Self-supervised learning and self-training

Samuel Cheng
Image credits: Ishan Misra, Amit Chaudhary

University of Oklahoma

April 3, 2024

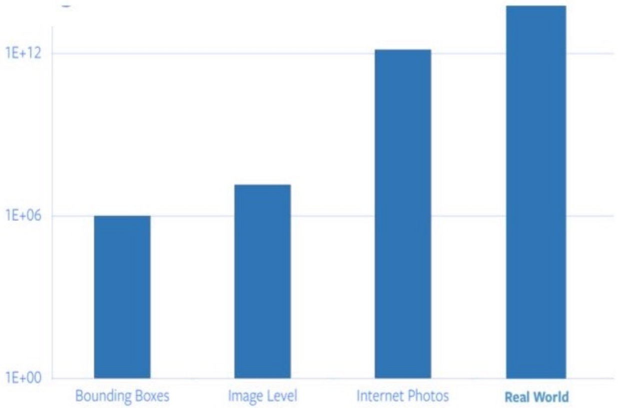
Self-Supervised Learning vs. Other Techniques

- Supervised learning: requires labeled data for training
- Unsupervised learning: learns representations without using any labels
- Semi-supervised learning: uses a mixture of labeled and unlabeled data for training
- Self-supervised learning: a subcategory of unsupervised learning that creates surrogate tasks by using the data itself as a source of supervision
    - A useful technique to achieve semi-supervised learning

Applications

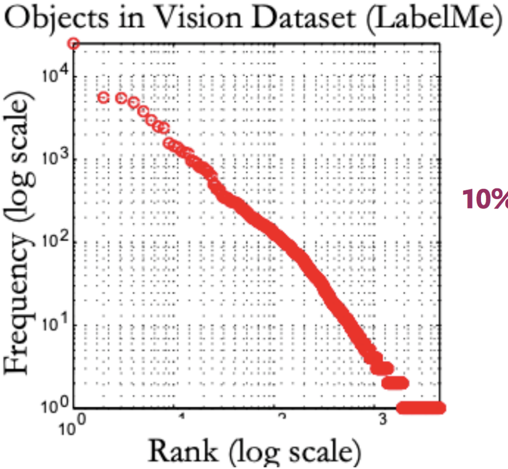
- Computer vision: image inpainting, object detection, segmentation.
- Natural language processing: language modeling, machine translation, sentiment analysis.
- Speech recognition: speaker identification, emotion recognition, speech enhancement.

Why self-supervised learning?



Variation in available data quantum basis annotation complexity

Why self-supervised learning?



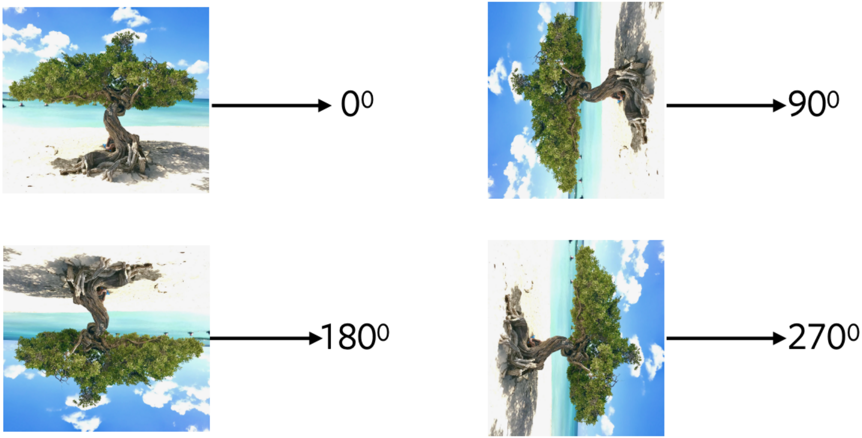
10% of the classes account for 93% of the data

Pretext tasks vs downstream tasks

- Pretext task: Auxiliary task to learn representations
- Examples:
  - Image rotation task
  - Image inpainting
  - Relative position task
  - Jigsaw puzzle solving
  - Contrastive learning

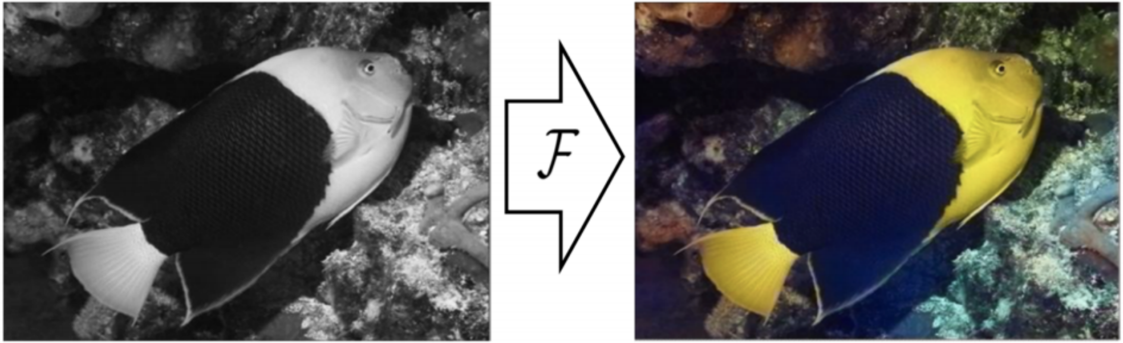
- Downstream task: Main task where learned representations are utilized
- Examples:
  - Image classification
  - Object detection
  - Semantic segmentation

Image rotation task

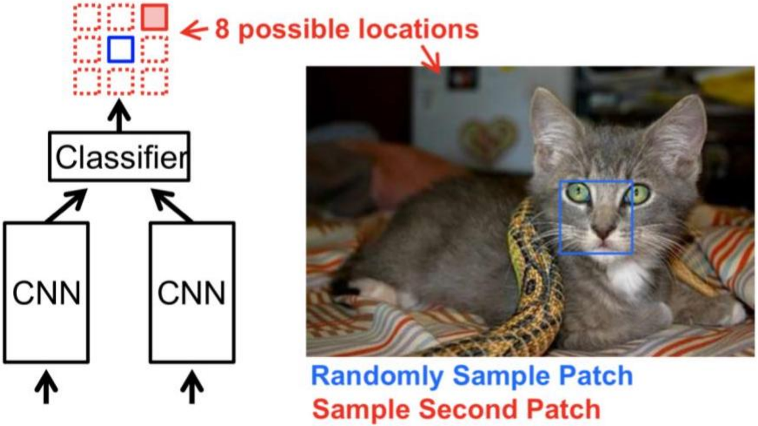


Gidaris et al., 2018, Predicting Image Rotations

Coloring task

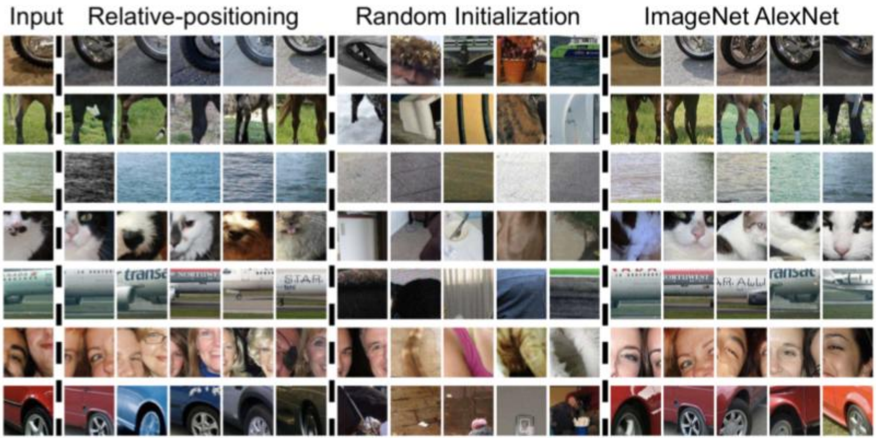


Relative position task



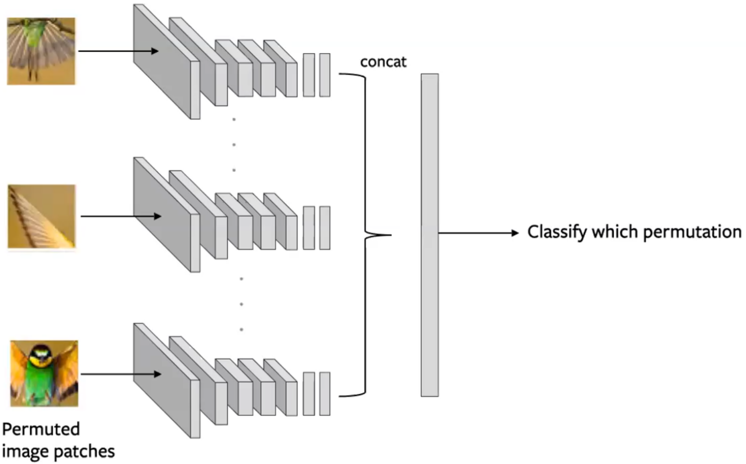
Pretext task for predicting relative position of patches

Evaluation with clustering

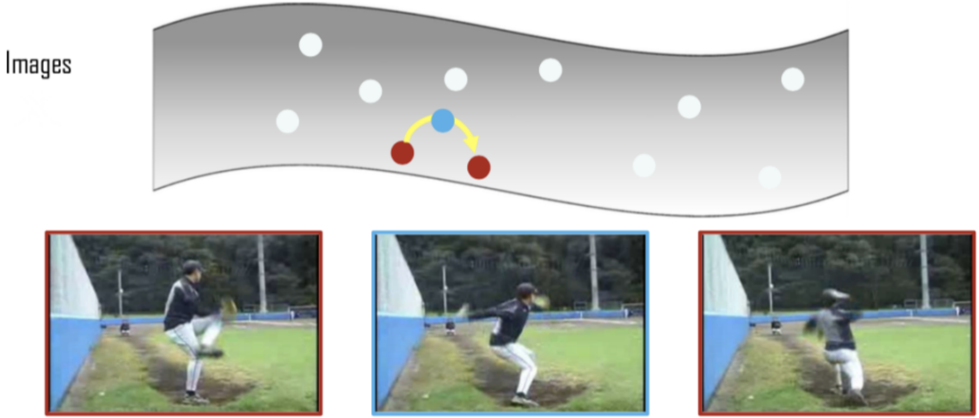


Comparison of nearest neighbours for relative positioning with ImageNet pretrained and random initialized networks

