Transfer learning and self-supervised learning

Andrei Bursuc
valeo.ai

slides: https://abursuc.github.io/slides/ensta/transfer-learning-2025.html

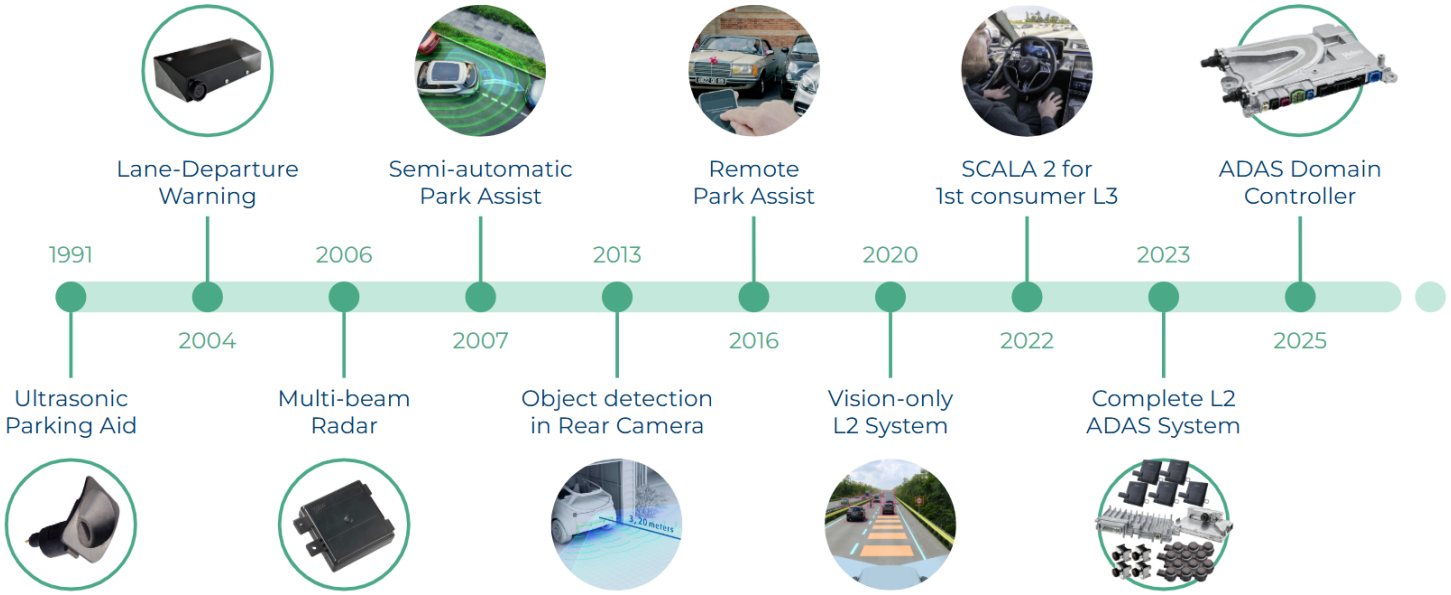
About me

- PhD at Mines ParisTech
- PostDoc at Inria Rennes and Inria Paris
- Research Scientist at Safran, Valeo
- Working on computer vision and machine learning for autunomous systems, *i.e.* any robot that moves/flies

Contact:

- email: `andrei.bursuc@gmail.com`
- web page: `https://abursuc.github.io/`
- twitter: `@abursuc`

Valeo's history in ADAS

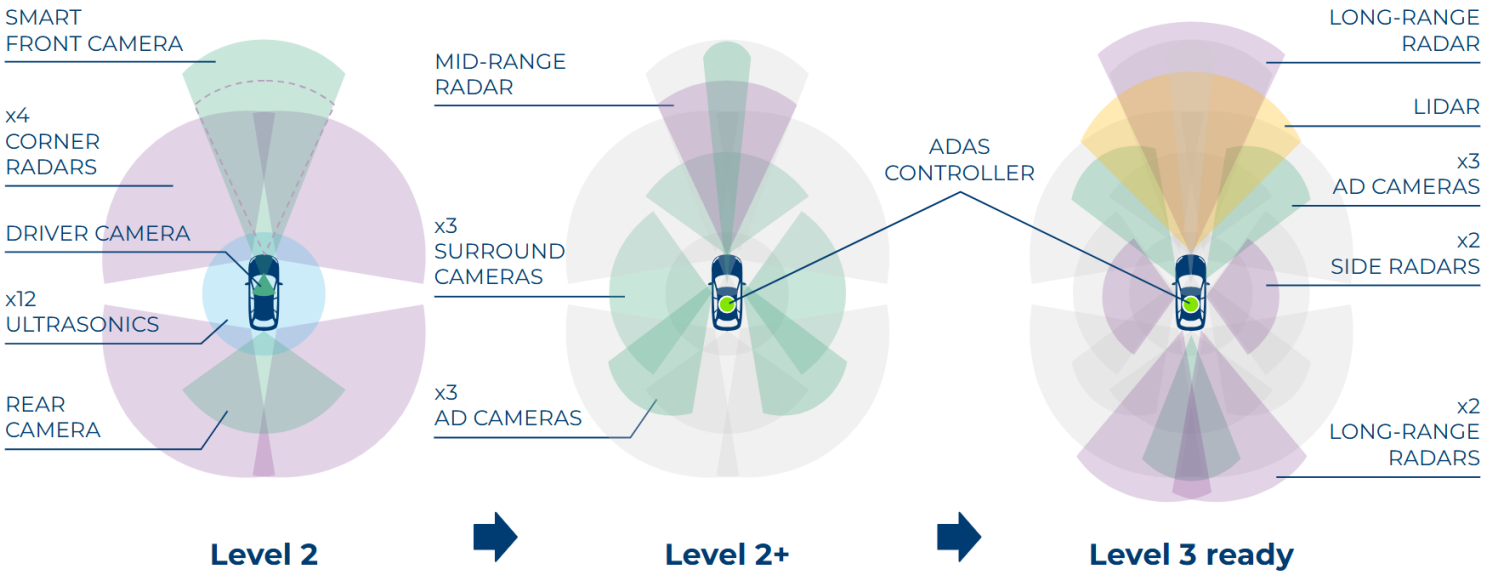


1.5+ Billion sensors shipped in 30 years

Another **1.5+ billion sensors** to be shipped in the next 5 years

Valeo sensor suite

Scalable system architecture



From ADAS to AD - spectrum of vehicle automatization

Driving Assistance

- Blind spot detection
- Cruise control

Forward collision warning + autobrake

↓ **56%** Front-to-rear crashes with injuries

Lane departure warning

↓ **21%** Injury crashes

*ADAS = Advanced Driving Assistance Systems | **AD = Autonomous Driving

From ADAS to AD - spectrum of vehicle automatization

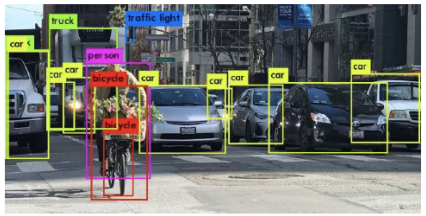
*ADAS = Advanced Driving Assistance Systems | **AD = Autonomous Driving

valeo.ai

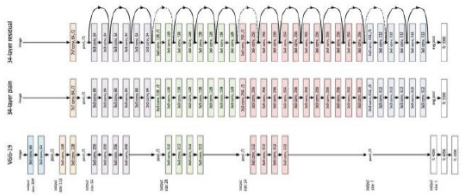
- ~25 researchers & PhDs
- Dedicated to open research
- 10s of academic collabs across France and Europe
- Offices: Paris, Prague
- Topics: perception, data efficiency, forecasting, reliability, explainability

https://valeoai.github.io

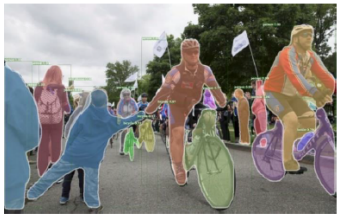
Deep Learning + Supervised Learning is a really cool and strong combo.



Faster R-CNN, Ren et al. 15



ResNet, He et al. 16



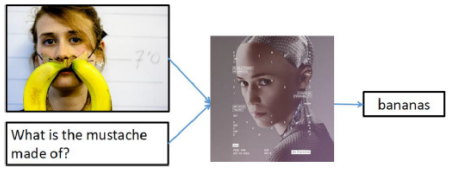
Mask R-CNN, He et al. 17

(a) Mobile phone query

(b) Retrieved image of same place

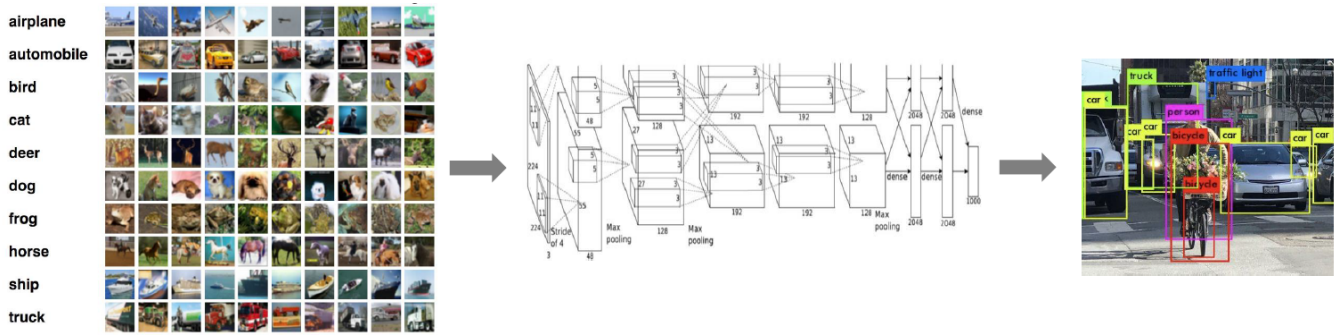
NetVlad: place recognition, Arandjelović et al. 16

Human pose estimation, Newel et al. 17



Visual Question Answering, Antol et al 15

Deep Learning: how it works?



- Predefine the set of visual concepts to be learned
- Collect diverse and large number of examples for each of them
- Train a deep model for several GPU hours or days

Deep Learning requires large amounts of carefully labeled data which is difficult to acquire and expensive to annotate.

Meanwhile, in the real world ...

Difficult to acquire and curate large human-annotated datasets

- Requires intense human labor
  - annotating + cleaning raw data
- Time consuming and expensive
- Error prone (human mistakes)

Annotating such image: ~1.5h

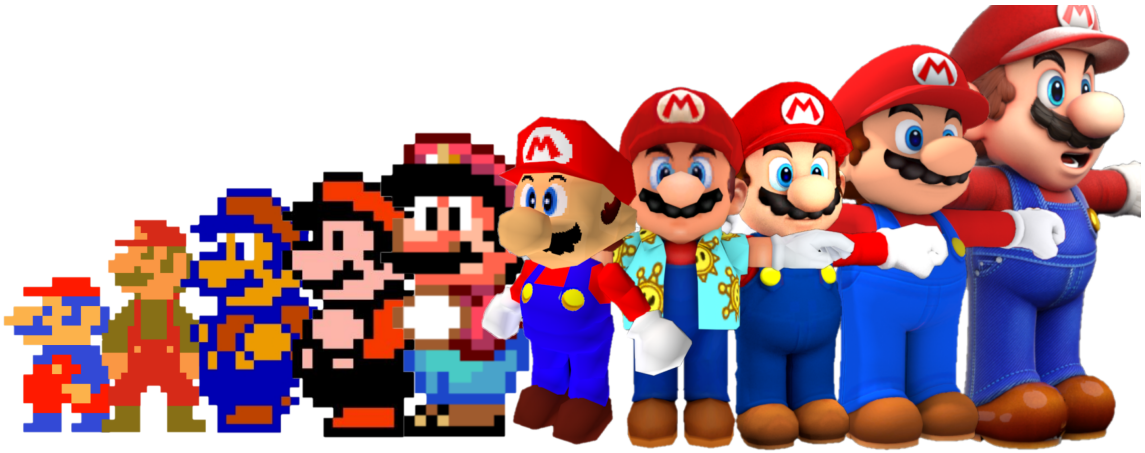
Difficult to keep the pace with an ever changing world

Men's fashion trends 1980-1989

- Data distributions shift all the time, e.g., fashion trends, new Instagram filters
- Infeasible to launch large annotation campaigns each time

Credit image: La Polo

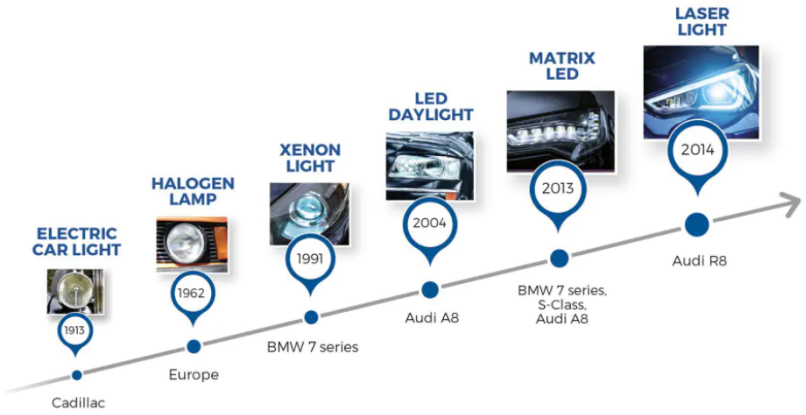
Difficult to keep the pace with an ever changing world



Super Mario from 1981 to 2017

- Sensors specs are frequently upgraded
- Infeasible to launch large annotation campaigns each time

Difficult to keep the pace with an ever changing world



Halogen vs. LED

- Sensors specs are frequently upgraded
- Headlamps change the appearance of the scenes and of the vehicles to detect

Outline

Transfer learning

Off-the shelf networks

Fine-tuning

(Task) transfer learning

Multi-task learning

Domain adaptation

Self-supervised learning

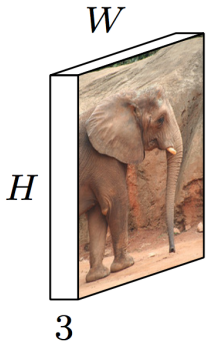
Transfer Learning

Transfer learning

- Assume two datasets S and T
- Dataset S is fully annotated, plenty of images and we can train a model CNN_S on it
- Dataset T is not as much annotated and/or with fewer images
    - annotations of T do not necessarily overlap with S
- We can use the model CNN_S to learn a better CNN_T
- This is transfer learning

Why using a pre-trained CNN (*off-the-shelf*) would be a good idea?

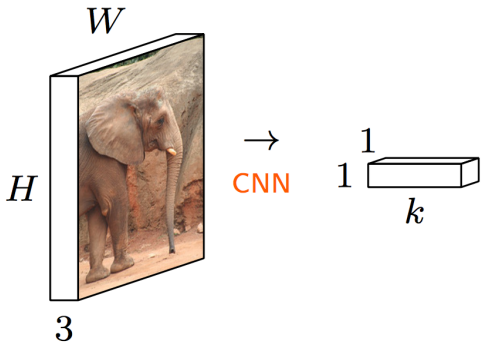
Image ranking by CNN features



- 3-chanel RGB input, 224×224
- AlexNet pre-trained on ImageNet for classification

A. Krizhevksy et al., Imagenet Classification with Deep Convolutional Neural Networks, NIPS 2012

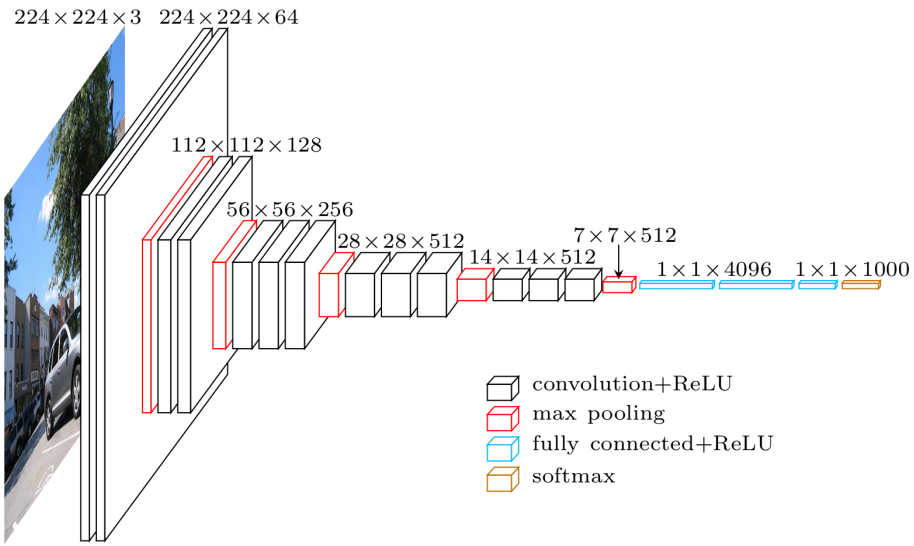
Image ranking by CNN features



- 3-chanel RGB input, 224×224
- AlexNet pre-trained on ImageNet for classification
- last fully connected layer (fc_6): global descriptor dimension $k = 4096$

A. Krizhevksy et al., Imagenet Classification with Deep Convolutional Neural Networks, NIPS 2012

VGG-16



K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, NIPS 2014

VGG-16

```
           Activation maps          Parameters
INPUT:     [224x224x3]   = 150K     0
CONV3-64:  [224x224x64]  = 3.2M     (3x3x3)x64    =       1,728
CONV3-64:  [224x224x64]  = 3.2M     (3x3x64)x64   =      36,864
POOL2:     [112x112x64]  = 800K     0
CONV3-128: [112x112x128] = 1.6M     (3x3x64)x128  =      73,728
CONV3-128: [112x112x128] = 1.6M     (3x3x128)x128 =     147,456
POOL2:     [56x56x128]   = 400K     0
CONV3-256: [56x56x256]   = 800K     (3x3x128)x256 =     294,912
CONV3-256: [56x56x256]   = 800K     (3x3x256)x256 =     589,824
CONV3-256: [56x56x256]   = 800K     (3x3x256)x256 =     589,824
POOL2:     [28x28x256]   = 200K     0
CONV3-512: [28x28x512]   = 400K     (3x3x256)x512 =   1,179,648
CONV3-512: [28x28x512]   = 400K     (3x3x512)x512 =   2,359,296
CONV3-512: [28x28x512]   = 400K     (3x3x512)x512 =   2,359,296
POOL2:     [14x14x512]   = 100K     0
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
POOL2:     [7x7x512]     =  25K     0
FC:        [1x1x4096]    = 4096     7x7x512x4096  = 102,760,448
FC:        [1x1x4096]    = 4096     4096x4096     =  16,777,216
FC:        [1x1x1000]    = 1000     4096x1000     =   4,096,000

TOTAL activations: 24M x 4 bytes ~=  93MB / image (x2 for backward)
TOTAL parameters: 138M x 4 bytes ~= 552MB (x2 for plain SGD, x4 for Adam)
```

K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, NIPS 2014

Slide credit: C. Ollion & O. Grisel

VGG-16

```
           Activation maps          Parameters
INPUT:     [224x224x3]   = 150K     0
CONV3-64:  [224x224x64]  = 3.2M     (3x3x3)x64    =       1,728
CONV3-64:  [224x224x64]  = 3.2M     (3x3x64)x64   =      36,864
POOL2:     [112x112x64]  = 800K     0
CONV3-128: [112x112x128] = 1.6M     (3x3x64)x128  =      73,728
CONV3-128: [112x112x128] = 1.6M     (3x3x128)x128 =     147,456
POOL2:     [56x56x128]   = 400K     0
CONV3-256: [56x56x256]   = 800K     (3x3x128)x256 =     294,912
CONV3-256: [56x56x256]   = 800K     (3x3x256)x256 =     589,824
CONV3-256: [56x56x256]   = 800K     (3x3x256)x256 =     589,824
POOL2:     [28x28x256]   = 200K     0
CONV3-512: [28x28x512]   = 400K     (3x3x256)x512 =   1,179,648
CONV3-512: [28x28x512]   = 400K     (3x3x512)x512 =   2,359,296
CONV3-512: [28x28x512]   = 400K     (3x3x512)x512 =   2,359,296
POOL2:     [14x14x512]   = 100K     0
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
POOL2:     [7x7x512]     =  25K     0
FC:        [1x1x4096]    = 4096     7x7x512x4096  = 102,760,448
FC:        [1x1x4096]    = 4096     4096x4096     =  16,777,216
FC:        [1x1x1000]    = 1000     4096x1000     =   4,096,000

TOTAL activations: 24M x 4 bytes ~=  93MB / image (x2 for backward)
TOTAL parameters: 138M x 4 bytes ~= 552MB (x2 for plain SGD, x4 for Adam)
```

K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, NIPS 2014

Slide credit: C. Ollion & O. Grisel

Image ranking by CNN features

- query images

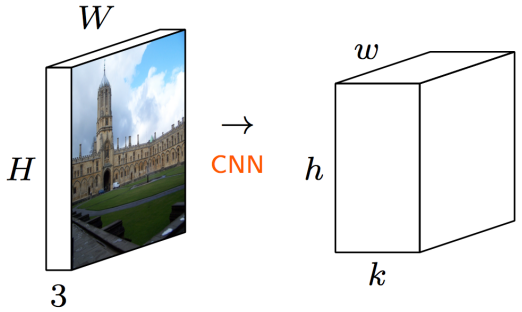
A. Krizhevksy et al., Imagenet Classification with Deep Convolutional Neural Networks, NIPS 2012

Image ranking by CNN features

- query images : nearest neighbors in ImageNet according to Euclidean distance, aka k-NN classifier

A. Krizhevksy et al., Imagenet Classification with Deep Convolutional Neural Networks, NIPS 2012

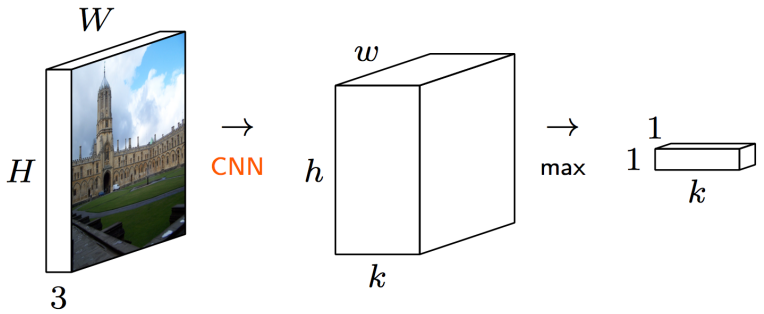
Sampling information from feature maps



- VGG-16 last convolutional layer, $k = 512$

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

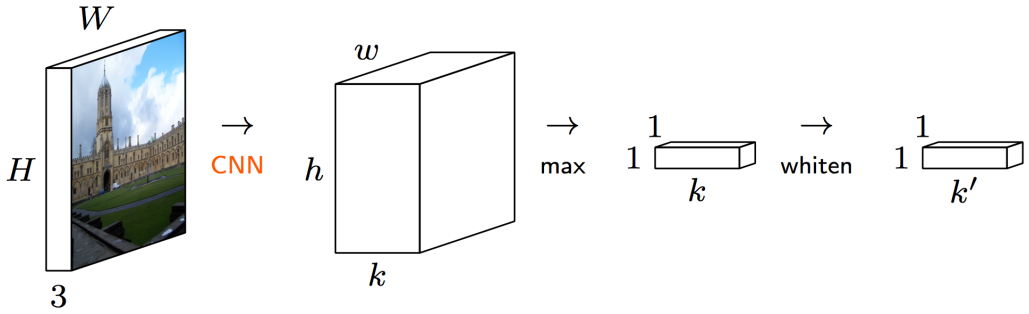
Sampling information from feature maps



- VGG-16 last convolutional layer, $k = 512$
- global spatial max-pooling/sum

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

Sampling information from feature maps



- VGG-16 last convolutional layer, $k = 512$
- global spatial max-pooling/sum
- ℓ_2 normalization, PCA-whitening, ℓ_2 normalization
- MAC: maximum activation from convolutions

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

VGG-16

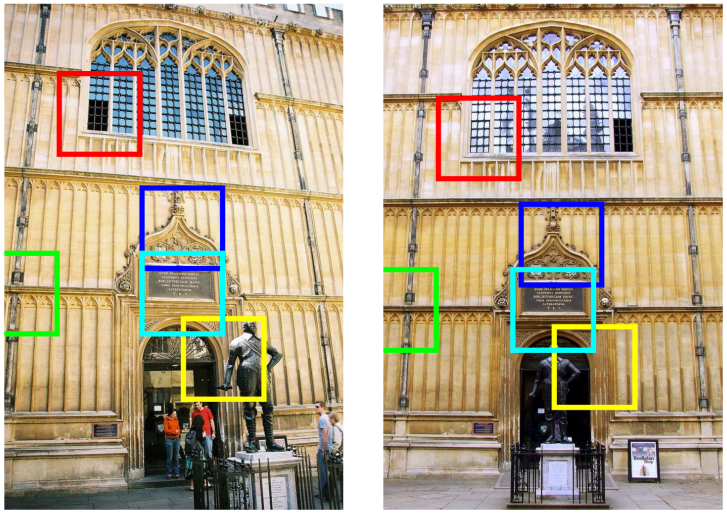
```
           Activation maps          Parameters
INPUT:     [224x224x3]   = 150K     0
CONV3-64:  [224x224x64]  = 3.2M     (3x3x3)x64    =       1,728
CONV3-64:  [224x224x64]  = 3.2M     (3x3x64)x64   =      36,864
POOL2:     [112x112x64]  = 800K     0
CONV3-128: [112x112x128] = 1.6M     (3x3x64)x128  =      73,728
CONV3-128: [112x112x128] = 1.6M     (3x3x128)x128 =     147,456
POOL2:     [56x56x128]   = 400K     0
CONV3-256: [56x56x256]   = 800K     (3x3x128)x256 =     294,912
CONV3-256: [56x56x256]   = 800K     (3x3x256)x256 =     589,824
CONV3-256: [56x56x256]   = 800K     (3x3x256)x256 =     589,824
POOL2:     [28x28x256]   = 200K     0
CONV3-512: [28x28x512]   = 400K     (3x3x256)x512 =   1,179,648
CONV3-512: [28x28x512]   = 400K     (3x3x512)x512 =   2,359,296
CONV3-512: [28x28x512]   = 400K     (3x3x512)x512 =   2,359,296
POOL2:     [14x14x512]   = 100K     0
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
CONV3-512: [14x14x512]   = 100K     (3x3x512)x512 =   2,359,296
POOL2:     [7x7x512]     =  25K     0
FC:        [1x1x4096]    = 4096     7x7x512x4096  = 102,760,448
FC:        [1x1x4096]    = 4096     4096x4096     =  16,777,216
FC:        [1x1x1000]    = 1000     4096x1000     =   4,096,000

TOTAL activations: 24M x 4 bytes ~=  93MB / image (x2 for backward)
TOTAL parameters: 138M x 4 bytes ~= 552MB (x2 for plain SGD, x4 for Adam)
```

K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, NIPS 2014

Slide credit: C. Ollion & O. Grisel

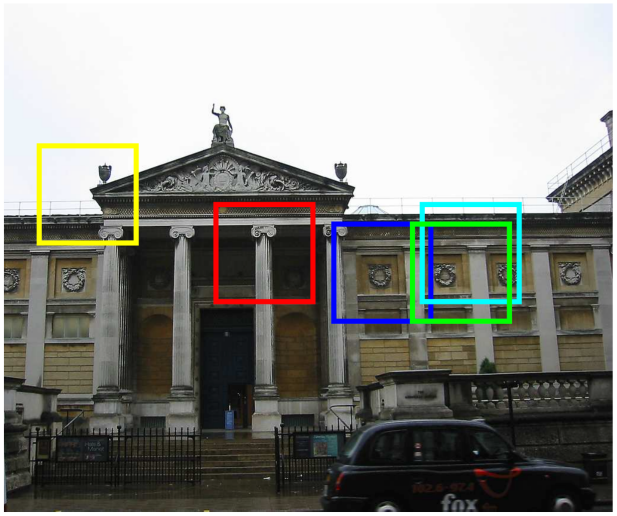
Global max-pooling: matching

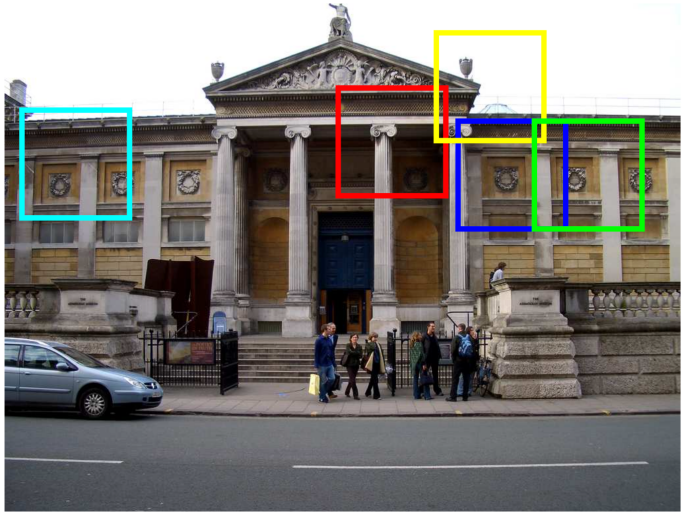


- receptive fields of 5 components of MAC vectors that contribute most to image similarity

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

Global max-pooling: matching





- receptive fields of 5 components of MAC vectors that contribute most to image similarity

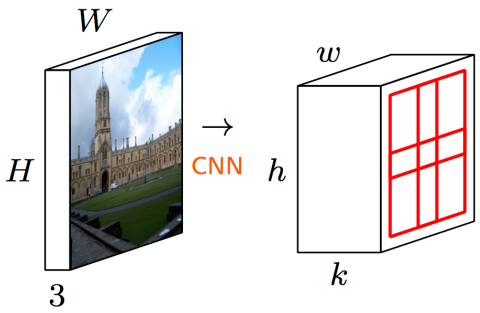
G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

Global max-pooling: matching

- receptive fields of 5 components of MAC vectors that contribute most to image similarity

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

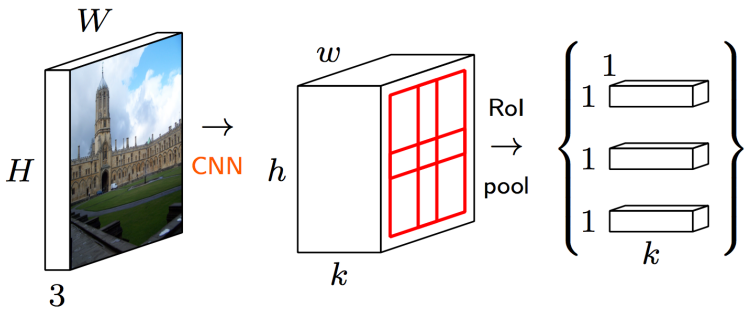
Regional max-pooling (R-MAC)



- VGG-16 last convolutional layer, $k = 512$
- fixed mulitscale overlapping regions

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

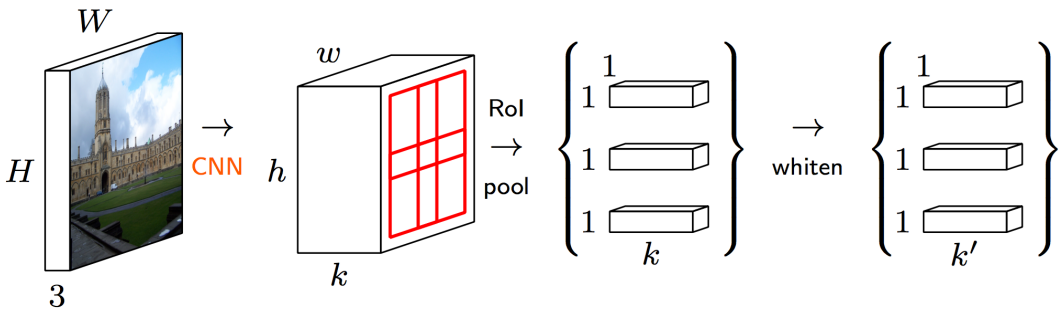
Regional max-pooling (R-MAC)



- VGG-16 last convolutional layer, $k = 512$
- fixed mulitscale overlapping regions, spatial max-pooling

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

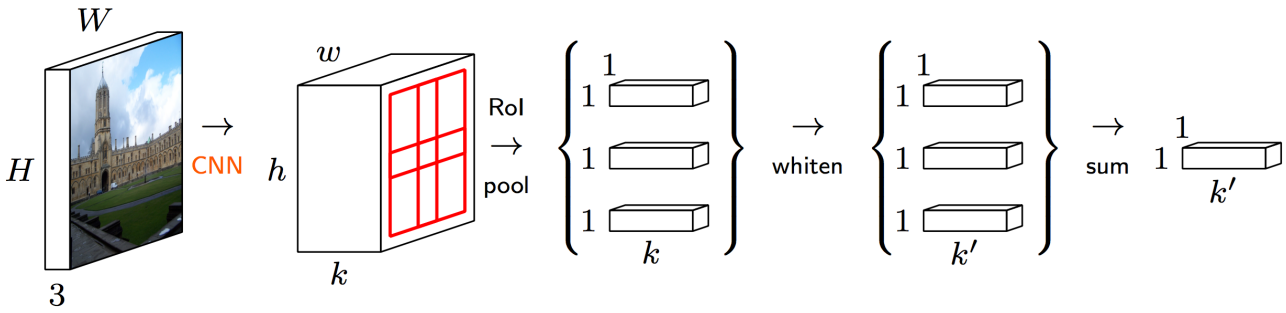
Regional max-pooling (R-MAC)



- VGG-16 last convolutional layer, $k = 512$
- fixed mulitscale overlapping regions, spatial max-pooling
- ℓ_2 normalization, PCA-whitening, ℓ_2 normalization

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

Regional max-pooling (R-MAC)

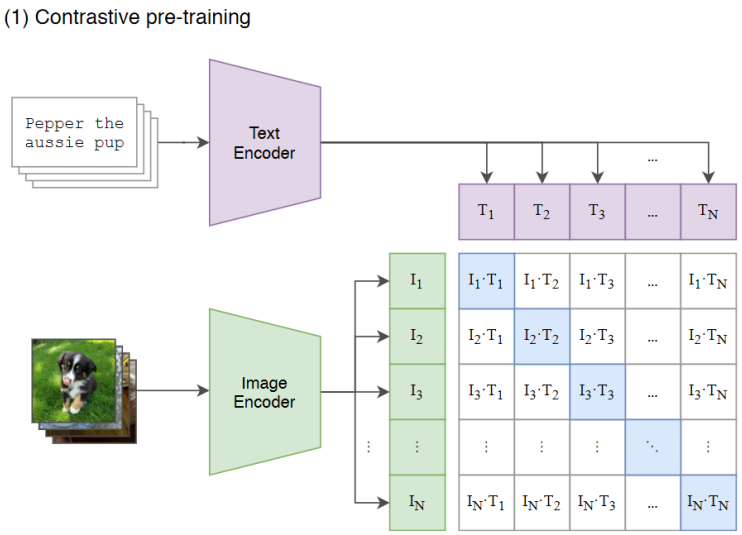


- VGG-16 last convolutional layer, $k = 512$
- fixed mulitscale overlapping regions, spatial max-pooling
- ℓ_2 normalization, PCA-whitening, ℓ_2 normalization
- sum-pooling over all descriptors, ℓ_2 normalization

G. Tolias et al., Particular object retrieval with integral max-pooling, ICLR 2016

Similar strategies can be used for ViT architectures where the `[cls]` token or `[avg]` of patch tokens can be used as image representations.

