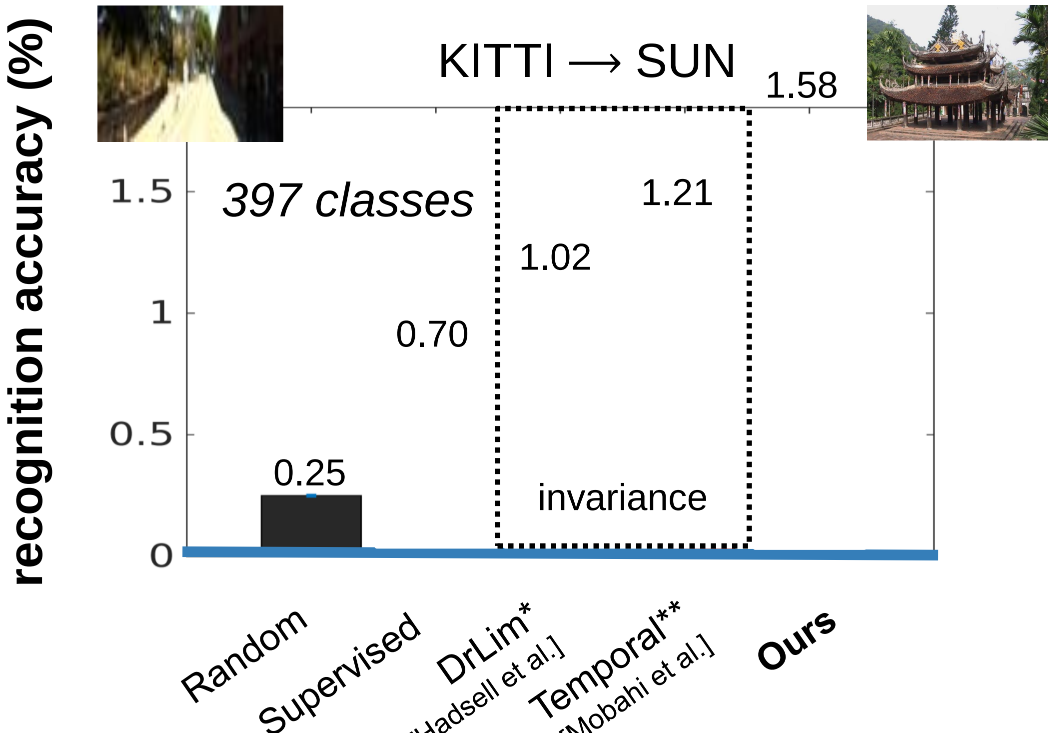
Exploit features for **static scene classification**
(SUN, 397 classes)

Xiao et al, CVPR '10

Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

Results: Recognition

Do ego-motion equivariant features improve recognition?



6 labeled training examples per class

Up to 30% accuracy increase over state of the art!

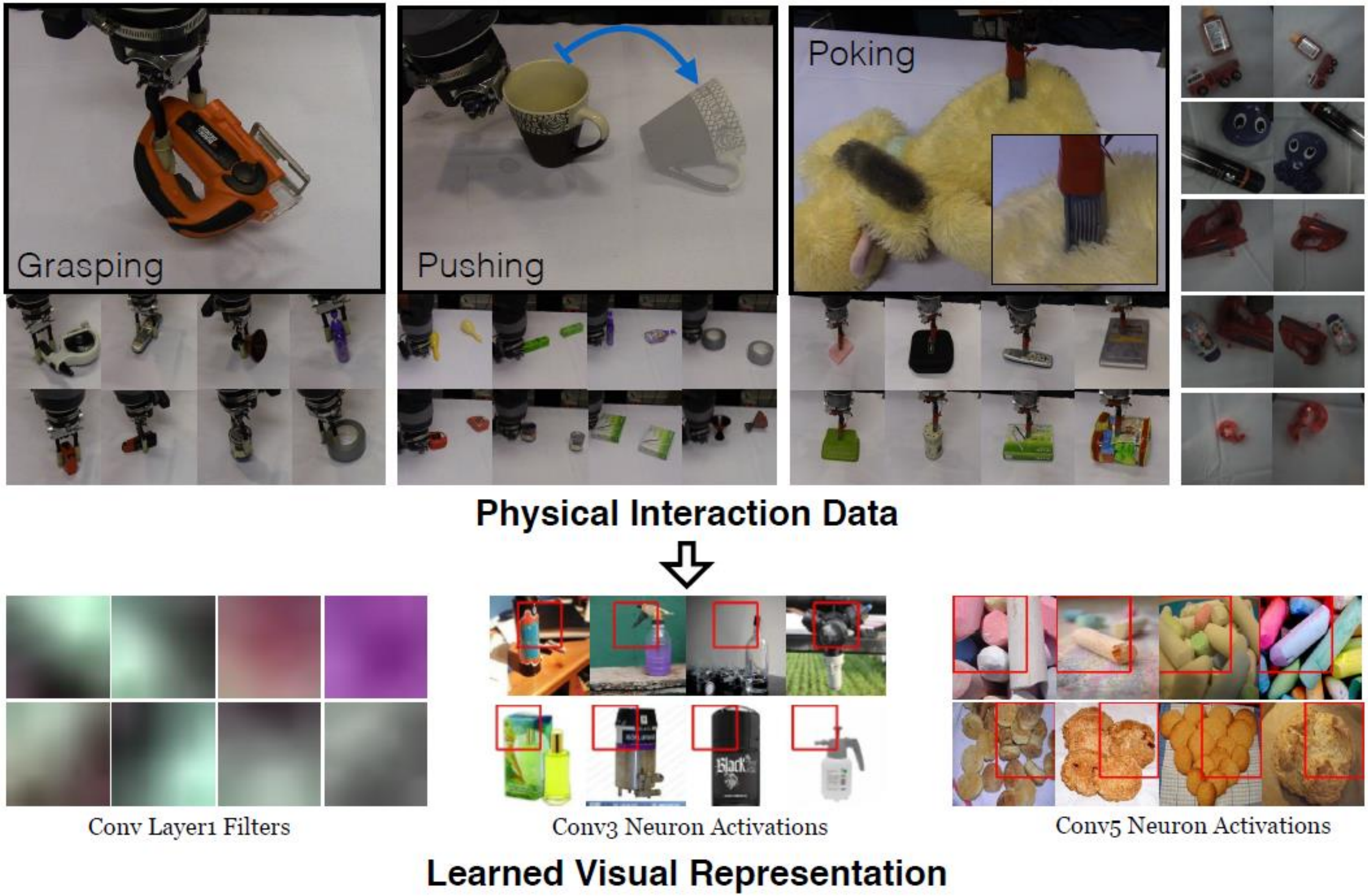
Jayaraman and Grauman, "Learning image representations tied to ego-motion", ICCV 2015

The Curious Robot: Learning Visual Representations via Physical Interactions

Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han,
Yong-Lae Park, and Abhinav Gupta

ECCV 2016

Embodied representations



Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Grasping

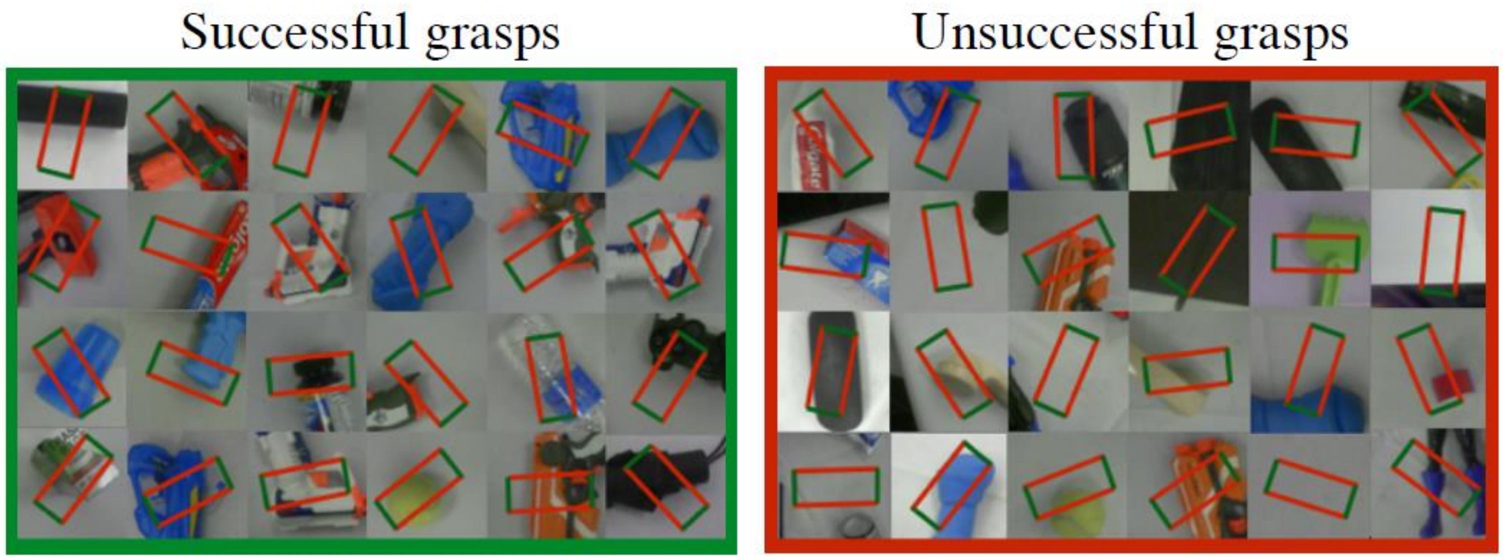


Fig. 2. Examples of successful (left) and unsuccesful grasps (right). We use a patch based representation: given an input patch we predict 18-dim vector which represents whether the center location of the patch is graspable at $0°$, $10°$, $\ldots 170°$.

Pushing

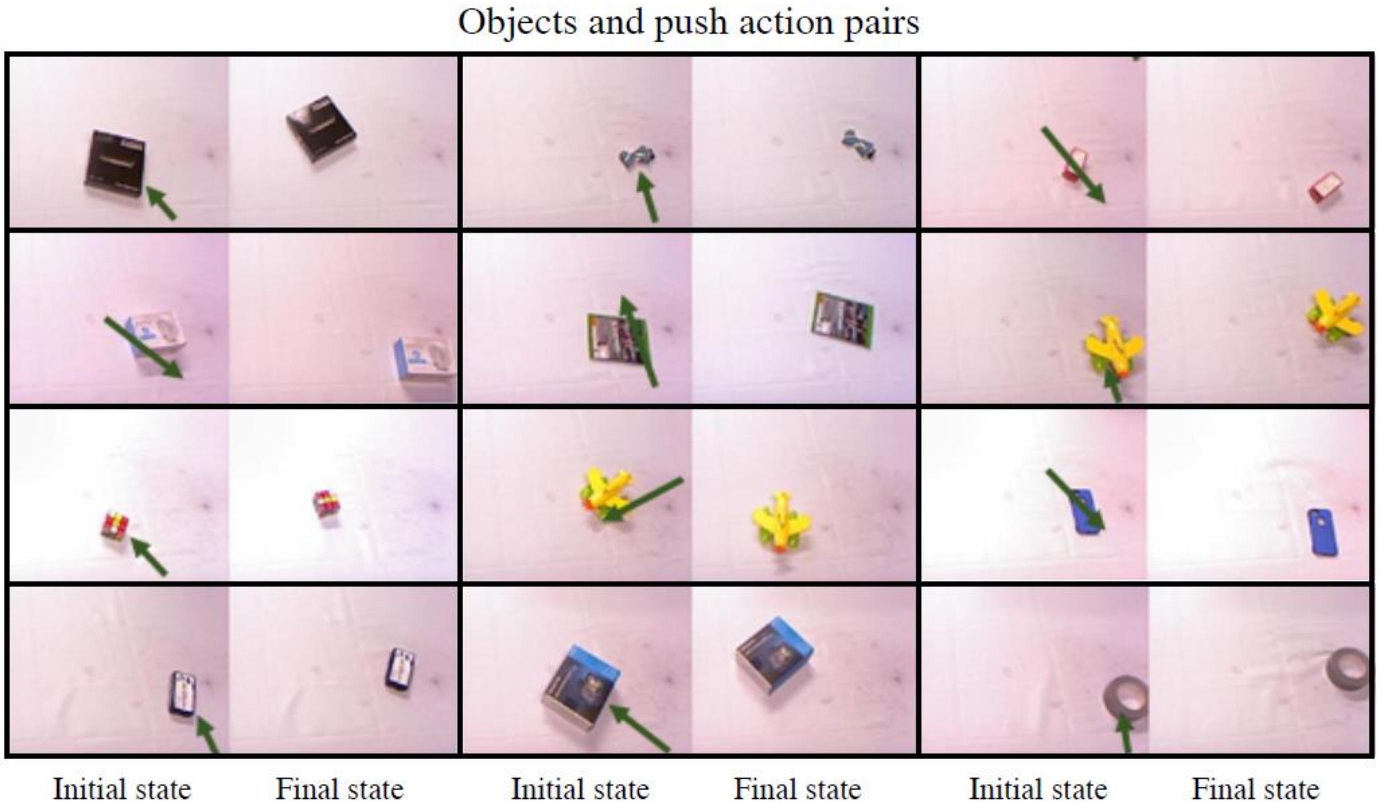


Fig. 4. Examples of initial state and final state images taken for the push action. The arrows demonstrate the direction and magnitude of the push action.