Jigsaw task

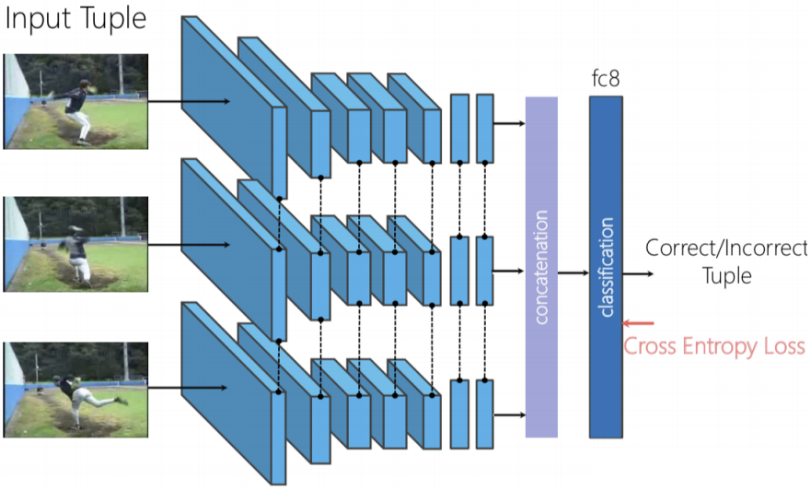

concat
Classify which permutation
Permuted
image patches

Interpolation task for videos

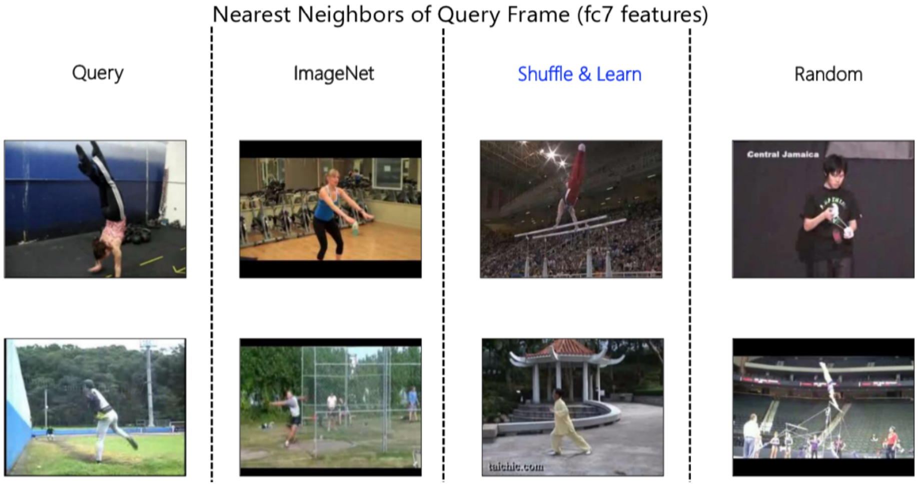


Given a start and an end, can this point lie in between?

“Shuffle & learn”



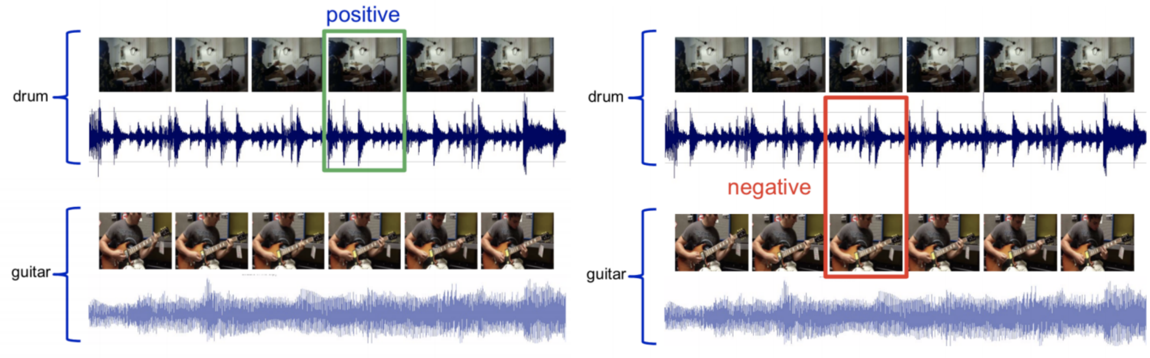
Return query with nearest neighbor output



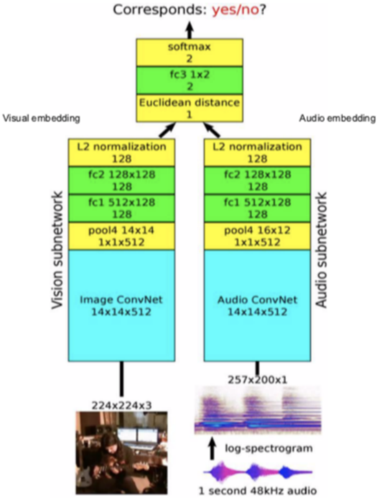
Great initialization for human pose estimation

Initialization (AlexNet)	End task	
	FLIC Dataset Keypoints AUC	MPII Dataset Keypoints AUC
ImageNet Supervised	**51.3**	47.2
Shuffle and Learn (Self-supervised)	49.6	**47.6**

Video + audio pretask



Video + audio pretask



Some Considerations of Pretext Tasks

- Pretext tasks vary in difficulty and predictive power
  - Relative position: easy, simple classification
  - Masking and fill-in: harder, better representation
  - Contrastive methods: more information, beyond pretext tasks
- A single pretext task may not be enough to learn self-supervised representations
  - How do we train multiple pre-training tasks?

Swapping Final Layers for Different Pretext Tasks

- Final fully-connected layer can be swapped based on batch type
- Example:
  - Feed a batch of black-and-white images, produce colored images
  - Switch final layer, feed a batch of patches, predict relative position
- How much should we train on a pretext task?

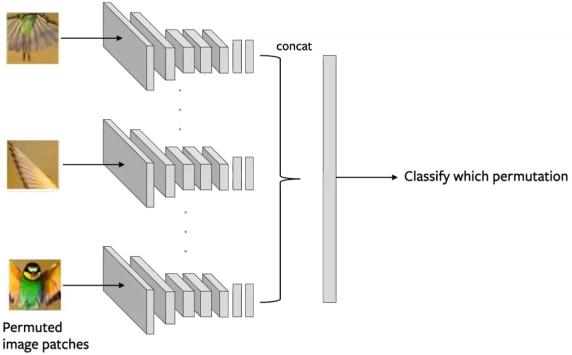
Multiple pretext tasks can boost performance

Initialization (ResNet101)	End task	
	ImageNet top-5 accuracy	VOC07 Detection mAP
Relative Position	59.2	66.8
Colorization	62.5	65.5
Relative Position + Colorization (Multi-task)	66.6	68.8

Some Considerations for Training Pretext Tasks

- Rule of thumb: Choose a difficult pretext task or multiple pretext tasks that improves the downstream task
- In development, train pretext tasks as part of the entire pipeline
- In practice, one usually do not re-train later
  - Retraining may lead to overfitting as downstream task does not necessary align with the pretext tasks

Scaling SSL: jigsaw puzzles



- Use a subset of permutations (e.g., from 9!, use 100)
- N-way ConvNet uses shared parameters
- Problem complexity depends on subset size
- Can perform better on downstream tasks than supervised methods

Evaluation: Fine-tuning vs. Linear Classifier

- Good representation should transfer with little training ⇒ Transfer learning evaluation
  - Fine-tuning: Use entire network as initialization, update all weights
  - Linear Classifier: Train a small linear classifier on top of pretext network
- How each layer is doing as a representation for the downstream task?

What does each layer learn?

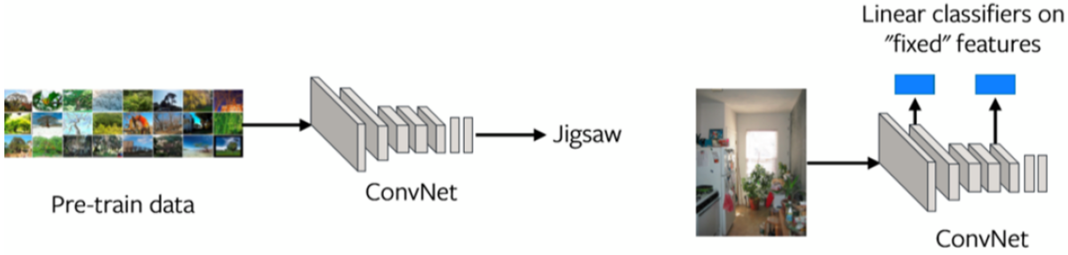


Fig. 1: Feature representations at each layer

What does each layer learn?