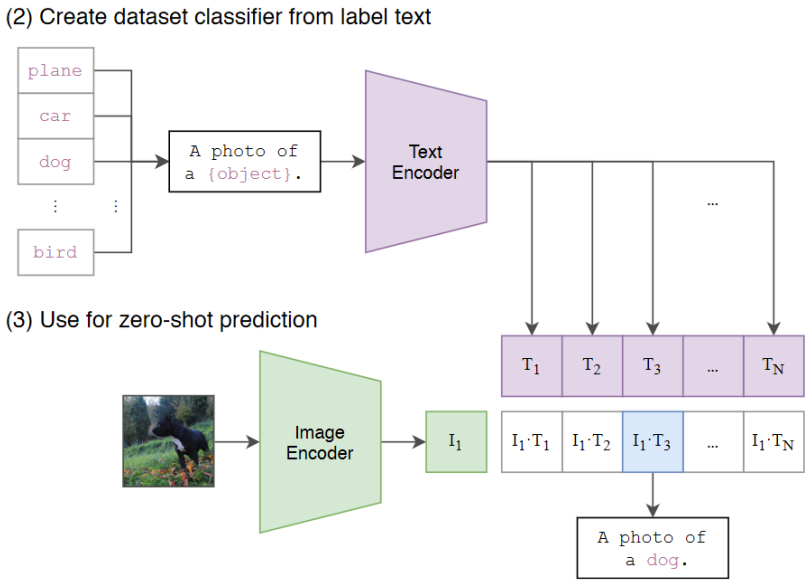
Zero-shot classification with CLIP



Training an image network and a text network together from $400M$ pairs of images and captions crawled from Internet

A. Radford et al., Learning Transferable Visual Models From Natural Language Supervision, arXiv 2021

Zero-shot classification with CLIP



Use networks and embeddings in zero-shot (or open-vocabulary) classification

A. Radford et al., Learning Transferable Visual Models From Natural Language Supervision, arXiv 2021

Implementation - numpy-like pseudocode

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)
# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

A. Radford et al., Learning Transferable Visual Models From Natural Language Supervision, arXiv 2021

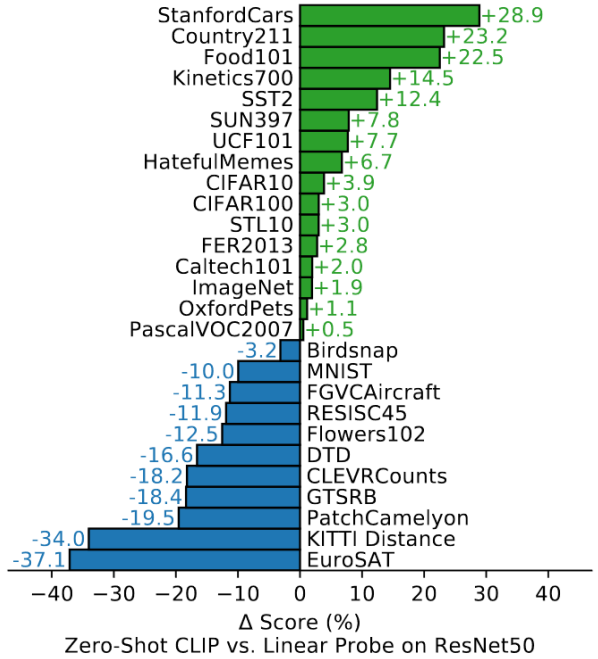
Training CLIP is very expensive:

- ResNet (RN50x64): 18 days on 592 V100 GPUS
- ViT (Vision Transformer): 12 days on 256 V100 GPUs
- batch-size: 32.768

Such models that are expensive to train but can be repurposed for other tasks, are referred to as foundation models.

R. Bommasani et al., On the Opportunities and Risks of Foundation Models, arXiv 2021

Works very well in practice



Zero-shot CLIP is competitive with a fully supervised baseline

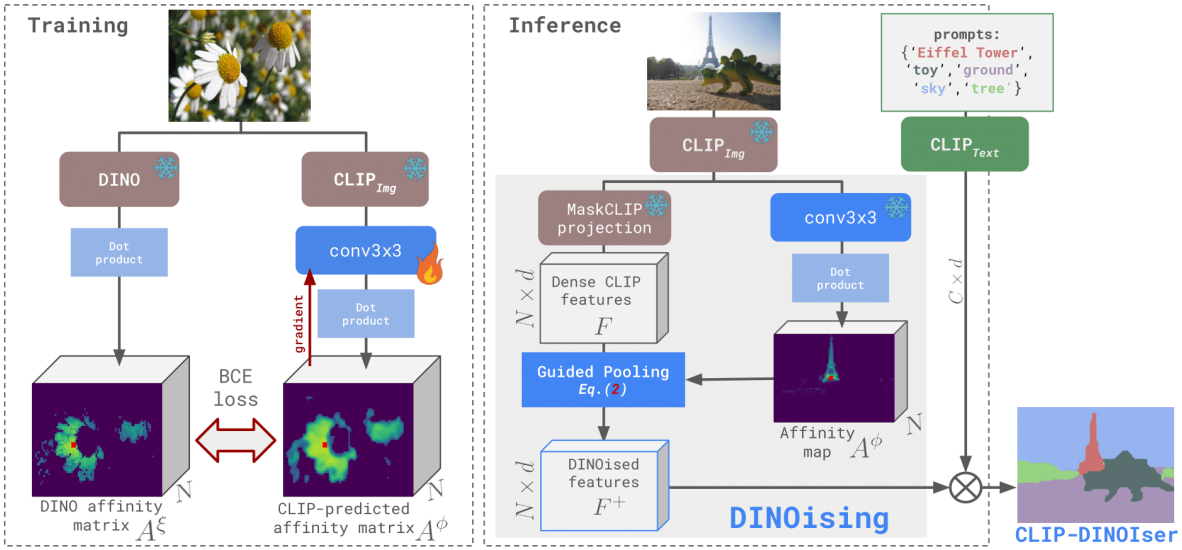
A. Radford et al., Learning Transferable Visual Models From Natural Language Supervision, arXiv 2021

CLIP zero-shot segmentation without human annotations

- leverage objectness information from another foundation model (DINO) to guide CLIP local pooling

M. Wysoczańska et al., CLIP-DINOiser: Teaching CLIP a few DINO tricks for open-vocabulary semantic segmentation, ECCV 2024

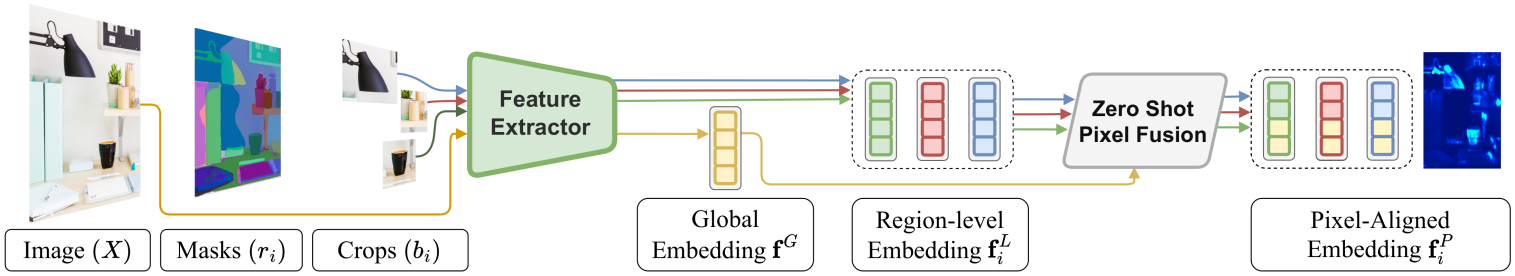
CLIP zero-shot segmentation without human annotations



- leverage objectness information from another foundation model (DINO) to guide CLIP local pooling

M. Wysoczańska et al., CLIP-DINOiser: Teaching CLIP a few DINO tricks for open-vocabulary semantic segmentation, ECCV 2024

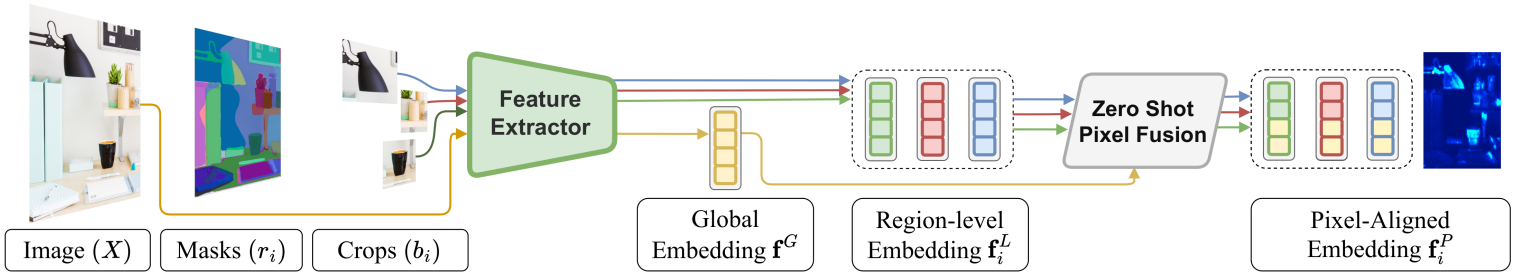
CLIP can be extended to pixel-level representations



- by default CLIP extract global image features (1 veector for the whole image)
- some methods fine-tune CLIP to get pixel-level features but they drift away from text representations
- Concept-Fusion proposes a trick to get pixel-level features without any fine-tuning (compatibility with text CLIP is preserved)

K.M. Jatavallabhula et al., ConceptFusion: Open-set Multimodal 3D Mapping, arXiv 2023

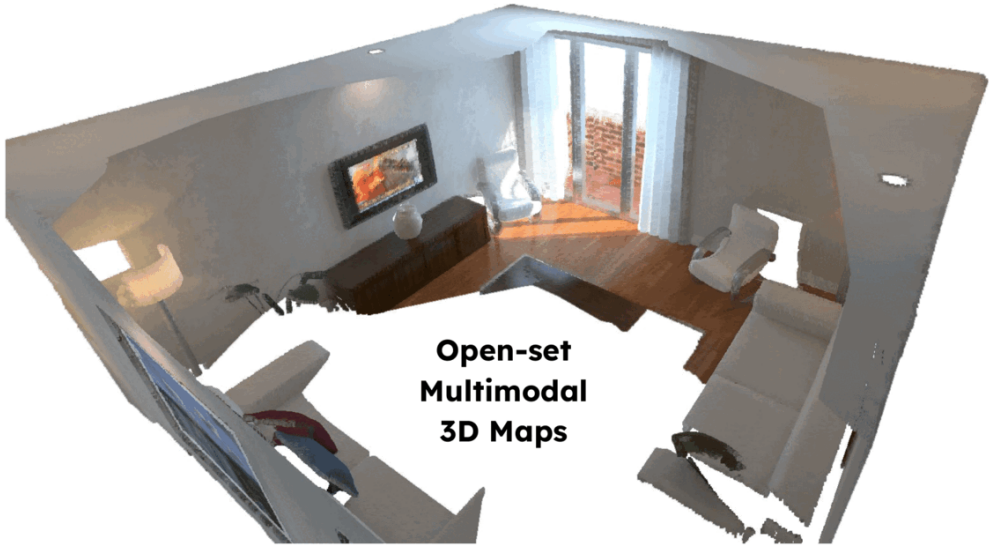
CLIP can be extended to pixel-level representations



- by default CLIP extract global image features (1 veector for the whole image)
- some methods fine-tune CLIP to get pixel-level features but they drift away from text representations
- Concept-Fusion proposes a trick to get pixel-level features without any fine-tuning (compatibility with text CLIP is preserved)

K.M. Jatavallabhula et al., ConceptFusion: Open-set Multimodal 3D Mapping, arXiv 2023

Multi-modal CLIP querying on 3D meshes



- the room layout is construction from a sequence of images from a camera walking around the room (computed with Dense SLAM)

K.M. Jatavallabhula et al., ConceptFusion: Open-set Multimodal 3D Mapping, arXiv 2023

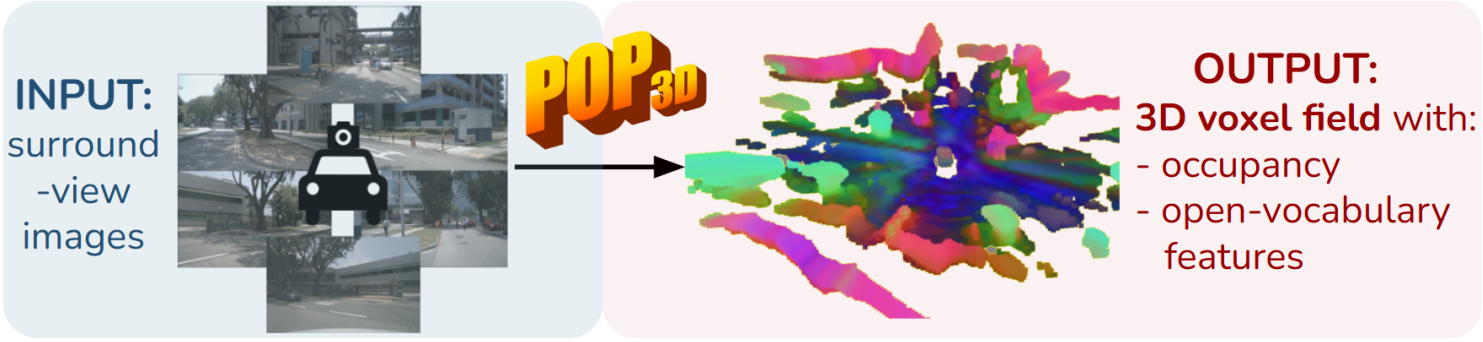
Multi-modal CLIP querying on 3D meshes

K.M. Jatavallabhula et al., ConceptFusion: Open-set Multimodal 3D Mapping, arXiv 2023

Clustering on Lidar point clouds

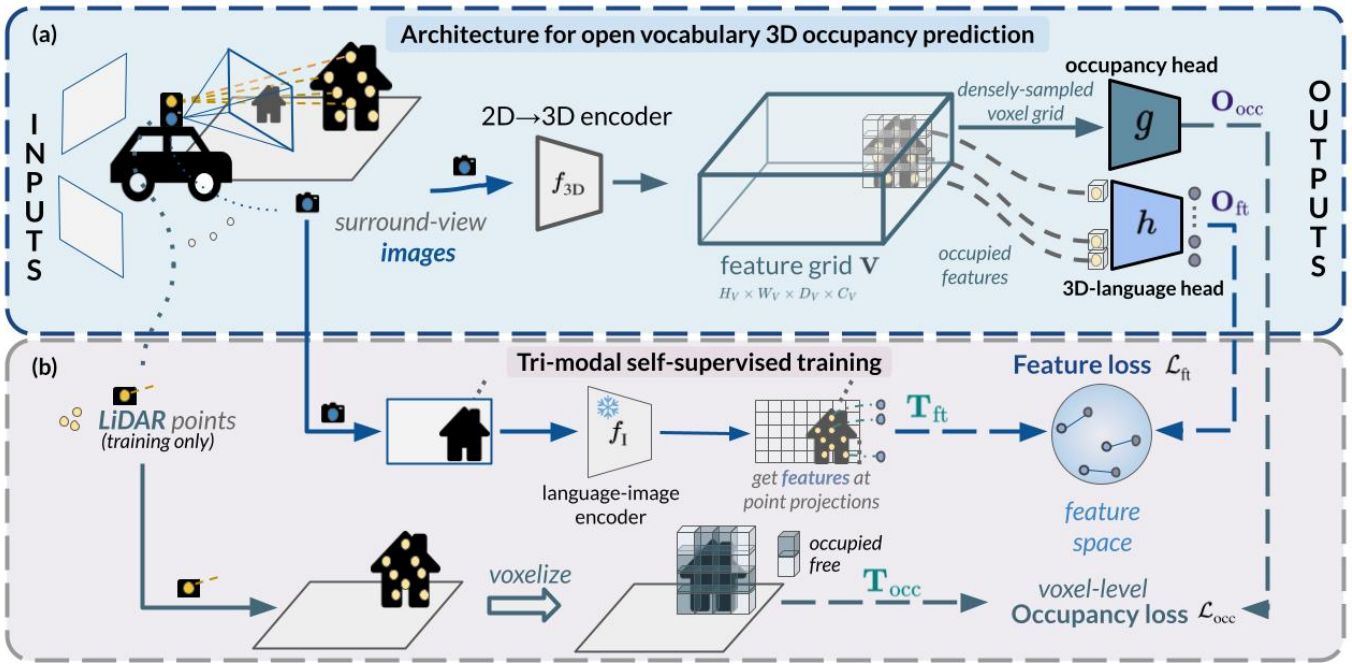
K.M. Jatavallabhula et al., ConceptFusion: Open-set Multimodal 3D Mapping, arXiv 2023

Open-Vocabulary 3D Occupancy Prediction from Images



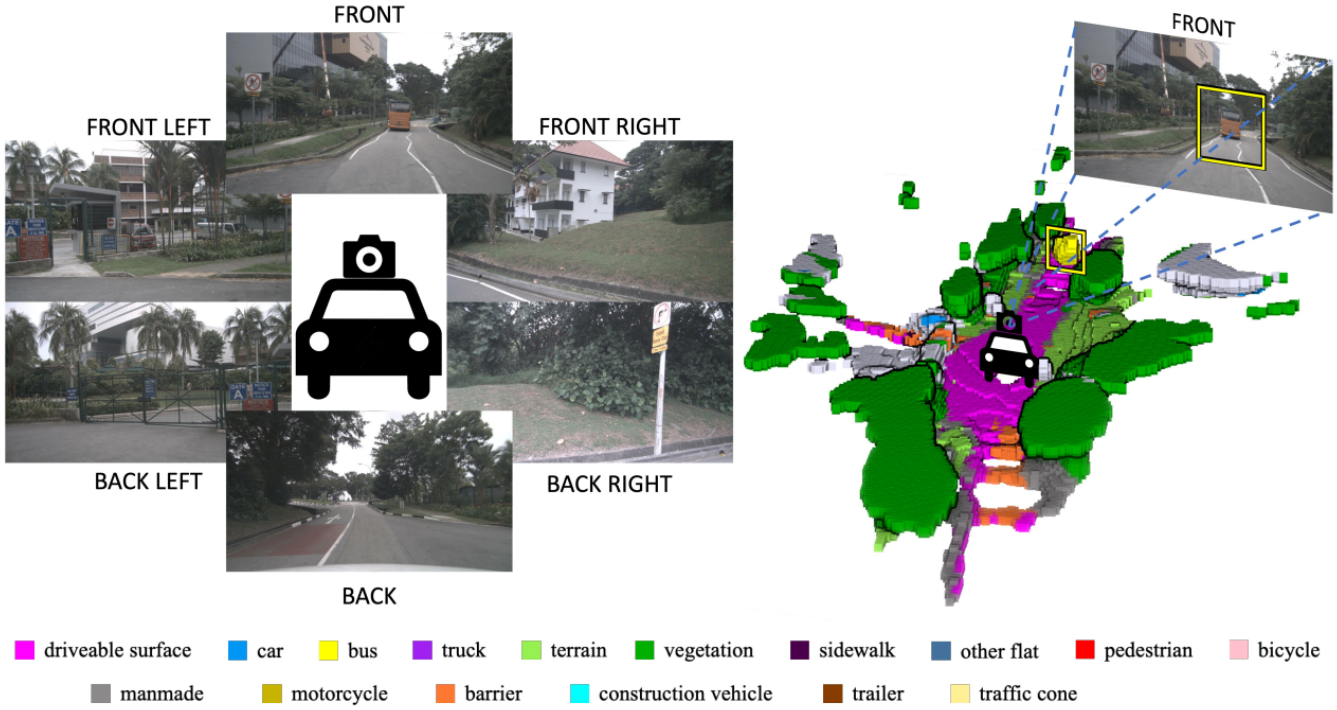
A. Vobecky et al., POP-3D: Open-Vocabulary 3D Occupancy Prediction from Images, NeurIPS 2023

POP-3D architecture



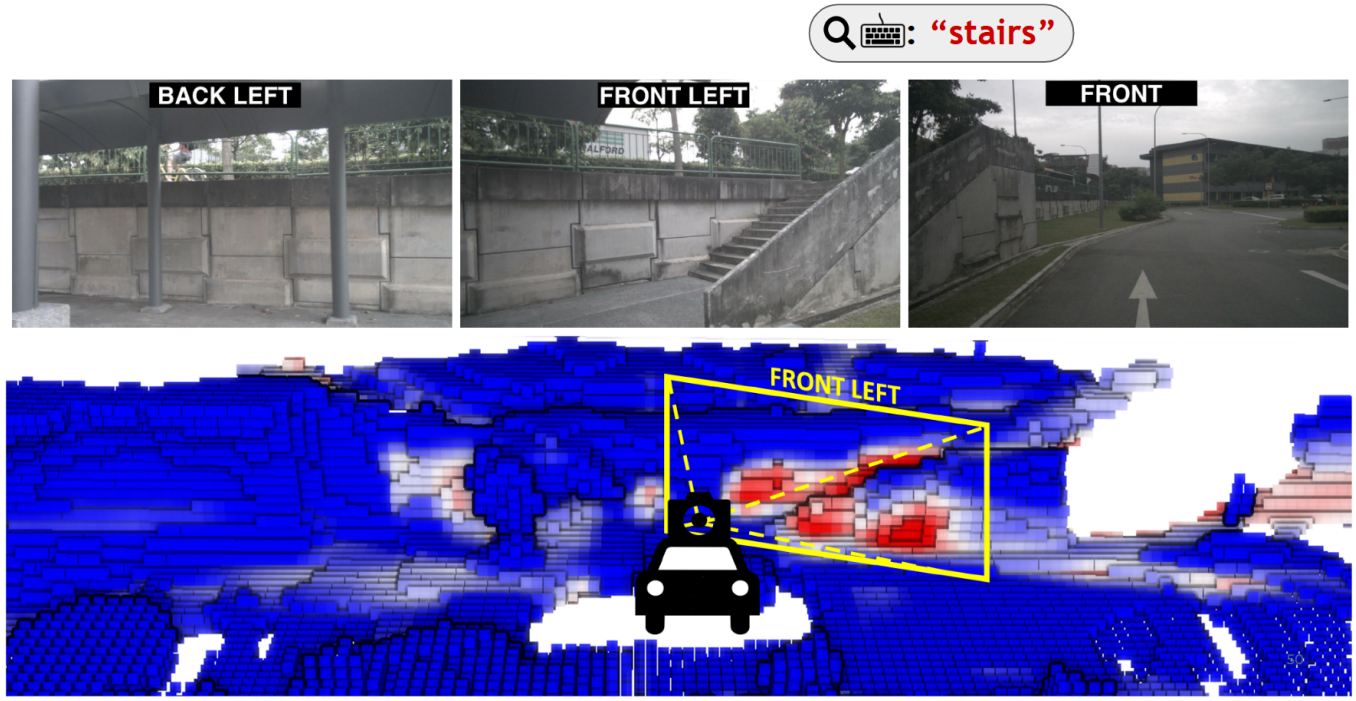
A. Vobecky et al., POP-3D: Open-Vocabulary 3D Occupancy Prediction from Images, NeurIPS 2023

POP-3D qualitative results: zero-shot semantic segmentation



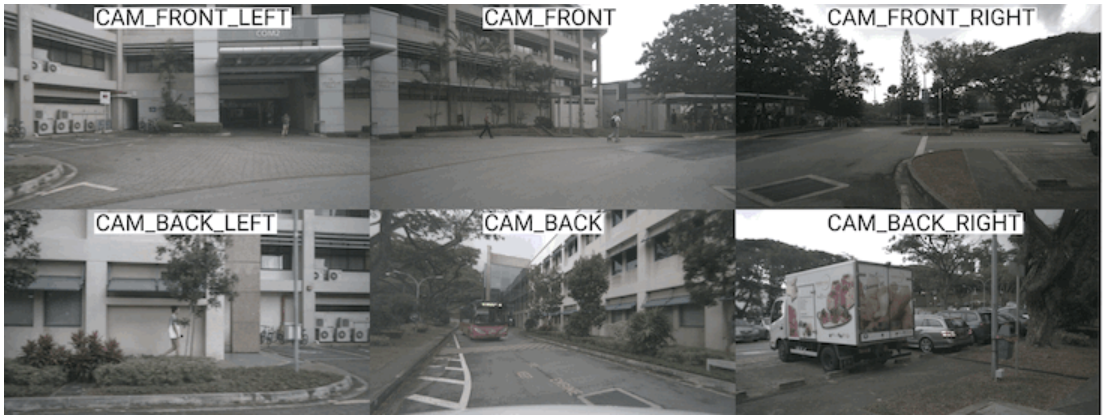
A. Vobecky et al., POP-3D: Open-Vocabulary 3D Occupancy Prediction from Images, NeurIPS 2023

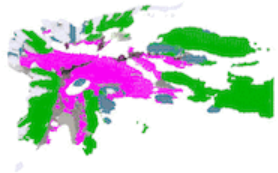
POP-3D qualitative results: retrieval



A. Vobecky et al., POP-3D: Open-Vocabulary 3D Occupancy Prediction from Images, NeurIPS 2023

POP-3D

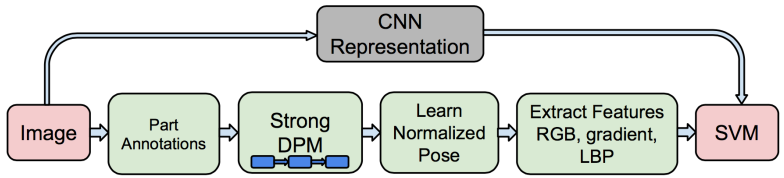


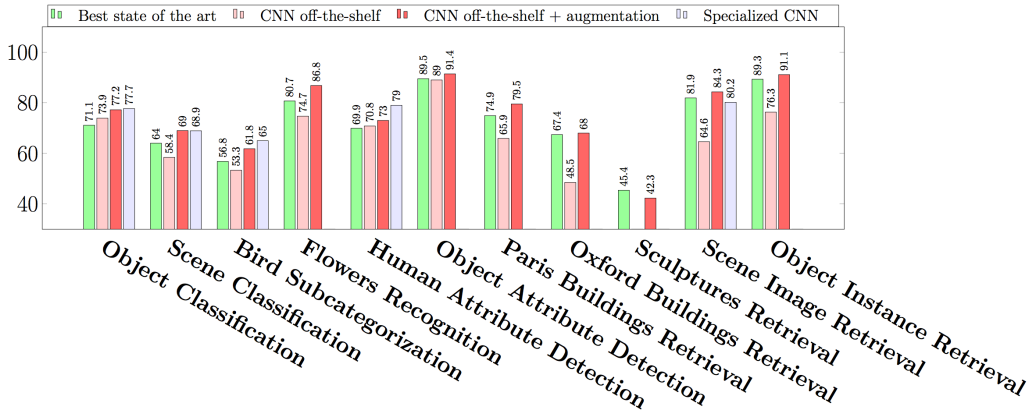


A. Vobecky et al., POP-3D: Open-Vocabulary 3D Occupancy Prediction from Images, NeurIPS 2023

Can we do more with a bit of training?