Pinto et al., “The Curious Robot: Learning Visual Representations via Physical Interactions”, ECCV 2016

Poking

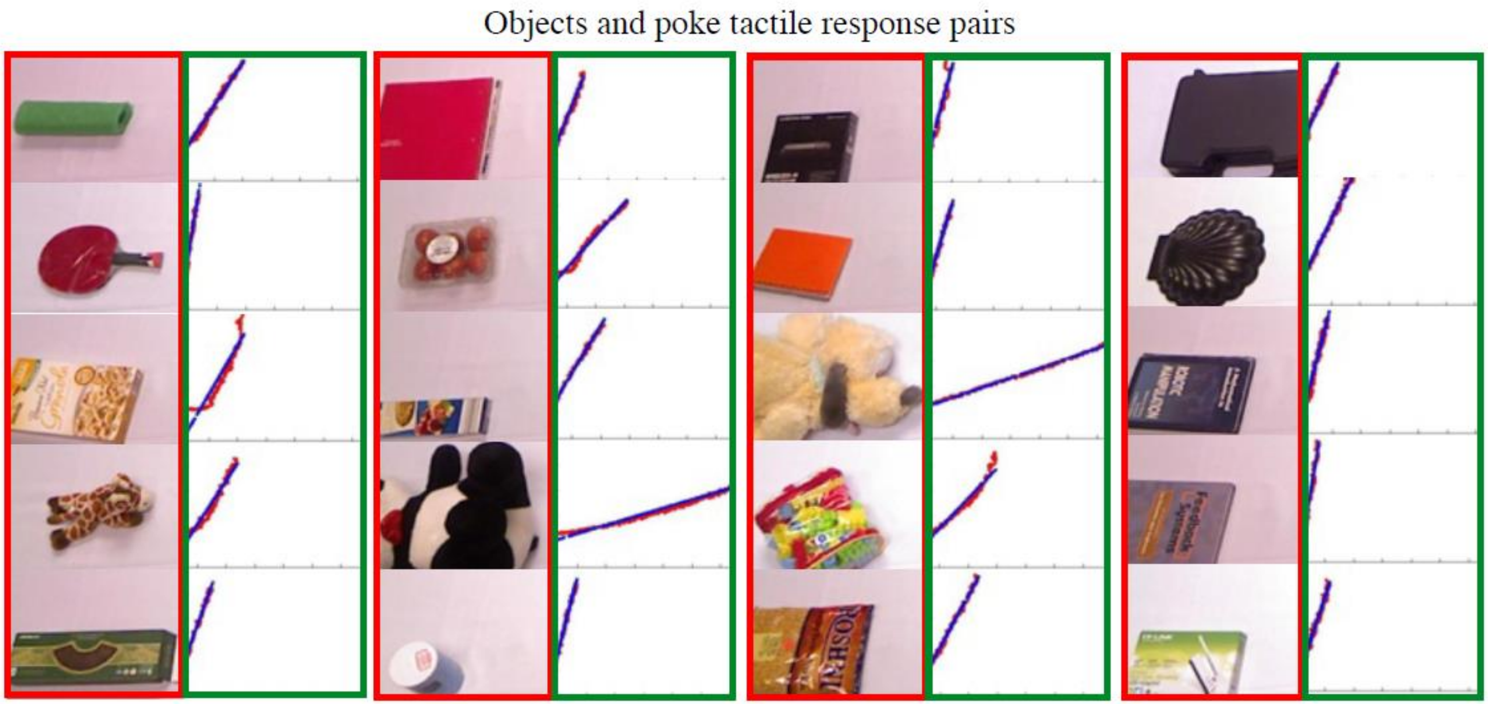


Fig. 6. Examples of the data collected by the poking action. On the left we show the object poked, and on the right we show force profiles as observed by the tactile sensor.

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Pose/viewpoint invariance

Fig. 7. Examples of objects in different poses provided to the embedding network.

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Representations from interactions

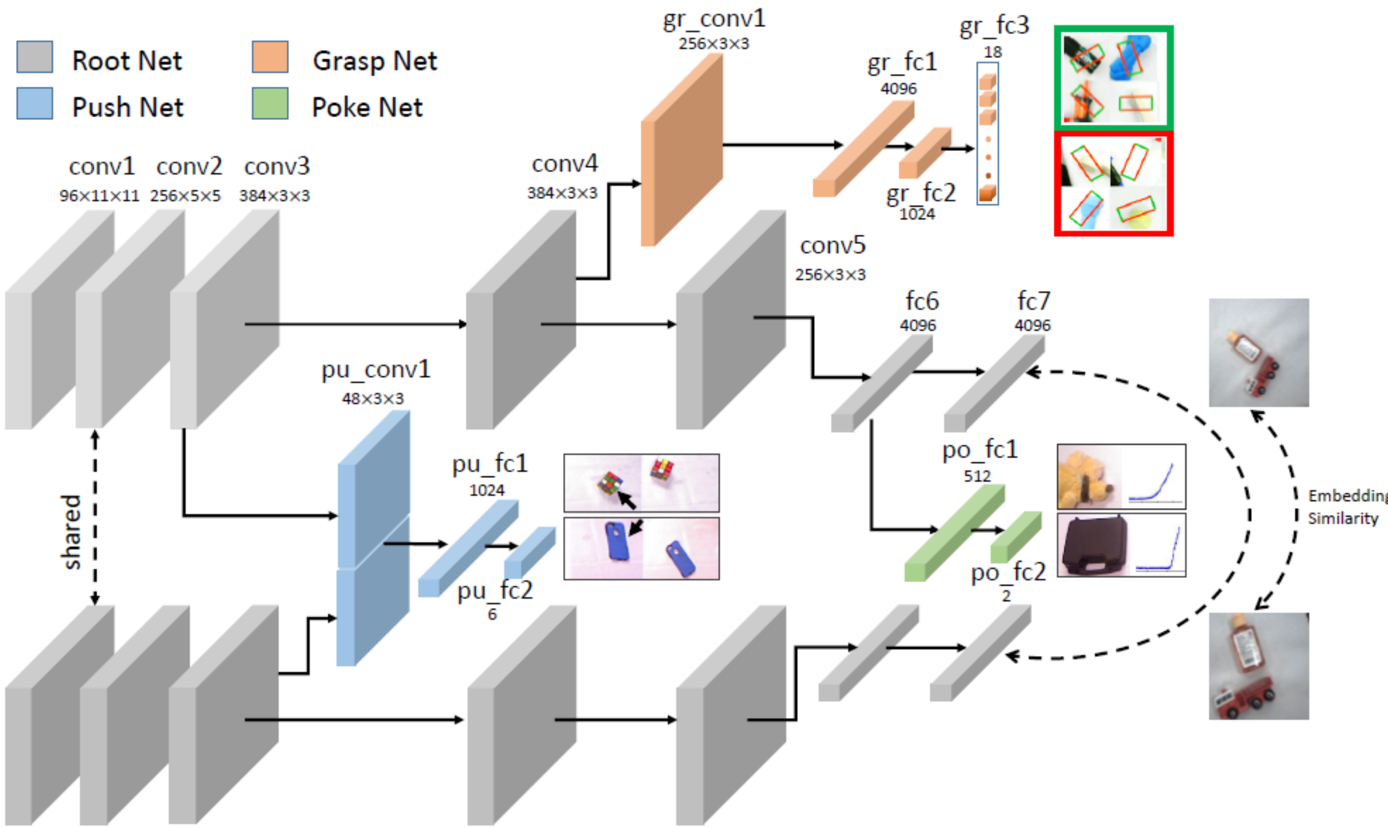


Fig. 8. Our shared convolutional architecture for four different tasks.

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Classification/retrieval performance

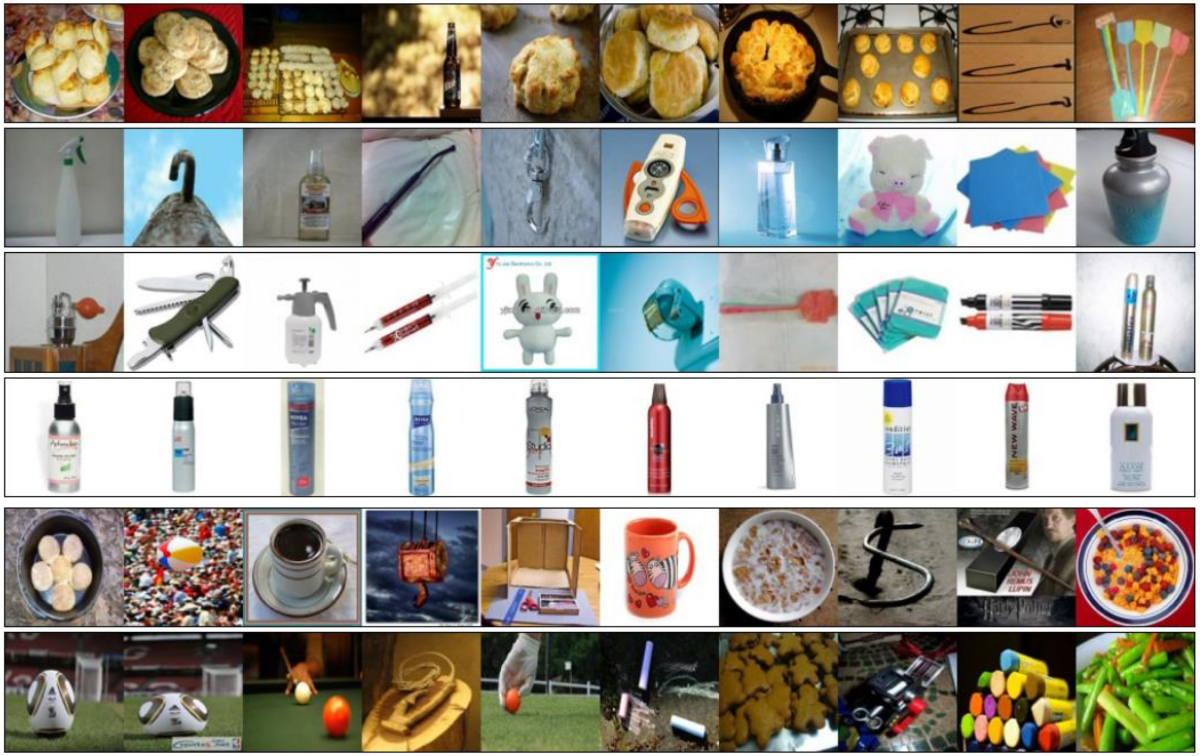


Fig. 10. The first column corresponds to query image and rest show the retrieval. Note how the network learns that cups and bowls are similar (row 5).