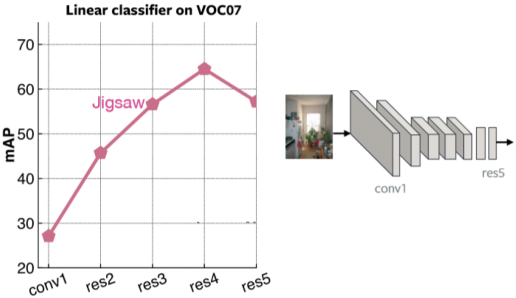
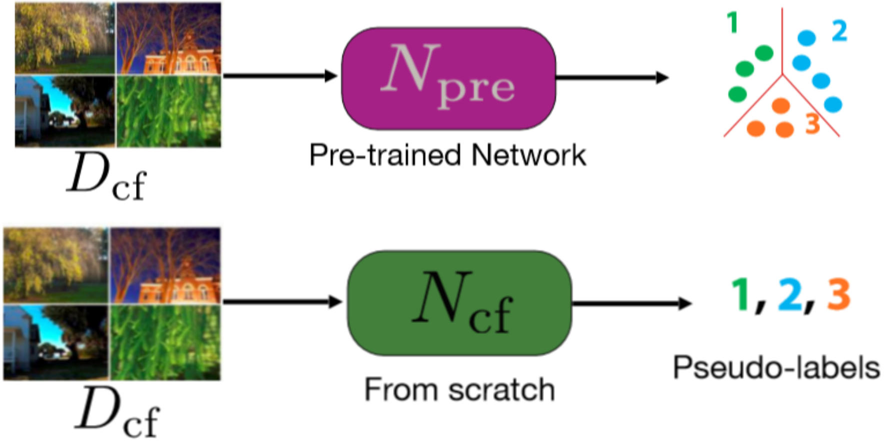


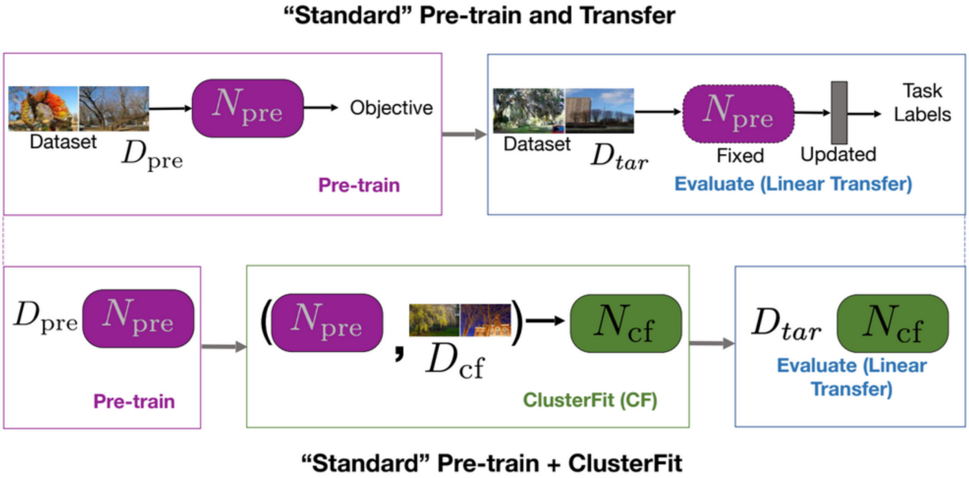
Fig. 2: Performance of Jigsaw based on each layer

- Deeper layers: Higher mAP on downstream tasks
- Final layer: Sharp drop in mAP, overly specialized
  - Contrasts with supervised networks: mAP generally increases with depth
  - Pretext task not well-aligned to downstream task

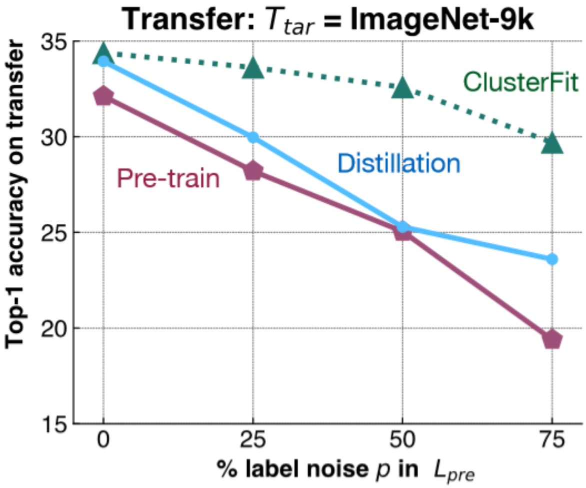
ClusterFit


N_{pre}
Pre-trained Network
D_{cf}
1
2
3
N_{cf}
From scratch
D_{cf}
1, 2, 3
Pseudo-labels

ClusterFit vs "standard" transfer learning

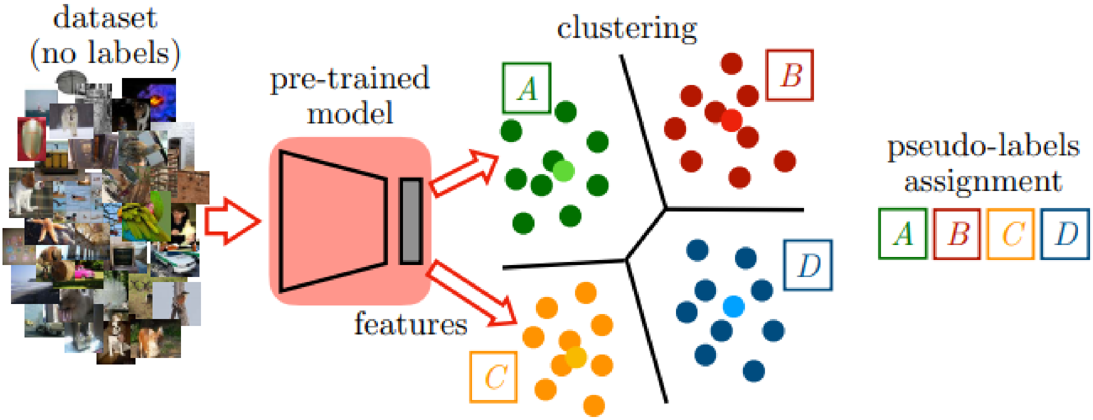


Gain over distillation when there is label noise



	Transfer Dataset			
Method	**ImageNet-1M**	**VOC07**	**Places205**	**iNaturalist**
Jigsaw Pretrain	46.0	66.1	39.9	22.1
Pretrain 2x	45.1	65.4	38.7	21.8
ClusterFit	**55.2** +9	**69.5** +3	**45.0** +5	**29.8** +7

Boosting self-supervised learning via knowledge transfer

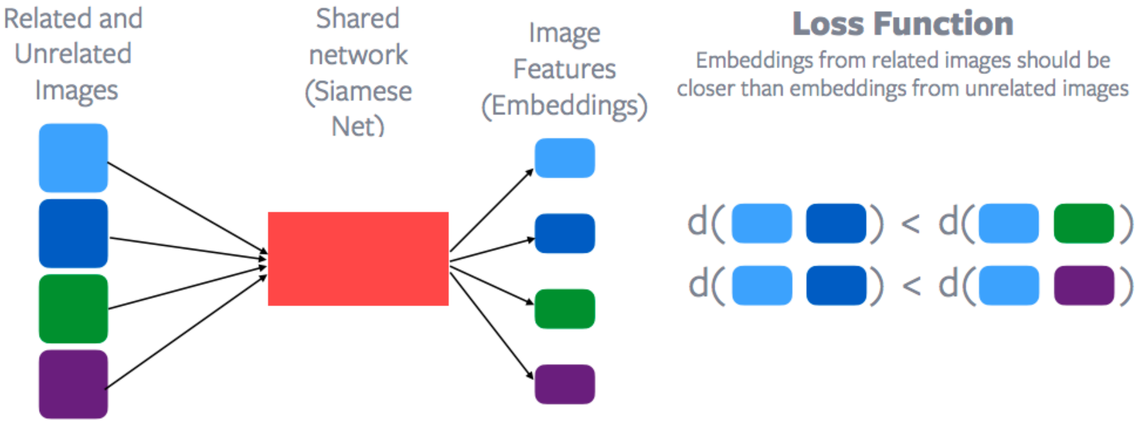


Boosting self-supervised learning via knowledge transfer

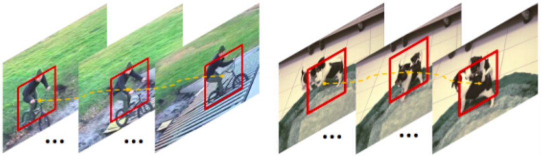
Task	Clustering	Pre-text architecture	Downstream arch.	Classification	Detection(SS)	Detection(MS)	Segmentation
Jigsaw	no	AlexNet	AlexNet	67.7	53.2	-	-
Jigsaw++	no	AlexNet	AlexNet	69.8	55.5	55.7	38.1
Jigsaw++	yes	AlexNet	AlexNet	69.9	55.0	55.8	40.0
Jigsaw++	yes	VGG-16	AlexNet	**72.5**	**56.5**	**57.2**	**42.6**

#clusters	**500**	**1000**	**2000**	**5000**	**10000**
mAP on voc-classification	69.1	69.5	69.9	69.9	70.0

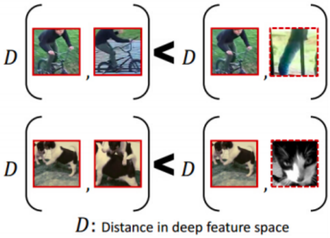
Contrastive learning



Video example

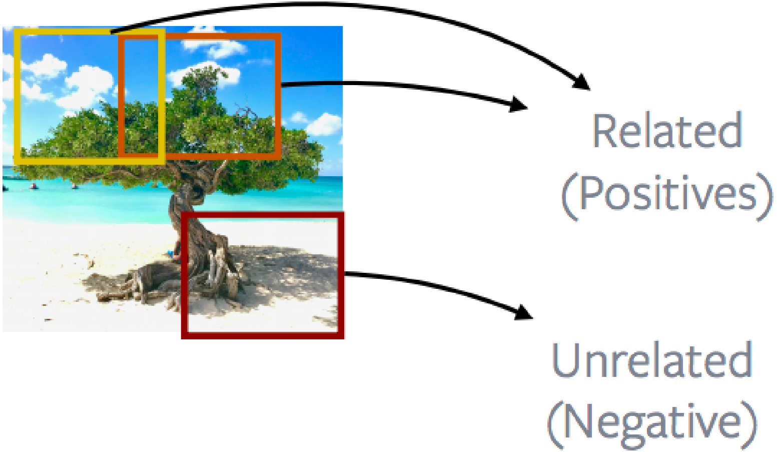


(a) Unsupervised Tracking in Videos

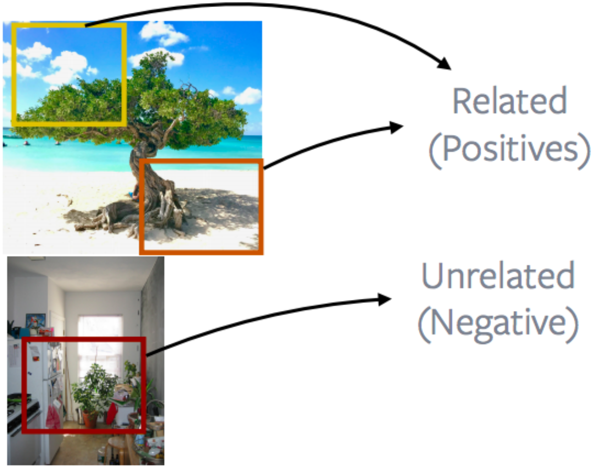


(c) Ranking Objective

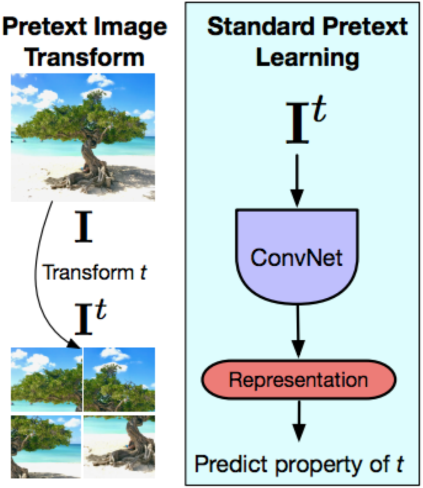
Nearby patches vs. distant patches of an Image