Training a task-specific linear classifier on top of CNN features led to SoTA or nearly-SoTA results





A. Razavian et al., CNN Features off-the-shelf: an Astounding Baseline for Recognition, arXiv 2014

Mikael Huss
@mikaelhuss

Follow

Stefan Carlsson, head of Computer Vision Group, KTH in Sthlm. "computer vision is now essentially solved" #sthlmDL

6:46 PM - 10 Mar 2015

Mikael Huss
@mikaelhuss

Follow

Stefan Carlsson, head of Computer Vision Group, KTH in Sthlm. "computer vision is now essentially solved" #sthlmDL

6:46 PM - 10 Mar 2015

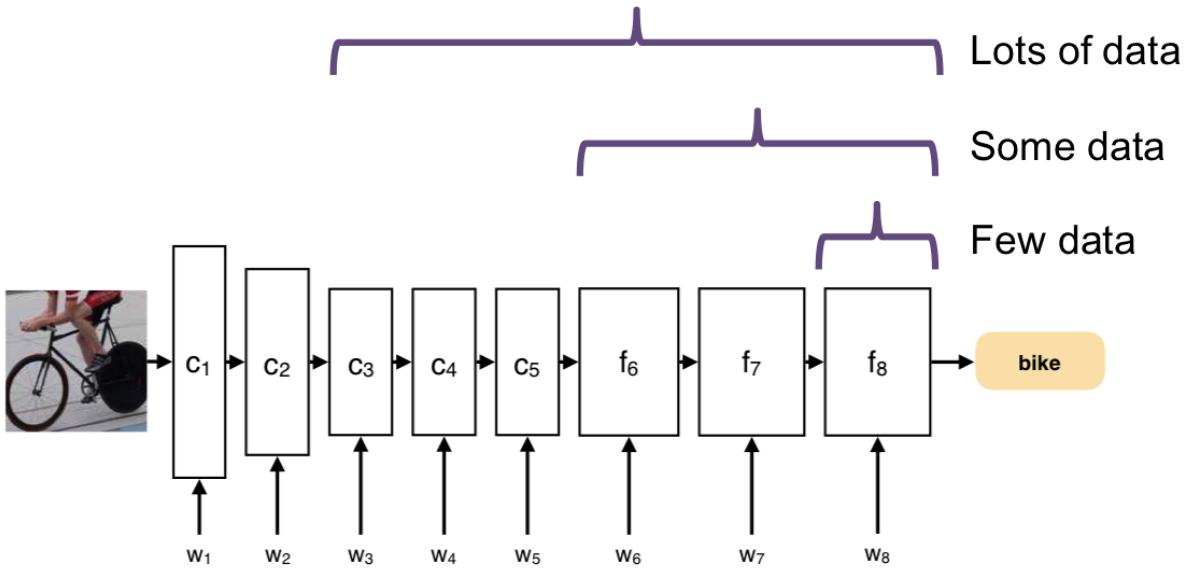
2025 edit: Not quite there yet

Fine-tuning

Fine-tuning

- Assume the parameters of CNN_S are already a good start near our final local optimum
- Use them as the initial parameters for our new CNN for the target dataset
- This is a good solution when the dataset T is relatively big
    - e.g. for Imagenet S with 1M images, T with a few thousand images

Fine-tuning



- Depending on the size of T decide which layer to freeze and which to finetune/replace
- Use lower learning rate when fine-tuning: about $\frac{1}{10}$ of original learning rate
  - for new layers use agressive learning rate
- If S and T are very similar,fine-tune only fully-connected layers
- If datasets are different and you have enough data, fine-tune all layers

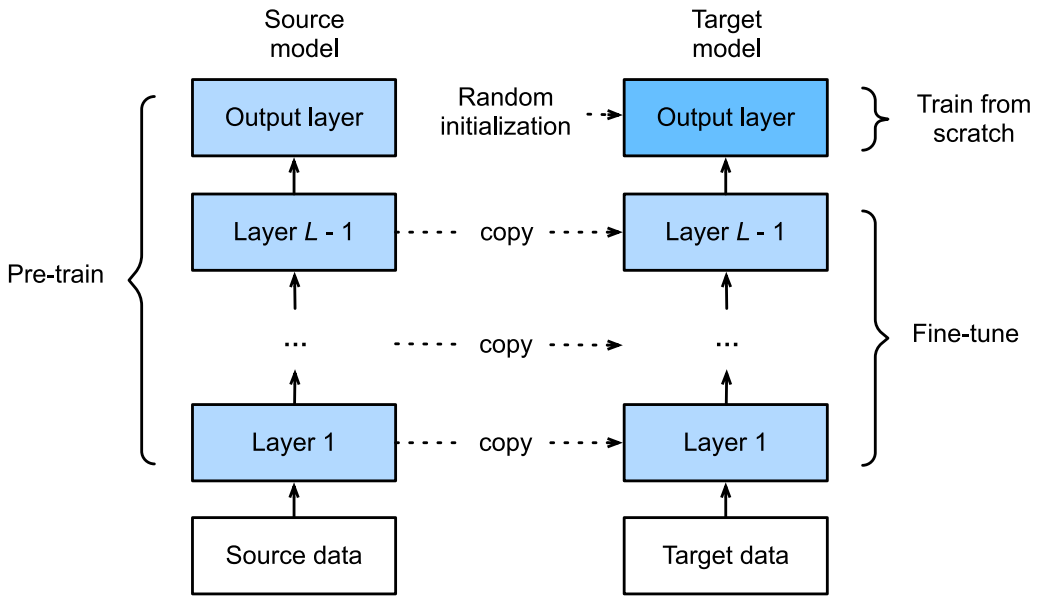


Figure credit: A. Zhang et al., Dive Into Deep Learning, 2020

For models pre-trained on ImageNet, transferred/fine-tuned networks usually work even when the input images for the new task are not photographs of objects or animals, such as biomedical images, satellite images or paintings.

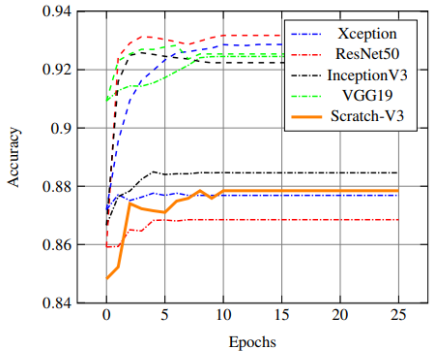
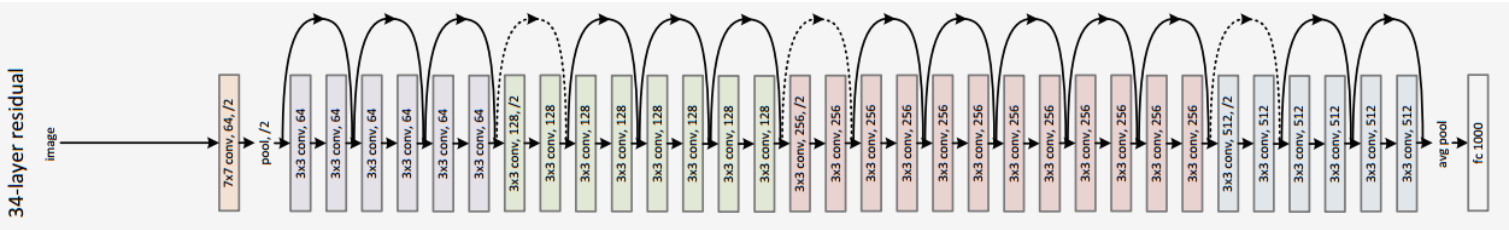


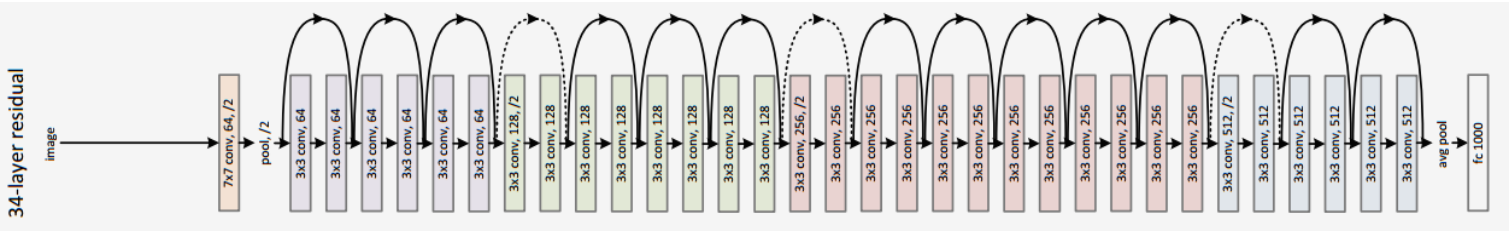
Fig. 2: Comparison between the fine tuning approach versus the off the shelf one when classifying the material of the heritage objects of the Rijksmuseum dataset. We observe how the first approach (as reported by the the dashed lines) leads to significant improvements when compared to the latter one (reported by the dash-dotted lines) for three out of four neural architectures. Furthermore, we can also observe how training a DCNN from scratch leads to worse results when compared to fine-tuned architectures which have been pre-trained on ImageNet (solid orange line).

M. Sabatelli et al, Deep Transfer Learning for Art Classification Problems, ECCV Workshops 2018

Fine-tuning a ResNet?



Fine-tuning a ResNet?



Preferably select layers to freeze main block

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

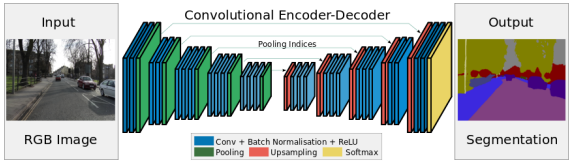
Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

Selecting which layer to freeze is less of a problem nowadays.
Common practice is to either freeze entire network or freeze only first two blocks (object detection) or fine-tune entire network.

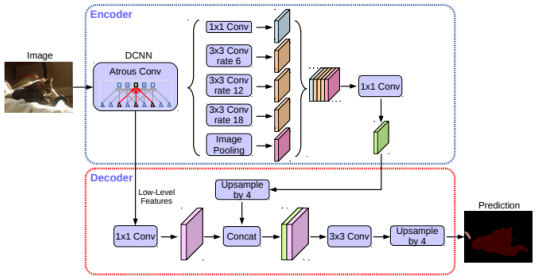
Pre-trained networks are the bread and butter in computer vision (at large) solutions, both academic and industrial.

Other communities are starting to follow this practice.

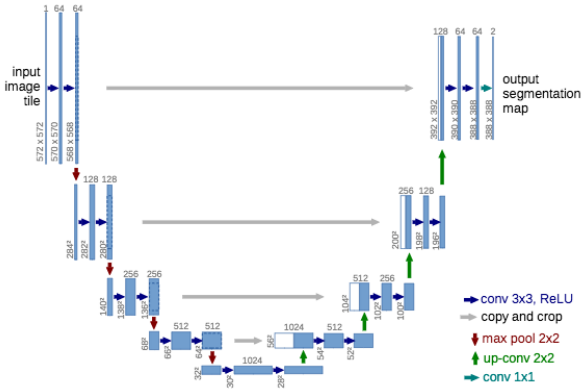
Pre-trained networks are the bread and butter of computer vision:
semantic segmentation



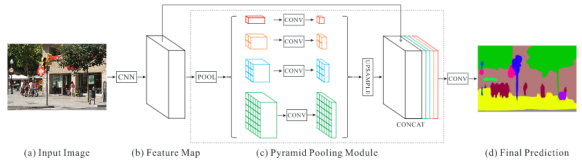
SegNet



DeepLabV3



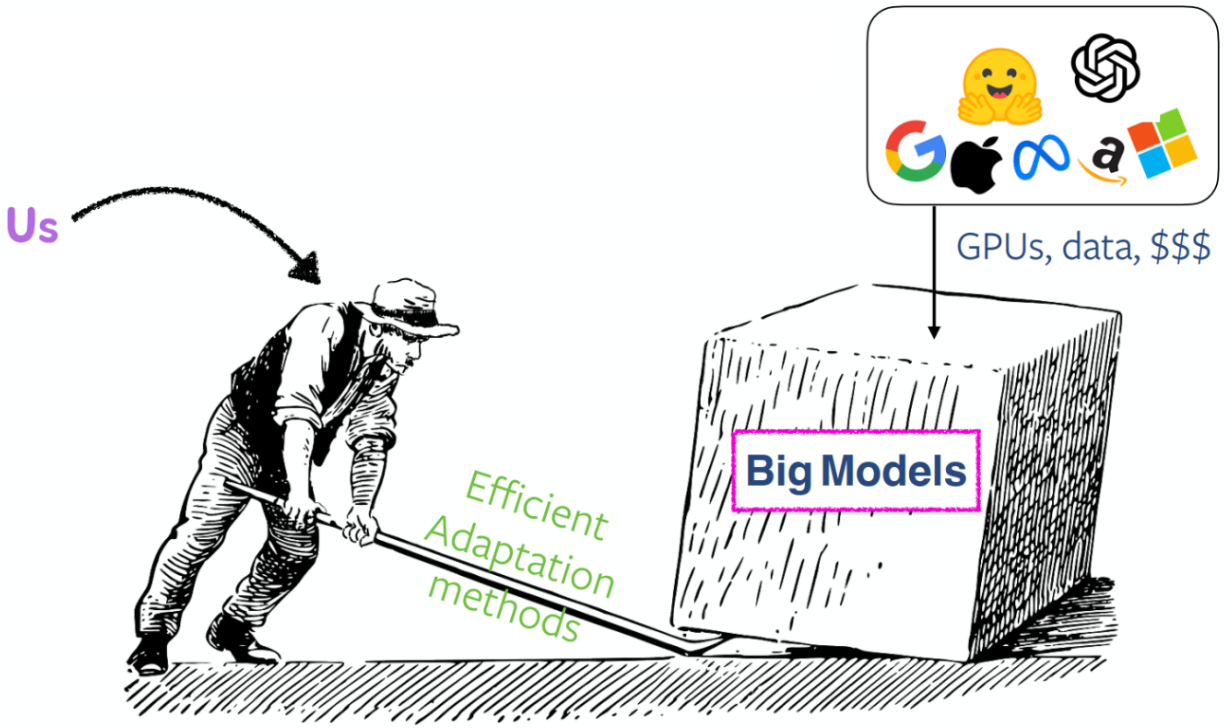
U-Net



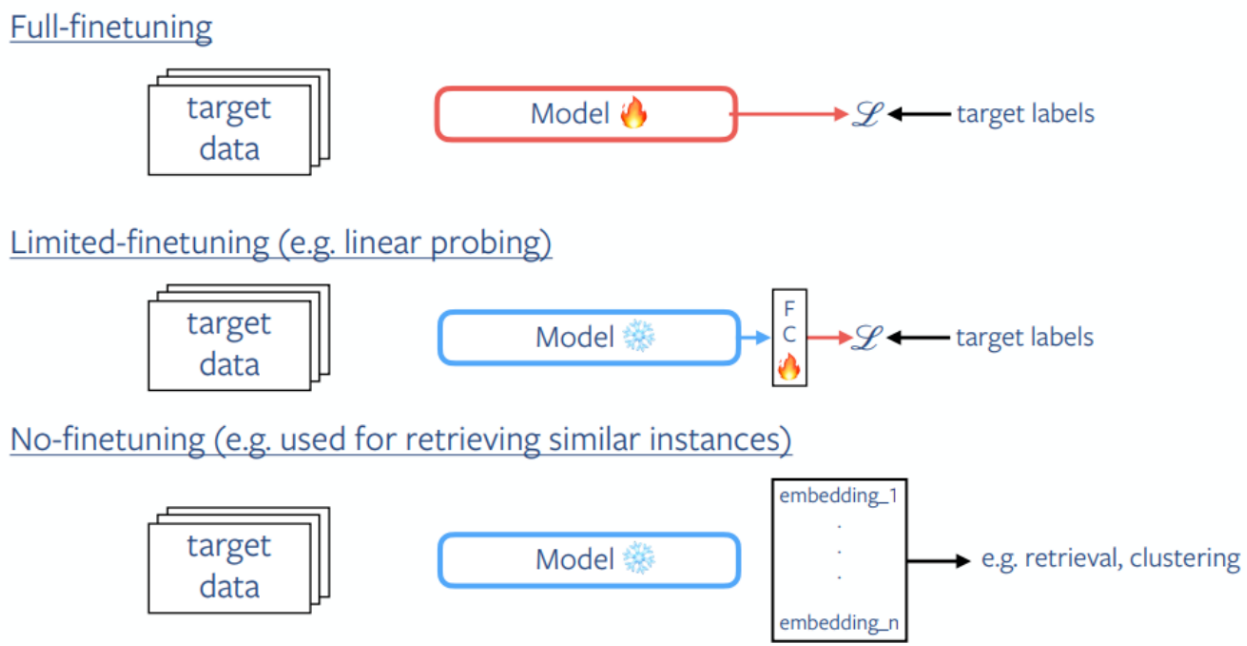
PSPNet

Parameter-efficient finetuning methods (PEFT)

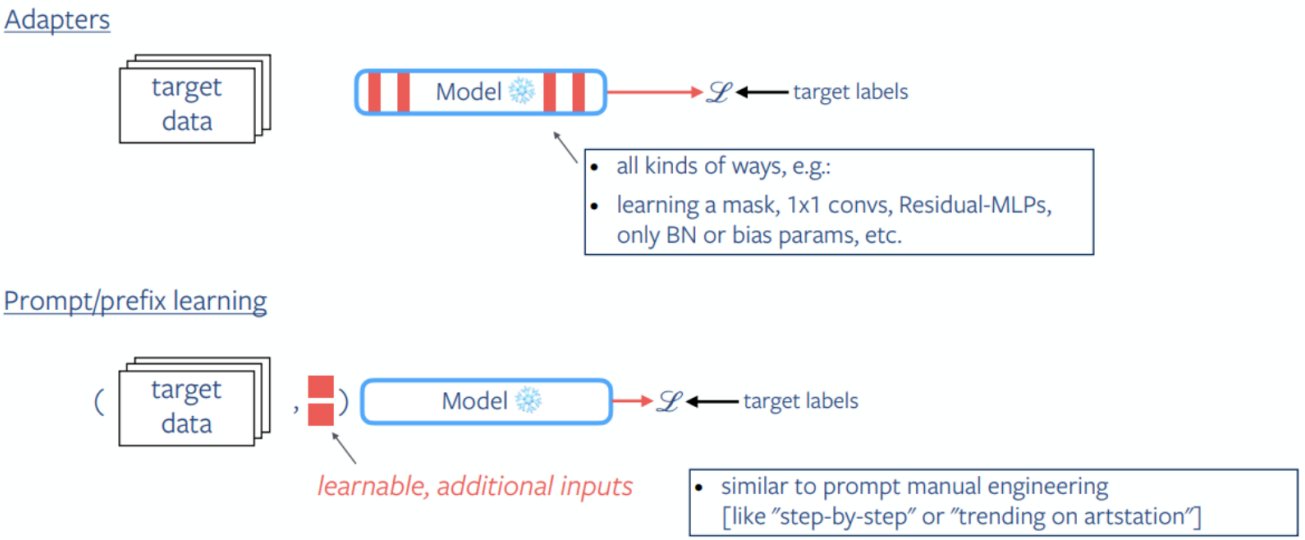
PEFTs



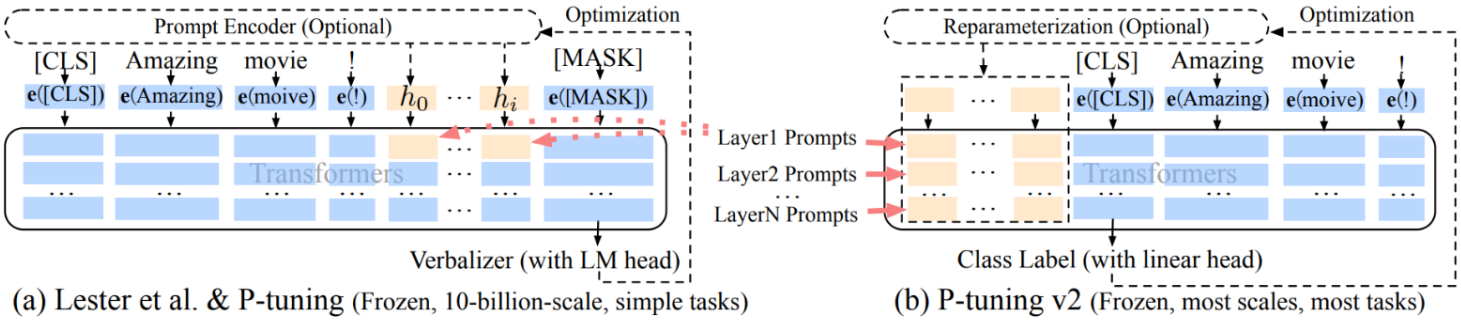
Main strategies to adapt models



Main strategies to adapt models



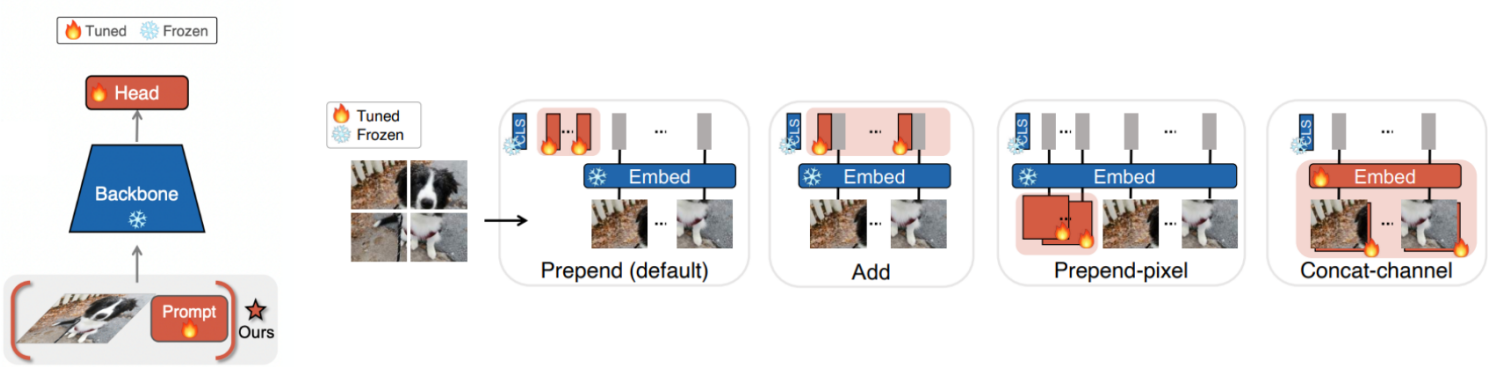
Prompt Learning



- prefixes are just learnable vectors
- extension to "deep prompt tuning"
- increases memory because of attention

X.L. Li et al., Prefix-Tuning: Optimizing Continuous Prompts for Generation, ACL 2021
X. Liu et al., P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks, ACL 2022

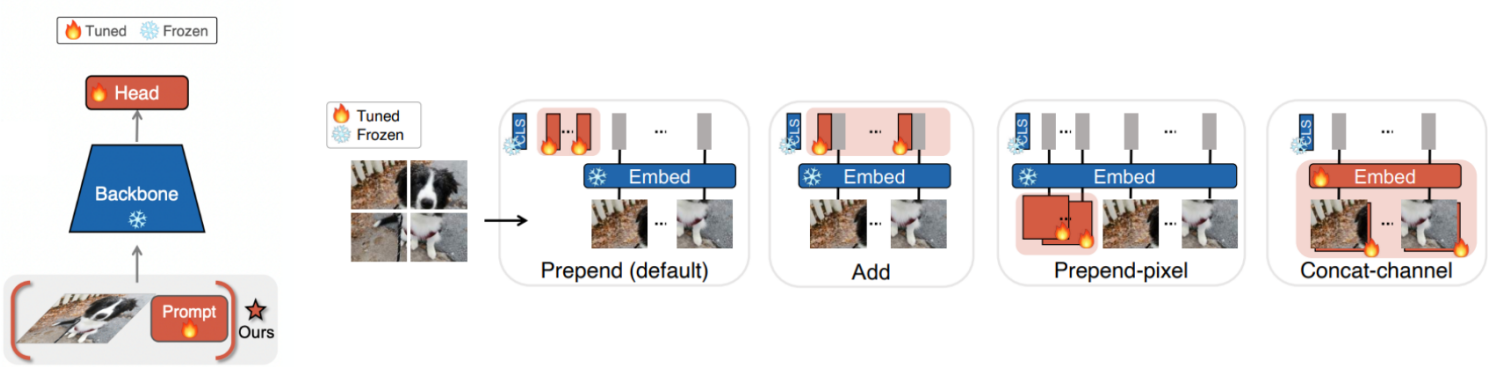
Prompt Learning



- explore various way of prompting for visual inputs
- also train a linear layer on top

M. Jia et al., Visual Prompt Tuning, ECCV 2022

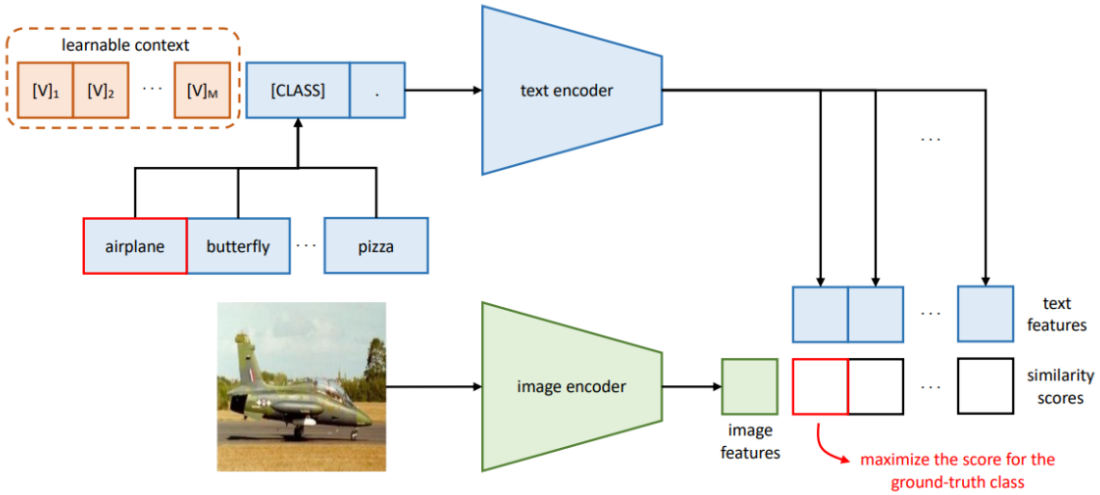
Prompt Learning



- explore various way of prompting for visual inputs
- also train a linear layer on top

M. Jia et al., Visual Prompt Tuning, ECCV 2022

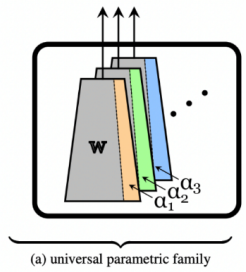
Prompt Learning



- CoOP: Context optimization
- learnable, vector-version of `"this is an [image/photograph/illustration] of {}"`

K. Zhou et al., Learning to Prompt for Vision-Language Models, IJCV 2022

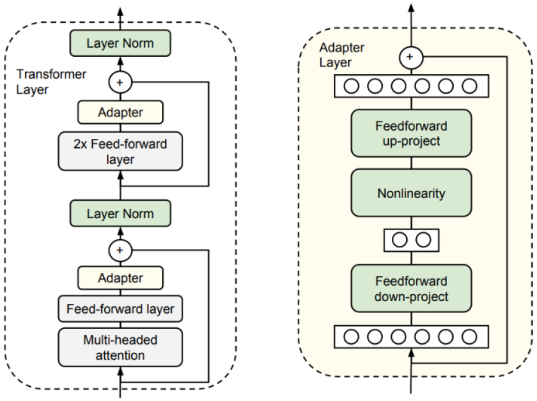
Adapters: changes in the middle of the DNN



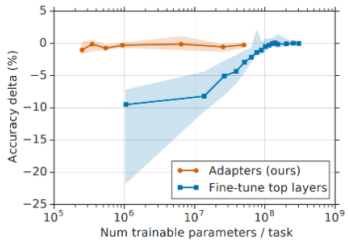
$$g(x;\alpha) = x + \alpha * x$$

Residual adapters

α can be conv1x1, BN scaling



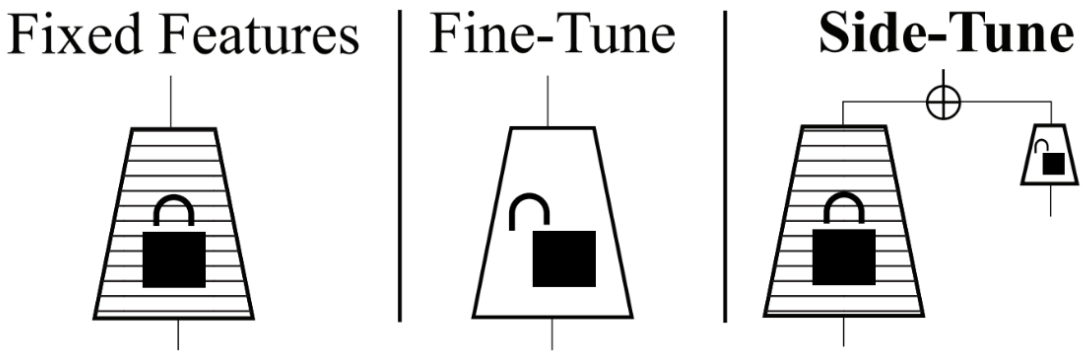
Transformer adapters



- 🙁: complex computation graph; inference time
- 🙂: limited memory to store, expressive, fast to learn

S. A. Rebuffi et al., Learning multiple visual domains with residual adapters, NeurIPS 2017
N. Houlsby et al., Parameter-Efficient Transfer Learning for NLP, ICML 2019

Side-tuning

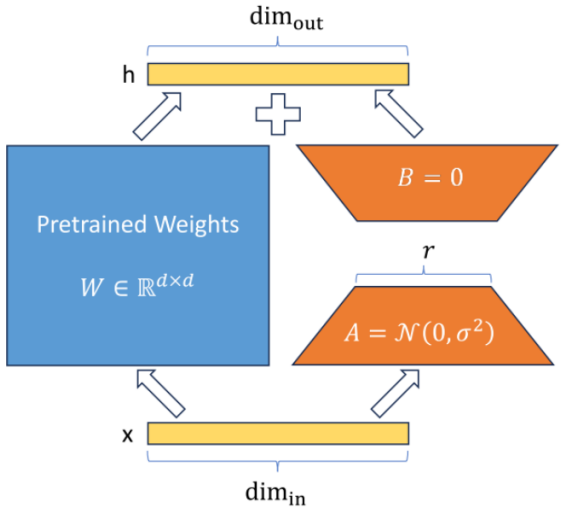


- Learn new network for missing knowledge
- Keep original network frozen

A. Sax et al., Side-Tuning: A Baseline for Network Adaptation via Additive Side Networks, ECCV 2020

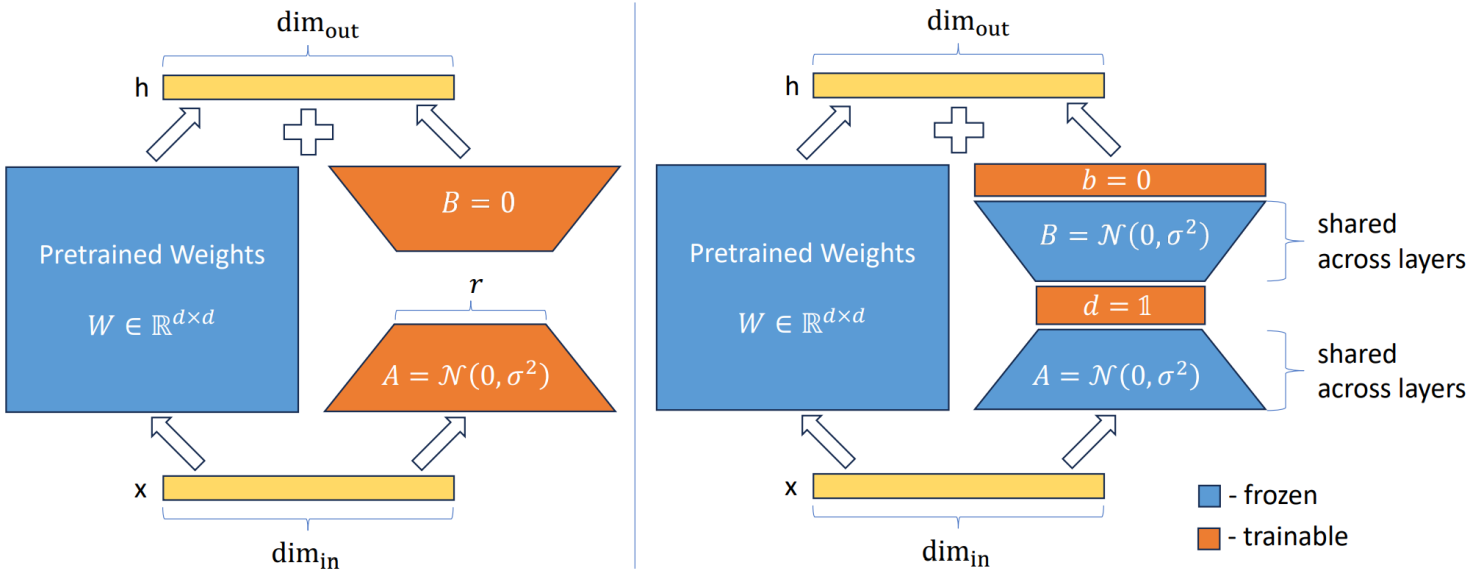
LoRA: Low-Rank Adaptation

- Adapting matrix multiplies in efficiently -> generalization of full fine-tuning
- Normal fully connected layer: $h = W_0 x$
- LoRA adapted: $h = W_0 x + \Delta W x = W_0 x + BAx$
- 🙂: linear operations, new weights can be fused with original weights
- 🙂: limited parameters trained, $< 1\%$ of parameters
- 🙁: less-expressive than adapters



E. Hu et al.,LoRA: Low-Rank Adaptation of Large Language Models, ICLR 2021

Many, many LoRA follow-ups out-there



D.W. Kopiczko et al., VeRA: Vector-based Random Matrix Adaptation, ICLR 2024

(Task) Transfer learning

Taskonomy: Disentangling Task Transfer Learning

A. Zamir et al., Taskonomy: Disentangling Task Transfer Learning, CVPR 2018

Question

- Some vision tasks are more inter-related than others:
  - depth estimation could help surface normals estimation?
  - scene layout could help object detection?
- for some task annotation data can be more easily obtainable than others
- could we find fully computational approach for modeling the structure of the space of visual tasks?