Pinto et al., "The Curious Robot: Learning Visual Representations via Physical Interactions", ECCV 2016

Classification/retrieval performance

Table 1. Classification accuracy on ImageNet Household, UW RGBD and Caltech-256

	Household	UW RGBD	Caltech-256
Root network with random init.	0.250	0.468	0.242
Root network trained on robot tasks (**ours**)	0.354	0.693	0.317
AlexNet trained on ImageNet	0.625	0.820	0.656

Table 2. Image Retrieval with Recall@k metric

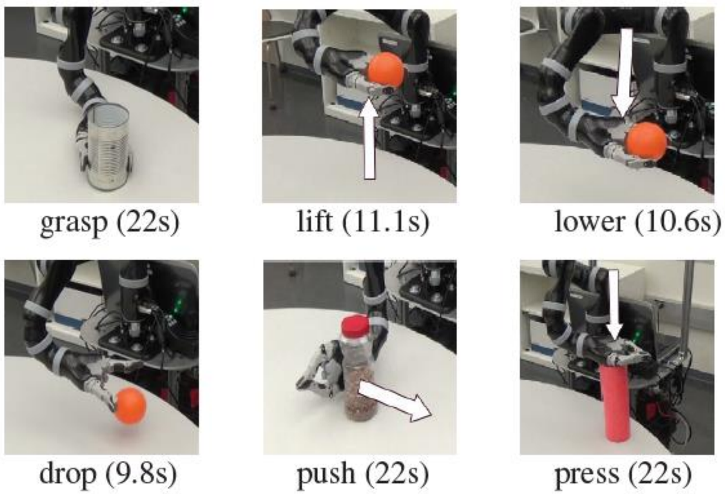
	Instance level				Category level			
	k=1	k=5	k=10	k=20	k=1	k=5	k=10	k=20
Random Network	0.062	0.219	0.331	0.475	0.150	0.466	0.652	0.800
Our Network	0.720	0.831	0.875	0.909	0.833	0.918	0.946	0.966
AlexNet	0.686	0.857	0.903	0.941	0.854	0.953	0.969	0.982

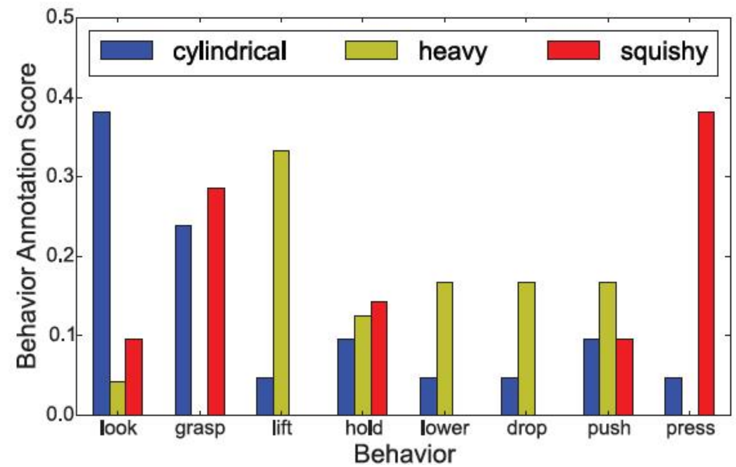
Pinto et al., “The Curious Robot: Learning Visual Representations via Physical Interactions”, ECCV 2016

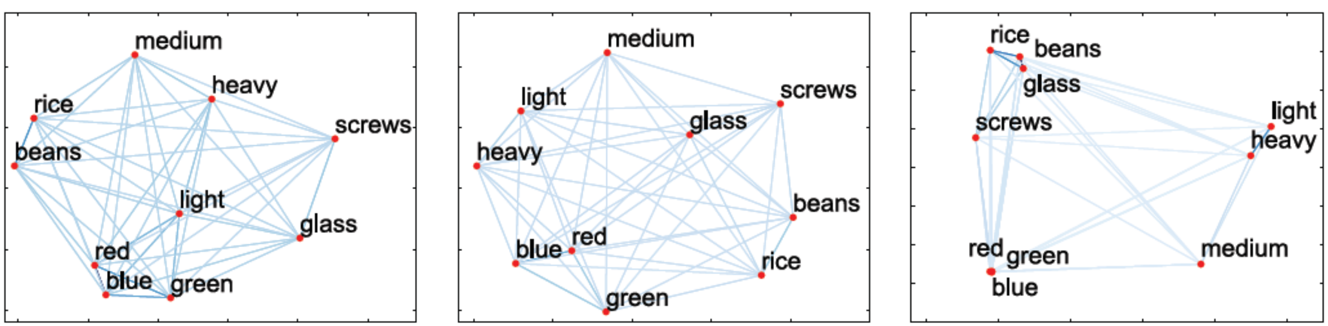
Guiding Exploratory Behaviors for Multi-Modal Grounding of Linguistic Descriptions

Jesse Thomason, Jivko Sinapov, Ray Mooney, Peter Stone

AAAI 2018







a) google news embedding b) lexsub embedding c) sensorimotor embedding

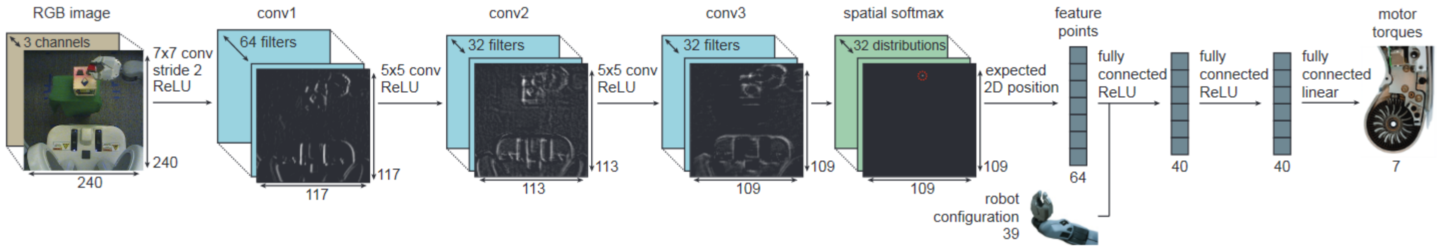


Figure 2: Visuomotor policy architecture. The network contains three convolutional layers, followed by a spatial softmax and an expected position layer that converts pixel-wise features to feature points, which are better suited for spatial computations. The points are concatenated with the robot configuration, then passed through three fully connect
to produce the torques.

End-to-End Training of Deep Visuomotor Policies

Sergey Levine, **Chelsea Finn**, Trevor Darrell, Pieter Abbeel

JMLR 2016

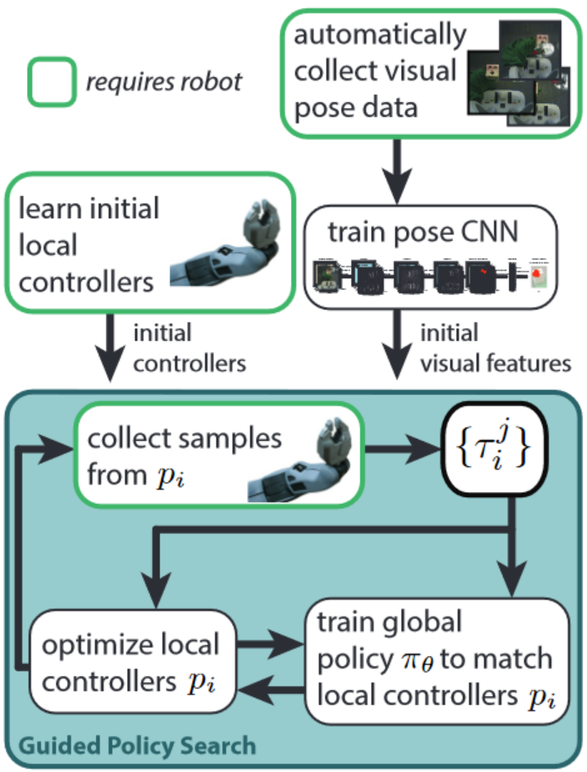


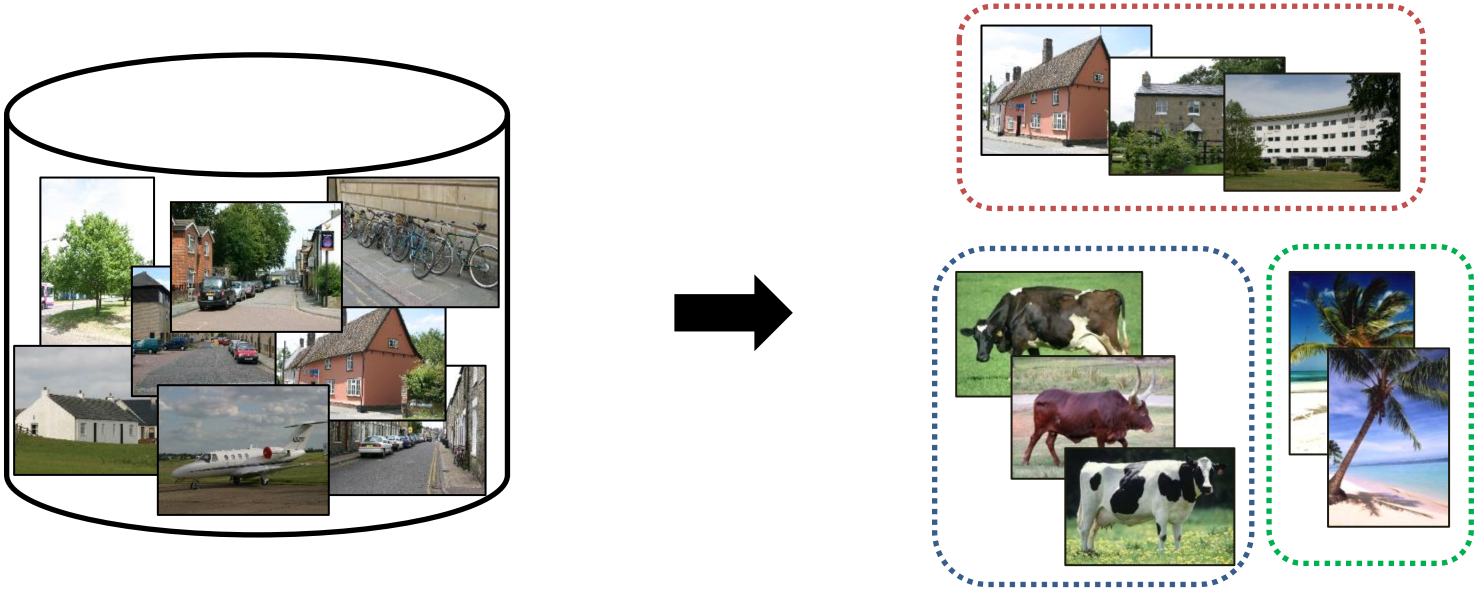
Figure 3: Diagram of our approach, including the main guided policy search phase and initialization phases.

Object-Graphs for Context-Aware Category Discovery

Yong Jae Lee and Kristen Grauman

CVPR 2010

Goal



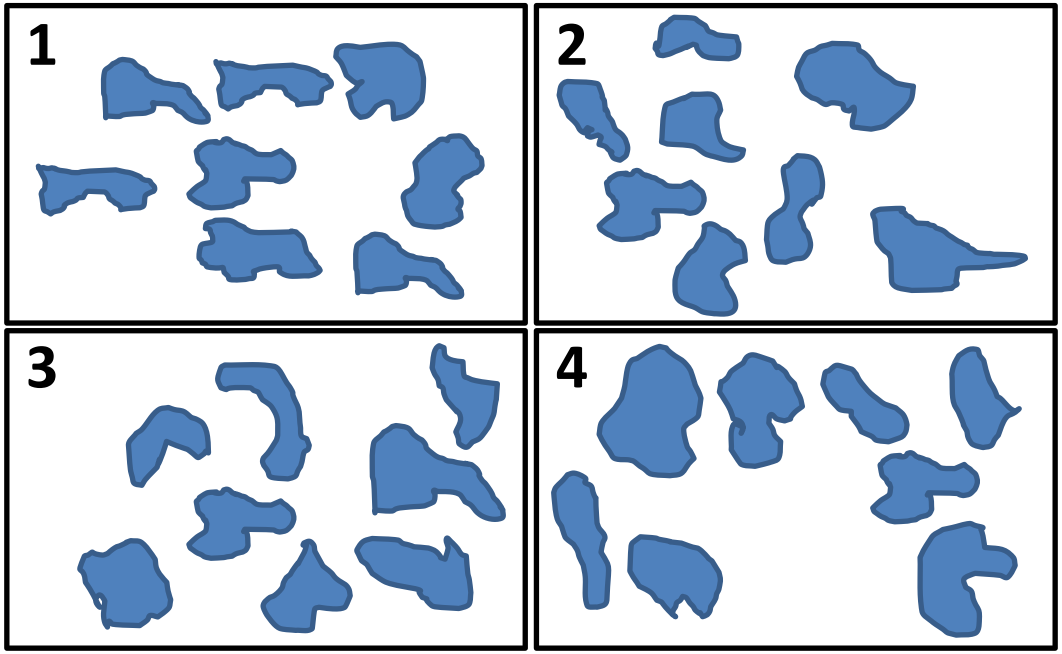
Unlabeled Image Data **Discovered categories**

- Discover *new* object categories, based on their relation to categories for which we have trained models

Lee and Grauman, "Object-Graphs for Context-Aware Category Discovery", CVPR 2010

Existing approaches

Previous work treats unsupervised visual discovery as an appearance-grouping problem.