Patches of an image vs. patches of other images



Pretext Image Transform and Standard Pretext Learning

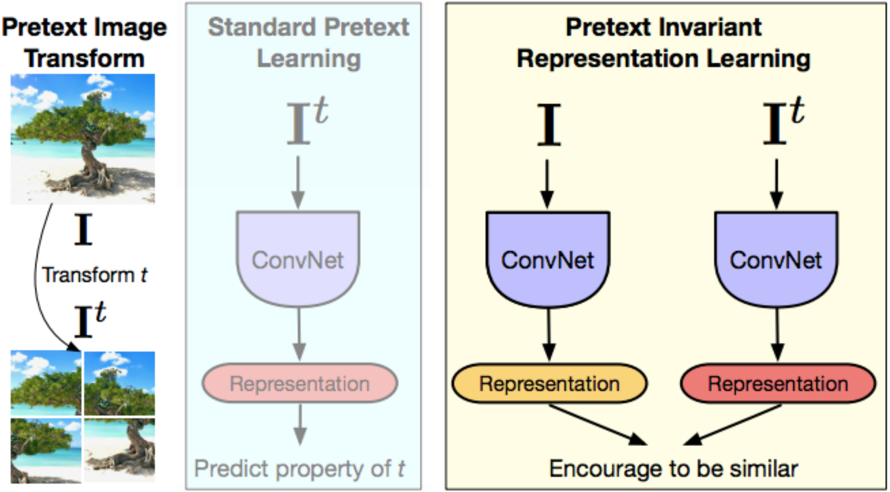


Consider the pretext task of predicting the property of a transform

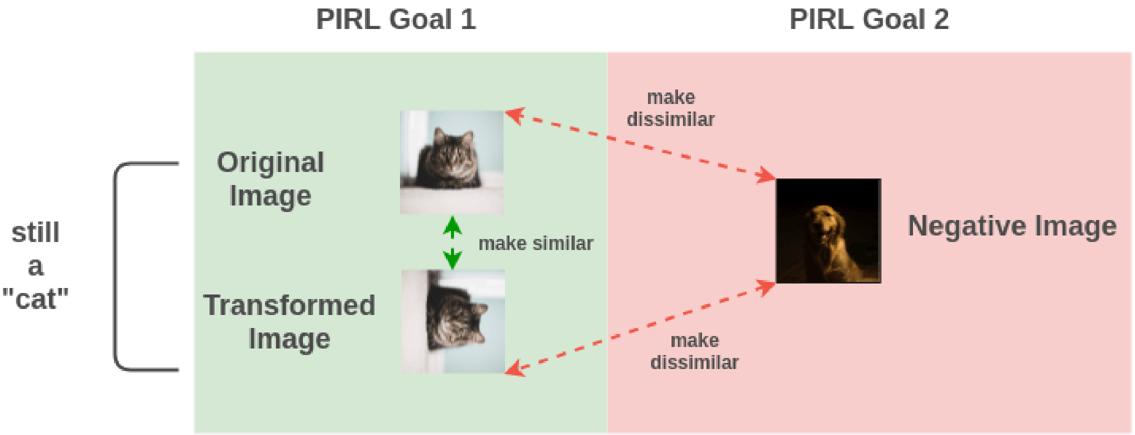
- The pretext task always reasons about a single image
- Pretext task captures some property of the transform
  - One usually wants representations to be invariant to transform
  - Representation from pretext task does the exact opposite thing

.

Pretext Invariant Representation Learning (PIRL)

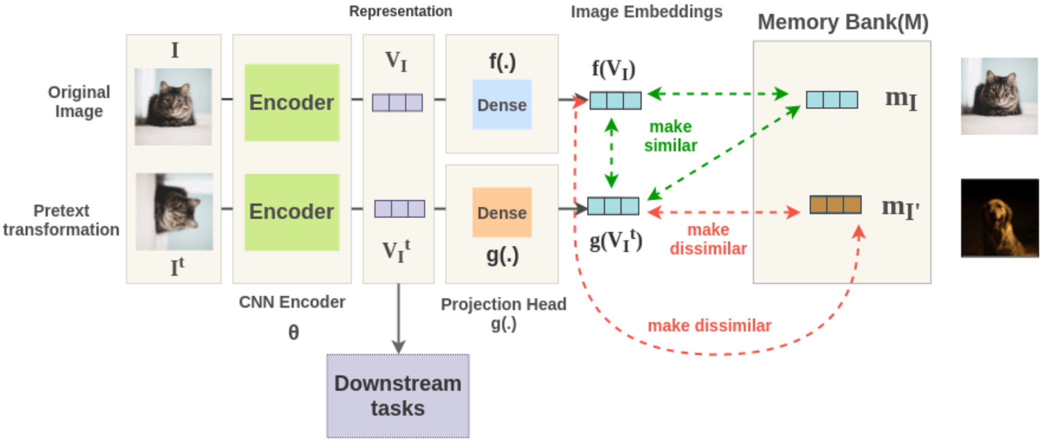


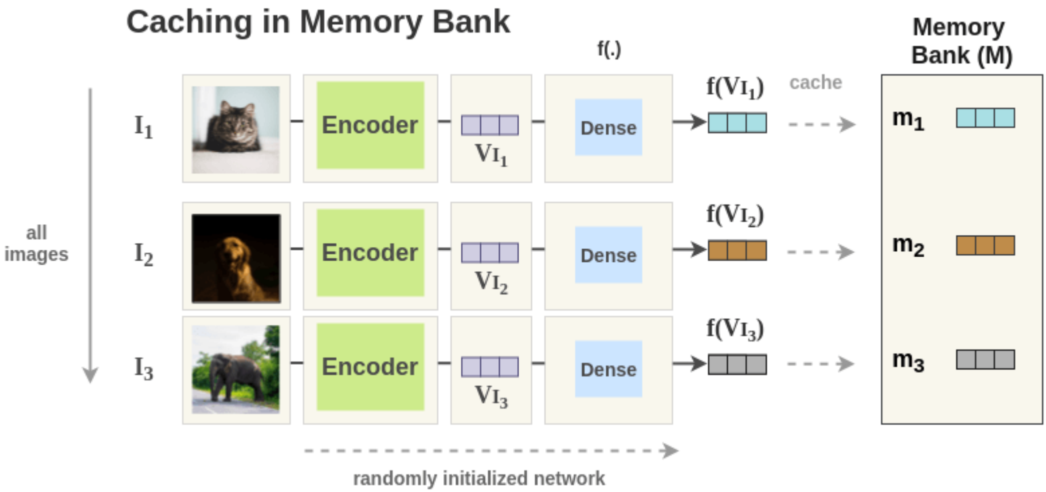
PIRL Goals



PIRL Goal 1
PIRL Goal 2
still a "cat"
Original Image
Transformed Image
make similar
make dissimilar
make dissimilar
Negative Image

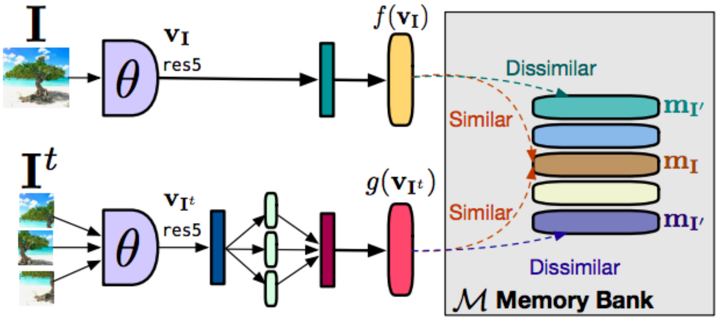
PIRL Generic Framework





Caching in Memory Bank
f(.)
Memory Bank (M)
all images
I1
I2
I3
Encoder
Encoder
Encoder
VI1
VI2
VI3
Dense
Dense
Dense
f(VI1)
f(VI2)
f(VI3)
cache
m1
m2
m3
randomly initialized network

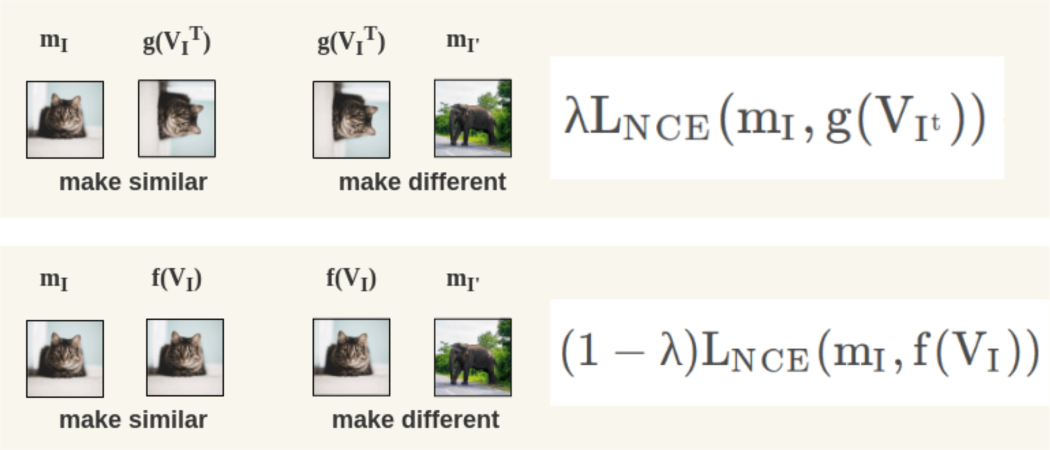
Net loss



$$L_{\text{NCE}}(g(\mathbf{v}_{\mathbf{I}^t}), \mathbf{m}_\mathbf{I}) + L_{\text{NCE}}(f(\mathbf{v}_\mathbf{I}), \mathbf{m}_\mathbf{I})$$