A. Zamir et al., Taskonomy: Disentangling Task Transfer Learning, CVPR 2018

Task bank/dictionary

- Task Bank
  - 26 semantic, 2d, 3d and other tasks
- Dataset
  - 4M real images
  - each image has the GT label for all tasks
- Task specific networks
  - 26 x

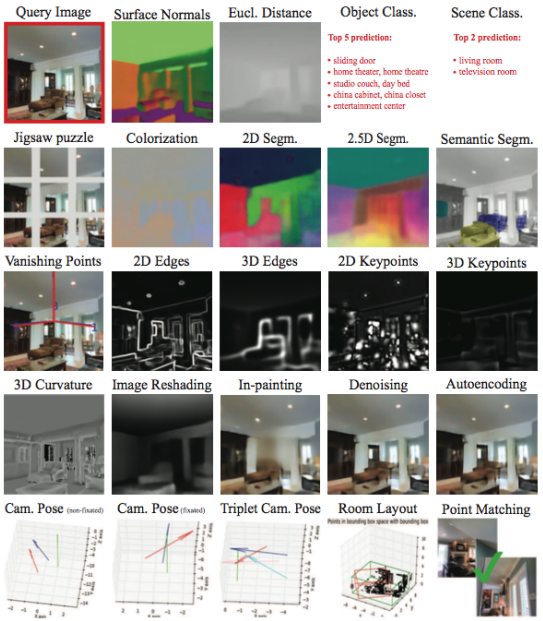
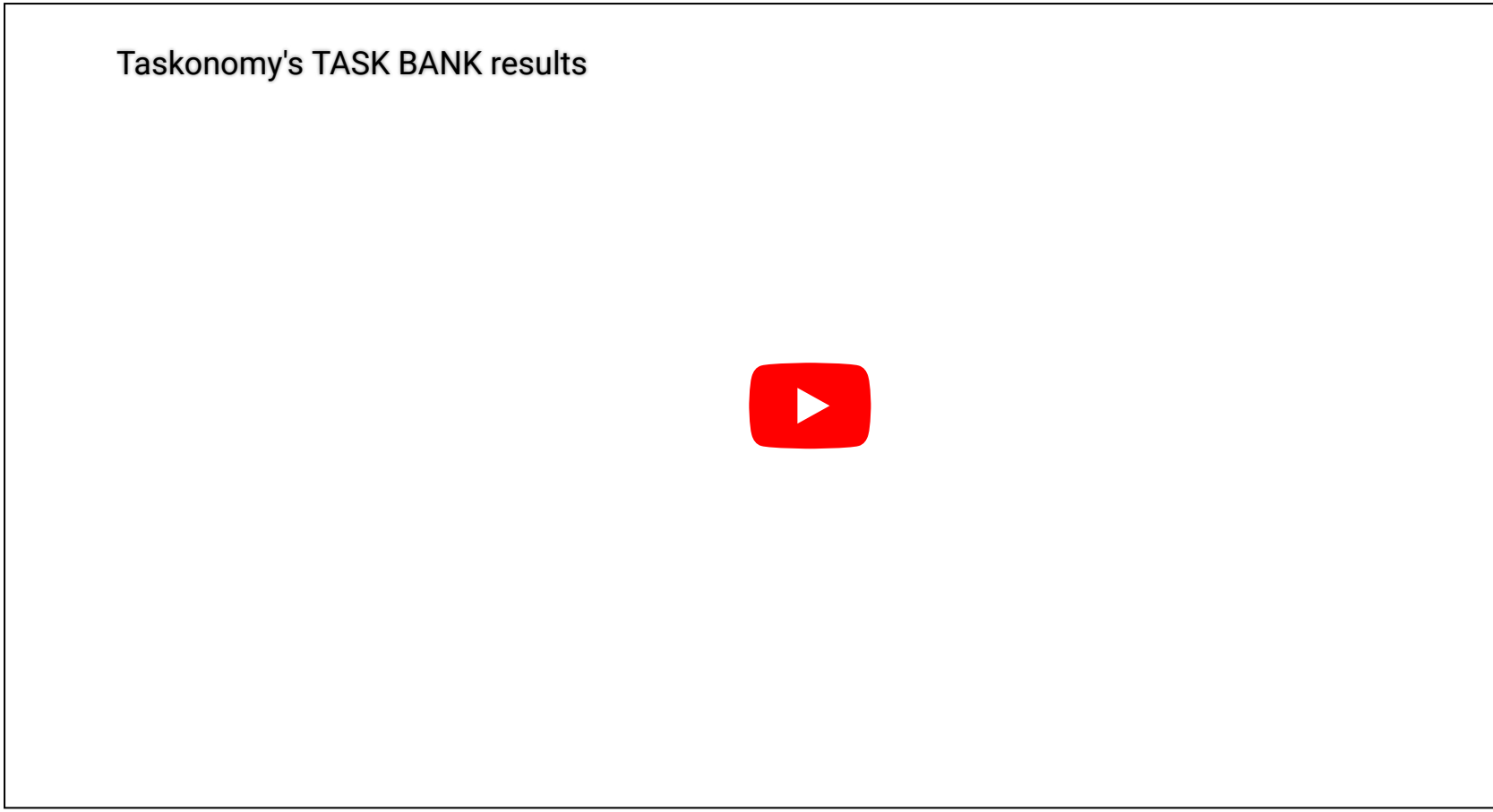


Figure 3: **Task Dictionary.** Outputs of 24 (of 26) task-specific networks for a query (top left). See results of applying frame-wise on a video here.

Task bank/dictionary



A. Zamir et al., Taskonomy: Disentangling Task Transfer Learning, CVPR 2018

Common architecture

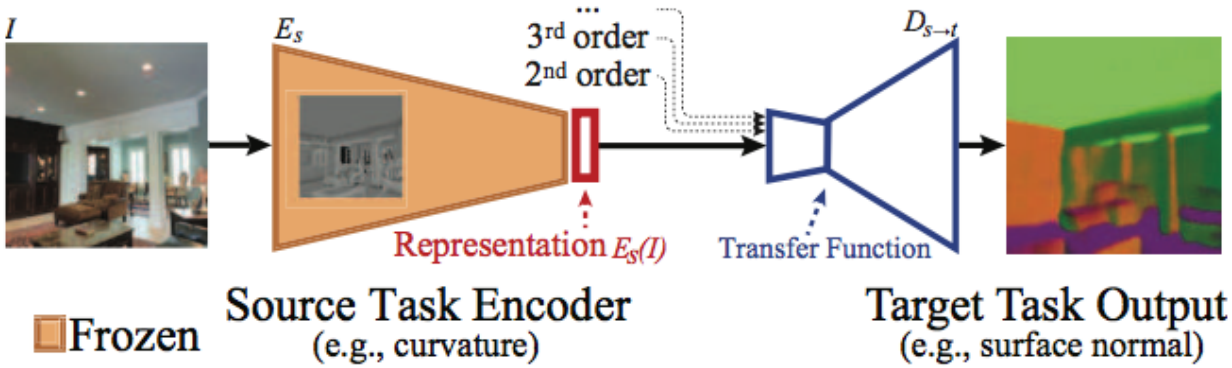


Figure 4: **Transfer Function**. A small readout function is trained to map representations of source task's frozen encoder to target task's labels. If order$>$ 1, transfer function receives representations from multiple sources.

- encoder: ResNet50
- transfer function: 2 conv layers
- decoder: 15 fully convolutional layers / 2-3 fully connected layers

A. Zamir et al., Taskonomy: Disentangling Task Transfer Learning, CVPR 2018

Common architecture

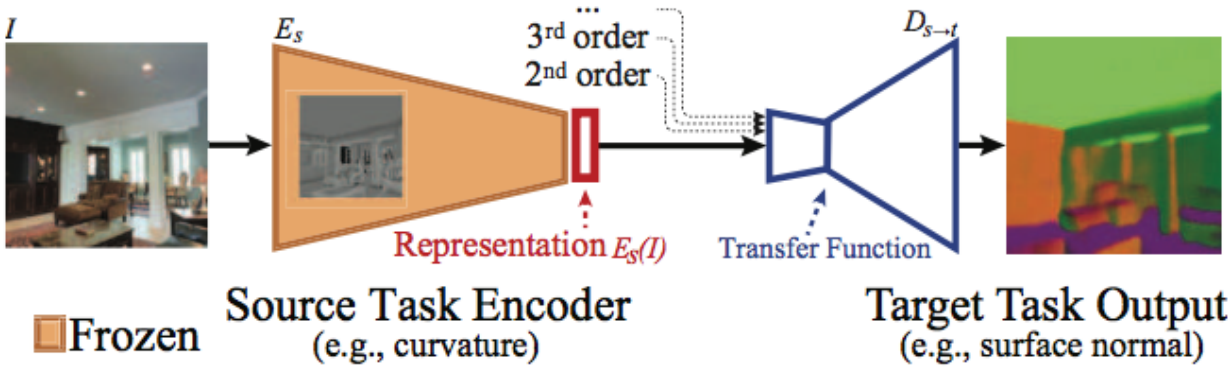


Figure 4: **Transfer Function**. A small readout function is trained to map representations of source task's frozen encoder to target task's labels. If order>1, transfer function receives representations from multiple sources.

- full training (gold standard): $120k$ training, $16k$ validation, $17k$ testing
- fine-tuning: $1-16k$

A. Zamir et al., Taskonomy: Disentangling Task Transfer Learning, CVPR 2018

Computational model

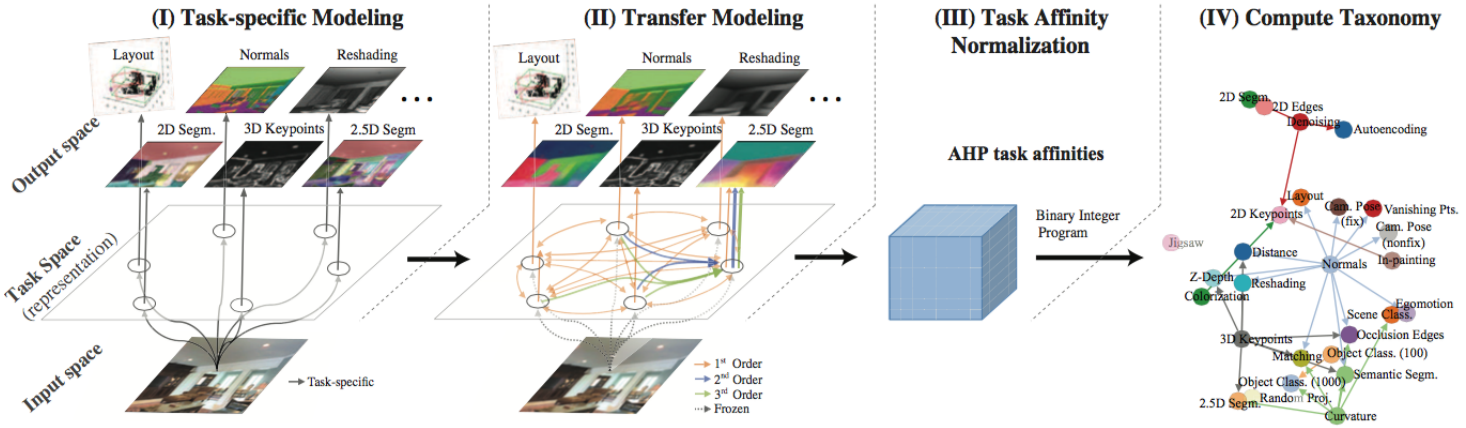


Figure 2: **Computational modeling of task relations and creating the taxonomy.** From left to right: I. Train task-specific networks. II. Train (first order and higher) transfer functions among tasks in a latent space. III. Get normalized transfer affinities using AHP (Analytic Hierarchy Process). IV. Find global transfer taxonomy using BIP (Binary Integer Program).

A. Zamir et al., Taskonomy: Disentangling Task Transfer Learning, CVPR 2018

Computed taxonomies

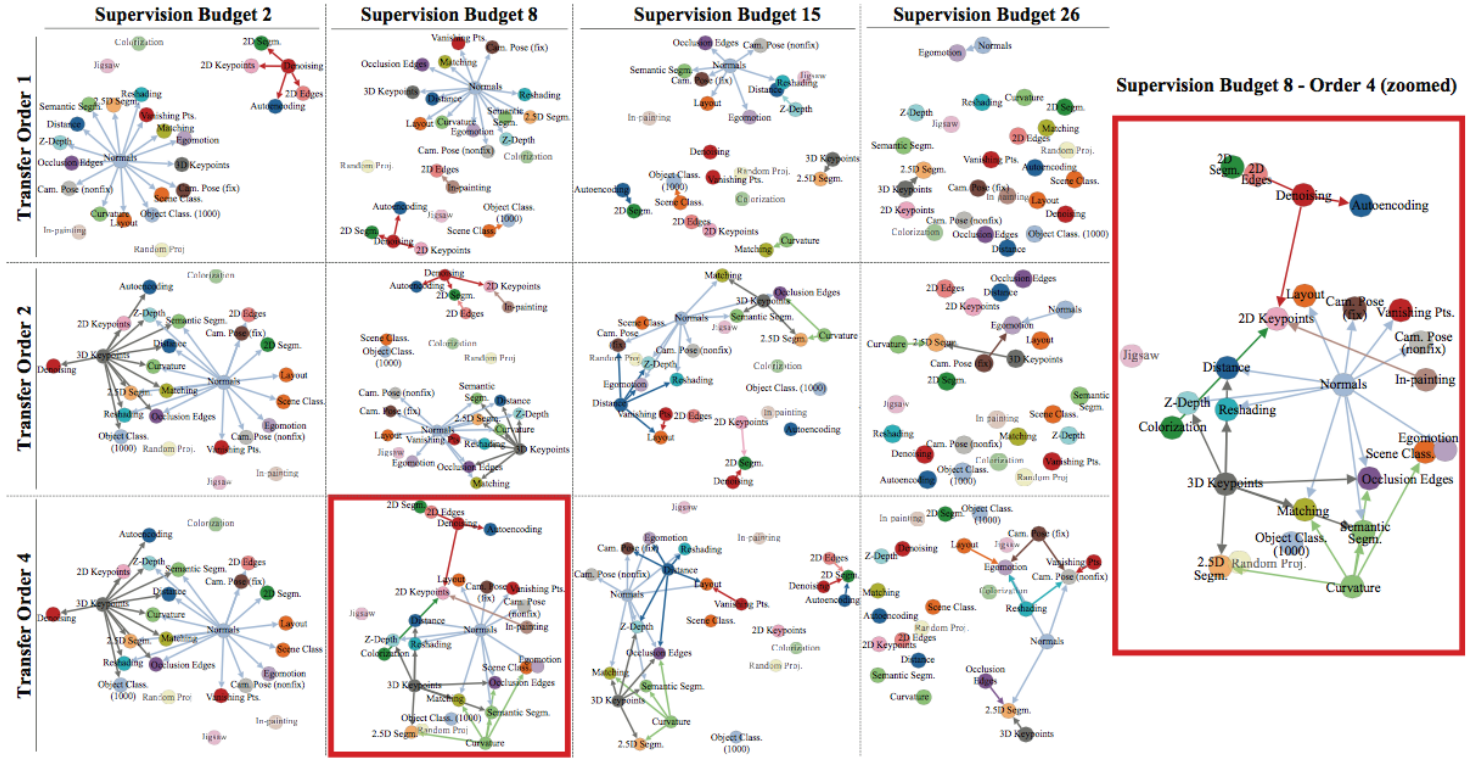


Figure 8: **Computed taxonomies** for solving 22 tasks given various supervision budgets (x-axes), and maximum allowed transfer orders (y-axes). One is magnified for better visibility. Nodes with incoming edges are target tasks, and the number of their incoming edges is the order of their chosen transfer function. Still transferring to some targets when tge budget is 26 (full budget) means certain transfers started performing better than their fully supervised task-specific counterpart. See the interactive solver website for color coding of the nodes by *Gain* and *Quality* metrics. Dimmed nodes are the source-only tasks, and thus, only participate in the taxonomy if found worthwhile by the BIP optimization to be one of the sources.

A. Zamir et al., Taskonomy: Disentangling Task Transfer Learning, CVPR 2018

Multi-Task Learning (MTL)

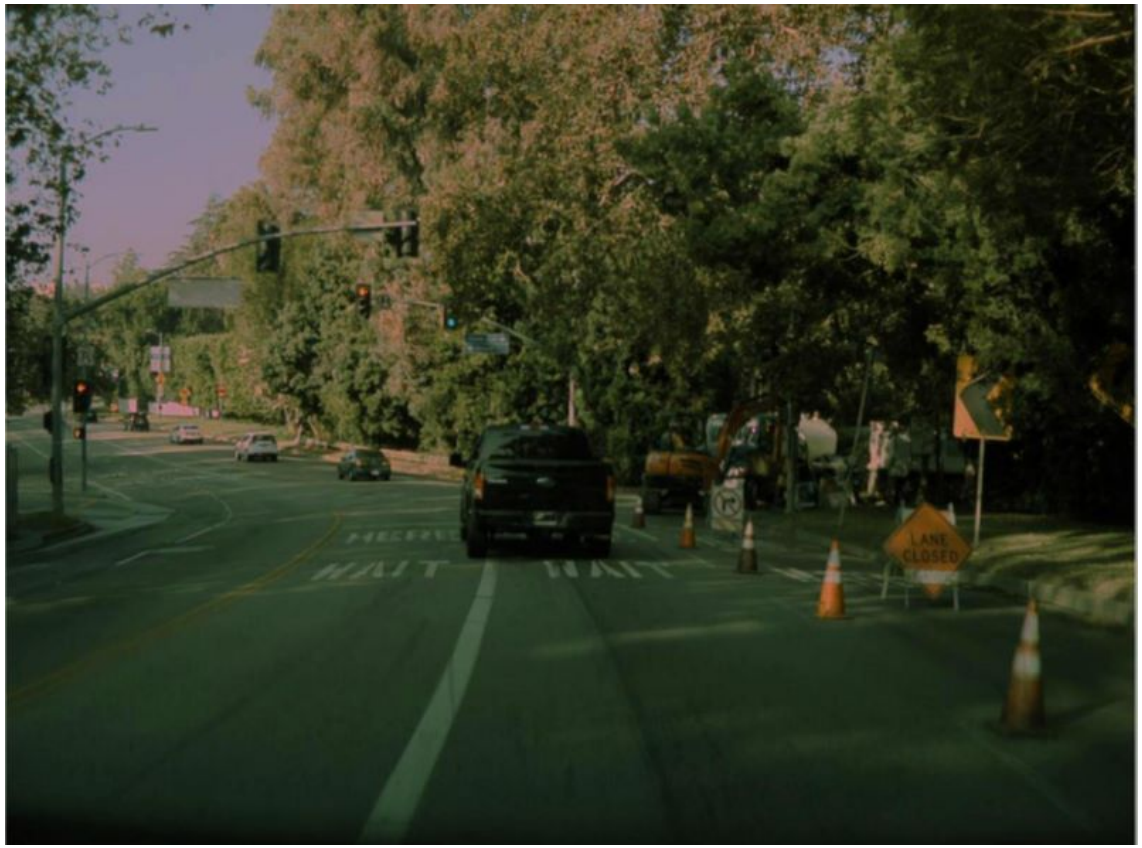


Image credit: A. Karpathy

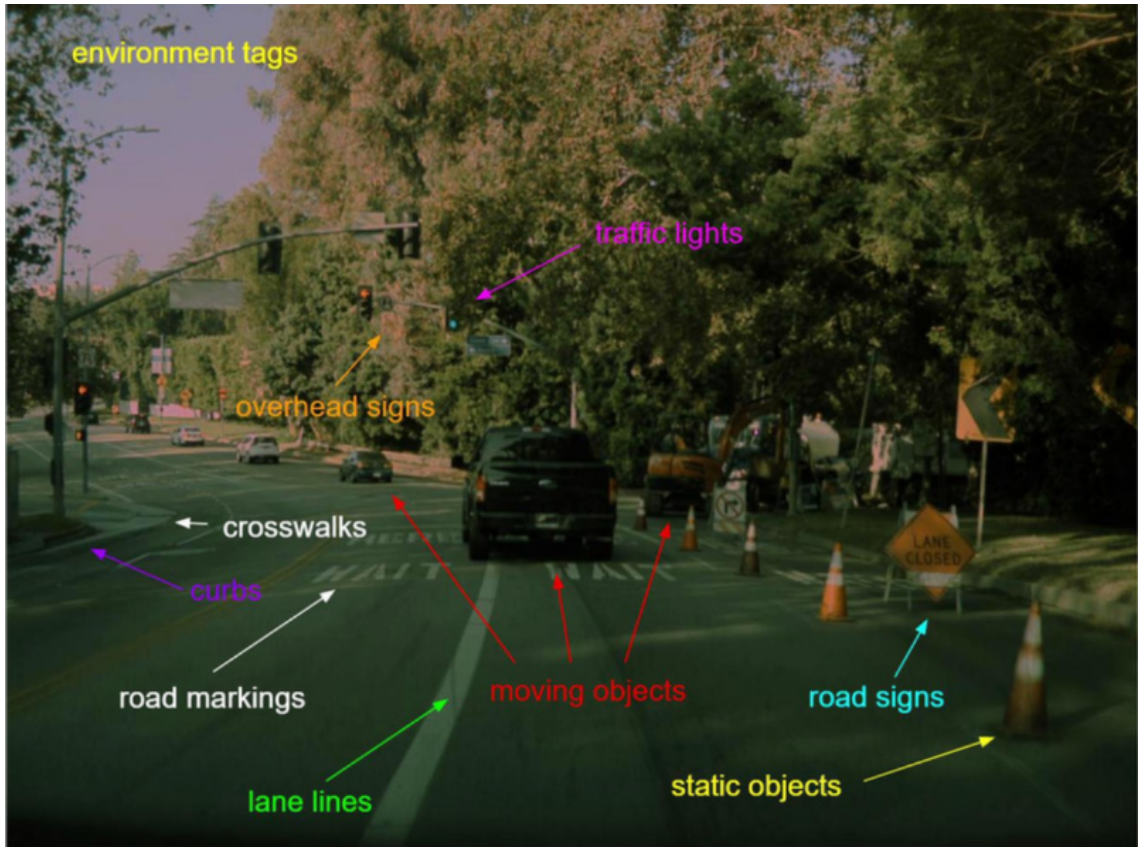


Image credit: A. Karpathy

In an autonomous vehicle there are plenty of perception tasks that need to be simultaneously adddressed.

In an autonomous vehicle there are plenty of perception tasks that need to be simultaneously adddressed.

We have two options:

1. train multiple models (one for each task)

In an autonomous vehicle there are plenty of perception tasks that need to be simultaneously adddressed.

We have two options:

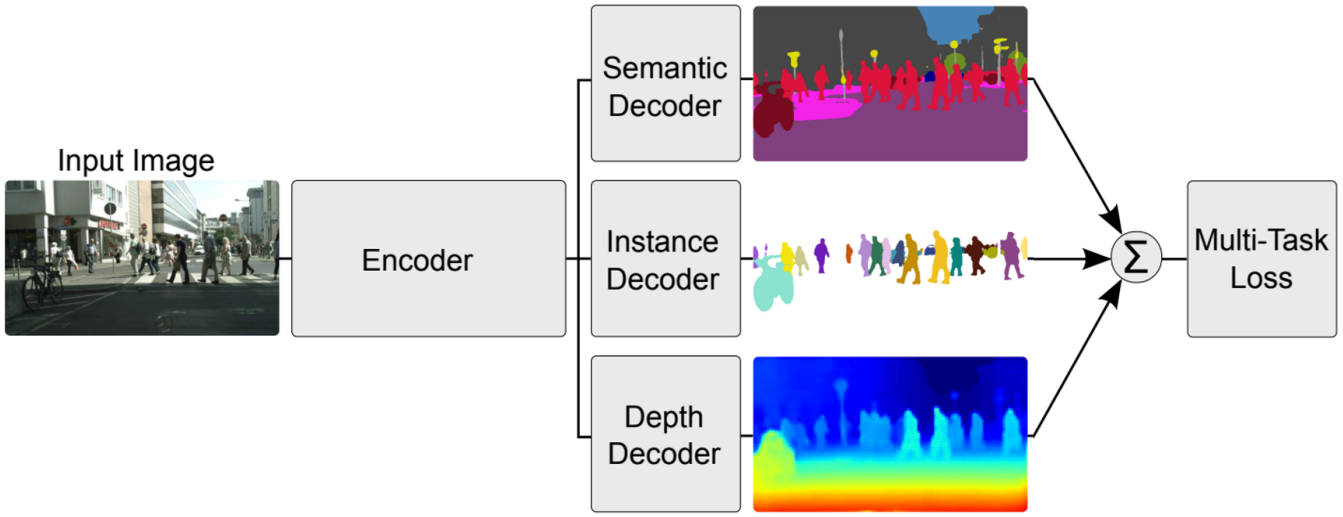
1. train multiple models (one for each task)
2. inject all knowledge in a single model

In an autonomous vehicle there are plenty of perception tasks that need to be simultaneously adddressed.

We have two options:

1. train multiple models (one for each task)
2. inject all knowledge in a single model

Multi-Task Learning

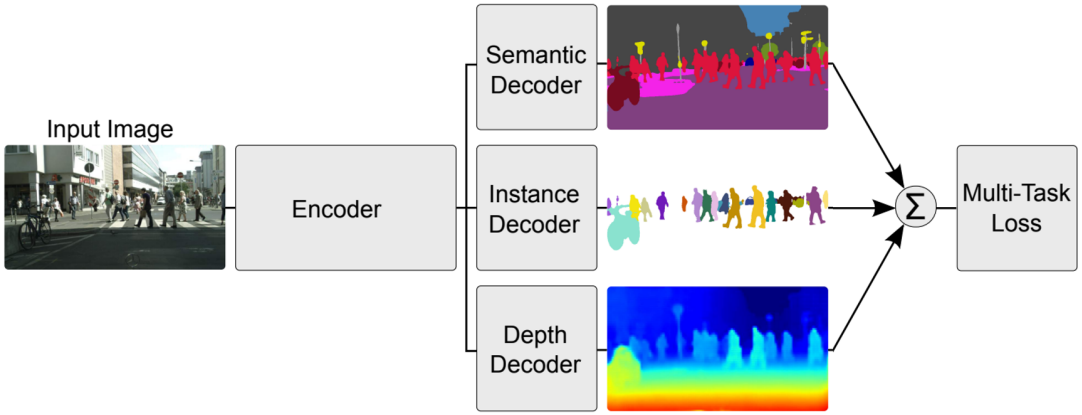


In MTL we usually have a *shared backbone* (encoder) and multiple *"heads"*, 1+ for each task.

The encoder learns useful features for all tasks, while the *"heads"* are specialized.

Image credit: A. Kendall

Multi-Task Learning



$$L_{total} = \sum_i w_i L_i$$

$$\sum_i w_i = 1$$

Each task has its own loss and the total loss is a weighted combination.