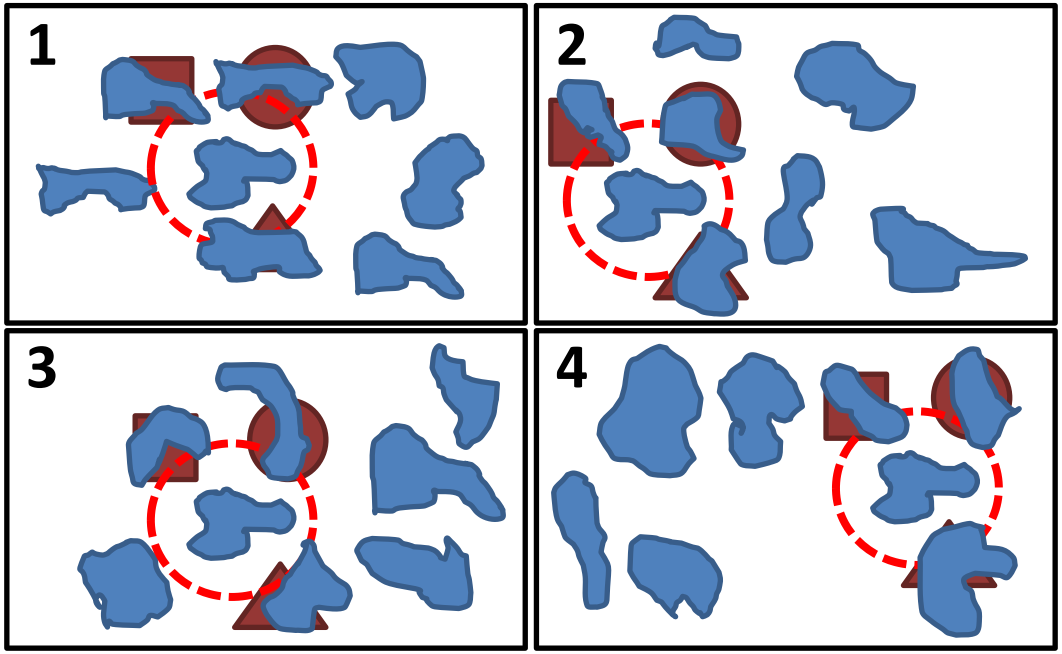


Can you identify the recurring pattern?

Lee and Grauman, "Object-Graphs for Context-Aware Category Discovery", CVPR 2010

Our idea

How can seeing previously learned objects in novel images help to discover *new* categories?

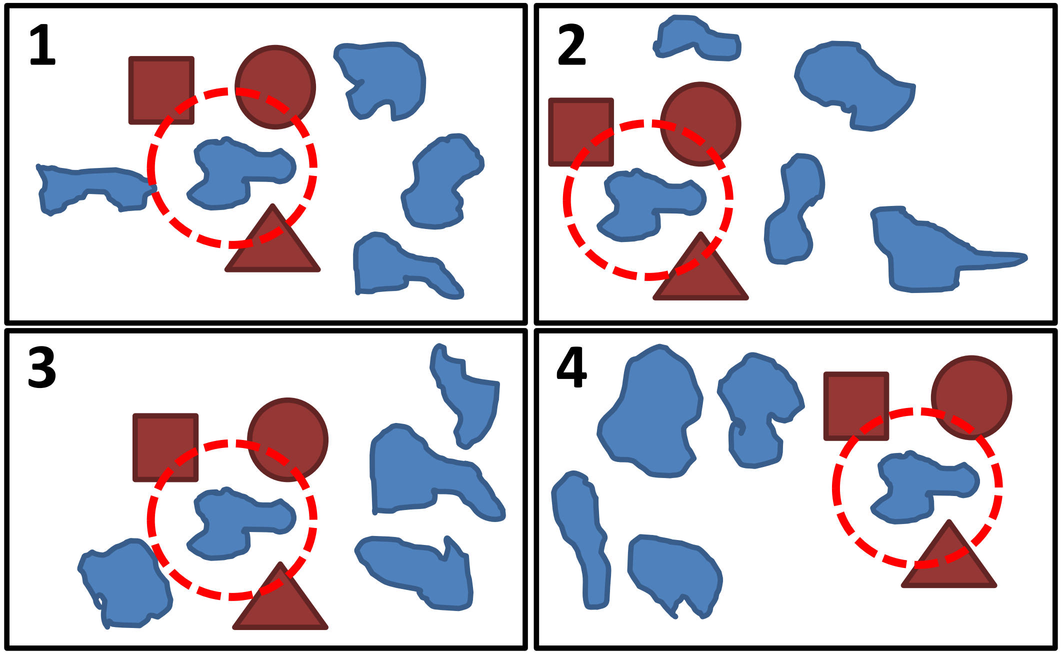


Can you identify the recurring pattern?

Lee and Grauman, "Object-Graphs for Context-Aware Category Discovery", CVPR 2010

Our idea

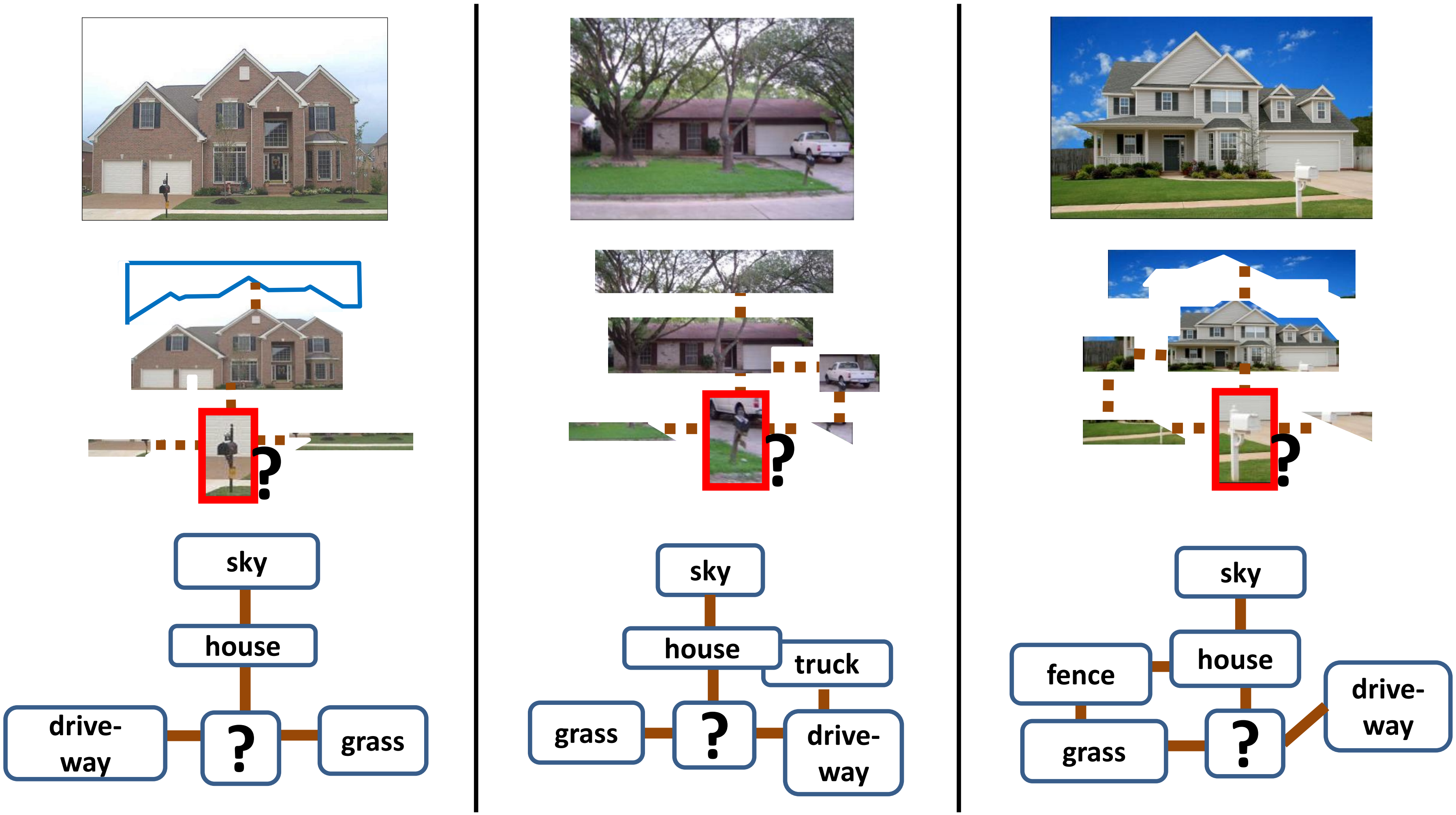
Discover visual categories within unlabeled images by modeling interactions between the unfamiliar regions and familiar objects.



Can you identify the recurring pattern?

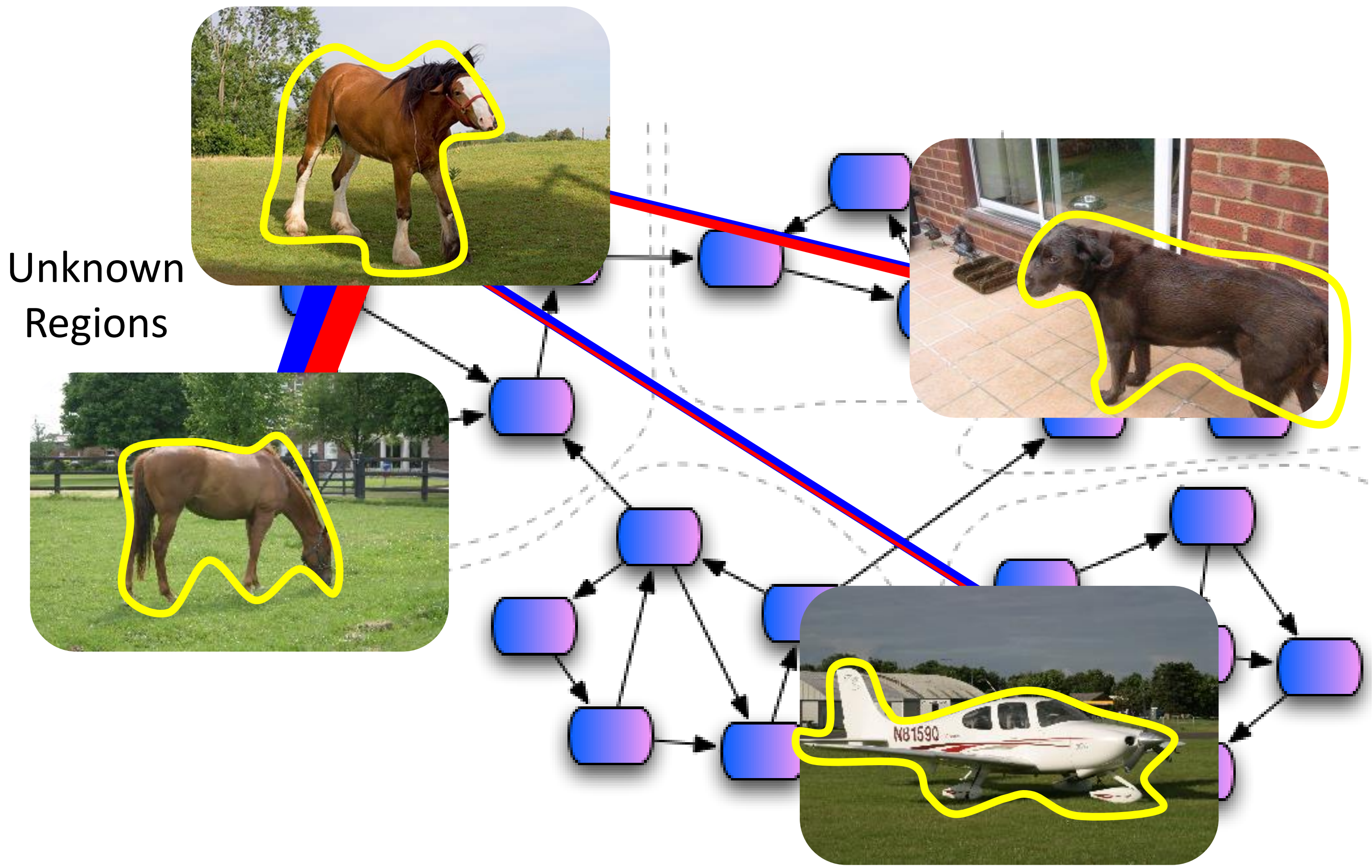
Lee and Grauman, "Object-Graphs for Context-Aware Category Discovery", CVPR 2010