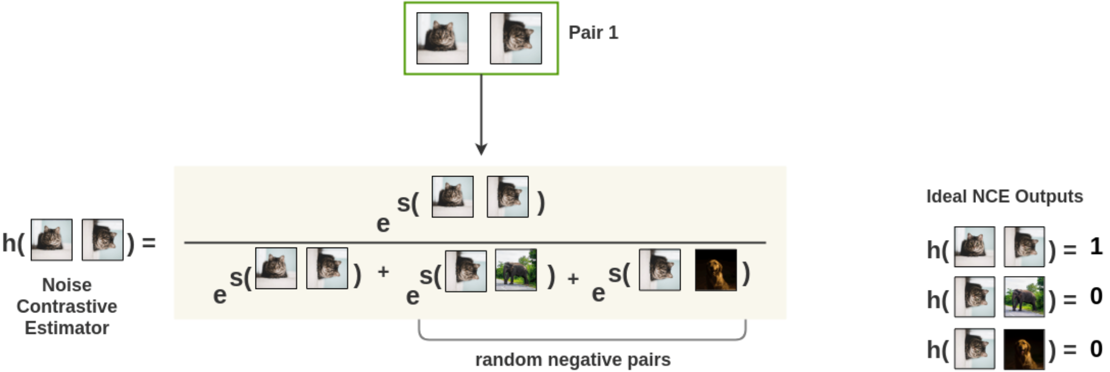
Rep. $\mathbf{I}^t$ and $\mathbf{m}_\mathbf{I}$ should be similar   Rep. $\mathbf{I}$ and $\mathbf{m}_\mathbf{I}$ should be similar

Net loss



* m: from memory bank

Noise contrastive estimator loss

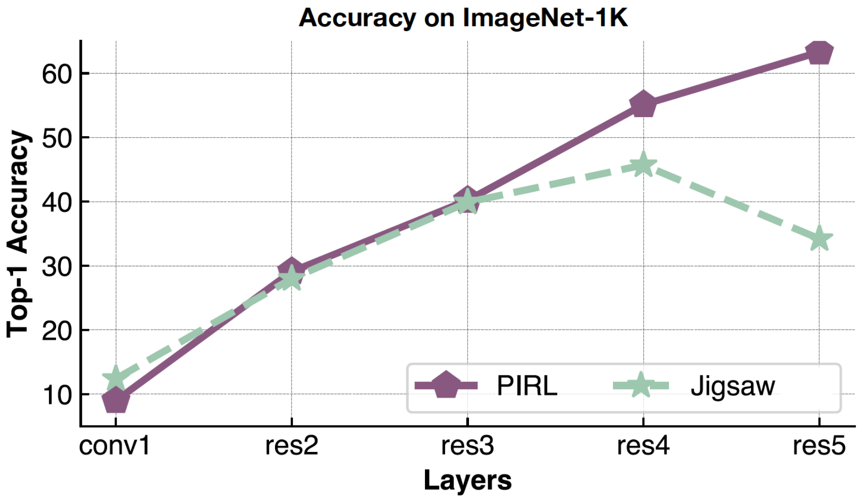


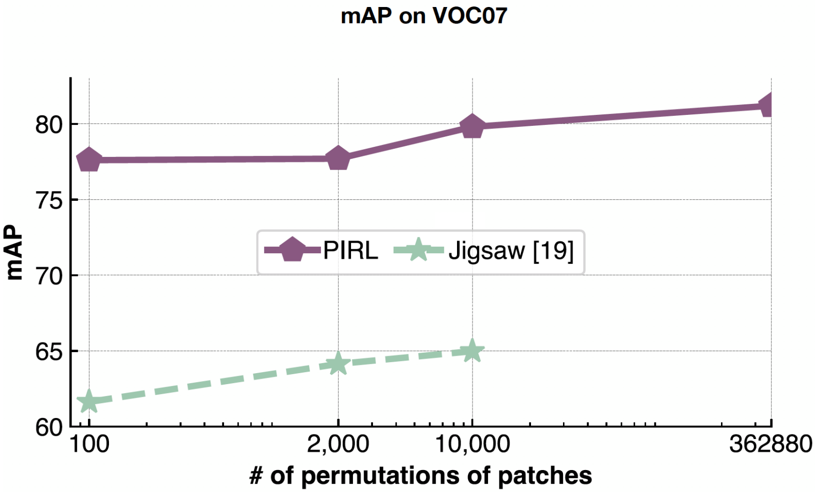
$$L_{\mathrm{NCE}}(f_1, f_2) = -\log[h(f_1, f_2)] - \sum_{f'} \log[1 - h(f', f_2)]$$

Initialization	VOC07+12			VOC07		
	AP^{all}	AP^{50}	AP^{75}	AP^{all}	AP^{50}	AP^{75}
ImageNet Supervised	52.6	**81.1**	57.4	43.8	**74.5**	45.9
PIRL	**54.0**	80.7	**59.7**	**44.7**	73.4	**47.0**

+1.4 +2.3 +1.1

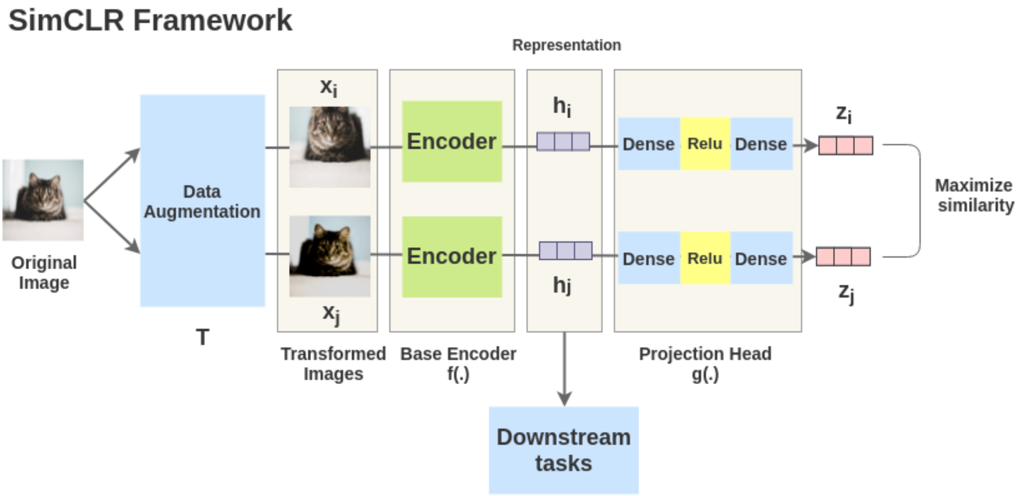
Method	# Pretrain Images	Evaluation	
		ImageNet	Places-205
DeeperCluster (Caron et al., 2019)	100M	45.6	42.1
Jigsaw (Goyal et al., 2019)	100M	48.3	44.8
PIRL	1M $\frac{1}{100}$	**57.8** +9	**51.0** +6

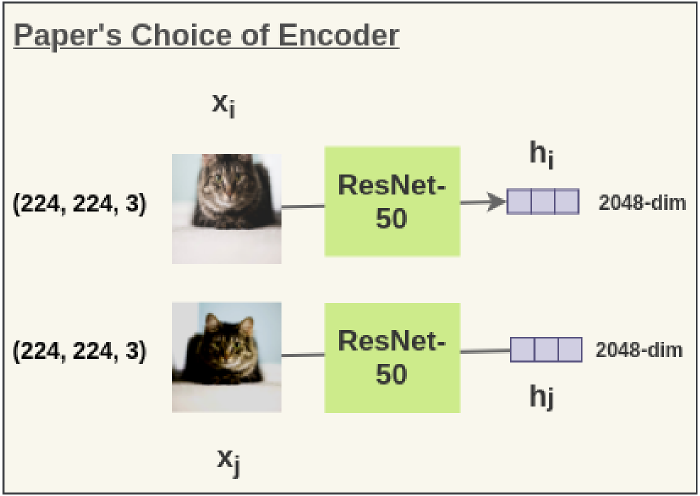

Accuracy on ImageNet-1K
Top-1 Accuracy
60
50
40
30
20
10
conv1
res2
res3
res4
res5
Layers
PIRL
Jigsaw


mAP on VOC07
mAP
80
75
70
65
60
PIRL
Jigsaw [19]
100
2,000
10,000
362880
# of permutations of patches

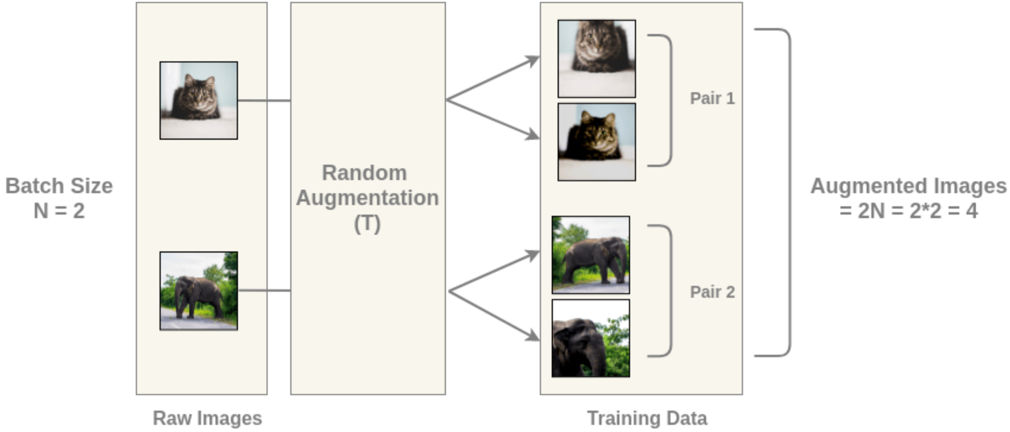
	Transfer Dataset			
Method	**ImageNet-1M**	**VOC07**	**Places205**	**iNaturalist**
Jigsaw	46.0	66.1	41.4	22.1
Rotation	48.9	63.9	47.6	23
PIRL (Rot)	60.2	77.1	47.6	31.2
PIRL (Jigsaw + Rot)	**63.1**	**80.3**	**49.7**	**33.6**