Image credit: A. Kendall

Multi-Task Learning in PyTorch

```
class MTLNet(nn.Module):
    def __init__(self):
        super(MTLNet, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(input_size, encoder_size),
            nn.ReLU()
        )

        self.head1 = nn.Sequential(
            nn.Linear(encoder_size, h1_size),
            nn.ReLU(),
            nn.Linear(h1_size, output1_size)
        )

        self.head2 = nn.Sequential(
            nn.Linear(encoder_size, h2_size),
            nn.ReLU(),
            nn.Linear(h2_size, output2_size)
        )

    def forward(self, x):
        shared_features = self.encoder(x)
        out1 = self.head1(shared_features)
        out2 = self.head2(shared_features)
        return out1, out2
```

Multi-Task Learning in PyTorch

```python
class MTLNet(nn.Module):
    def __init__(self):
        super(MTLNet, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(input_size, encoder_size),
            nn.ReLU()
        )

        self.head1 = nn.Sequential(
            nn.Linear(encoder_size, h1_size),
            nn.ReLU(),
            nn.Linear(h1_size, output1_size)
        )

        self.head2 = nn.Sequential(
            nn.Linear(encoder_size, h2_size),
            nn.ReLU(),
            nn.Linear(h2_size, output2_size)
        )

    def forward(self, x):
        shared_features = self.encoder(x)
        out1 = self.head1(shared_features)
        out2 = self.head2(shared_features)
        return out1, out2
```

Multi-Task Learning in PyTorch

```python
class MTLNet(nn.Module):
    def __init__(self):
        super(MTLNet, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(input_size, encoder_size),
            nn.ReLU()
        )

        self.head1 = nn.Sequential(
            nn.Linear(encoder_size, h1_size),
            nn.ReLU(),
            nn.Linear(h1_size, output1_size)
        )

        self.head2 = nn.Sequential(
            nn.Linear(encoder_size, h2_size),
            nn.ReLU(),
            nn.Linear(h2_size, output2_size)
        )

    def forward(self, x):
        shared_features = self.encoder(x)
        out1 = self.head1(shared_features)
        out2 = self.head2(shared_features)
        return out1, out2
```

```python
for i, (x, y_task1, y_task2) in enumerate(train_loader
    ...
    y_pred1, y_pred2 = mtl_net(x)

    loss1 = criterion(y_pred1, y_task1)
    loss2 = criterion(y_pred2, y_task2)
    loss = weight1 * loss1 + weight2 * loss2

    optimizer.zero_grad()

    loss.backward()

    optimizer.step()
    ...
```

Multi-Task Learning in PyTorch

```python
class MTLNet(nn.Module):
    def __init__(self):
        super(MTLNet, self).__init__()

        self.encoder = nn.Sequential(
            nn.Linear(input_size, encoder_size),
            nn.ReLU()
        )

        self.head1 = nn.Sequential(
            nn.Linear(encoder_size, h1_size),
            nn.ReLU(),
            nn.Linear(h1_size, output1_size)
        )

        self.head2 = nn.Sequential(
            nn.Linear(encoder_size, h2_size),
            nn.ReLU(),
            nn.Linear(h2_size, output2_size)
        )

    def forward(self, x):
        shared_features = self.encoder(x)
        out1 = self.head1(shared_features)
        out2 = self.head2(shared_features)
        return out1, out2
```

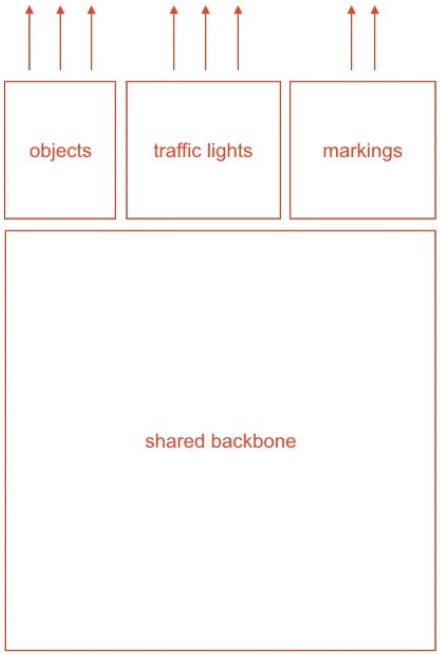
```python
for i, (x, y_task1, y_task2) in enumerate(train_loader
    ...
    y_pred1, y_pred2 = mtl_net(x)

    loss1 = criterion(y_pred1, y_task1)
    loss2 = criterion(y_pred2, y_task2)
    loss = weight1 * loss1 + weight2 * loss2

    optimizer.zero_grad()

    loss.backward()

    optimizer.step()
    ...
```



- Usually there are $2-5$ tasks per network
- In some cases, on autonomous vehicles you can have up to 50 tasks per network.

Image credit: A. Karpathy

Pros

- highly practical solution
- shared encoder can be shared among many tasks
- MTL can lead to better regularization *[Caruana (1993)]*

R. Caruana, Multitask learning: A knowledge-based source of inductive bias, ICML 1993

Pros

- highly practical solution
- shared encoder can be shared among many tasks
- MTL can lead to better regularization *[Caruana (1993)]*

Cons

- balancing difficulties and impact of each task is non-trivial:
  - normalize gradients from all heads *[Chen et al. (2018)]*
  - learn per task weights *[Kendall et al. (2018), Leang et al. (2020)]*
  - learn a scheduling for training heads *[Leang et al. (2020)]*
  - no weighting at all, but more regularization *[Kurin et al. (2022)]*

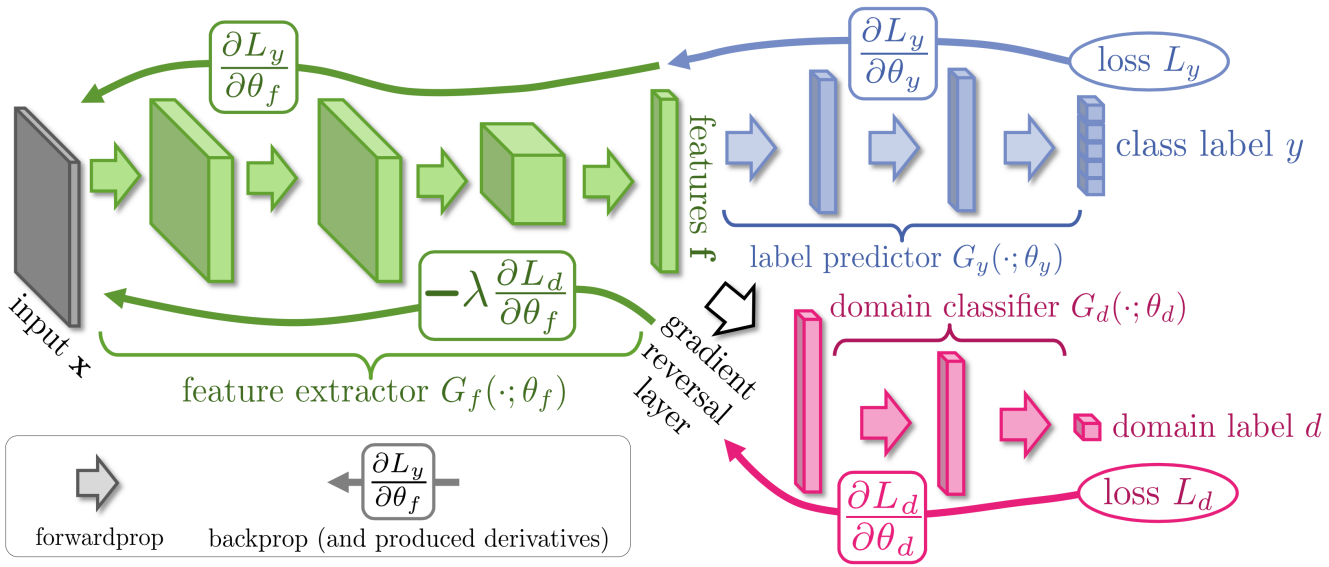
R. Caruana, Multitask learning: A knowledge-based source of inductive bias, ICML 1993
Z. Chen et al., Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, ICML 2018.
A. Kendall et al., Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, CVPR 2018
I. Leang et al., Dynamic Task Weighting Methods for Multi-task Networks in Autonomous Driving Systems, ITSC 2020
V. Kurin et al., In Defense of the Unitary Scalarization for Deep Multi-Task Learning, arXiv 2022

Domain Adaptation

Example: Internet images → webcam

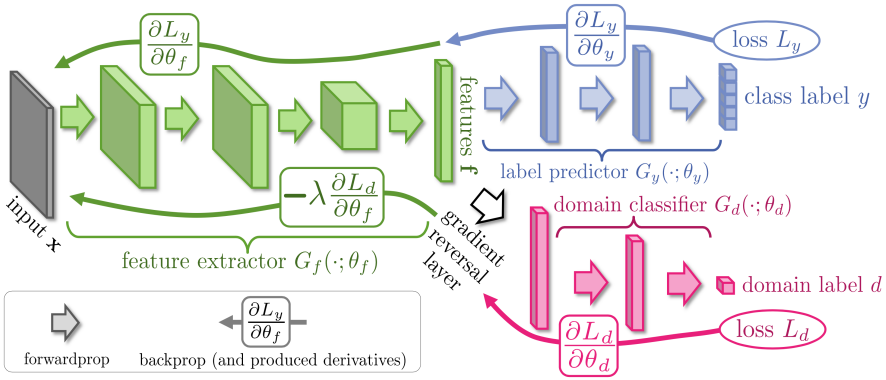
Example: (semi-)synthetic → real

Domain-adversarial networks



Y. Ganin et al., Domain-Adversarial Training of Neural Networks, JMLR 2016

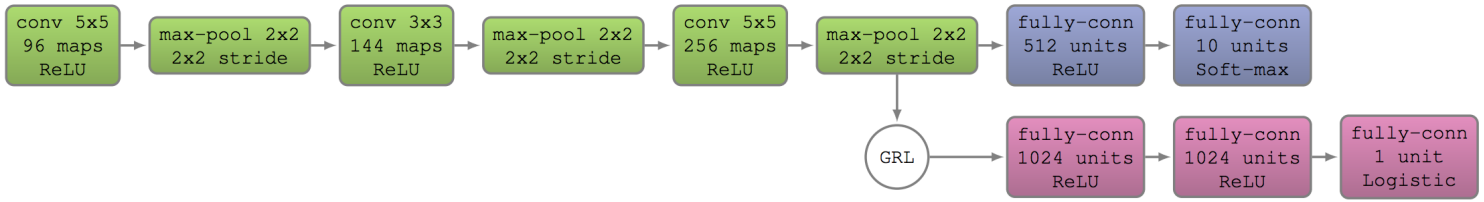
Domain-adversarial networks



- build a network
- train feature extractor + class predictor on source data
- train feature extractor + domain classifier on source+target data
- use feature extractor + class predictor at test time

Y. Ganin et al., Domain-Adversarial Training of Neural Networks, JMLR 2016

Domain-adversarial networks

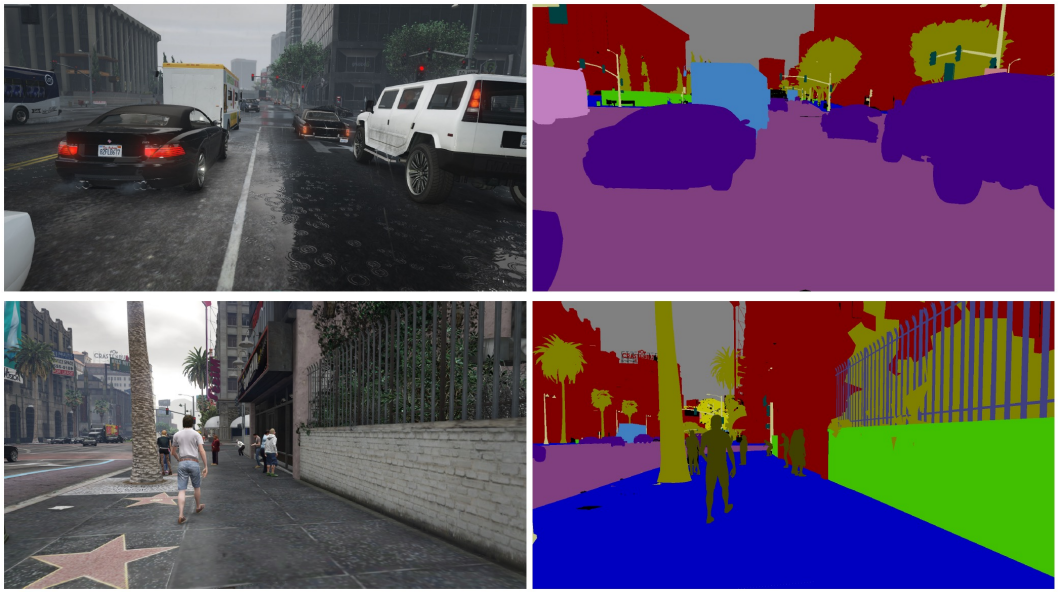


- build a network
- train feature extractor + class predictor on source data
- train feature extractor + domain classifier on source+target data
- use feature extractor + class predictor at test time

Y. Ganin et al., Domain-Adversarial Training of Neural Networks, JMLR 2016

Large-gap domain adaptation

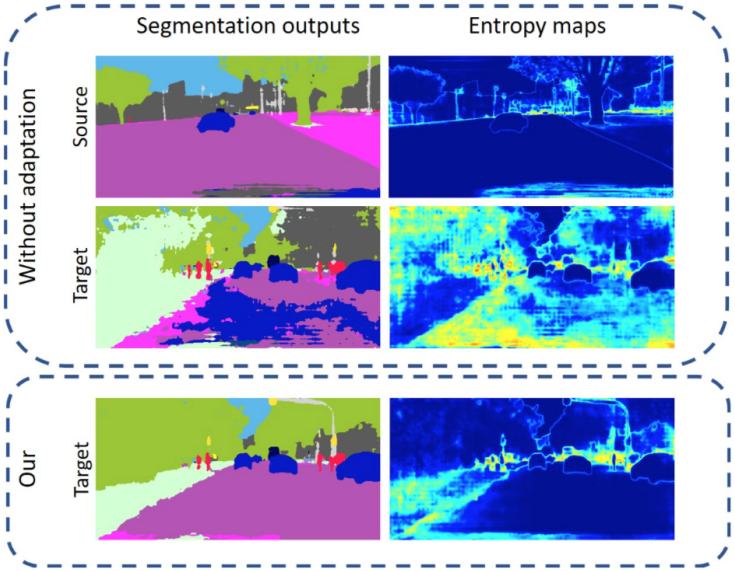
Using video games to generate training data



GTA V

S. Richter et al., Playing for Data: Ground Truth from Computer Games, ECCV 2016

Unsupervised domain adaptation (UDA)

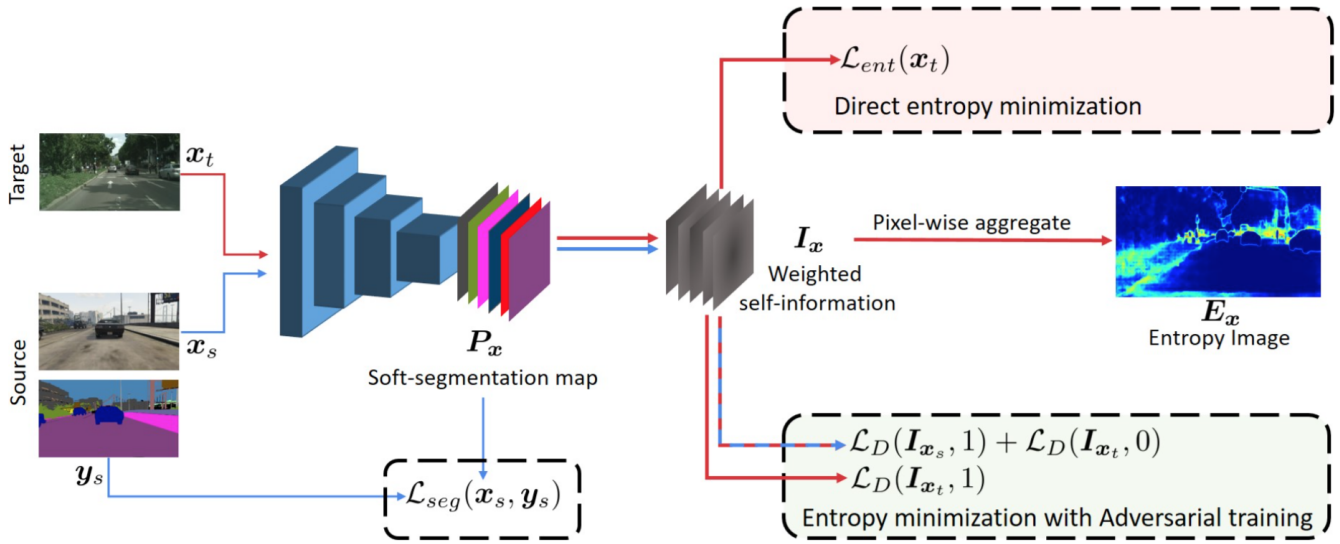


Directly testing on the target data is not quite optimal

Predictions on target domain have higher entropy ⟶ what if we just minimize entropy during training?

T. Vu et al., ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation, CVPR 2019

Unsupervised domain adaptation (UDA)



ADVENT - Adversarial Entropy Minimization for Domain Adaptation

If you squint a bit you might recognize a form of *multi-task learning* here.

T. Vu et al., ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation, CVPR 2019

Unsupervised domain adaptation (UDA)

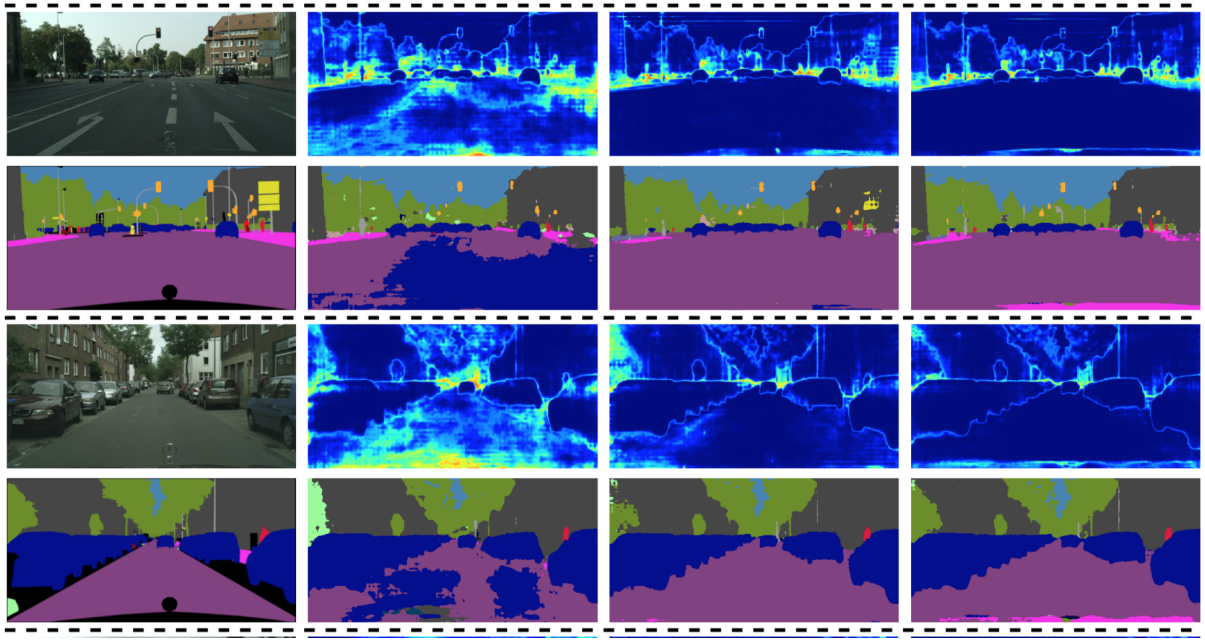
(a) GTA5 → Cityscapes

Method	UDA Model	Oracle	mIoU Gap (%)
FCNs in the Wild [15]	27.1	64.6	-37.5
CyCADA [14]	28.9	60.3	-31.4
Adapt-SegNet [41]	35.0	61.8	-25.2
Ours (single model)	35.6	61.8	-24.6
Adapt-SegNet [41]	42.4	65.1	-22.7
Ours (single model)	43.6	65.1	-21.5
Ours (two models)	44.8	65.1	**-20.3**

GAP w.r.t. oracle, i.e. when training on the target data.
Top: VGG-16 encoder; bottom ResNet-101 encoder.

T. Vu et al., ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation, CVPR 2019

Unsupervised domain adaptation (UDA)



Semantic segmentation: from GTA V → Cityscapes

T. Vu et al., ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation, CVPR 2019

Unsupervised domain adaptation (UDA)

Object detection: from Cityscapes → Foggy Cityscapes (synthetic fog)

T. Vu et al., ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation, CVPR 2019

A small detour

A common trick used in UDA and semi-supervised learning is self-training.

A common trick used in UDA and semi-supervised learning is self-training.

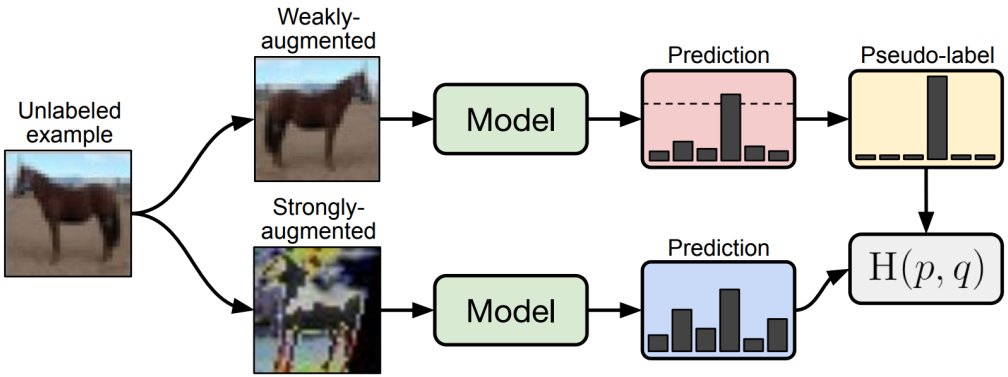
In self-training a network is trained using as ground-truth its own predictions or predictions from another network. Such labels are called pseudo-labels.

A common trick used in UDA and semi-supervised learning is self-training.

In self-training a network is trained using as ground-truth its own predictions or predictions from another network. Such labels are called pseudo-labels.

In semi-supervised learning we assume we have a dataset with
labeled samples and
a (bigger) dataset with unlabeled images.

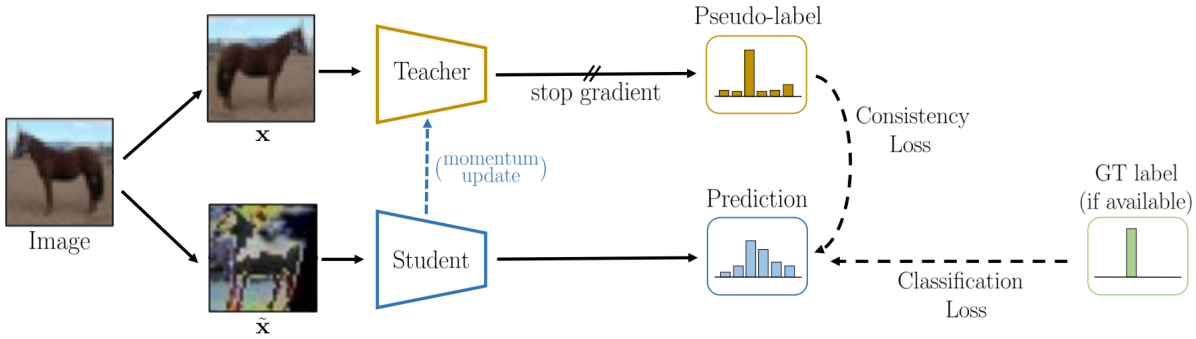
FixMatch



- apply data augmentation to unlabeled and labeled images
- use pseudo-labels as ground-truth for unlabeled images
- for labeled images use classification loss

K. Sohn et al., FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence, NeurIPS 2020

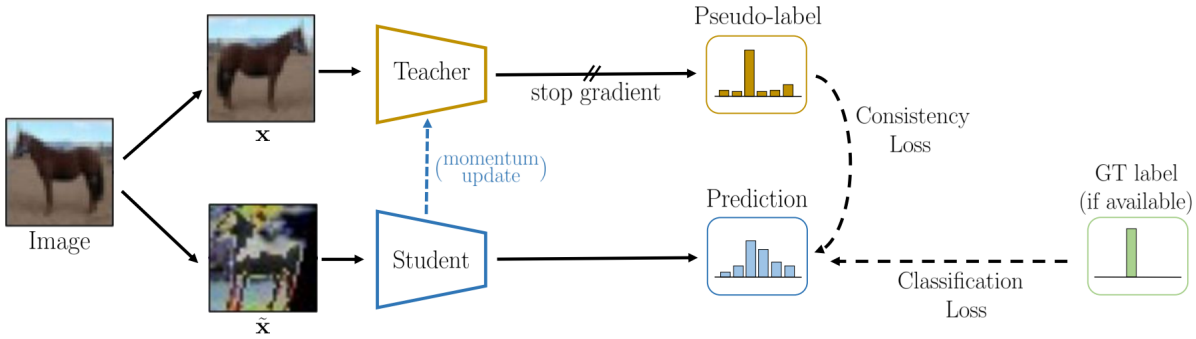
Teacher-student approaches



- teacher: generate target classification predictions from an image
- student: trained to predict this target given a different random view of the same images
- the teacher is a moving average of the student weights in time

A. Tarvainen and H. Valpola, Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, NeurIPS 2017

Teacher-student approaches

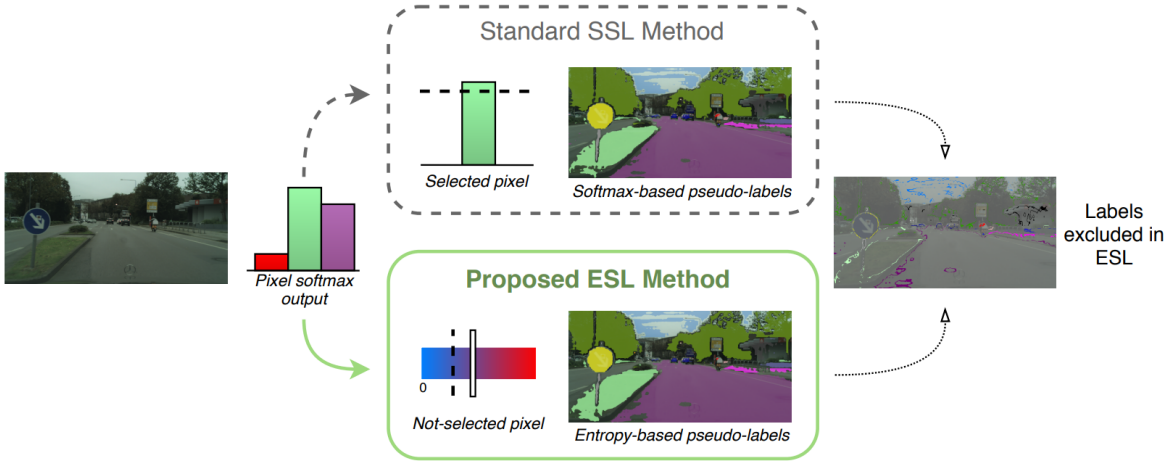


- teacher: generate target classification predictions from an image
- student: trained to predict this target given a different random view of the same images
- the teacher is a moving average of the student weights in time
- previuos methods averaged teacher ensemble predictions

A. Tarvainen and H. Valpola, Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, NeurIPS 2017

Back to our regular program

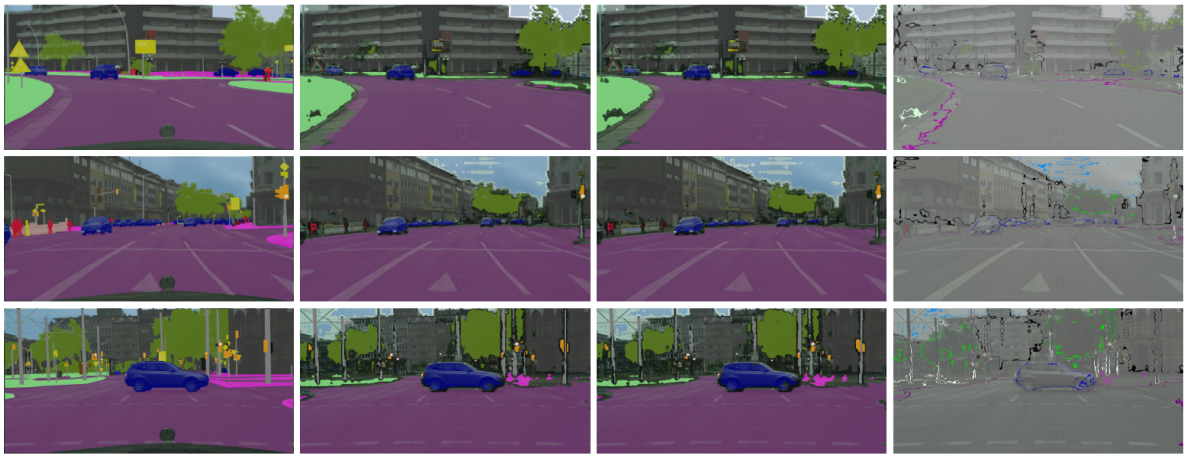
UDA + self-training



- principle: find confident pixels in target images, select them and use as ground truth
- selection criteria: entropy of predictions

A. Saporta et al., ESL: Entropy-guided Self-supervised Learning for Domain Adaptation in Semantic Segmentation, CVPRW 2020

UDA + self-training

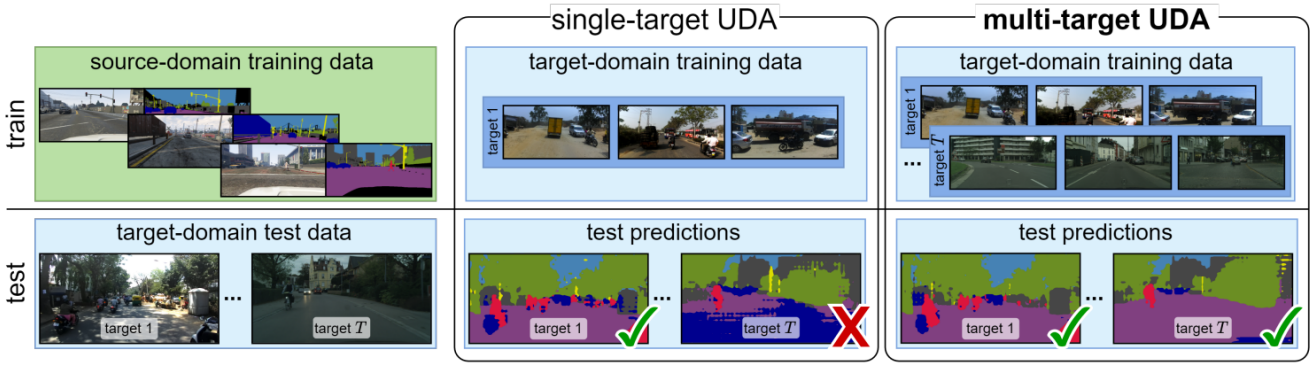


(a) Ground-truth labels (b) Softmax-based pseudo-labels (c) Entropy-based pseudo-labels (d) Excluded by entropy criterion

Table 4: Comparison of incorrect predictions (in %) selected in the pseudo-labels extracted (ADVENT [23])

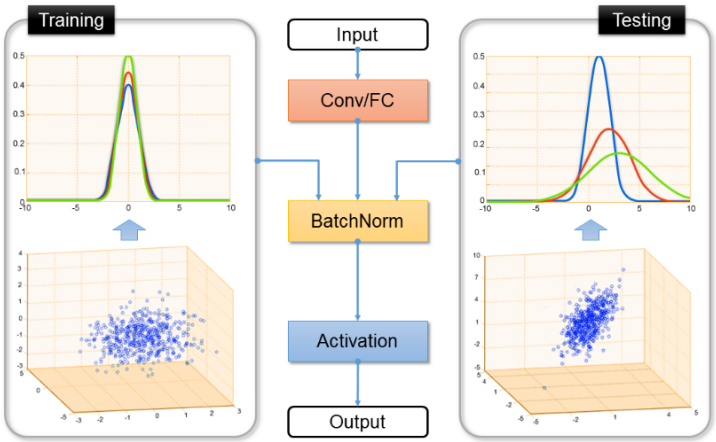
A. Saporta et al., ESL: Entropy-guided Self-supervised Learning for Domain Adaptation in Semantic Segmentation, CVPRW 2020

Multi-target UDA



A. Saporta et al., Multi-Target Adversarial Frameworks for Domain Adaptation in Semantic Segmentation, ICCV 2021

Domain Adaptation by BatchNorm



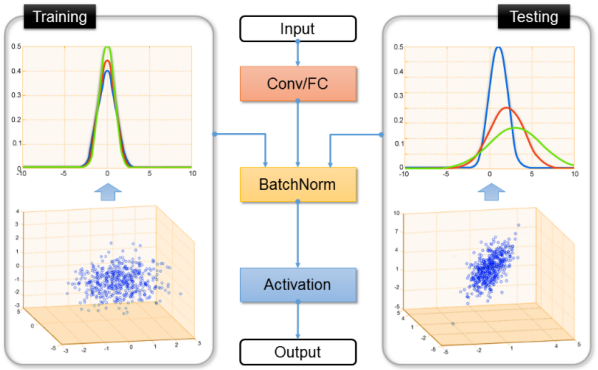
Nudge network towards test distribution via the BatchNorm statistics.