Context-aware visual discovery



Lee and Grauman, “Object-Graphs for Context-Aware Category Discovery”, CVPR 2010

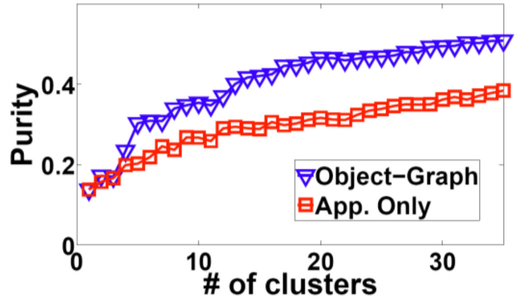
Clusters from region-region affinities



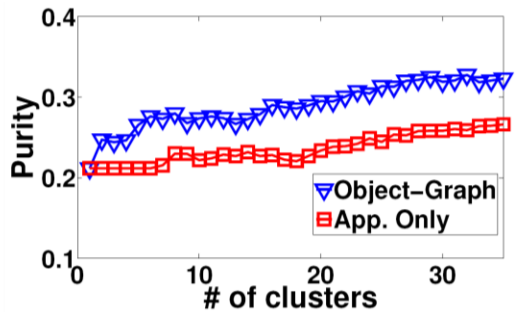
$$K(s_i, s_j) = \color{red}{K_{app}}(s_i, s_j) + \color{blue}{K_{obj-graph}}(s_i, s_j)$$

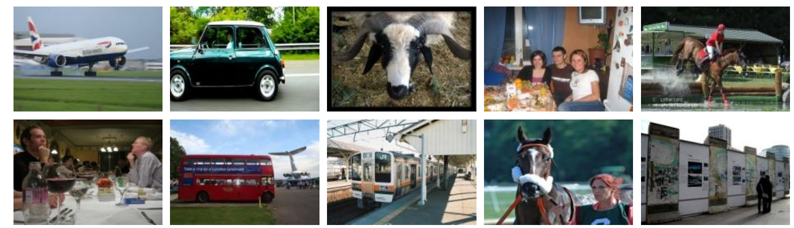
Lee and Grauman, “Object-Graphs for Context-Aware Category Discovery”, CVPR 2010

Object Discovery Accuracy

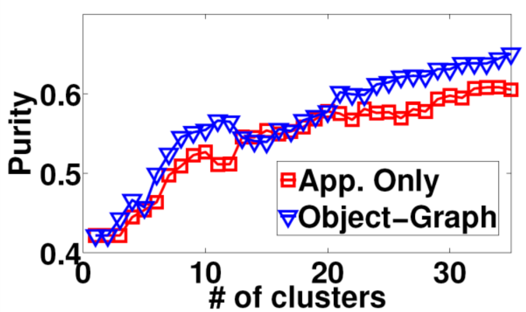


MSRC-v2

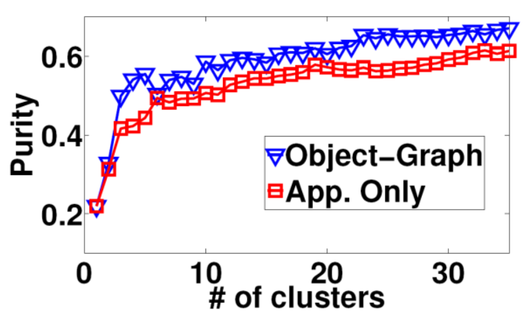




PASCAL 2008



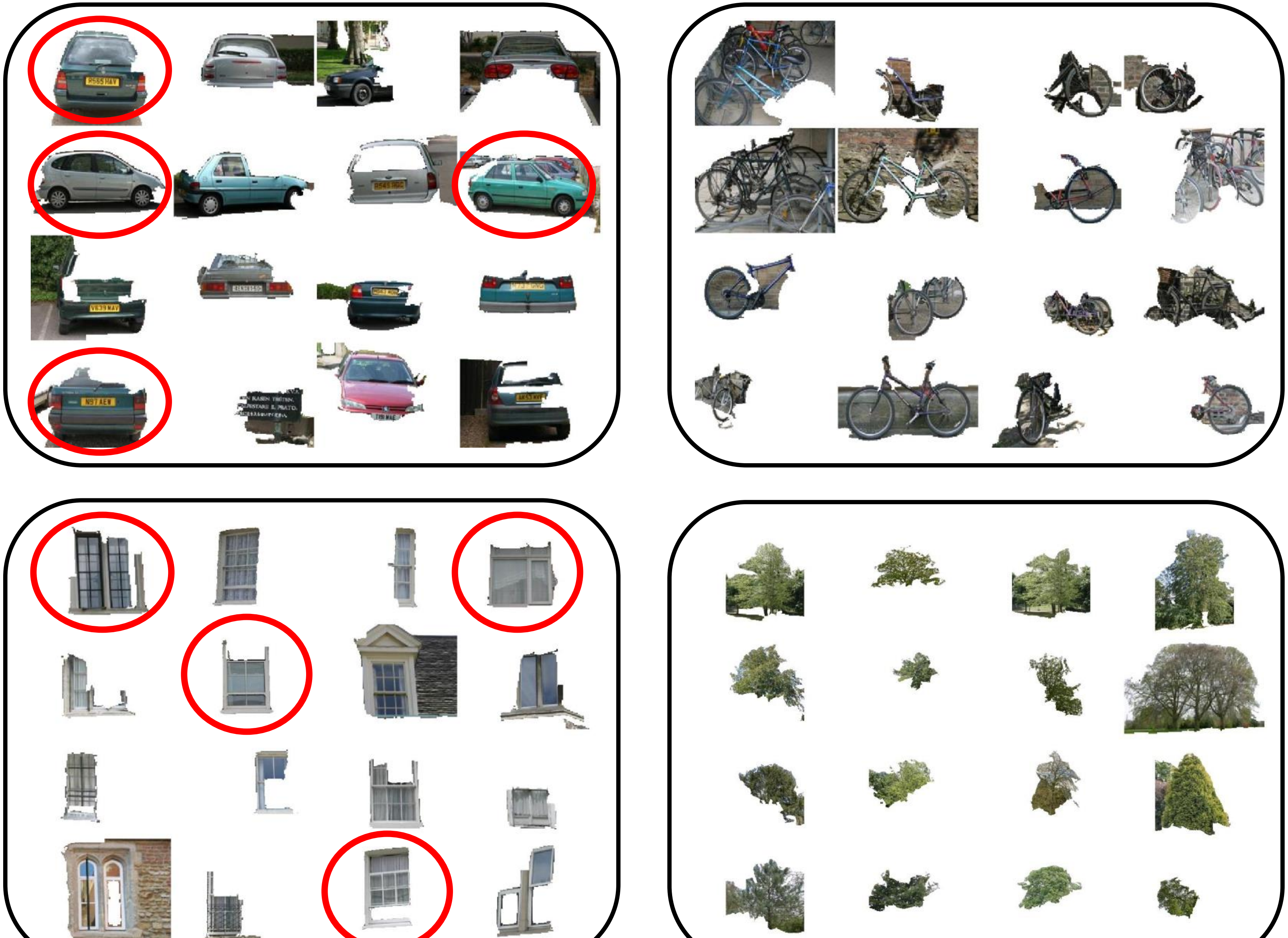
MSRC-v0



Corel

Lee and Grauman, “Object-Graphs for Context-Aware Category Discovery”, CVPR 2010

Examples of Discovered Categories



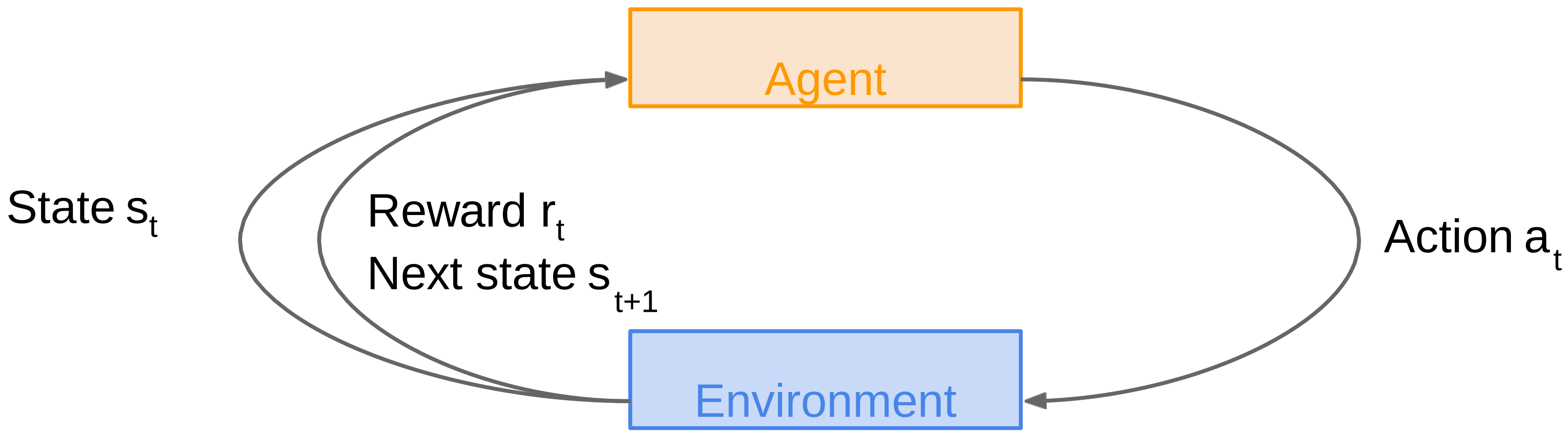
Lee and Grauman, "Object-Graphs for Context-Aware Category Discovery", CVPR 2010

Discussion

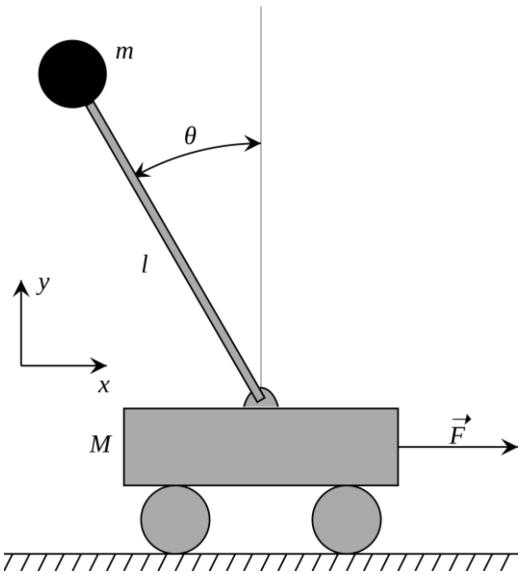
- Many types of supervision have been tried
- What other types of supervision “for free” can we use?
- What kind of data to use?
- How do we know if a certain supervision type would work?