SimCLR

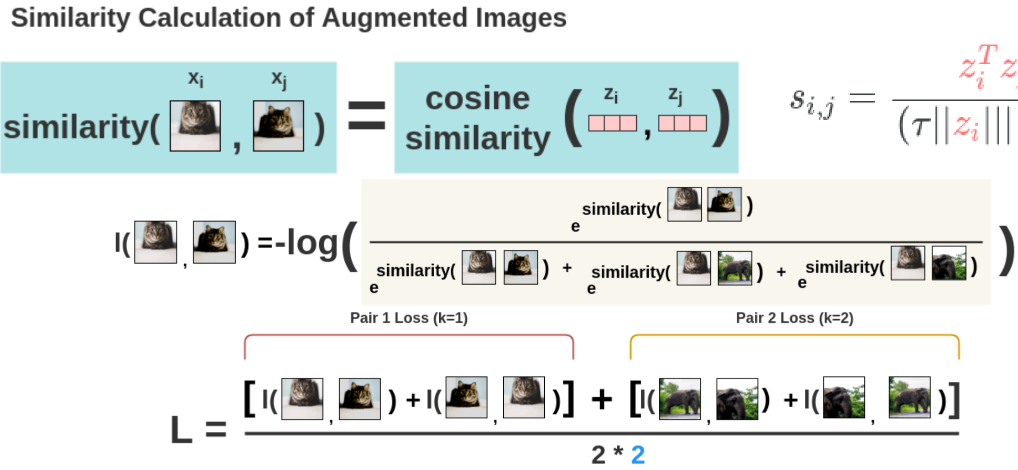

SimCLR Framework
Representation
x_i
h_i
z_i
Original Image
Data Augmentation
Encoder
Encoder
Dense
Relu
Dense
Dense
Relu
Dense
Maximize similarity
h_j
x_j
z_j
T
Transformed Images
Base Encoder f(.)
Projection Head g(.)
Downstream tasks


Paper's Choice of Encoder
x_i
(224, 224, 3)
ResNet-50
h_i
2048-dim
(224, 224, 3)
ResNet-50
2048-dim
hj
x_j

Preparing similar pairs in a batch

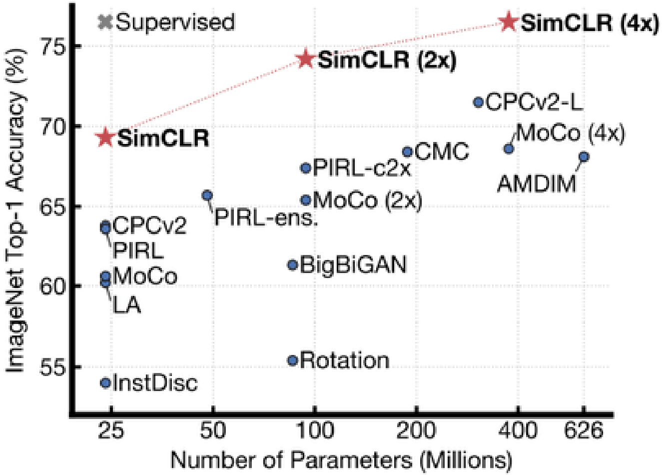


Loss



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

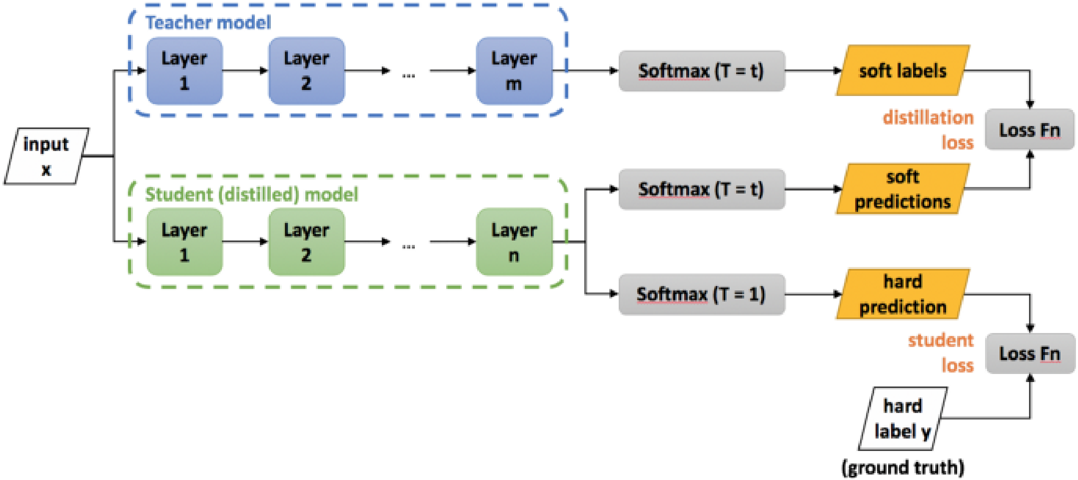
Result


Supervised
SimCLR (4x)
SimCLR (2x)
SimCLR
CPCv2-L
MoCo (4x)
CMC
PIRL-c2x
AMDIM
MoCo (2x)
PIRL-ens.
CPCv2
PIRL
BigBiGAN
MoCo
LA
Rotation
InstDisc
ImageNet Top-1 Accuracy (%)
75
70
65
60
55
25
50
100
200
400
626
Number of Parameters (Millions)

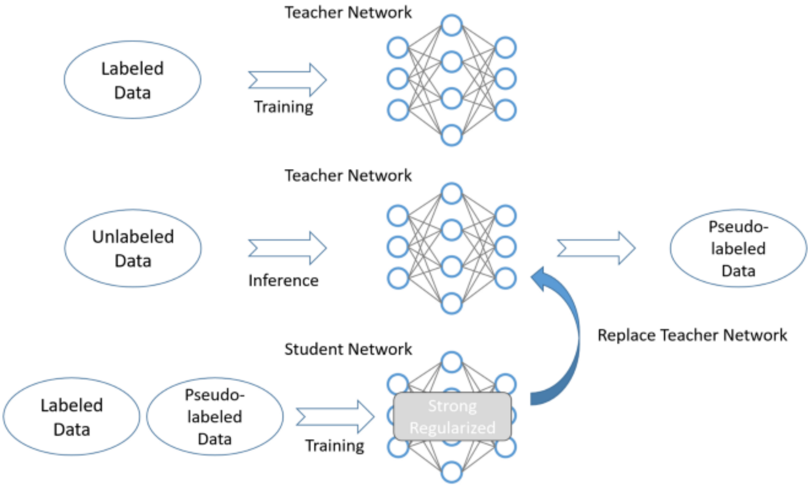
Self-training

- An effective technique to achieve semi-supervised learning (train a model with a mixture of labeled and unlabeled data)
    - A pre-trained model will learn from labeled data
    - Use the pre-trained model to label the unlabeled data
    - Refine the model with additional data
- Unlike self-supervised learning, no surrogate/pretext task is involved

Recall: Knowledge Distillation



Self-training with Noisy Student improves ImageNet classification



Difference from standard self-training

- Adding noise to training procedure to prevent student model from simply memorizing the training data
- Noise can be injected into the student model in two ways
  - input noise: data augmentation with RandAugment is used
  - model noise: dropout and stochastic depth are used. Dropout randomly drops out some of the neurons during training to prevent overfitting, while stochastic depth randomly skips some of the layers during training to improve generalization
- Unlike typical self-training cycle, iteratively increases the student model's capacity

Result

Method	# Params	Extra Data	Top-1 Acc.	Top-5 Acc.
ResNet-50 [30]	26M	-	76.0%	93.0%
ResNet-152 [30]	60M	-	77.8%	93.8%
DenseNet-264 [36]	34M	-	77.9%	93.9%
Inception-v3 [81]	24M	-	78.8%	94.4%
Xception [15]	23M	-	79.0%	94.5%
Inception-v4 [79]	48M	-	80.0%	95.0%
Inception-resnet-v2 [79]	56M	-	80.1%	95.1%
ResNeXt-101 [92]	84M	-	80.9%	95.6%
PolyNet [100]	92M	-	81.3%	95.8%
SENet [35]	146M	-	82.7%	96.2%
NASNet-A [104]	89M	-	82.7%	96.2%
AmoebaNet-A [65]	87M	-	82.8%	96.1%
PNASNet [50]	86M	-	82.9%	96.2%
AmoebaNet-C [17]	155M	-	83.5%	96.5%
GPipe [38]	557M	-	84.3%	97.0%
EfficientNet-B7 [83]	66M	-	85.0%	97.2%
EfficientNet-L2 [83]	480M	-	85.5%	97.5%
ResNet-50 Billion-scale [93]	26M	3.5B images labeled with tags	81.2%	96.0%
ResNeXt-101 Billion-scale [93]	193M		84.8%	-
ResNeXt-101 WSL [55]	829M		85.4%	97.6%
FixRes ResNeXt-101 WSL [86]	829M		86.4%	98.0%
Big Transfer (BiT-L) [43]†	928M	300M weakly labeled images from JFT	87.5%	98.5%
Noisy Student Training (EfficientNet-L2)	480M	300M unlabeled images from JFT	**88.4%**	**98.7%**

Table 2: Top-1 and Top-5 Accuracy of Noisy Student Training and previous state-of-the-art methods on ImageNet. EfficientNet-L2 with Noisy Student Training is the result of iterative training for multiple iterations by putting back the student model as the new teacher. It has better tradeoff in terms of accuracy and model size compared to previous state-of-the-art models. †: Big Transfer is a concurrent work that performs transfer learning from the JFT dataset.

Noisy student increases robustness

Method	Top-1 Acc.	Top-5 Acc.
ResNet-101 [32]	4.7%	-
ResNeXt-101 [32] (32x4d)	5.9%	-
ResNet-152 [32]	6.1%	-
ResNeXt-101 [32] (64x4d)	7.3%	-
DPN-98 [32]	9.4%	-
ResNeXt-101+SE [32] (32x4d)	14.2%	-
ResNeXt-101 WSL [55, 59]	61.0%	-
EfficientNet-L2	49.6%	78.6%
Noisy Student Training (L2)	**83.7%**	**95.2%**

Table 3: Robustness results on ImageNet-A.

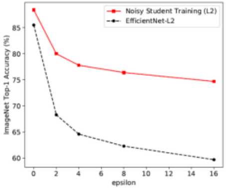


Figure 4: Noisy Student Training improves adversarial robustness against an FGSM attack though the model is not optimized for adversarial robustness. The accuracy is improved by 11% at $\epsilon = 2$ and gets better as ϵ gets larger.