Y. Li et al., Revisiting Batch Normalization For Practical Domain Adaptation, arXiv 2016
V. Besnier et al., This dataset does not exist, ICASSP 2020

Domain Adaptation by BatchNorm

(1) Compute mean and variance for the new test set:

$$\mu_{\text{test}}^{(\tau+1)} = \alpha\mu_{\text{test}}^{(\tau)} + (1-\alpha)\hat{\mu}_{\text{batch}}$$
$$\sigma_{\text{test}}^{(\tau+1)} = \alpha\sigma_{\text{test}}^{(\tau)} + (1-\alpha)\hat{\sigma}_{\text{batch}}$$



Y. Li et al., Revisiting Batch Normalization For Practical Domain Adaptation, arXiv 2016
V. Besnier et al., This dataset does not exist, ICASSP 2020

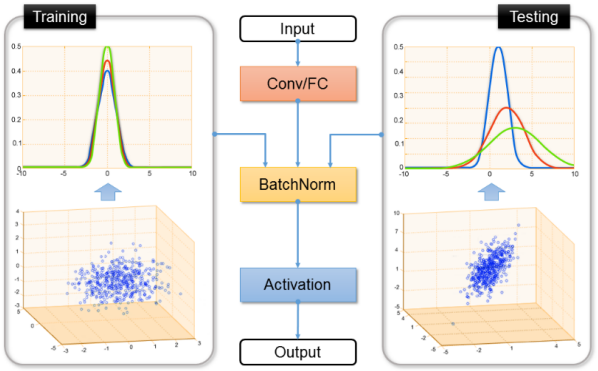
Domain Adaptation by BatchNorm

(1) Compute mean and variance for the new test set:

$$\mu_{\text{test}}^{(\tau+1)} = \alpha\mu_{\text{test}}^{(\tau)} + (1-\alpha)\hat{\mu}_{\text{batch}}$$
$$\sigma_{\text{test}}^{(\tau+1)} = \alpha\sigma_{\text{test}}^{(\tau)} + (1-\alpha)\hat{\sigma}_{\text{batch}}$$

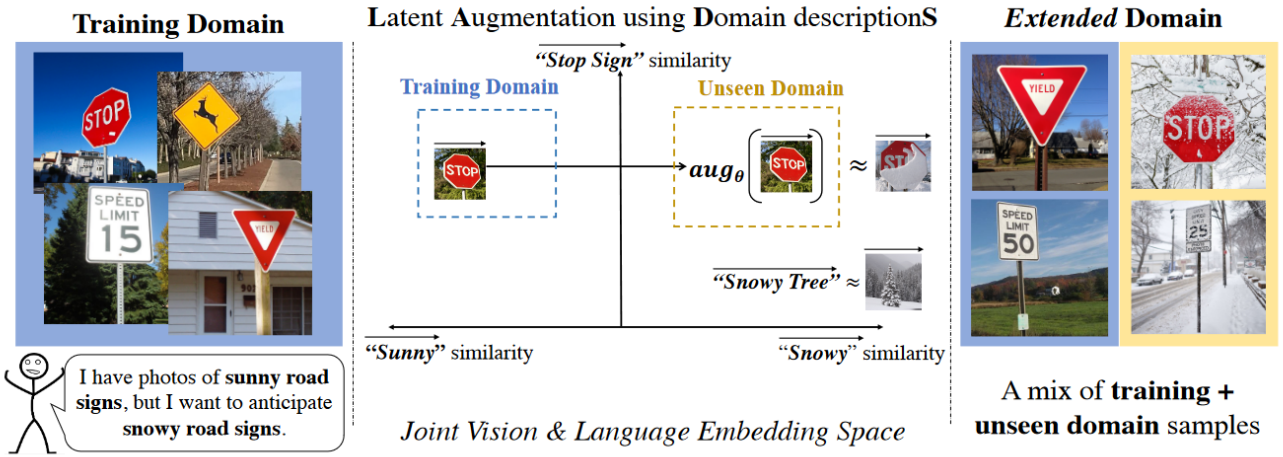
(2) At runtime use test-specific statistics:

$$\mathbf{y} = \gamma \odot (\mathbf{u} - \mu_{\text{test}}) \odot \frac{1}{\sqrt{\sigma_{\text{test}}^2 + \epsilon}} + \beta$$



Y. Li et al., Revisiting Batch Normalization For Practical Domain Adaptation, arXiv 2016
V. Besnier et al., This dataset does not exist, ICASSP 2020

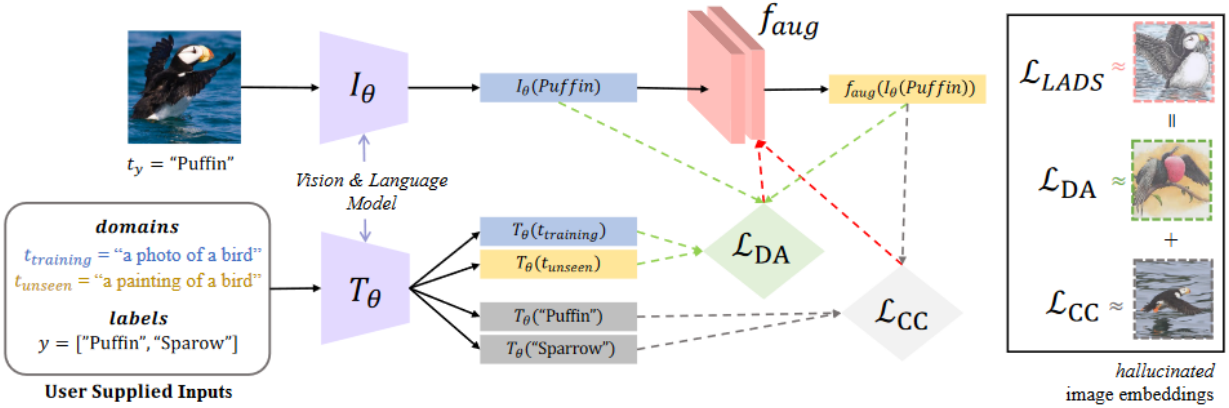
Prompt-driven Zero-shot Domain Adaptation



Use textual information to generate features for domains we have seen yet

Dunlap et al., Using Language to Extend to Unseen Domains, ICLR 2023
Fahes et al., PØDA: Prompt-driven Zero-shot Domain Adaptation, ICCV 2023

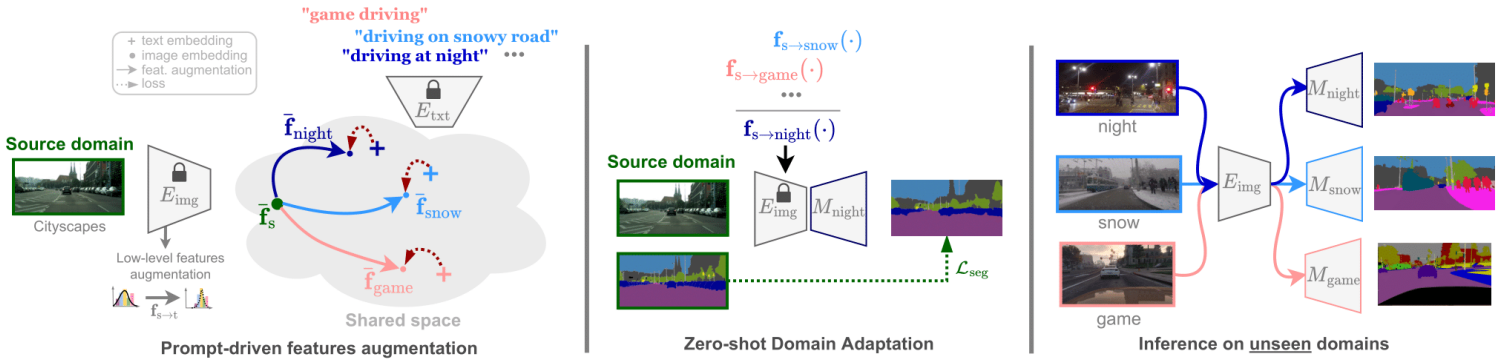
Prompt-driven Zero-shot Domain Adaptation



Use textual information to generate features for domains we have seen yet

Dunlap et al., Using Language to Extend to Unseen Domains, ICLR 2023
Fahes et al., PØDA: Prompt-driven Zero-shot Domain Adaptation, ICCV 2023

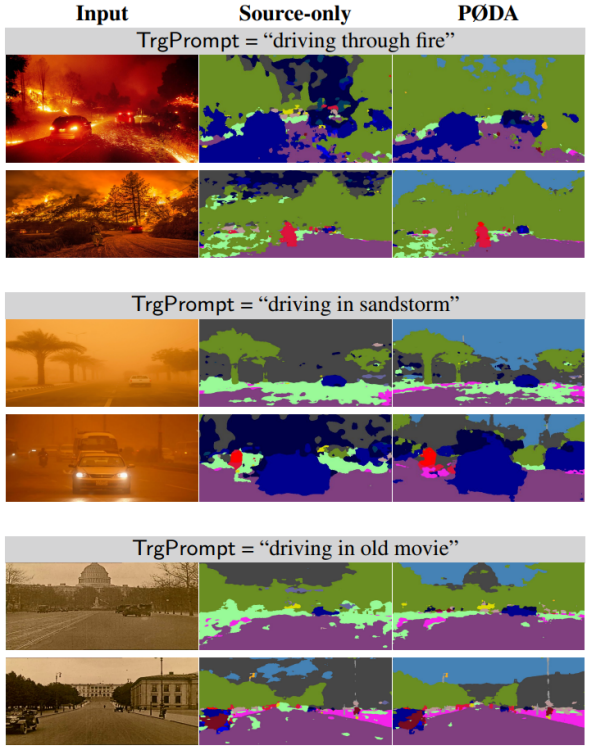
Prompt-driven Zero-shot Domain Adaptation



Use textual information to generate features for domains we have seen yet

Dunlap et al., Using Language to Extend to Unseen Domains, ICLR 2023
Fahes et al., PØDA: Prompt-driven Zero-shot Domain Adaptation, ICCV 2023

PØDA qualitative results



Fahes et al., PØDA: Prompt-driven Zero-shot Domain Adaptation, ICCV 2023

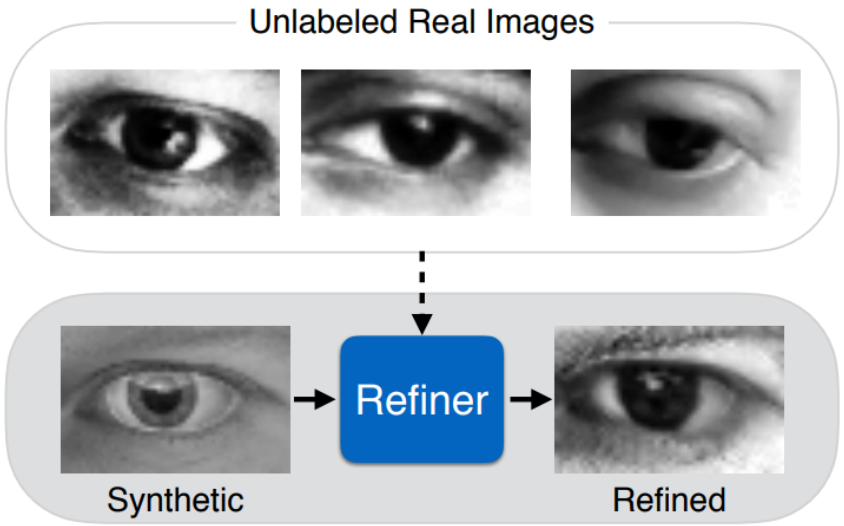
Another approach for domain adaptation is to focus on the content of the images instead of the learned representations themselves.

Another approach for domain adaptation is to focus on the content of the images instead of the learned representations themselves.

To this end, researchers take inspiration from generative methods.

Image translation

Instead of learning domain invariant features, we can modify the synthetic data to "look" more realistic.

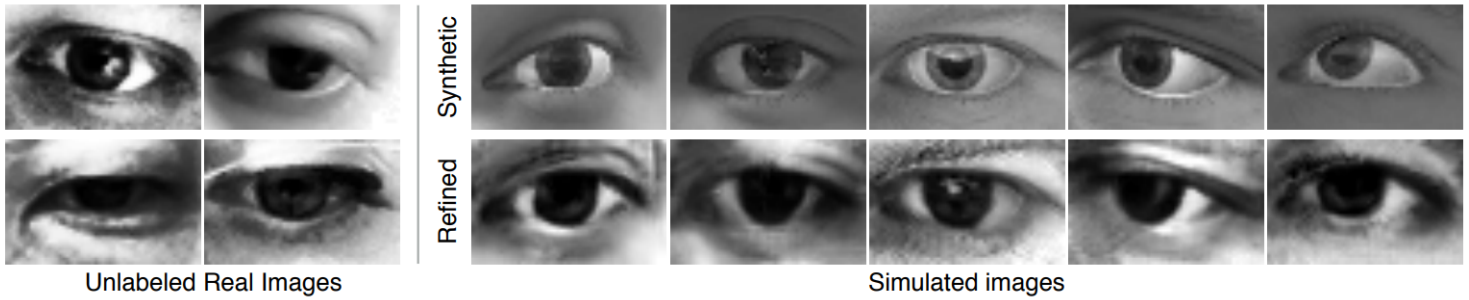


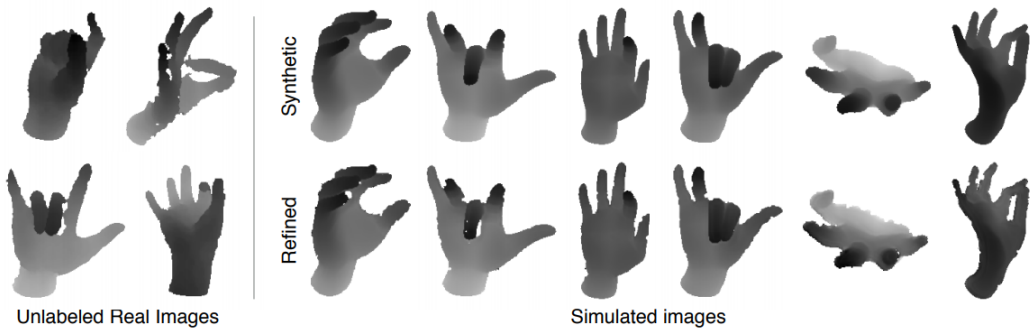
A. Srivastava et al., Learning from Simulated and Unsupervised Images through Adversarial Training, CVPR 2017

Image translation

A. Srivastava et al., Learning from Simulated and Unsupervised Images through Adversarial Training, CVPR 2017

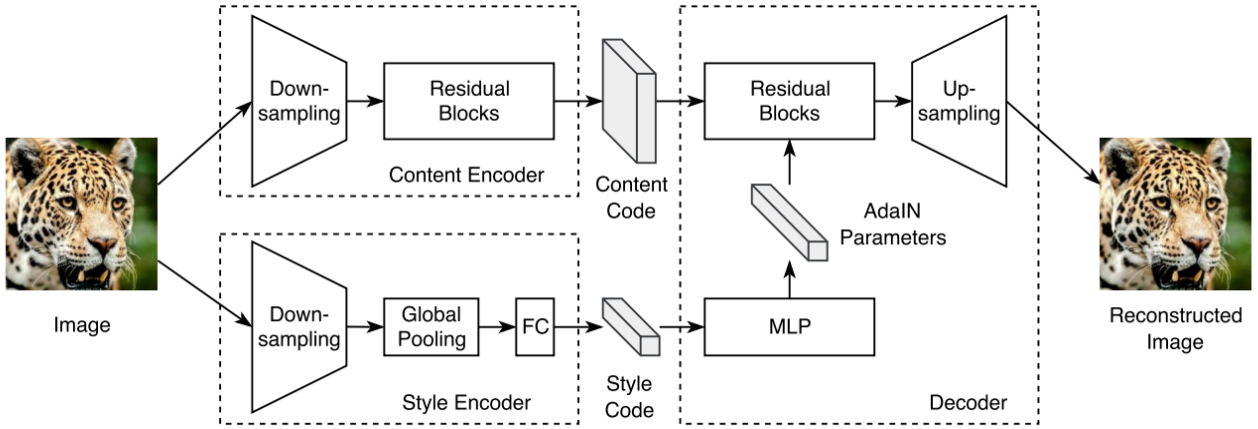
Image translation





A. Srivastava et al., Learning from Simulated and Unsupervised Images through Adversarial Training, CVPR 2017

Image translation

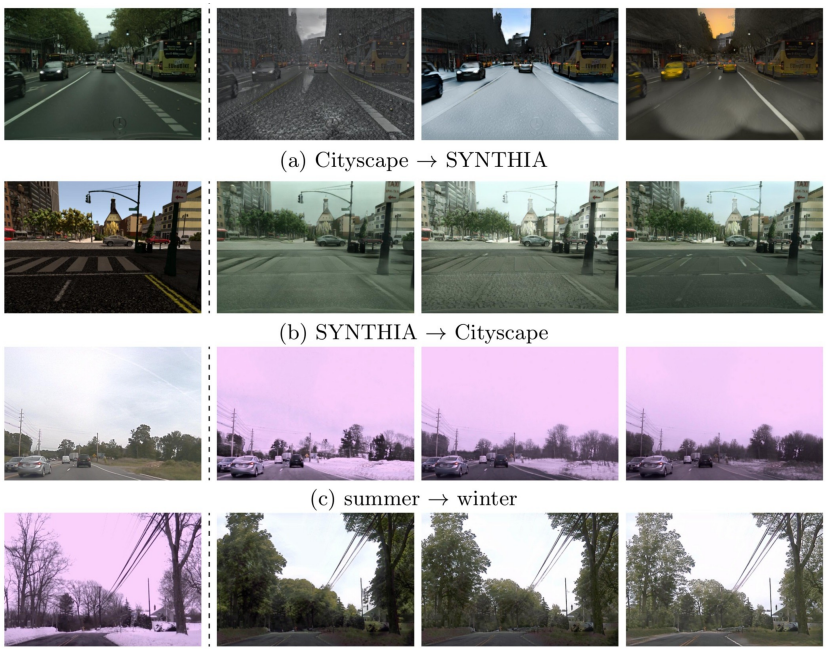


MUNIT

- In MUNIT we assume that the difference between domains is in style, while the content is similar
- We learn to extract separately content and style features
- In the decoding part we can inject the style from other domains.

X. Huang et al., MUNIT: Multimodal UNsupervised Image-to-image Translation, ECCV 2018

Image translation



(a) Cityscape → SYNTHIA

(b) SYNTHIA → Cityscape

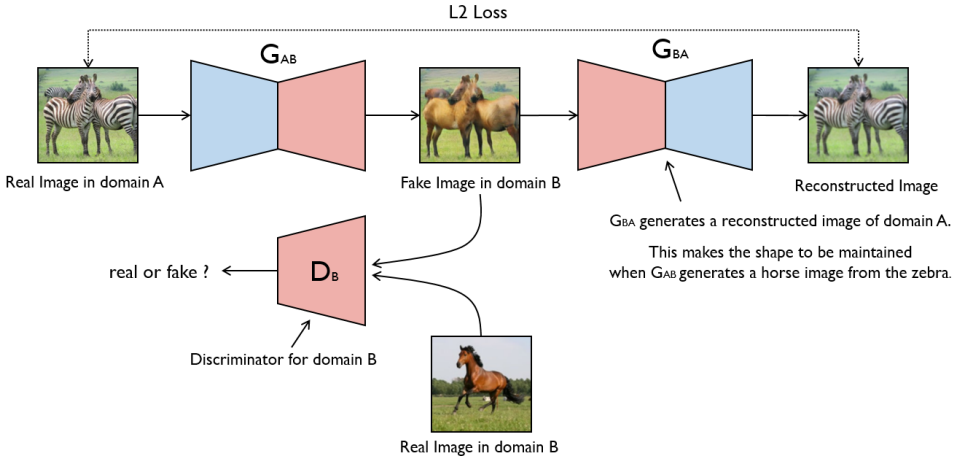
(c) summer → winter

(d) winter → summer

Qualitative examples from MUNIT

X. Huang et al., MUNIT: Multimodal UNsupervised Image-to-image Translation, ECCV 2018

Image translation

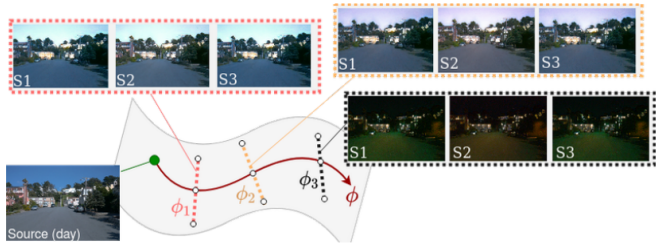


- Landmark work allowing to translate from one domain to another without having paired samples
- Drawback: images are not always realistic and the ground truth is not preserved

J.Y. Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017

Image translation

Translations (dark circle) of a source day image (center) exhibit both high variability and similarities with target data (outer circle).



Translations along dimensions (red) and style (dotted). For a given ϕ (sun elevation angle), the styles vary slightly (notice hue and brightness), proving disentanglement of ϕ and style.

`tl;dr:` CoMoGAN = CycleGAN + MUNIT + physical priors

F. Pizzati et al., CoMoGAN: continuous model-guided image-to-image translation, CVPR 2021

Image translation

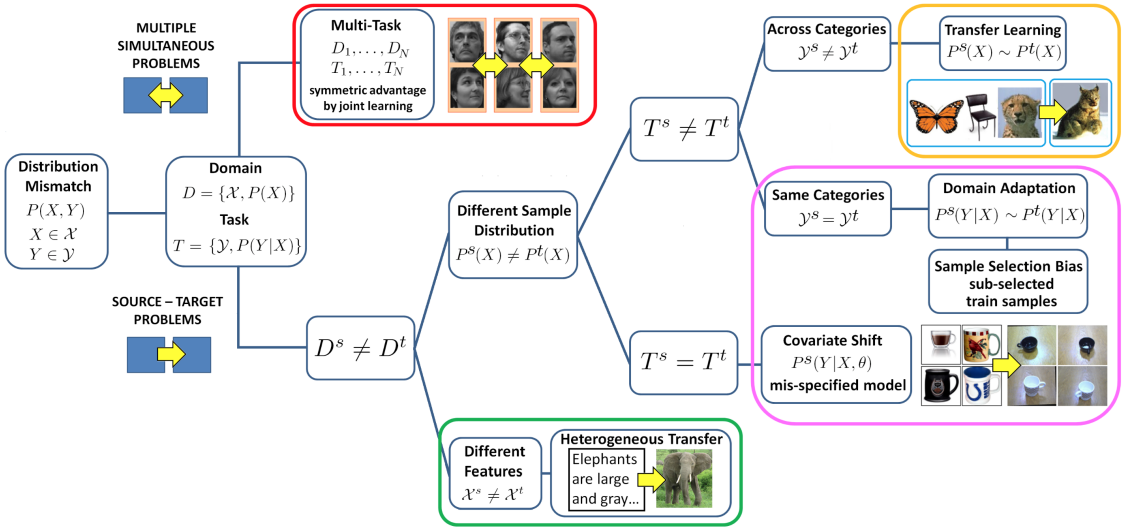
Qualitative results

`tl;dr:` CoMoGAN = CycleGAN + MUNIT + physical priors

F. Pizzati et al., CoMoGAN: continuous model-guided image-to-image translation, CVPR 2021

Transfer learning

Many flavors of transfer learning

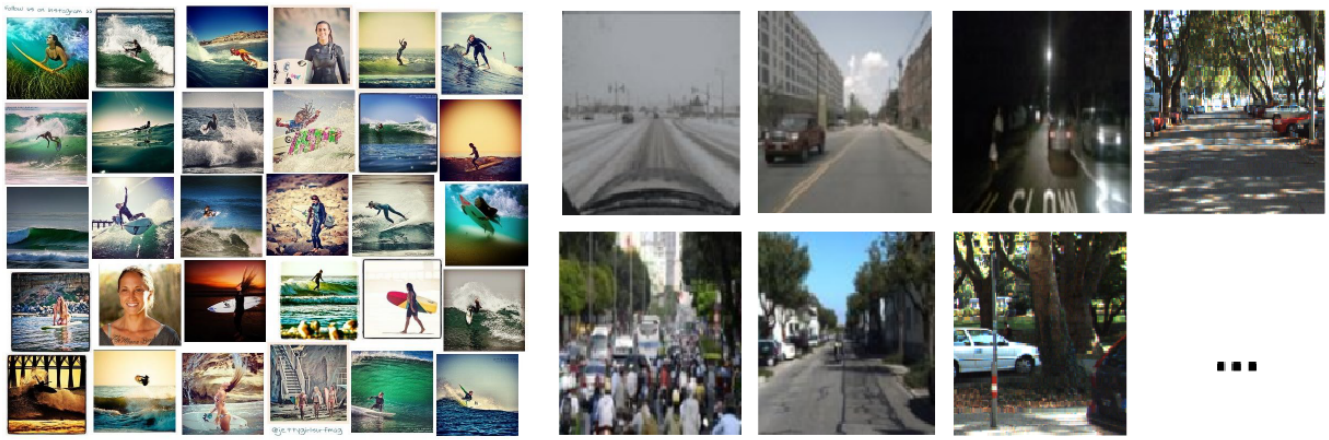


(Tommasi, PhD thesis, 2012)

Self-supervised learning

Can we exploit anything from raw data?

Exploiting raw unlabeled data



- Acquiring raw unlabeled data is usually easy
- However, typical supervised methods cannot exploit them

Inspiring success from self-supervision in NLP, e.g., word2vec

Input: The man went to the $[MASK]_1$. He bought a $[MASK]_2$ of milk .
Labels: $[MASK]_1$ = store; $[MASK]_2$ = gallon

Missing word prediction task.

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

Next sentence prediction task.

T. Mikolov et al., Efficient estimation of word representations in vector space, ArXiv 2013
T. Mikolov et al., Distributed representations of words and phrases and their compositionality, NeurIPS 2013
J. Devlin, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, ArXiv 2018

What is self-supervision?

- A form of unsupervised learning where the data (not the human) provides the supervision signal

What is self-supervision?

- A form of unsupervised learning where the data (not the human) provides the supervision signal
- Usually, define a pretext task for which the network is forced to learn what we really care about

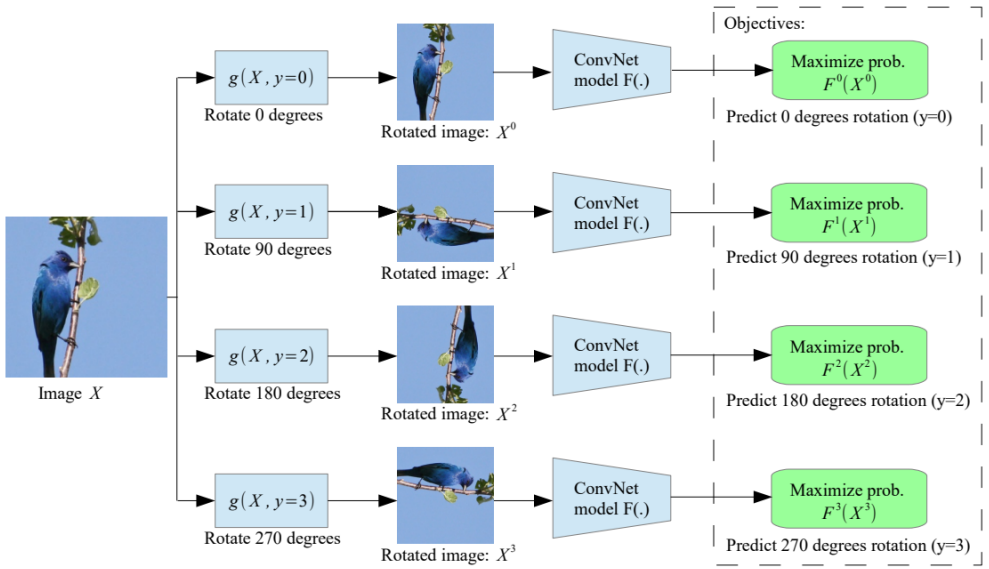
What is self-supervision?

- A form of unsupervised learning where the data (not the human) provides the supervision signal
- Usually, define a pretext task for which the network is forced to learn what we really care about
- For most pretext tasks, a part of the data is withheld and the network has to predict it

What is self-supervision?

- A form of unsupervised learning where the data (not the human) provides the supervision signal
- Usually, define a pretext task for which the network is forced to learn what we really care about
- For most pretext tasks, a part of the data is withheld and the network has to predict it
- The features/representations learned on the pretext task are subsequently used for a different downstream task, usually where some annotations are available.