Embodied learning

Reinforcement Learning



Cart-Pole Problem



Objective: Balance a pole on top of a movable cart

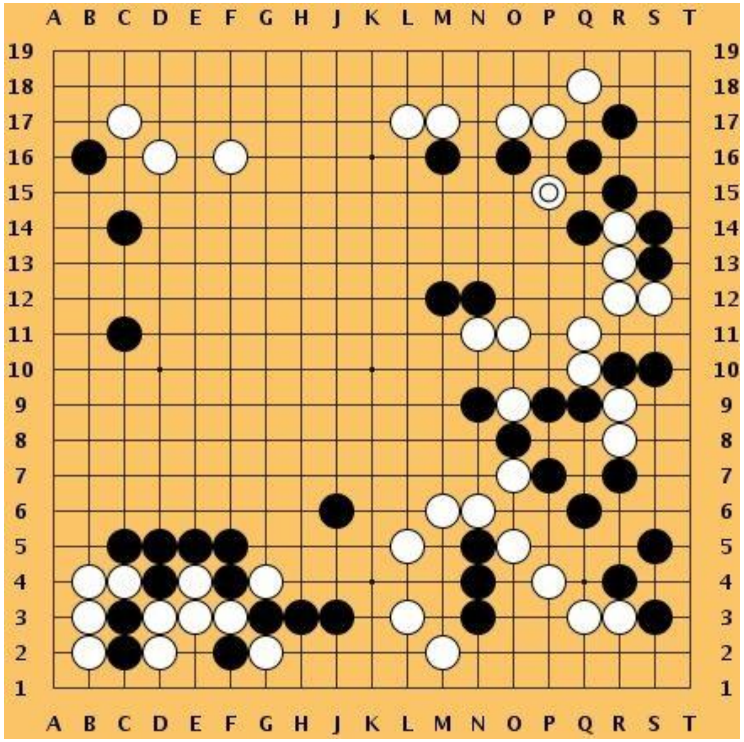
State: angle, angular speed, position, horizontal velocity
Action: horizontal force applied on the cart
Reward: 1 at each time step if the pole is upright

Atari Games

Objective: Complete the game with the highest score

State: Raw pixel inputs of the game state
Action: Game controls e.g. Left, Right, Up, Down
Reward: Score increase/decrease at each time step

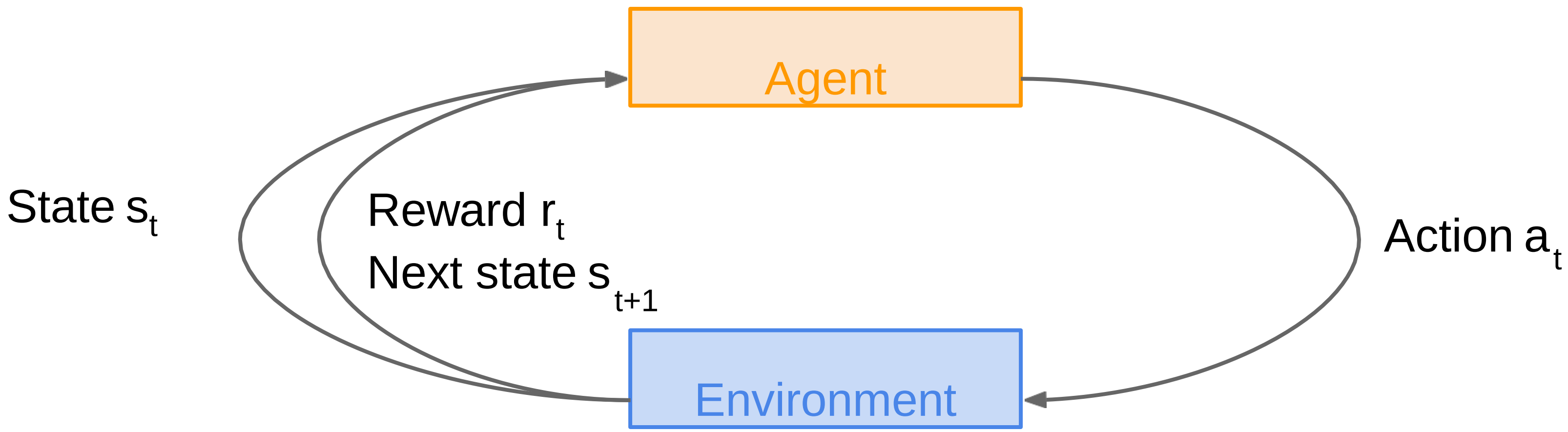
Go



Objective: Win the game!

State: Position of all pieces
Action: Where to put the next piece down
Reward: 1 if win at the end of the game, 0 otherwise

How can we mathematically formalize the RL problem?



Markov Decision Process

- Mathematical formulation of the RL problem
- **Markov property**: Current state completely characterises the state of the world

Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

Markov Decision Process

- At time step t=0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for t=0 until done:
  - Agent selects action a_t
  - Environment samples reward $r_t \sim R(. \mid s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim P(. \mid s_t, a_t)$
  - Agent receives reward r_t and next state s_{t+1}

- A policy u is a function from S to A that specifies what action to take in each state
- **Objective**: find policy u* that maximizes cumulative discounted reward: $\sum_{t \geq 0} \gamma^t r_t$

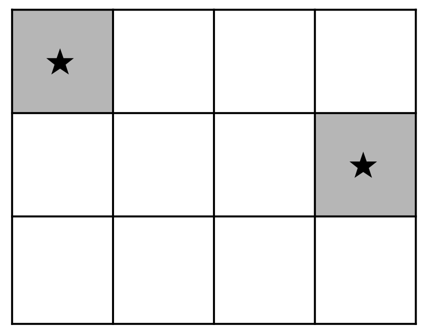
A simple MDP: Grid World

actions = {

1. right
2. left
3. up
4. down

}

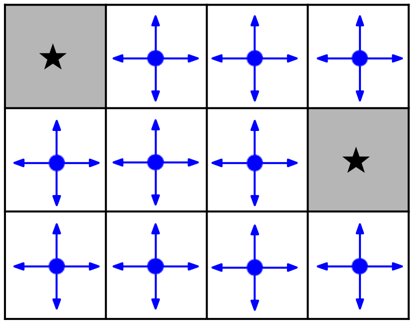
states



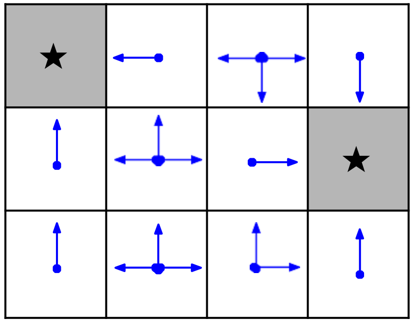
Set a negative "reward" for each transition (e.g. $r = -1$)

Objective: reach one of terminal states (greyed out) in least number of actions

A simple MDP: Grid World



Random Policy



Optimal Policy

The optimal policy u*

We want to find optimal policy u* that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?

The optimal policy u*

We want to find optimal policy u* that maximizes the sum of rewards.

How do we handle the randomness (initial state, transition probability…)?
Maximize the **expected sum of rewards!**

Formally: $\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right]$ with $s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)$

Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) s_0, a_0, r_0, s_1, a_1, r_1, …

How good is a state?

The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

How good is a state-action pair?

The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Bellman equation

The optimal Q-value function Q* is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Q* satisfies the following **Bellman equation**:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s, a\right]$$

Intuition: if the optimal state-action values for the next time-step Q*(s’,a’) are known, then the optimal strategy is to take the action that maximizes the expected value of
$r + \gamma Q^*(s', a')$

The optimal policy u* corresponds to taking the best action in any state as specified by Q*

Solving for the optimal policy: Q-learning

Q-learning: Use a function approximator to estimate the action-value function

$$Q(s, a; \theta) \approx Q^*(s, a)$$

function parameters (weights)

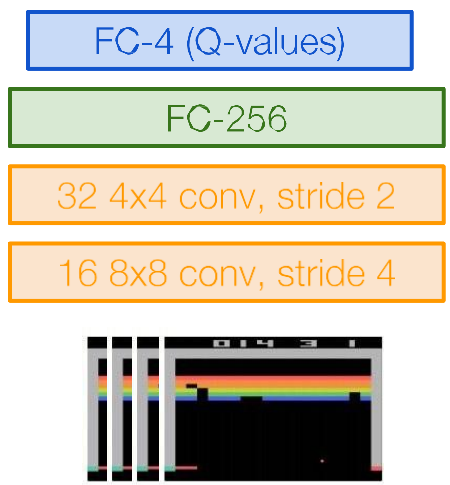
If the function approximator is a deep neural network => **deep q-learning**!

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Q-network Architecture

$Q(s, a; \theta)$: neural network with weights θ

A single feedforward pass to compute Q-values for all actions from the current state => efficient!



Last FC layer has 4-d output (if 4 actions), corresponding to $Q(s_t, a_1)$, $Q(s_t, a_2)$, $Q(s_t, a_3)$, $Q(s_t,a_4)$

Number of actions between 4-18 depending on Atari game

Current state s_t: 84x84x4 stack of last 4 frames
(after RGB->grayscale conversion, downsampling, and cropping)

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Initialize replay memory, Q-network

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do** ← Play M episodes (full games)
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Initialize state (starting game screen pixels) at the beginning of each episode

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

For each timestep t of the game

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

With small probability, select a random action (explore), otherwise select greedy action from current policy

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Take the action (a_t), and observe the reward r_t and next state s_{t+1}

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Store transition in replay memory

[Mnih et al. NIPS Workshop 2013; Nature 2015]

Putting it together: Deep Q-Learning with Experience Replay

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
    Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    for $t = 1, T$ **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
    end for
end for

Experience Replay:
Sample a random minibatch of transitions from replay memory and perform a gradient descent step

Policy Gradients

What is a problem with Q-learning?
The Q-function can be very complicated!

Example: a robot grasping an object has a very high-dimensional state => hard to learn exact value of every (state, action) pair

But the policy can be much simpler: just close your hand
Can we learn a policy directly, e.g. finding the best policy from a collection of policies?

Policy Gradients

Formally, let's define a class of parameterized policies: $\Pi = \{\pi_\theta, \theta \in \mathbb{R}^m\}$

For each policy, define its value:

$$J(\theta) = \mathbb{E}\left[\sum_{t \geq 0} \gamma^t r_t | \pi_\theta\right]$$

We want to find the optimal policy $\theta^* = \arg\max_\theta J(\theta)$

How can we do this?

Gradient ascent on policy parameters!

REINFORCE Algorithm (Williams 1992)

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta (a_t | s_t)$$

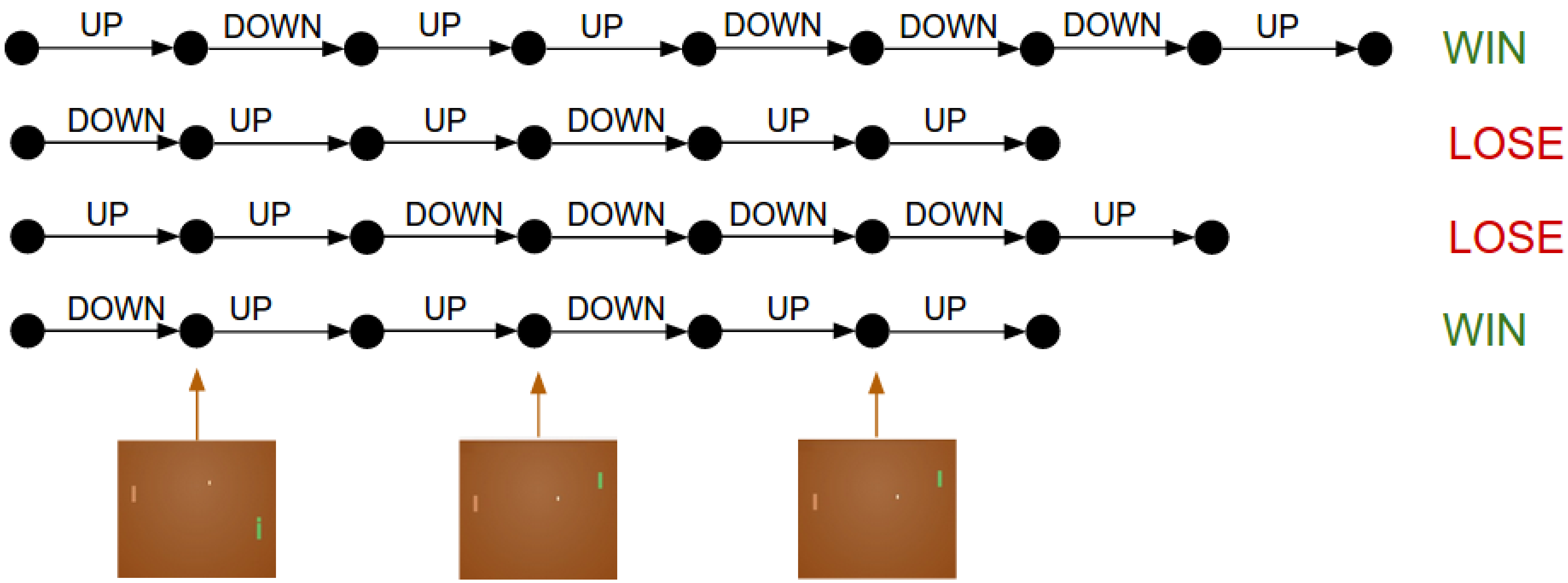
Interpretation:

- If r(τ) is high, push up the probabilities of the actions seen
- If r(τ) is low, push down the probabilities of the actions seen

Might seem simplistic to say that if a trajectory is good then all its actions were good. But in expectation, it averages out!