Method	Res.	Top-1 Acc.	mCE
ResNet-50 [31]	224	39.0%	76.7
SIN [23]	224	45.2%	69.3
Patch Gaussian [51]	299	52.3%	60.4
ResNeXt-101 WSL [55, 59]	224	-	45.7
EfficientNet-L2	224	62.6%	47.5
Noisy Student Training (L2)	224	76.5%	30.0
EfficientNet-L2	299	66.6%	42.5
Noisy Student Training (L2)	299	**77.8%**	**28.3**

Table 4: Robustness results on ImageNet-C. mCE is the weighted average of error rate on different corruptions, with AlexNet's error rate as a baseline (lower is better).

Method	Res.	Top-1 Acc.	mFR
ResNet-50 [31]	224	-	58.0
Low Pass Filter Pooling [99]	224	-	51.2
ResNeXt-101 WSL [55, 59]	224	-	27.8
EfficientNet-L2	224	80.4%	27.2
Noisy Student Training (L2)	224	85.2%	14.2
EfficientNet-L2	299	81.6%	23.7
Noisy Student Training (L2)	299	**86.4%**	**12.2**

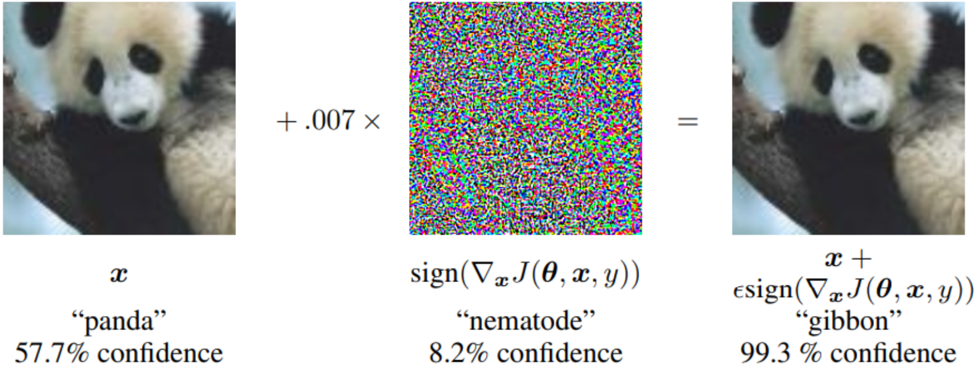
Table 5: Robustness results on ImageNet-P, where images are generated with a sequence of perturbations. mFR measures the model's probability of flipping predictions under perturbations with AlexNet as a baseline (lower is better).

- ImageNet-A: naturally adversarial examples
- ImageNet-C: Corruption from noise, blur, pixelate (intentional downsampling + upsampling)
- ImageNet-P: Perturbations (distortion from motion)
- FGSM: Fast gradient sign method

$$X_{Adversarial} = X + \varepsilon \, . sign(\nabla_X J(X, Y)),$$

Explaining and harnessing adversarial examples (FGSM)

$$X_{Adversarial} = X + \varepsilon \, . sign\left(\nabla_X J(X, Y)\right),$$



Revisiting Knowledge Distillation via Label Smoothing Regularization

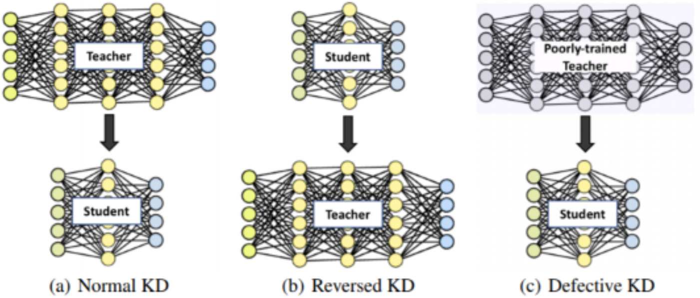


Figure 1: (a) Normal KD framework. (b)(c) Diagrams of exploratory experiments we conduct.

Table 2. Re-KD experiment results (accuracy, mean±std over 3 runs in %) on CIFAR10.

Teacher: baseline	Student: baseline	Normal KD (T→S)	Re-KD (S→T)
ResNet18: 95.12	Plain CNN: 87.14	87.67±0.17 (**+0.53**)	95.33±0.12 (**+0.21**)
	MobileNetV2: 90.98	91.69±0.14 (**+0.71**)	95.71±0.11 (**+0.59**)
MobileNetV2: 90.98	Plain CNN: 87.14	87.45±0.18 (**+0.31**)	91.81±0.23 (**+0.92**)
ResNeXt29: 95.76	ResNet18: 95.12	95.80±0.13 (**+0.68**)	96.49±0.15 (**+0.73**)

Table 3. Re-KD experiment results (accuracy, in %) on Tiny-ImageNet.

Teacher: baseline	Student: baseline	Normal KD (T→S)	Re-KD (S→T)
ResNet18: 63.44	MobileNetV2: 55.06	56.70 (**+1.64**)	64.12 (**+0.68**)
	ShuffleNetV2: 60.51	61.19 (**+0.68**)	64.35 (**+0.91**)
ResNet50: 67.47	MobileNetV2: 55.06	56.02 (**+0.96**)	67.68 (**+0.21**)
	ShuffleNetV2: 60.51	60.79 (**+0.28**)	67.62 (**+0.15**)
	ResNet18: 63.44	64.23 (**+0.79**)	67.89 (**+0.42**)

Table 4. De-KD accuracy (in %) on two datasets. Pt-Teacher is "Poorly-trained Teacher".

Dataset	Pt-Teacher: baseline	Student: baseline	De-KD
CIFAR100	ResNet18: 15.48	MobileNetV2: 68.38	70.65±0.35 (**+2.27**)
		ShuffleNetV2: 70.34	71.82±0.11 (**+1.48**)
	ResNet50: 45.82	MobileNetV2: 68.38	71.45±0.23 (**+3.09**)
		ShuffleNetV2: 70.34	72.11±0.09 (**+1.77**)
		ResNet18: 75.87	77.23±0.11 (**+1.23**)
	ResNeXt29: 51.94	MobileNetV2: 68.38	71.52±0.27 (**+3.14**)
		ShuffleNetV2:70.34	72.26±0.36 (**+1.92**)
		ResNet18: 75.87	77.28±0.17 (**+1.41**)
Tiny-ImageNet	ResNet18: 9.41	MobileNetV2: 55.06	56.22 (**+1.16**)
		ShuffleNetV2: 60.51	60.66 (**+0.15**)
	ResNet50: 31.01	MobileNetV2:55.06	56.02 (**+0.96**)
		ShuffleNetV2: 60.51	61.09 (**+0.58**)

Rethinking Pre-training and Self-training

This work experiments self-training and compare with pre-training

- Dataset:
  - Pretext: ImageNet/OpenImage
  - Target: COCO
- Control parameters:
  - Data augmentation:
    - Augment-S1: flip and scale jitter
    - Augment-S2: AutoAugment + Augment-S1
    - Augment-S3: Large scale jittering + Augment-S2
    - Augment-S4: Use RandAugment rather than AutoAugment in Augment-S3
  - Pre-training: Rand Init (no-pretraining) $<$ ImageNet Init $<$ ImageNet++ Init (better checkpoint)
  - Target domain labeled data quantity: 20%, 50%, 100%

Observation 1

Pretraining can be counterproductive when target domain data $\gg 0$ and augmentation $\gg 0$

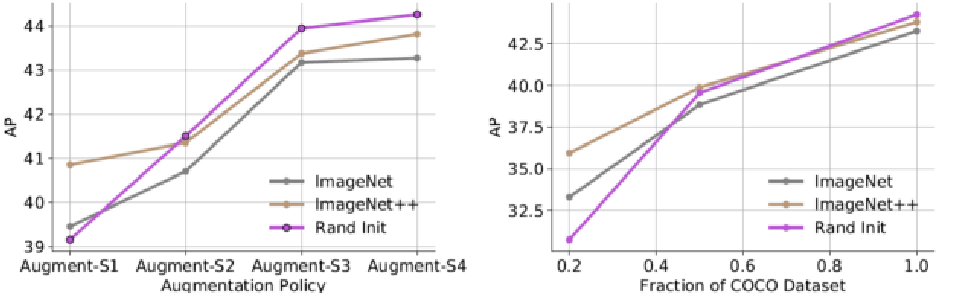


Figure 1: The effects of data augmentation and dataset size on pre-training. **Left**: Supervised object detection performance under various ImageNet pre-trained checkpoint qualities and data augmentation strengths on COCO. **Right**: Supervised object detection performance under various COCO dataset sizes and ImageNet pre-trained checkpoint qualities. All models use Augment-S4 (for similar results with other augmentation methods see Appendix C).

Observation 2

Self-training is better than trained from scratch and pre-training

Setup	Augment-S1	Augment-S2	Augment-S3	Augment-S4
Rand Init	39.2	41.5	43.9	44.3
ImageNet Init	(+0.3) 39.5	(-0.7) 40.7	(-0.8) 43.2	(-1.0) 43.3
Rand Init w/ ImageNet Self-training	(+1.7) 40.9	(+1.5) 43.0	(+1.5) 45.4	(+1.3) 45.6

Table 2: In regimes where pre-training hurts, self-training with the same data source helps. All models are trained on the full COCO dataset.

Setup	20% Dataset	50% Dataset	100% Dataset
Rand Init	30.7	39.6	44.3
Rand Init w/ ImageNet Self-training	(+3.4) 34.1	(+1.8) 41.4	(+1.3) 45.6
ImageNet Init	33.3	38.8	43.3
ImageNet Init w/ ImageNet Self-training	(+2.7) 36.0	(+1.7) 40.5	(+1.3) 44.6
ImageNet++ Init	35.9	39.9	43.8
ImageNet++ Init w/ ImageNet Self-training	(+1.3) 37.2	(+1.6) 41.5	(+0.8) 44.6

Table 3: Self-training improves performance for all model initializations across all labeled dataset sizes. All models are trained on COCO using Augment-S4.

Observation 3

Self-supervised pre-training and supervised pre-training is similar in performance

Setup	COCO AP
Rand Init	41.1
ImageNet Init (Supervised)	(-0.7) 40.4
ImageNet Init (SimCLR)	(-0.7) 40.4
Rand Init w/ Self-training	(+0.8) 41.9

Table 4: Self-supervised pre-training (SimCLR) hurts performance on COCO just like standard supervised pre-training. Performance of ResNet-50 backbone model with different model initializations on full COCO. All models use Augment-S4.