Example: Rotation prediction

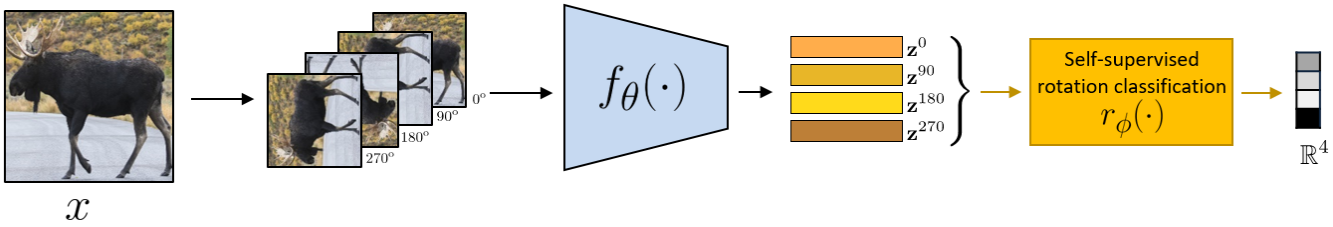


Predict the orientation of the image

S. Gidaris et al., Unsupervised Representation Learning by Predicting Image Rotations, ICLR 2018

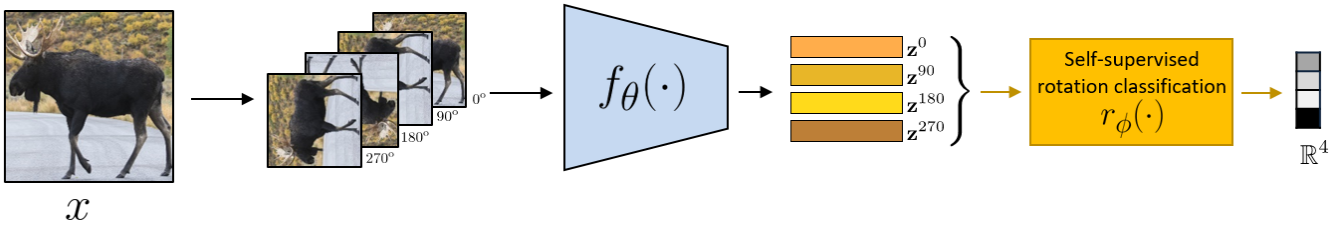
Self-supervised learning pipeline

Stage 1: Train network on pretext task (without human labels)

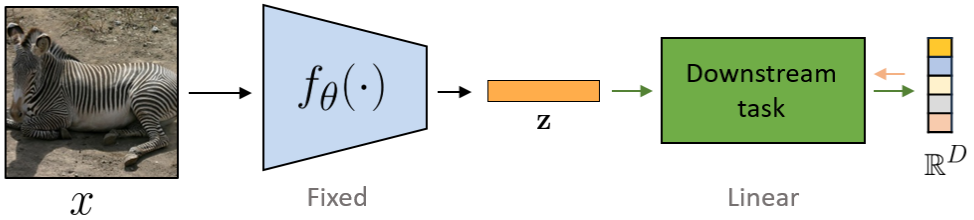


Self-supervised learning pipeline

Stage 1: Train network on pretext task (without human labels)

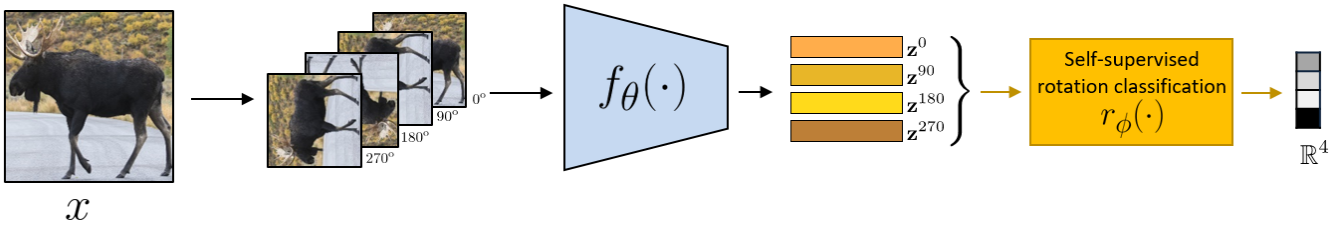


Stage 2: Train classifier on learned features for new task with fewer labels

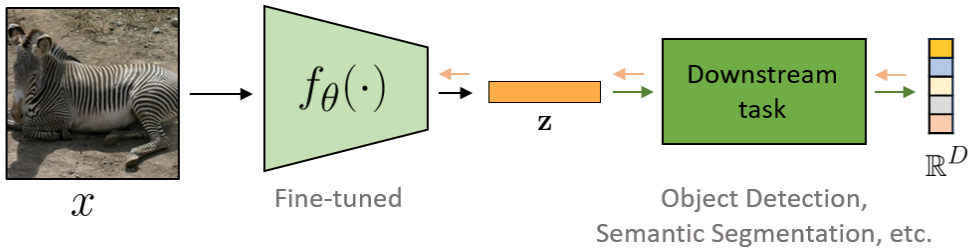


Self-supervised learning pipeline

Stage 1: Train network on pretext task (without human labels)



Stage 2: Fine-tune network for new task with fewer labels



Karate Kid and Self-Supervised Learning

The Karate Kid (1984)

Stage 1: Train *muscle memory* on pretext tasks

Stage 1: Train *muscle memory* on pretext tasks

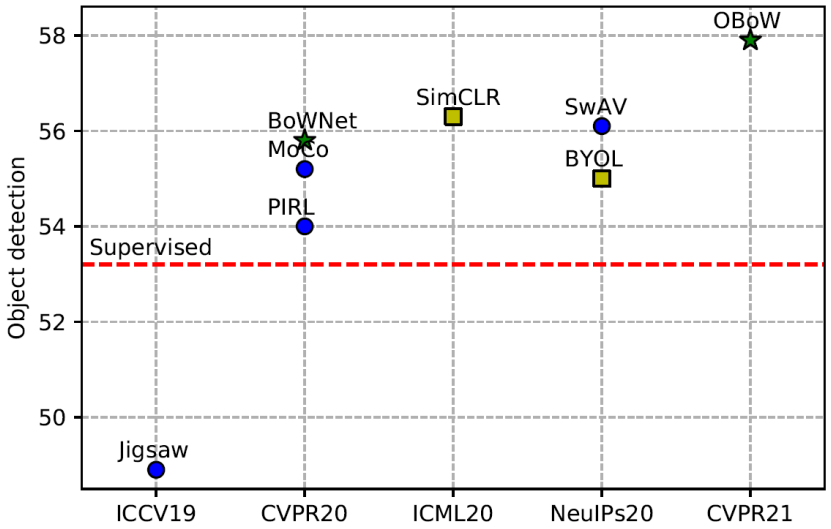
Mr. Miyagi = Deep Learning Practitioner daily chores = pretext tasks

Daniel LaRusso = ConvNet learning karate = downstream task

Stage 2: Fine-tune skills rapidly

Is this actually useful in practice?

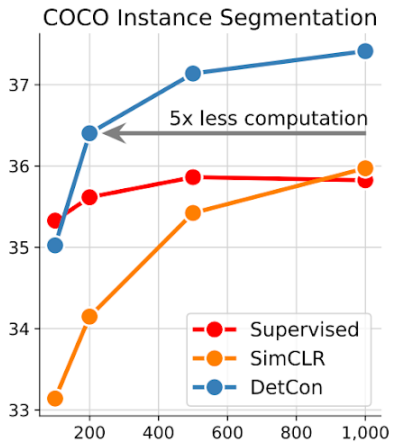
Transfer learning - object detection

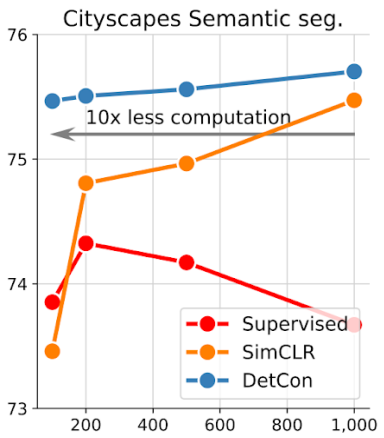


Object detection with Faster R-CNN fine-tuned on VOC trainval07+12 and evaluated on test07. Networks are pre-trained with self-supervision on ImageNet.

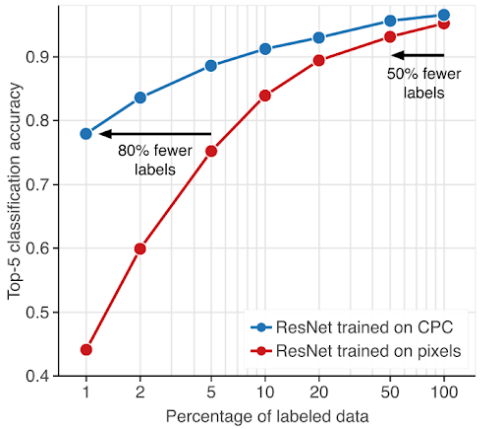
- Rapid progress in self-supervised learning
- Self-supervised methods are starting to outperform supervised methods
- This is a key milestone for self-supervised methods as they are finally showing their effectiveness to complex downstream tasks.

SSL methods are often more efficient than supervised methods





Efficiency in terms of number of epochs for ImageNet pretraining (SimCLR and DetCon do no use human annotated labels)



Data-efficiency of SSL and supervised learning methods

A tour of pretext tasks for Self-Supervised Learning

Pretext tasks:

- Inferring structure
- Transformation prediction
- Input reconstruction
- Exploiting time
- Multimodal

Pretext tasks:

- Inferring structure
- Transformation prediction
- Input reconstruction
- Exploiting time
- Multimodal

Context prediction

Can you guess the spatial configuration for the two pairs of patches?

Question 1:

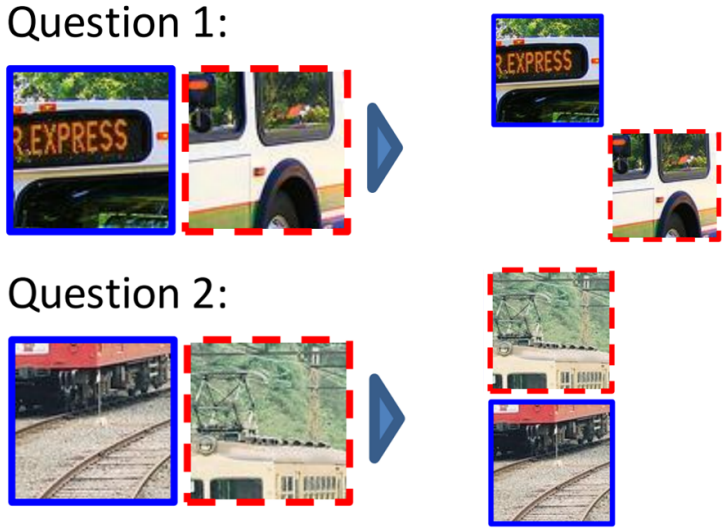
Question 2:

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

Context prediction

Can you guess the spatial configuration for the two pairs of patches?

Much easier if you recognize the object!

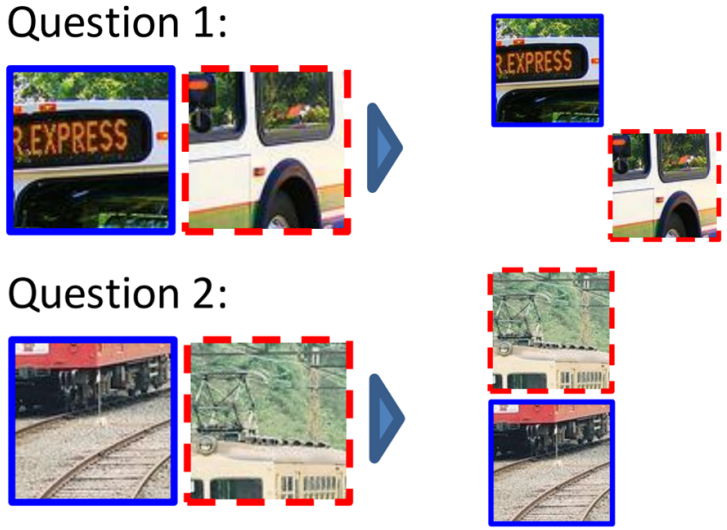


C. Doersch et al., Unsupervised Visual Representation Learning by Context Prediction, ICCV 2015

Context prediction

Can you guess the spatial configuration for the two pairs of patches?

Much easier if you recognize the object!

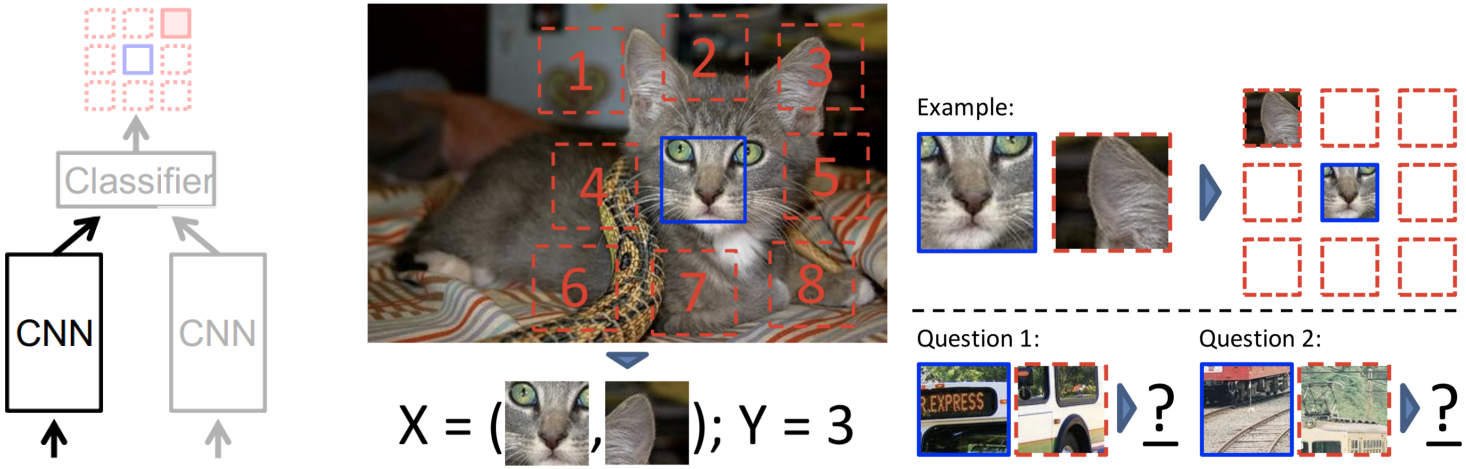


Intuition:

- The network should learn to recognize object parts and their spatial relations

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

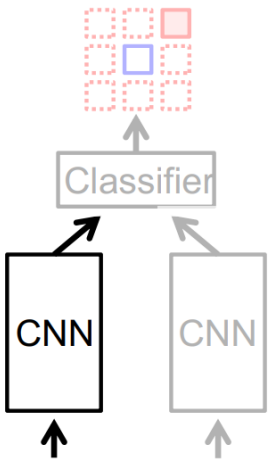
Context prediction



Predict the location of one patch relative to the center patch

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

Context prediction



Pros

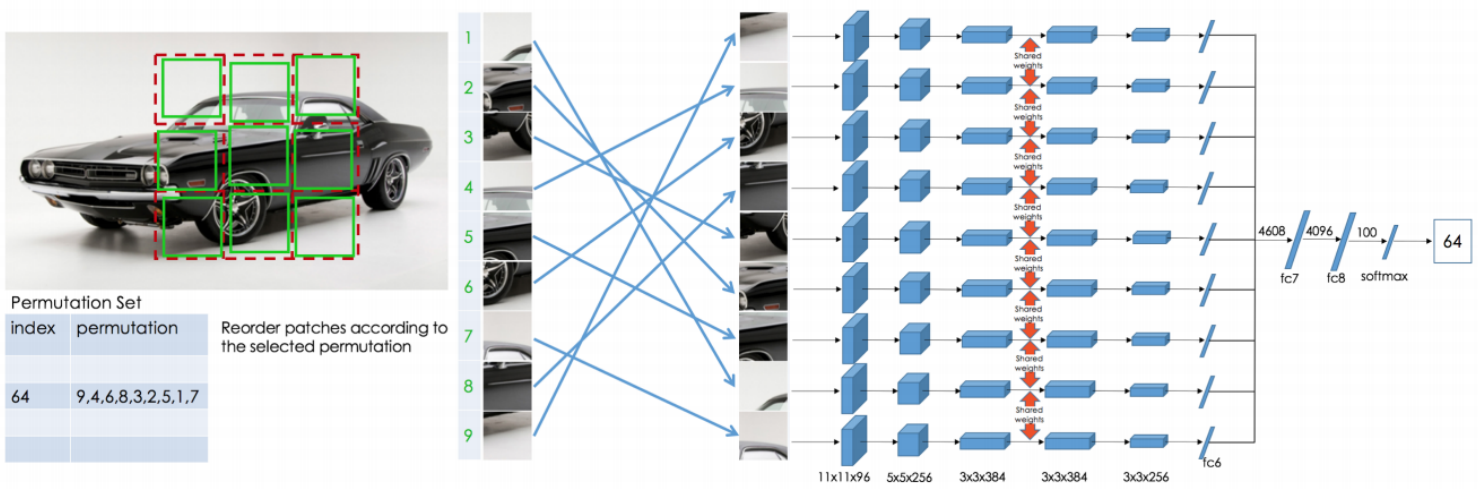
- The first self-supervised method
- Intuitive task that should enable learning about object parts

Cons

- Assumes training images are photographed with canonical orientations (and canonical orientations exist)
- Networks can “cheat” so special care is needed
- Training on patches, but trying to learn image representations
- Not fine-grained enough due to no negatives from other images
- Small output space - 8 cases (positions) to distinguish?

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

Jigsaw puzzles



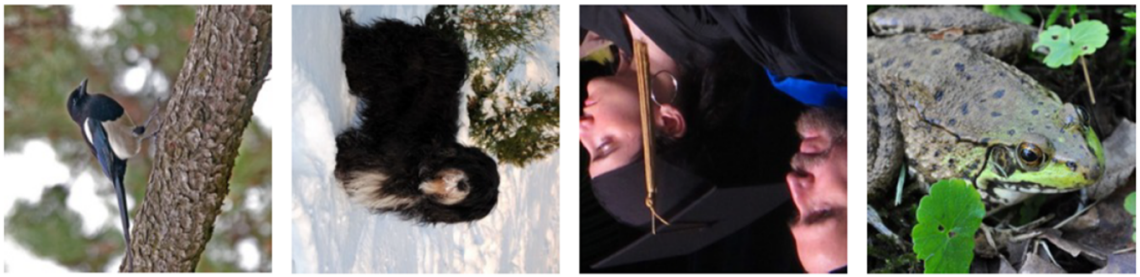
M. Noroozi and P. Favaro, Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles, ECCV 2016

Pretext tasks:

- Inferring structure
- Transformation prediction
- Input reconstruction
- Exploiting time
- Multimodal

Rotation prediction

Can you guess how much rotated is applied?

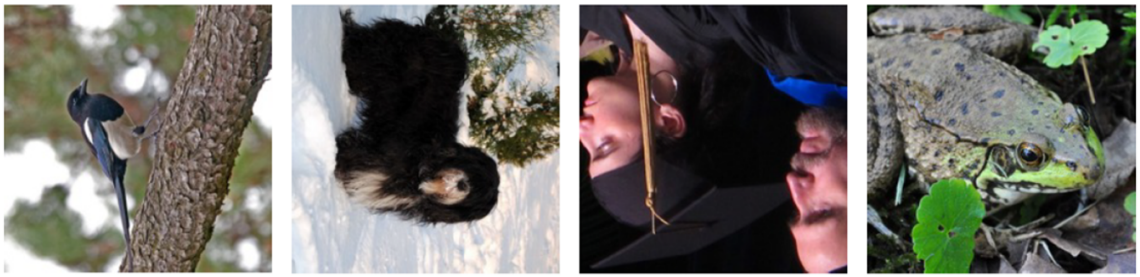


S. Gidaris et al., Unsupervised representation learning by predicting image rotations, ICLR 2018

Rotation prediction

Can you guess how much rotated is applied?

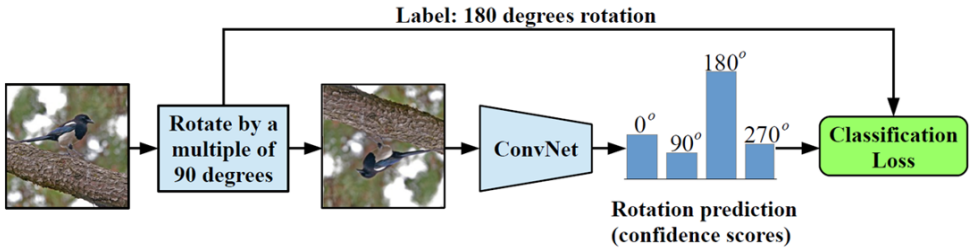
Much easier if you recognize the object!



90° rotation | 270° rotation | 180° rotation | 0° rotation

S. Gidaris et al., Unsupervised representation learning by predicting image rotations, ICLR 2018

Rotation prediction



Pros

- Very simple to implement and use, while being quite effective

Cons

- Assumes training images are photographed with canonical orientations (and canonical orientations exist)
- Train-eval gap: no rotated images at eval
- Not fine-grained enough due to no negatives from other images
- Small output space - 4 cases (rotations) to distinguish

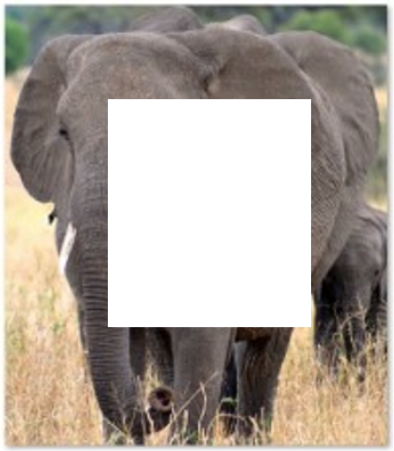
S. Gidaris et al., Unsupervised representation learning by predicting image rotations, ICLR 2018

Pretext tasks:

- Inferring structure
- Transformation prediction
- Input reconstruction
- Exploiting time
- Multimodal

Context encoders

What goes in the middle?

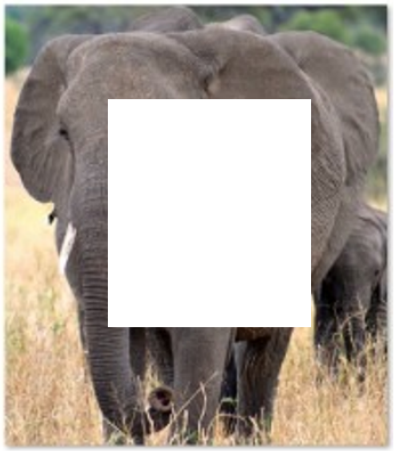


D. Pathak et al., Context Encoders: Feature Learning by Inpainting, CVPR 2016

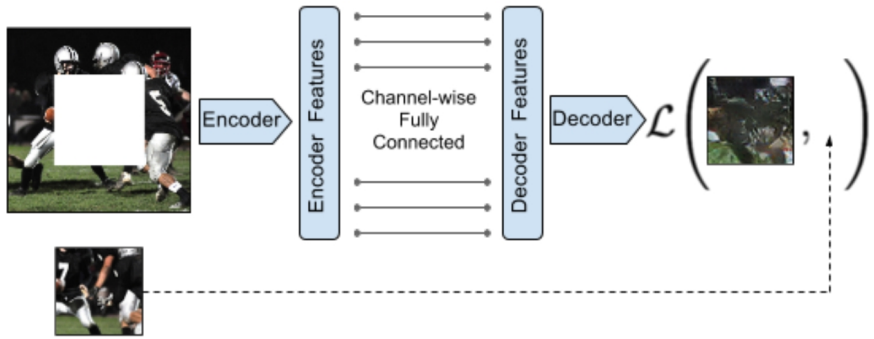
Context encoders

What goes in the middle?

Much easier if you recognize the objects!



D. Pathak et al., Context Encoders: Feature Learning by Inpainting, CVPR 2016



Pros

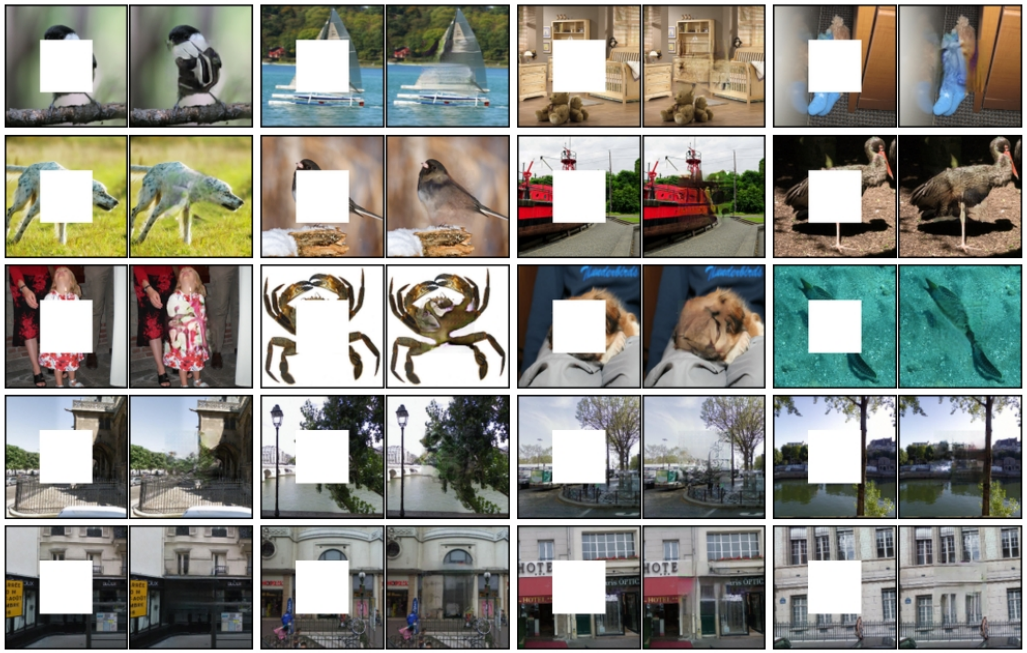
- Requires preservation of fine-grained information

Cons

- Train-eval gap: no masking at eval
- Input reconstruction is too hard and ambiguous
- Lots of effort spent on “useless” details: exact colour, good boundary, etc.

D. Pathak, Context Encoders: Feature Learning by Inpainting, CVPR 2016

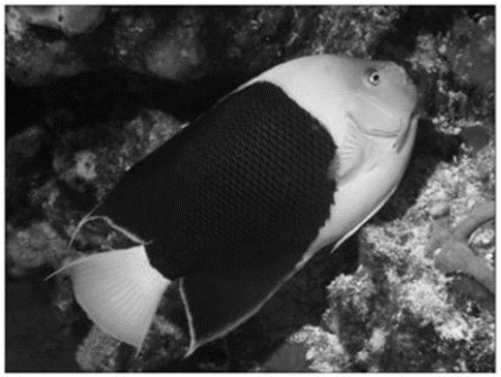
Context encoders



D. Pathak, Context Encoders: Feature Learning by Inpainting, CVPR 2016

Colorization

What is the colour of every pixel?

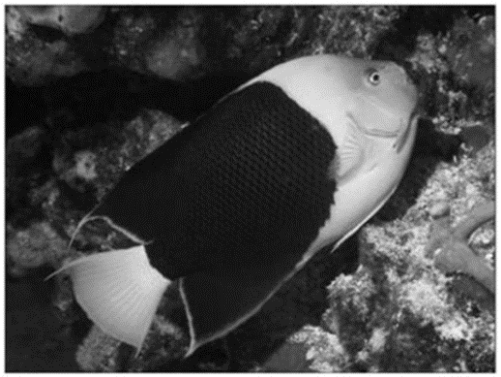


R. Zhang et al., Colorful image colorization, ECCV 2016

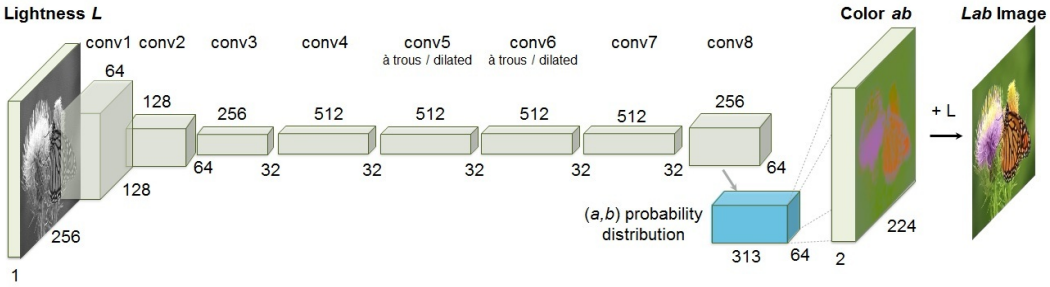
Colorization

What is the colour of every pixel?

Hard if you don't recognize the object!



R. Zhang et al., Colorful image colorization, ECCV 2016



Pros

- Requires preservation of fine-grained information

Cons

- Input reconstruction is too hard and ambiguous
- Lots of effort spent on “useless” details: exact colour, good boundary, etc
- Forced to evaluate on greyscale images, losing information

R. Zhang et al., Colorful image colorization, ECCV 2016

Pretext tasks:

- Inferring structure
- Transformation prediction
- Input reconstruction
- Exploiting time
- Multimodal

Pretext tasks:

- Inferring structure
- Transformation prediction
- Input reconstruction
- Exploiting time
- Multimodal

Tracking by colorization

Given an earlier frame, colourize the new one?

C. Vondrick et al., Tracking emerges by colorizing videos, ECCV 2018

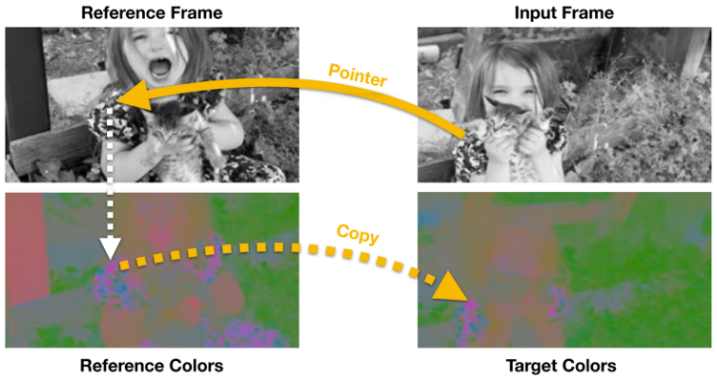
Tracking by colorization

Given an earlier frame, colourize the new one?

Easy if everything can be tracked!