However, this also suffers from high variance because credit assignment is really hard. Can we help the estimator?

Policy Gradients



Andrej Karpathy

Variance Reduction

Gradient estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance Reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Second idea: Use discount factor γ to ignore delayed effects

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance Reduction: Baseline

Problem: The raw value of a trajectory isn't necessarily meaningful. For example, if rewards are all positive, you keep pushing up probabilities of actions.

What is important then? Whether a reward is better or worse than what you expect to get

Idea: Introduce a baseline function dependent on the state.
Concretely, estimator is now:

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} - b(s_t) \right) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

How to choose the baseline?

Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Intuitively, we are happy with an action a_t in a state s_t if $Q^\pi(s_t, a_t) - V^\pi(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Using this, we get the estimator:

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t | s_t)$$

Policy Gradients

- Objective: $\sum_i A_i \log p(y_i|x_i)$
- x_i = state
- y_i = sampled action
- A_i = “advantage” e.g. +1/-1 for win/lose in simplest version, or discounted, or improvement over “baseline”

Adapted from Andrej Karpathy

Policy Gradients vs Q-Learning

- Policy gradients suffers from high variance and instability; might want to make gradients smaller (e.g. relative to a baseline)
- Policy gradients can handle continuous action spaces (Gaussian policy)
- Estimating exact value of state-action pairs vs choosing what actions to take (value not important)
- Step-by-step (did I correctly estimate the reward at this time) vs delayed feedback (run policy and wait until game terminates)

Actor-Critic Algorithm

We can combine Policy Gradients and Q-learning by training both an **actor** (the policy) and a **critic** (the Q-function).

- The actor decides which action to take, and the critic tells the actor how good its action was and how it should adjust
- Also alleviates the task of the critic as it only has to learn the values of (state, action) pairs generated by the policy
- Can also incorporate Q-learning tricks e.g. experience replay
- **Remark:** we can define by the **advantage function** how much an action was better than expected $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$

Example Q Network

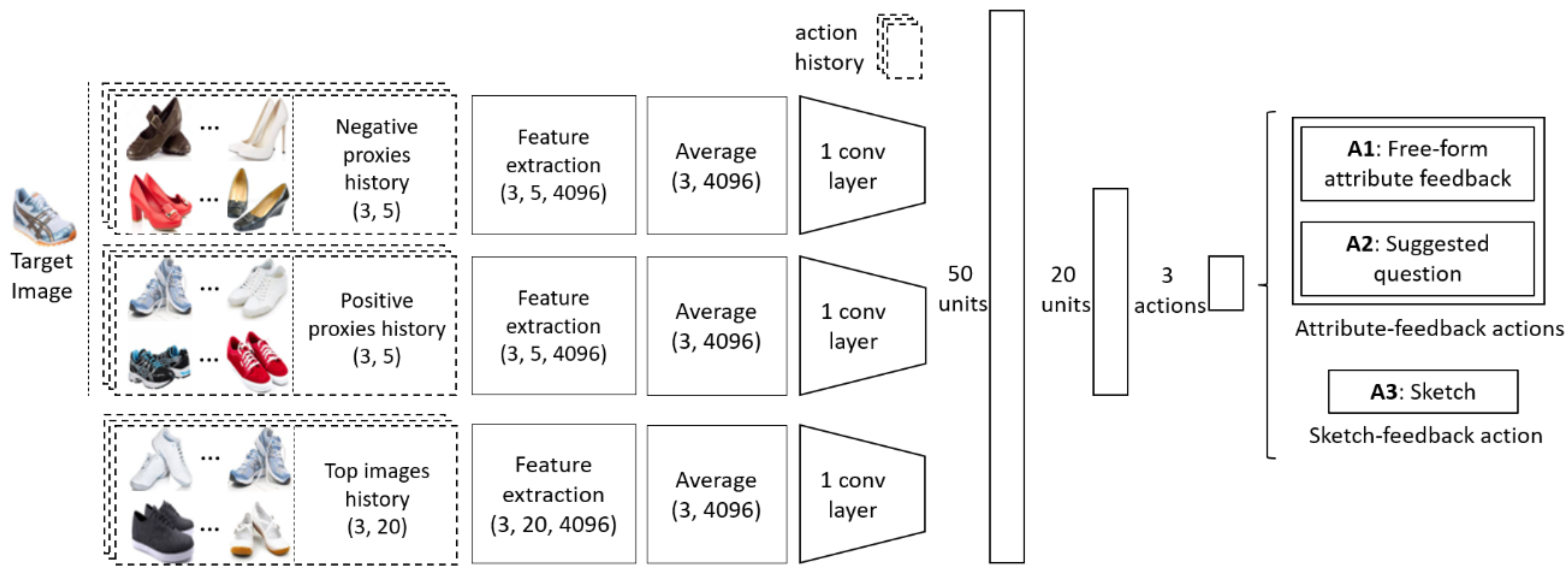


Figure 3: Architecture of our proposed Q-network. It receives histories of top-ranked images, positive and negative proxy images, and taken actions. It predicts the best action given a specific state. Inputs are denoted with dotted lines. Please see text for further explanation.

Murrugarra-Llerena and Kovashka, "Image Retrieval with Mixed Initiative and Multimodal Feedback", BMVC 2018

RL for navigation

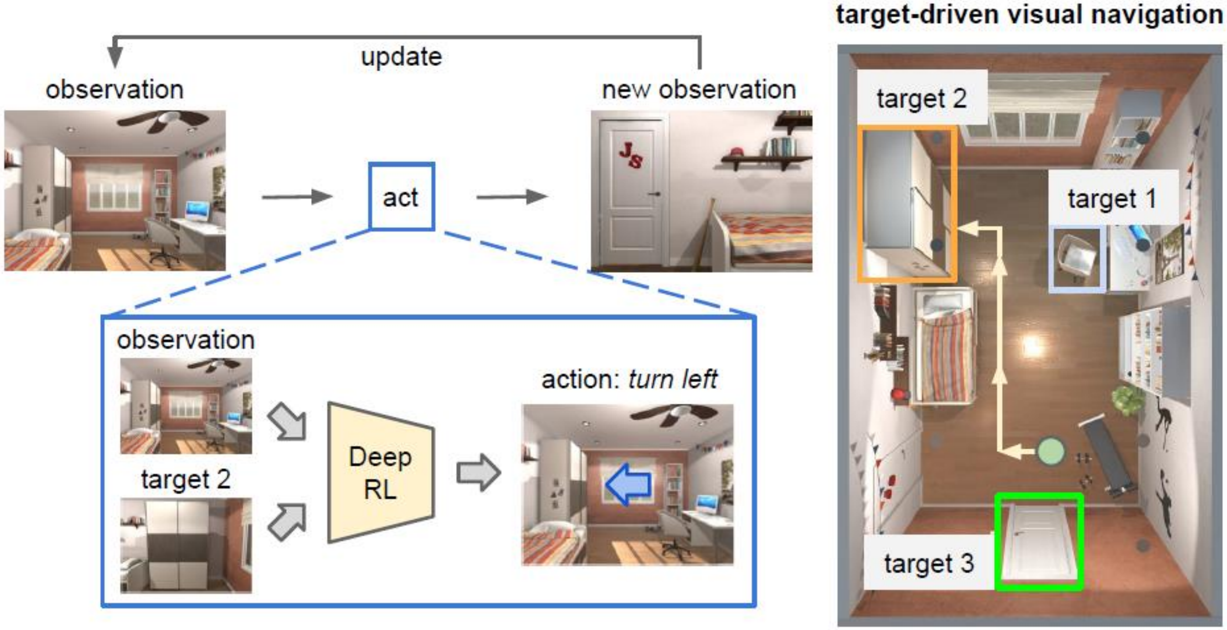


Fig. 1. The goal of our deep reinforcement learning model is to navigate towards a visual target with a minimum number of steps. Our model takes the current observation and the image of the target as input and generates an action in the 3D environment as the output. Our model learns to navigate to different targets in a scene without re-training.

Zhu et al., "Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning", ICRA 2017

RL for navigation

Figure 1: Our goal is to use scene priors to improve navigation in unseen scenes and towards novel objects. (a) There is no mug in the field of view of the agent, but the likely location for finding a mug is the cabinet near the coffee machine. (b) The agent has not seen a mango before, but it infers that the most likely location for finding a mango is the fridge since similar objects such as apple appear there as well. The most likely locations are shown with the orange box.

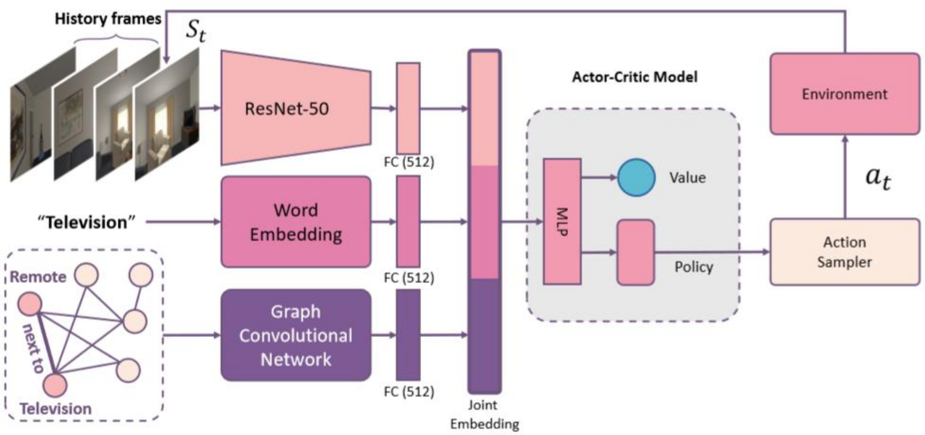


Figure 2: **Overview of the architecture.** Our model to incorporate semantic knowledge into semantic navigation. Specifically, we learn a policy network that decides an action based on the visual features of the current state, the semantic target category feature and the features extracted from the knowledge graph. We extract features from the parts of the knowledge graph that are activated.