Observation 4

Even targeted pre-training might not help (OpenImages have bounding box label)

Method	Pretrain Dataset	Iters	mmAP
FPN	None	540K	39.4
FPN	ImageNet	90K	36.4
FPN	ImageNet	180K	38.3
FPN	ImageNet	540K	39.3
FPN	OpenImages	90k	37.4
FPN	Objects365 w/o COCO 80	90K	39.6
FPN	Objects365	90K	42.0
FPN	ImageNet -> Objects365	90K	**42.3**
RetinaNet	ImageNet	180K	37.0
RetinaNet	Objects365	180K	39.5
RetinaNet	ImageNet -> Objects365	90K	**41.0**

Table 6. Generalization ability of general object detection results on the COCO dataset. “Iters” denotes the number of iterations for finetuning the models on the COCO dataset.

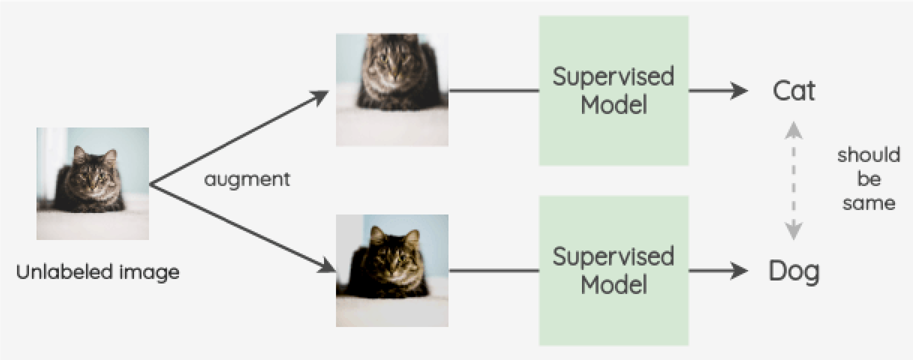
Observation 5

Joint training probably is helpful

Setup	Sup. Training	w/ Self-training	w/ Joint Training	w/ Self-training w/ Joint Training
Rand Init	30.7	(+3.4) 34.1	(+2.9) 33.6	(+4.4) 35.1
ImageNet Init	33.3	(+2.7) 36.0	(+0.7) 34.0	(+3.3) 36.6

Table 7: Comparison of pre-training, self-training and joint-training on COCO. All three methods use ImageNet as the additional dataset source. All models are trained on 20% COCO with Augment-S4.

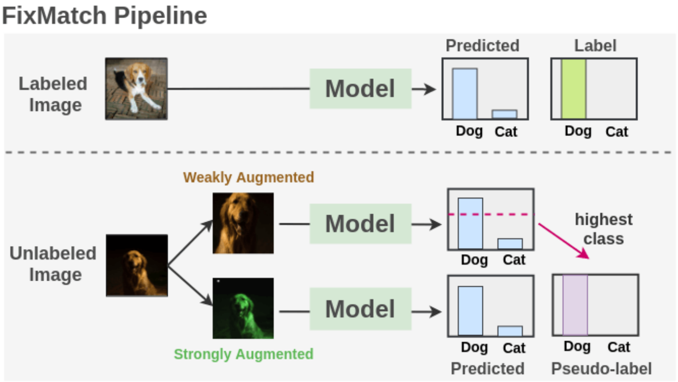
Consistency regularization



Similar to contrastive learning, but focus on classifier outputs rather than encoded features

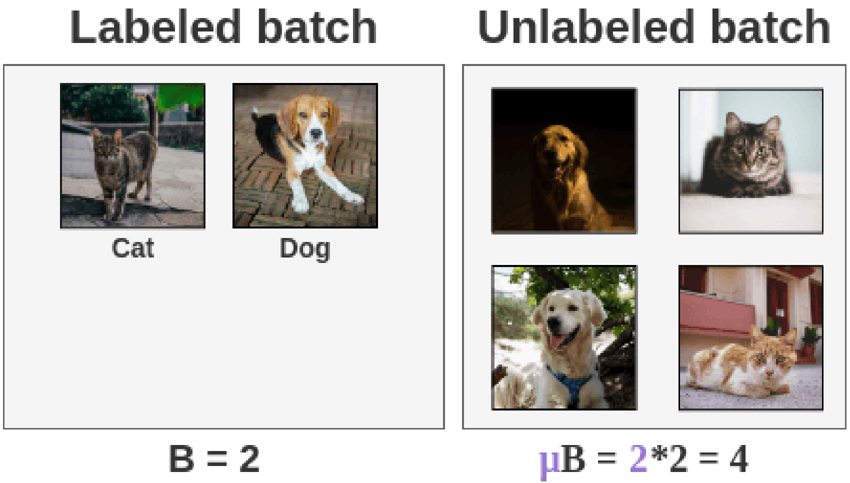
FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence

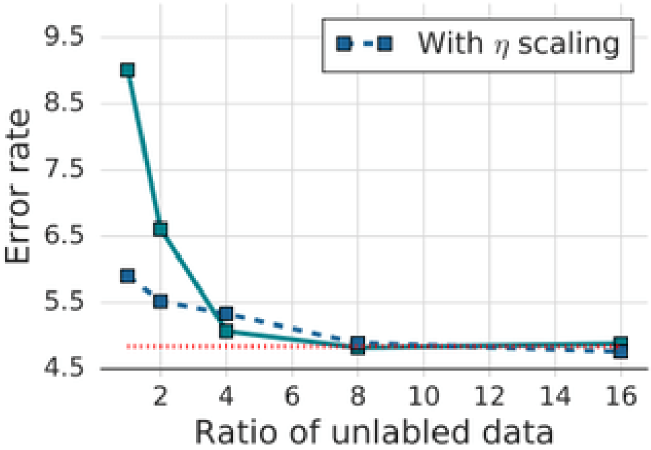
Main idea: combining consistency regularization with knowledge distillation



- Weakly Augmented:
  - Random flipping
  - Random translation (up to 12.5%)
- Strongly Augmented
  - Cutout
  - RandAugment
  - CTAugment

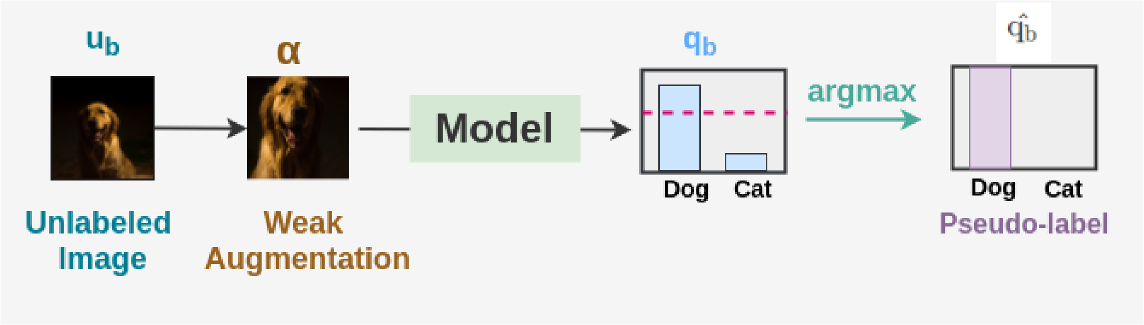
Step 1: preparing batches



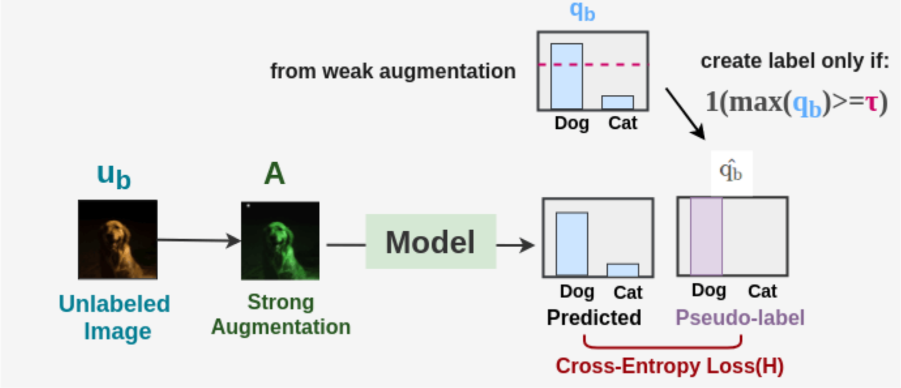

With η scaling
Error rate
9.5
8.5
7.5
6.5
5.5
4.5
2
4
6
8
10
12
14
16
Ratio of unlabled data

Steps 2 and 3: Supervised learning and pseudo-labeling

Pseudo-label generation



Step 4: Consistency regularization



$$l_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} 1(max(q_b) >= \tau)\ H(\hat{q}_b, p_m(y|A(u_b)\ ) \qquad loss = l_s + \lambda_u l_u$$

Result

STL-10

- MNIST of unsupervised learning
- 96x96 pixels, 10 classes
- 5,000 labeled images
- 100,000 unlabeled images

SOTA on STL-10 dataset

Table 3: Error rates for STL-10 on 1000-label splits. All baseline models are tested using the same codebase.

Method	Error rate	Method	Error rate
Π-Model	26.23±0.82	MixMatch	10.41±0.61
Pseudo-Labeling	27.99±0.80	UDA	7.66±0.56
Mean Teacher	21.43±2.39	ReMixMatch	**5.23**±0.45
FixMatch (RA)	7.98±1.50	FixMatch (CTA)	**5.17**±0.63

Summary

- Self-supervised learning: create surrogate task for pre-training
  - Pretext tasks
    - Coloring
    - Jigsaw
  - Sample clustering (e.g., ClusterFit)
  - Contrastive learning
    - PIRL
    - SimCLR
- Self-training: no surrogate task
  - Noisy student with knowledge distillation
  - Given sufficient unlabeled data, self-training usually works better than pre-training
  - Joint training is usually helpful
  - Fixmatch: consistency regularization + knowledge distillation

Links

- Self-supervised learning - pretext tasks
- Self-supervised learning - ClusterFit and PIRL
- Knowledge Transfer in Self Supervised Learning
- FixMatch semi-supervised learning