Yang et al., "Visual Semantic Navigation using Scene Priors", ICLR 2019

RL for question-answering

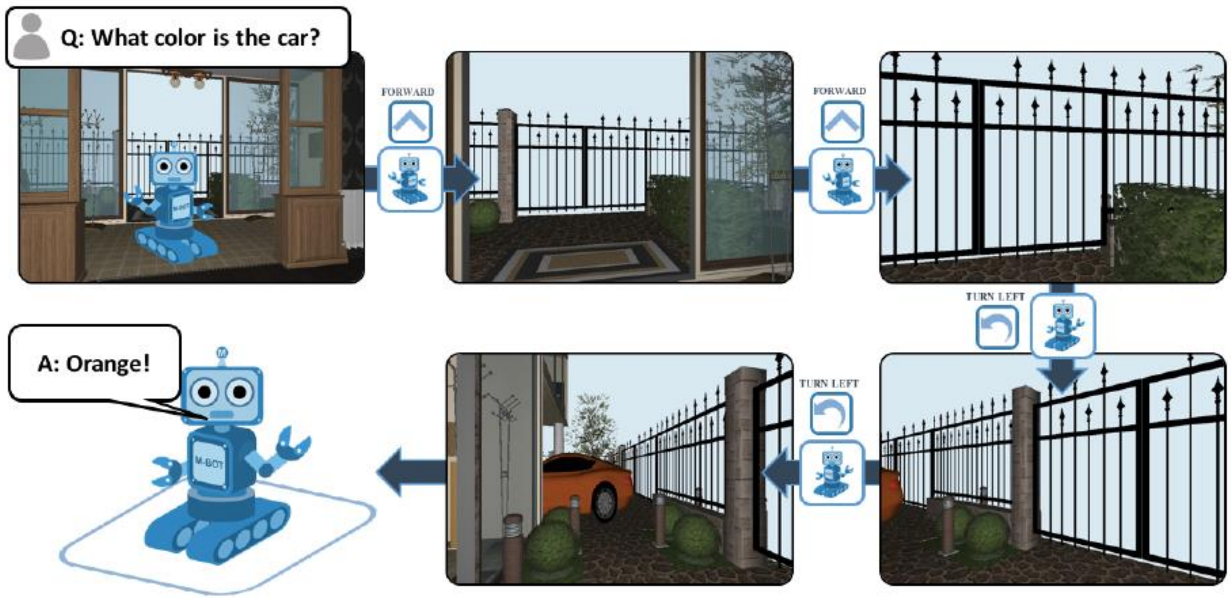


Figure 1: Embodied Question Answering – EmbodiedQA– tasks agents with navigating rich 3D environments in order to answer questions. These agents must jointly learn language understanding, visual reasoning, and goal-driven navigation to succeed.

Das et al., "Embodied Question Answering", CVPR 2018

RL for question-answering

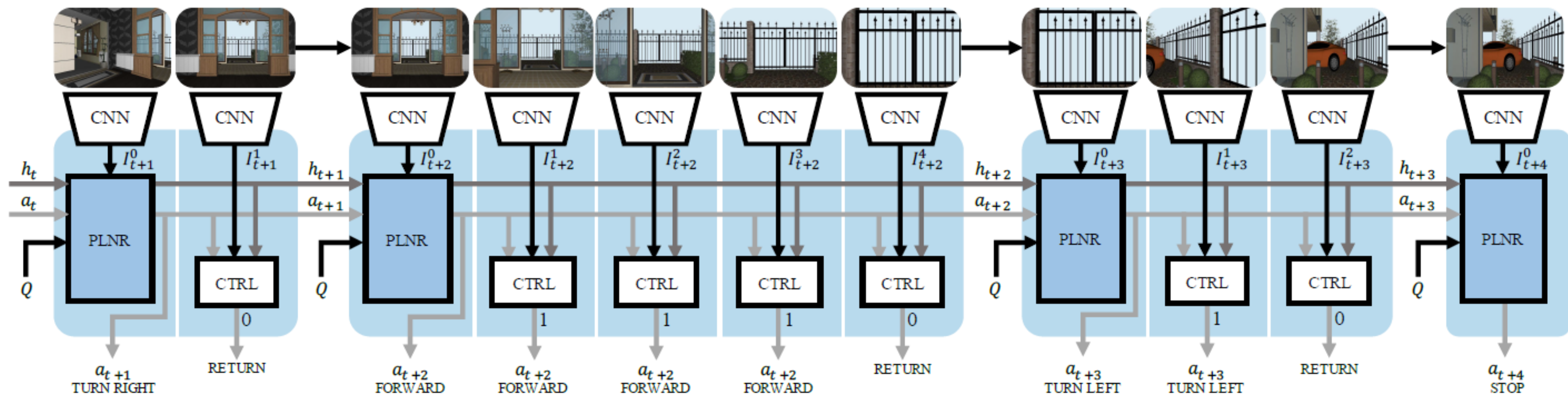
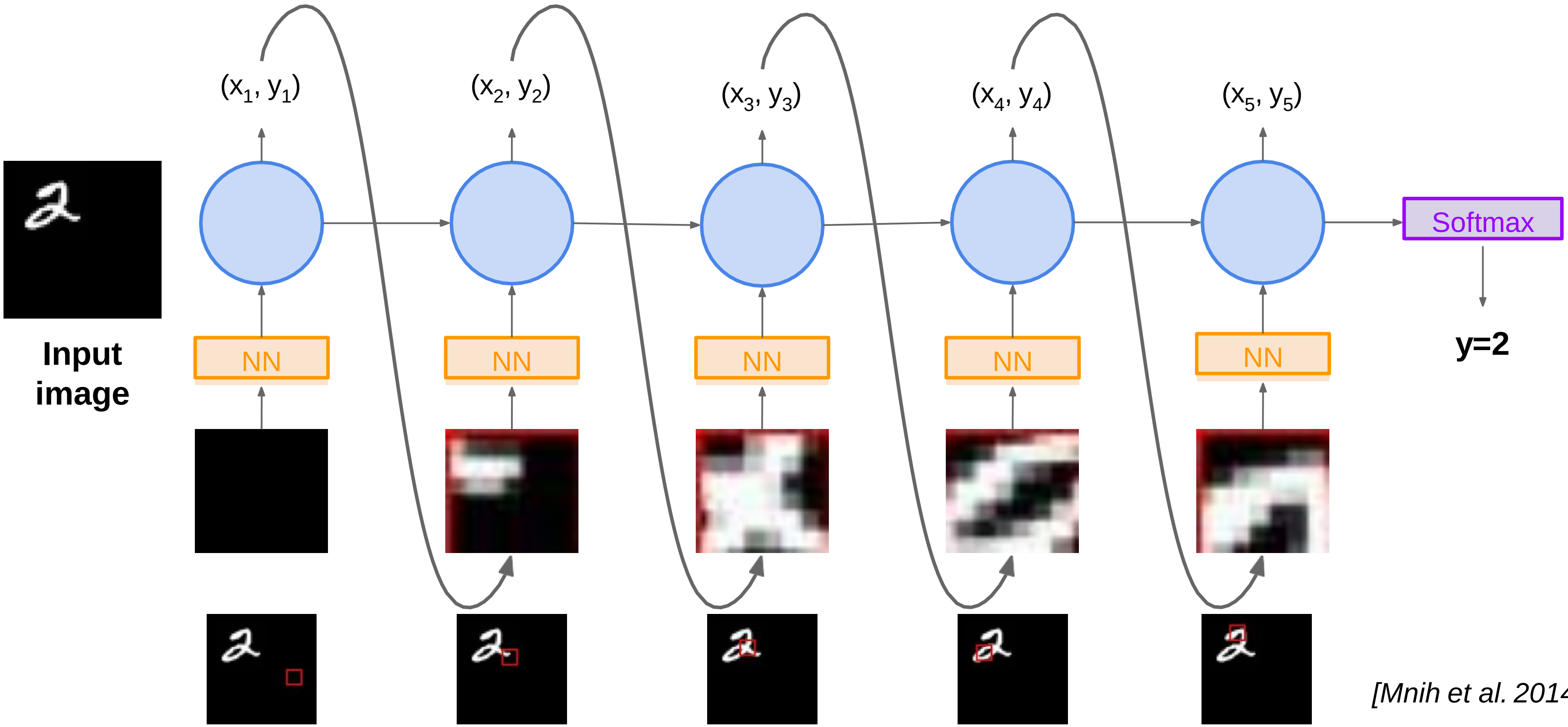


Figure 4: Our PACMAN navigator decomposes navigation into a planner and a controller. The planner selects actions and the controller executes these actions a variable number of times. This enables the planner to operate on shorter timescales, strengthening gradient flows.

Das et al., “Embodied Question Answering”, CVPR 2018

Recurrent Attention Model



[Mnih et al. 2014]

RL for object detection

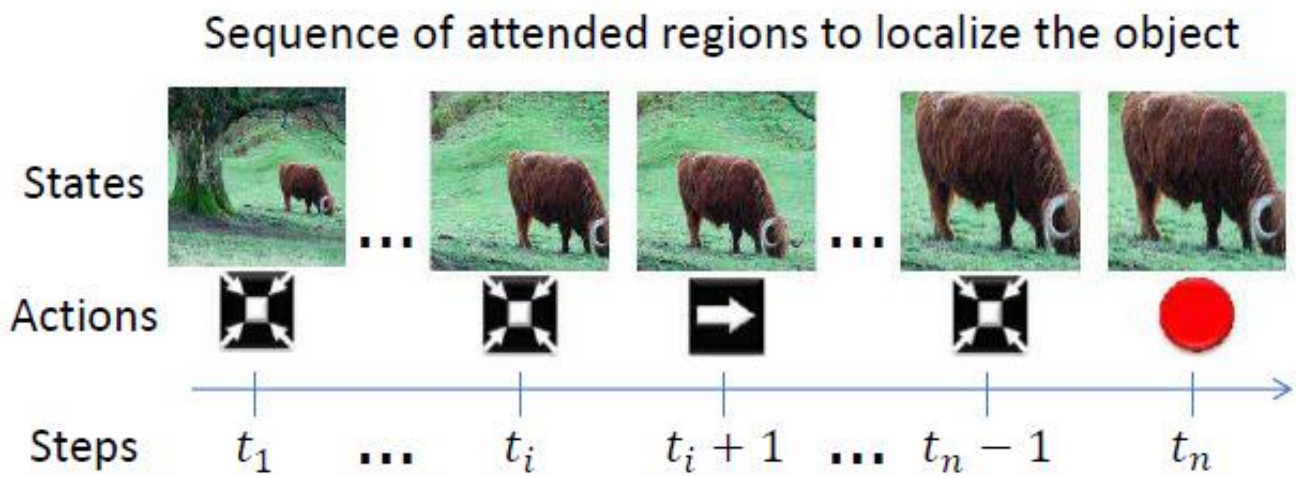


Figure 1. A sequence of actions taken by the proposed algorithm to localize a cow. The algorithm attends regions and decides how to transform the bounding box to progressively localize the object.

Caicedo and Lazebnik, "Active Object Localization with Deep Reinforcement Learning", ICCV 2015

RL for object detection

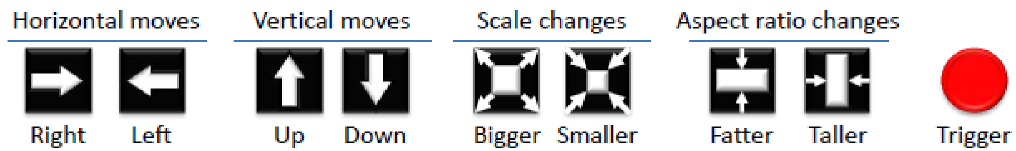
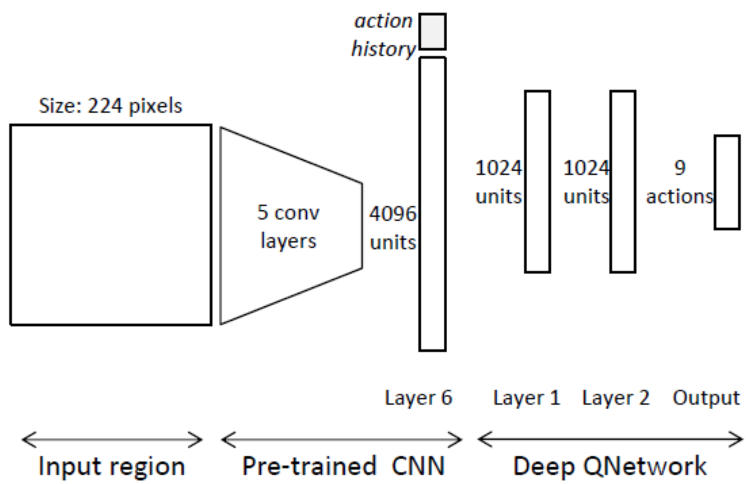


Figure 2. Illustration of the actions in the proposed MDP, giving 4 degrees of freedom to the agent for transforming boxes.

$$R_a(s, s') = sign\left(IoU(b', g) - IoU(b, g)\right)$$

$$R_\omega(s, s') = \begin{cases} +\eta & \text{if } IoU(b, g) \geq \tau \\ -\eta & \text{otherwise} \end{cases}$$



Caicedo and Lazebnik, "Active Object Localization with Deep Reinforcement Learning", ICCV 2015