C. Vondrick et al., Tracking emerges by colorizing videos, ECCV 2018

Tracking by colorization



Pros

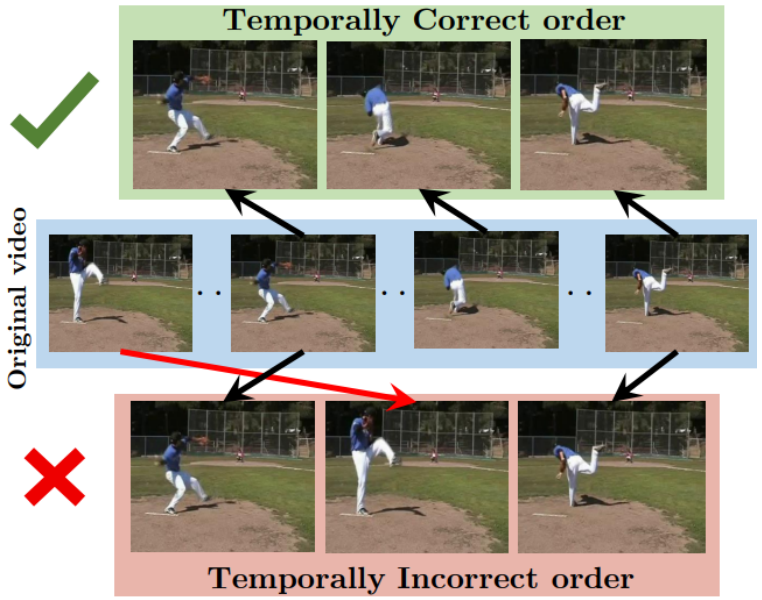
- Emerging behaviour: tracking, matching, optical flow, segmentation

Cons

- Low level cues are effective - less emphasis on semantics
- Forced to evaluate on greyscale frames, losing information

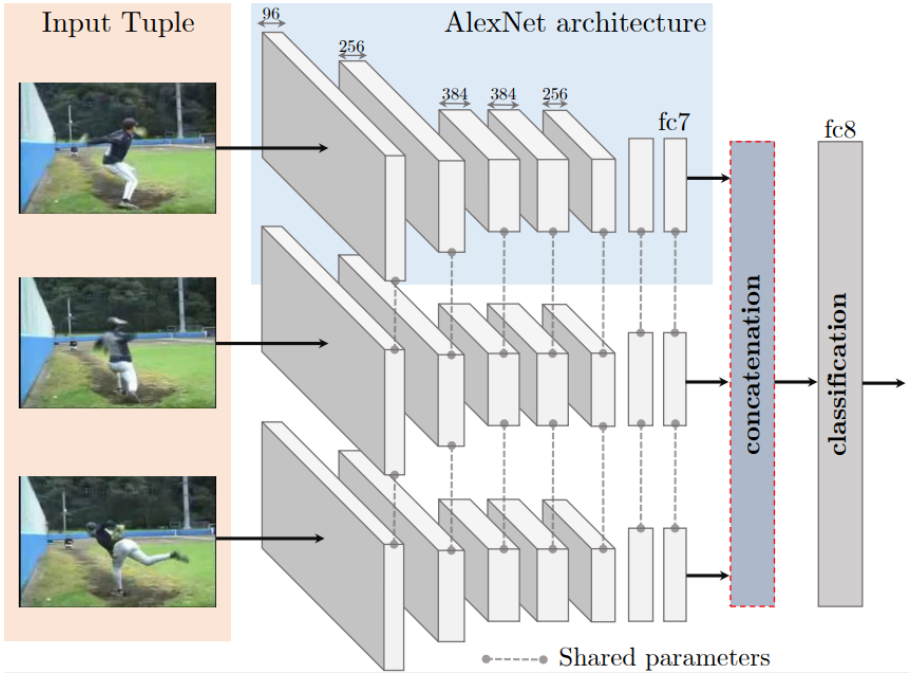
C. Vondrick et al., Tracking emerges by colorizing videos, ECCV 2018

Predicting the correct order of time



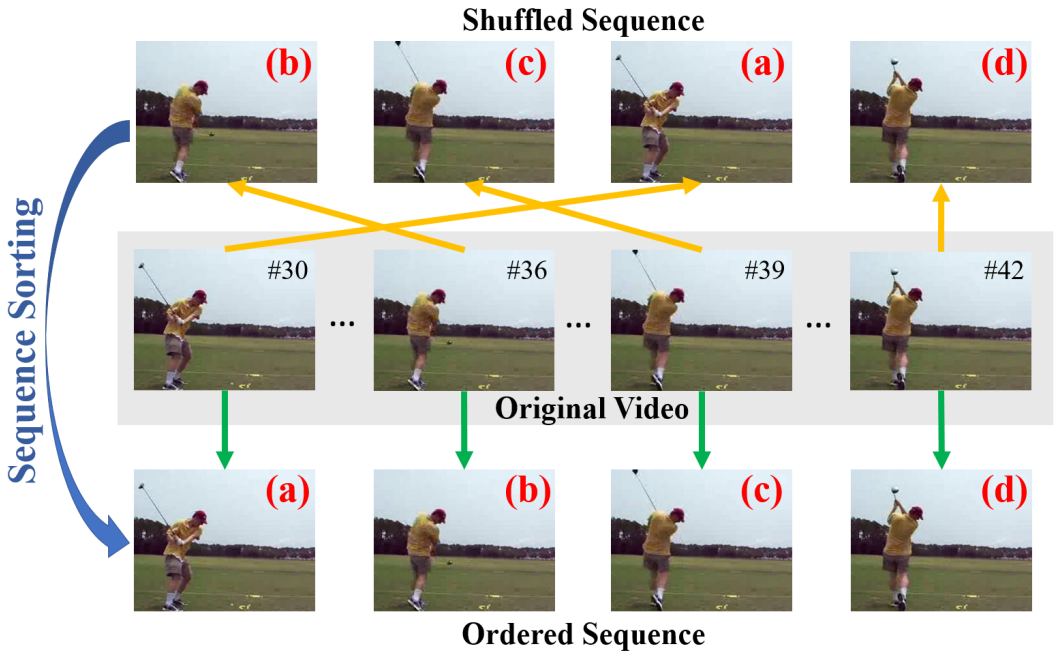
I. Misra et al., Shuffle and Learn: Unsupervised Learning using Temporal Order Verification, ECCV 2016

Predicting the correct order of time



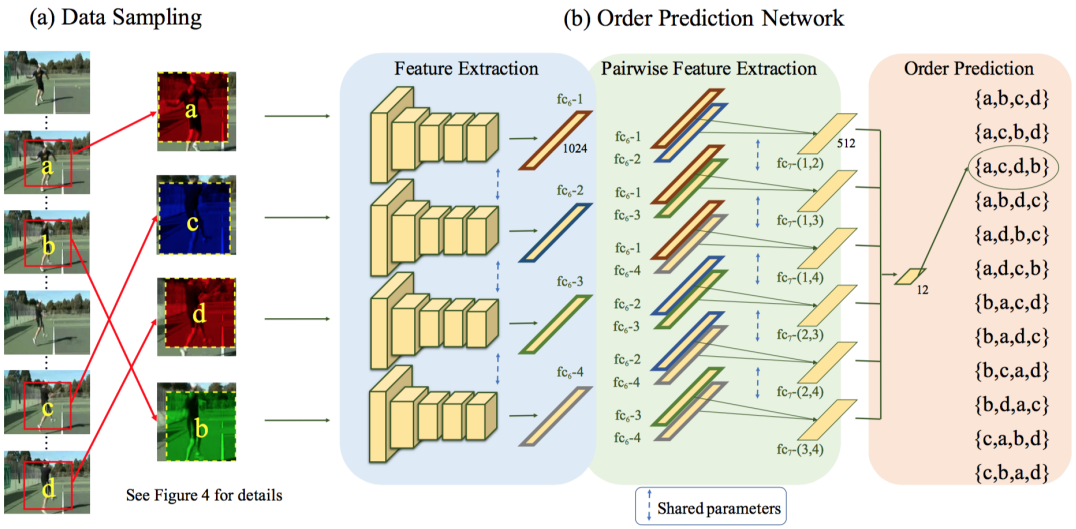
I. Misra et al., Shuffle and Learn: Unsupervised Learning using Temporal Order Verification, ECCV 2016

Sorting video sequences



H.Y. Lee et al., Unsupervised Representation Learning by Sorting Sequences, ICCV 2017

Sorting video sequences



H.Y. Lee et al., Unsupervised Representation Learning by Sorting Sequences, ICCV 2017

Pretext tasks:

- Inferring structure
- Transformation prediction
- Input reconstruction
- Exploiting time
- Multimodal

Audio-visual correspondence

Can audio and video learn from each other?

R. Arandjelovic, Look, Listen and Learn, ICCV 2017

Slide credit: A. Zisserman

Audio-visual correspondence

R. Arandjelovic, Look, Listen and Learn, ICCV 2017

Slide credit: A. Zisserman

Audio-visual correspondence

R. Arandjelovic, Look, Listen and Learn, ICCV 2017

Slide credit: A. Zisserman

Audio-visual correspondence

R. Arandjelovic, Look, Listen and Learn, ICCV 2017

Slide credit: A. Zisserman

Audio-visual correspondence

R. Arandjelovic, Look, Listen and Learn, ICCV 2017

Slide credit: A. Zisserman

Audio-visual correspondence

R. Arandjelovic, Look, Listen and Learn, ICCV 2017

Audio-visual correspondence

R. Arandjelovic, Look, Listen and Learn, ICCV 2017

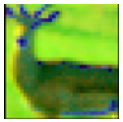
The first generation of self-supervised approaches achieve interesting results. However performance is still far from the performance of their supervised counterparts.

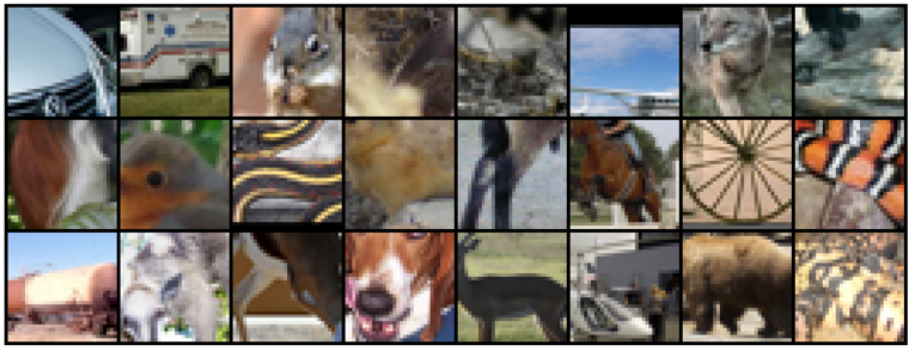
The recent line of approaches, contrastive and feature reconstruction, achieved remarkable results outperforming supervised variants on several benchmarks.

Contrastive methods

Exemplar networks

The image on the left is a distorted crop extracted from an image, which of these crops has the same source image?



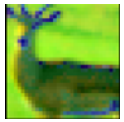


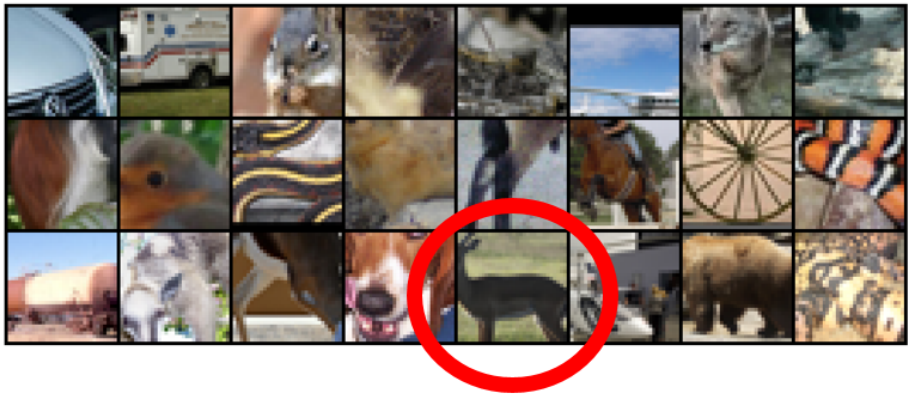
A. Dosovitskiy et al., Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks, PAMI 2015

Exemplar networks

The image on the left is a distorted crop extracted from an image, which of these crops has the same source image?

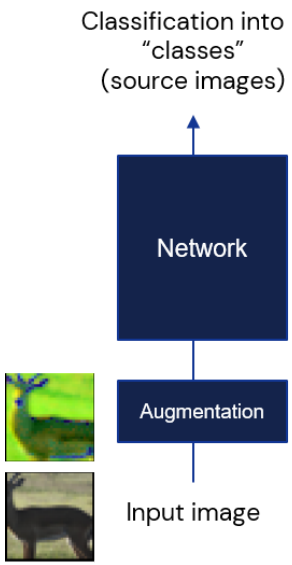
Easy if robust to the desired transformations (geometry and colour)





A. Dosovitskiy et al., Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks, PAMI 2015

Exemplar networks



Pros

- Representations are invariant to desired transformations
- Requires preservation of fine-grained information

Cons

- Choosing the augmentations is important
- Exemplar based: images of the same class or instance are negatives
  - Nothing prevents it from focusing on the background
- Original formulation is not scalable (number of “classes” = dataset size)

A. Dosovitskiy et al., Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks, PAMI 2015

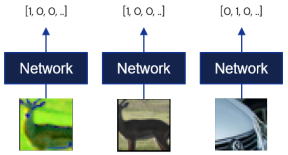
Some notations

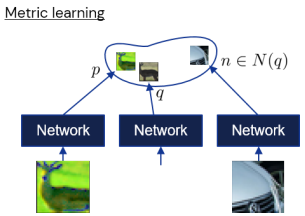
Let $x \in \mathrm{X}$ be input data and $y \in \{1, \ldots, L\}$ and $f_\theta(\cdot) : \mathrm{X} \rightarrow \mathrm{R}^D$ a network generating an embedding vector $f_\theta(x)$.

We denote:

- $q = f_\theta(x)$ (query)
- $\{x^i\}$ a set of samples from X.
- $k_i = f_\theta(x^i)$ the embeddings of $\{x^i\}$ as keys (representations)

Exemplar networks





Exemplar ConvNets are not scalable (number of "classes" = number of training images)

- Using w_i as class prototype prevents explicit comparison between instances, i.e. individual samples
- We can use instead a non-parametric variant that replaces $q^\top w_j$ with $q^\top k_i$

$$\mathrm{L}_{\text{softmax}}(q, c(q)) = -\log \frac{\exp(q^\top w_{c(q)})}{\sum_{c \in C} \exp(q^\top w_c)}$$

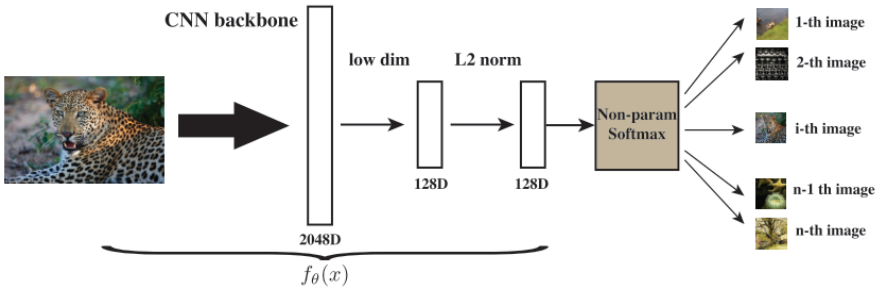
$$\downarrow$$

$$\mathrm{L}_{\text{non-param-softmax}}(q) = -\log \frac{\exp(q^\top k_q)}{\sum_{i \in N} \exp(q^\top k_i)}$$

- N is the number of training samples; C is the number of classes
- $c(q)$ is the class index of q

A. Dosovitskiy et al., Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks, PAMI 2015

Non-Parametric Classifier



Self-supervised learning as image instance-level discrimination

$$\mathrm{L}_{\text{non-param-softmax}}(q) = -\log \frac{\exp(q^\top k_q)}{\sum_{i \in N} \exp(q^\top k_i)}$$

The learning objective focuses now entirely on feature representation, instead of class-specific representations.

This loss if commonly named InfoNCE loss.

Z. Wu et al., Unsupervised Feature Learning via Non-Parametric Instance Discrimination, CVPR 2018

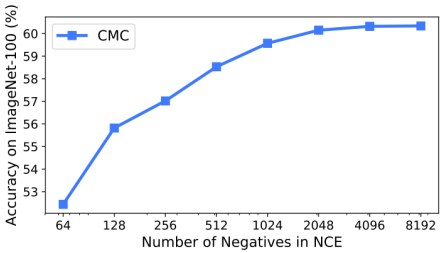
In recent literature InfoNCE is among the most popular and, best performing loss for contrastive self-supervised learning.

We will outline the main approaches that revolve around how to better use negatives.

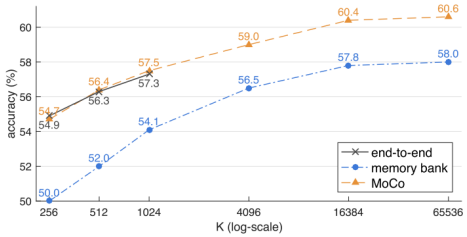
Dealing with negative samples

Multiple works have observed the influence of negative samples and proposed different heuristics to increase their number.

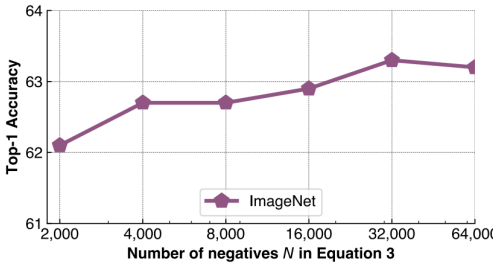
CMC [Tian et al. (2019)]



MoCo [He et al. (2020)]

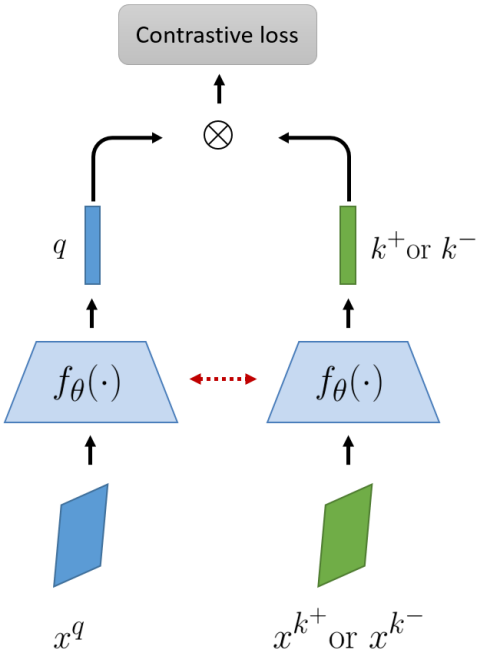


PiRL [Misra and van der Maaten (2020)]



Y. Tian et al., Contrastive Multiview Coding, ArXiv 2019
K. He et al., Momentum Contrast for Unsupervised Visual Representation Learning, CVPR 2020
I. Misra and L. van der Maaten, Self-Supervised Learning of Pretext-Invariant Representations, CVPR 2020

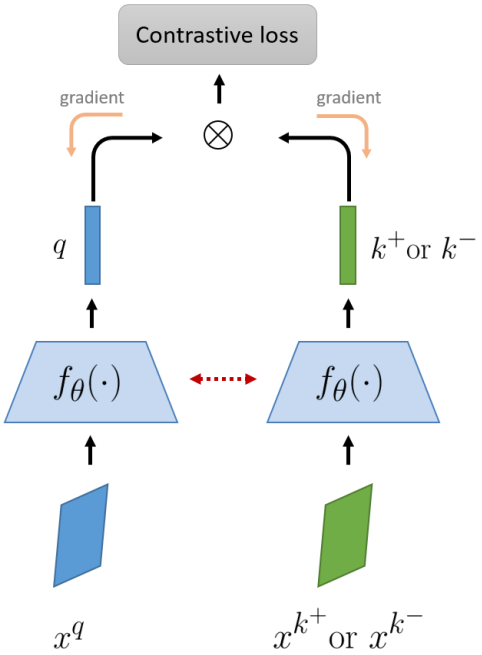
End-to-End training



- Derived from siamese networks in metric learning
- Shared encoder for queries and keys

R. Hadsell et al., Dimensionality reduction by learning an invariant mapping, CVPR 2006; F. Schroff et al., FaceNet: A unified embedding for face recognition and clustering, CVPR 2015

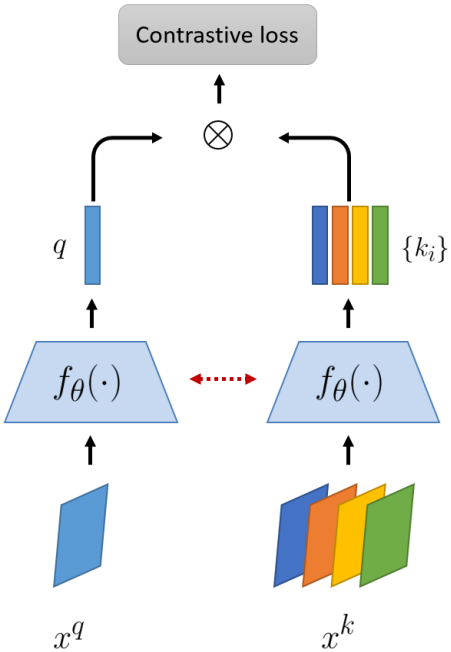
End-to-End training



- Derived from siamese networks in metric learning
- Shared encoder for queries and keys

R. Hadsell et al., Dimensionality reduction by learning an invariant mapping, CVPR 2006; F. Schroff et al., FaceNet: A unified embedding for face recognition and clustering, CVPR 2015

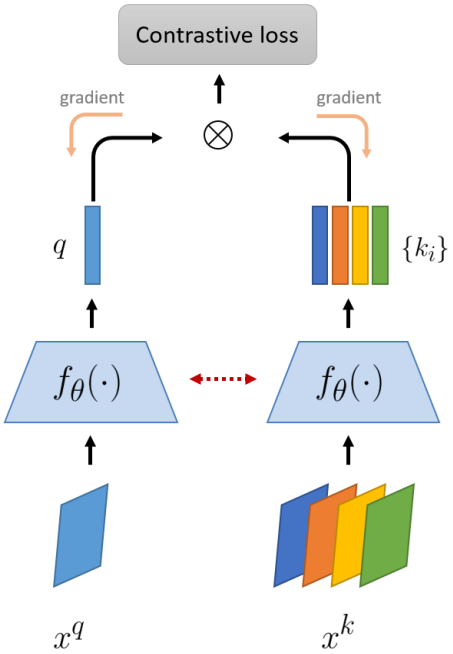
End-to-End training



- Derived from siamese networks in metric learning
- Shared encoder for queries and keys

R. Hadsell et al., Dimensionality reduction by learning an invariant mapping, CVPR 2006; F. Schroff et al., FaceNet: A unified embedding for face recognition and clustering, CVPR 2015

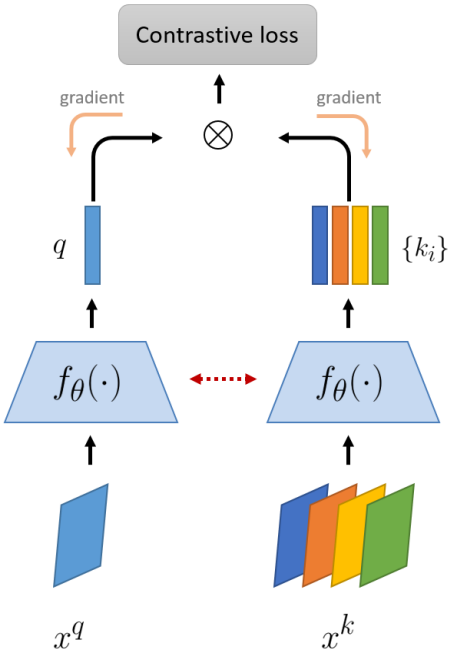
End-to-End training



- Derived from siamese networks in metric learning
- Shared encoder for queries and keys
- The encoder is updated by backpropagation through all samples

R. Hadsell et al., Dimensionality reduction by learning an invariant mapping, CVPR 2006; F. Schroff et al., FaceNet: A unified embedding for face recognition and clustering, CVPR 2015

End-to-End training



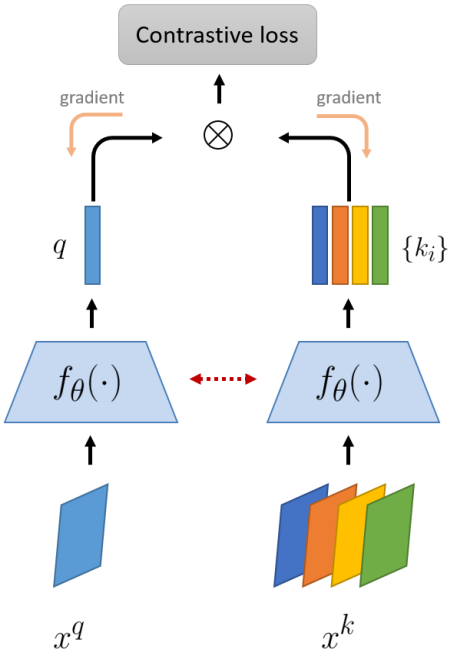
- Derived from siamese networks in metric learning
- Shared encoder for queries and keys
- The encoder is updated by backpropagation through all samples

Pros

- Consistent q and $\{k_i\}$

R. Hadsell et al., Dimensionality reduction by learning an invariant mapping, CVPR 2006; F. Schroff et al., FaceNet: A unified embedding for face recognition and clustering, CVPR 2015

End-to-End training



- Derived from siamese networks in metric learning
- Shared encoder for queries and keys
- The encoder is updated by backpropagation through all samples

Pros

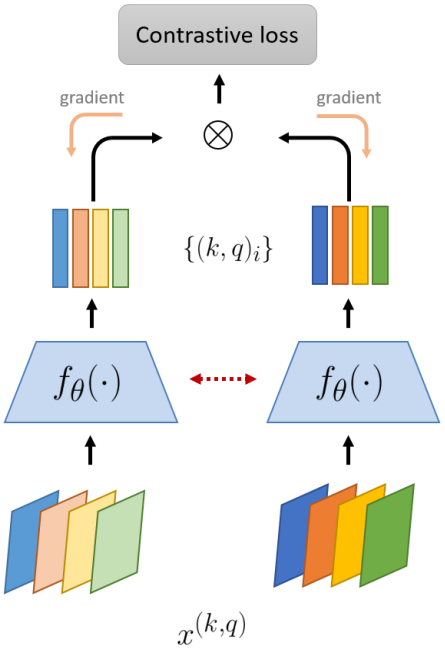
- Consistent q and $\{k_i\}$

Cons

- The amount of negatives limited by GPU memory

R. Hadsell et al., Dimensionality reduction by learning an invariant mapping, CVPR 2006; F. Schroff et al., FaceNet: A unified embedding for face recognition and clustering, CVPR 2015

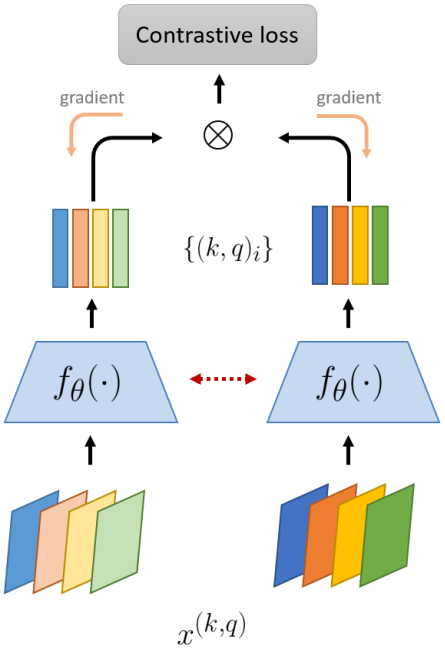
End-to-End training - SimCLR



- Rely on large mini-batches ($2k - 8k$ samples) to ensure plenty of negatives, e.g., $16,382$ negatives from a batch of $8,192$, i.e., $2(N-1)$ negatives
- The loss is computed across all positive pairs in a mini-batch

T. Chen et al., A Simple Framework for Contrastive Learning of Visual Representations, ICML 2020

End-to-End training - SimCLR



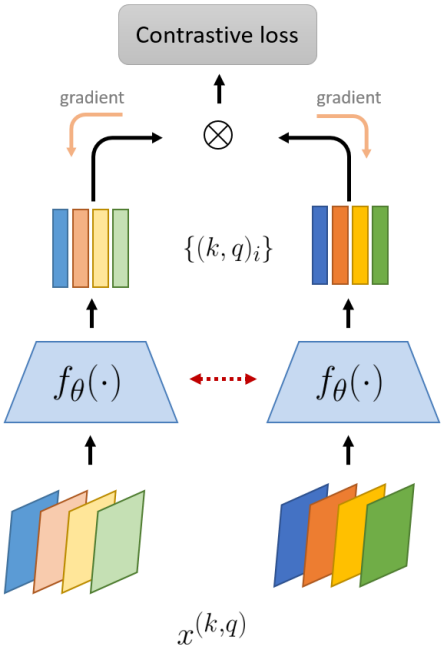
- Rely on large mini-batches ($2k - 8k$ samples) to ensure plenty of negatives, e.g., $16,382$ negatives from a batch of $8,192$, i.e., $2(N-1)$ negatives
- The loss is computed across all positive pairs in a mini-batch

Pros

- Lots of negatives

T. Chen et al., A Simple Framework for Contrastive Learning of Visual Representations, ICML 2020

End-to-End training - SimCLR



- Rely on large mini-batches ($2k - 8k$ samples) to ensure plenty of negatives, e.g., $16,382$ negatives from a batch of $8,192$, i.e., $2(N-1)$ negatives
- The loss is computed across all positive pairs in a mini-batch

Pros

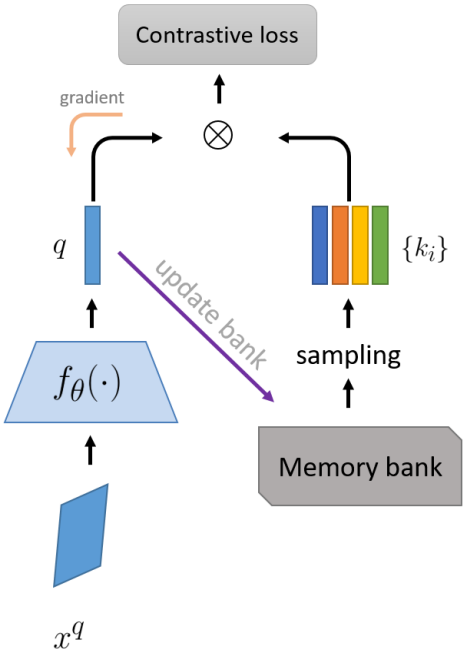
- Lots of negatives

Cons

- Lots of GPUs/TPUs

T. Chen et al., A Simple Framework for Contrastive Learning of Visual Representations, ICML 2020

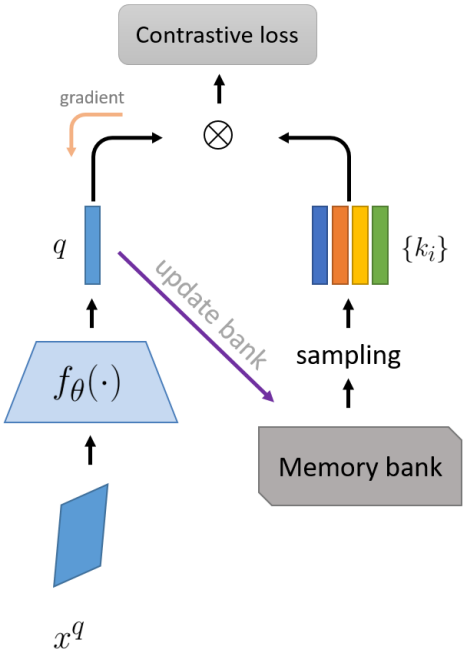
Memory bank



- Keys are randomly sampled from a memory bank of cached features
- The memory bank contains features or all images in the dataset
- There is no backpropagation through the memory bank
- Keys are updated via exponential moving average

Z. Wu et al., Unsupervised Feature Learning via Non-Parametric Instance Disc., CVPR 2018; I. Misra et al., Self-Sup. Learning of Pretext-Invariant Representations, CVPR 2020

Memory bank



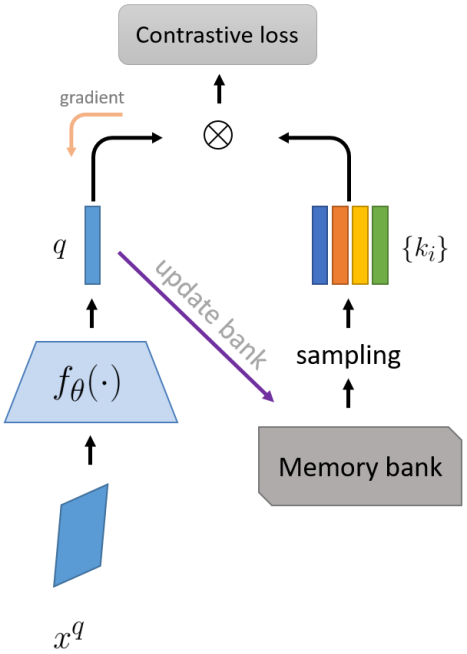
- Keys are randomly sampled from a memory bank of cached features
- The memory bank contains features or all images in the dataset
- There is no backpropagation through the memory bank
- Keys are updated via exponential moving average

Pros

- Many negatives ($K = 65,536$) while GPU memory efficient

Z. Wu et al., Unsupervised Feature Learning via Non-Parametric Instance Disc., CVPR 2018; I. Misra et al., Self-Sup. Learning of Pretext-Invariant Representations, CVPR 2020

Memory bank



- Keys are randomly sampled from a memory bank of cached features
- The memory bank contains features or all images in the dataset
- There is no backpropagation through the memory bank
- Keys are updated via exponential moving average

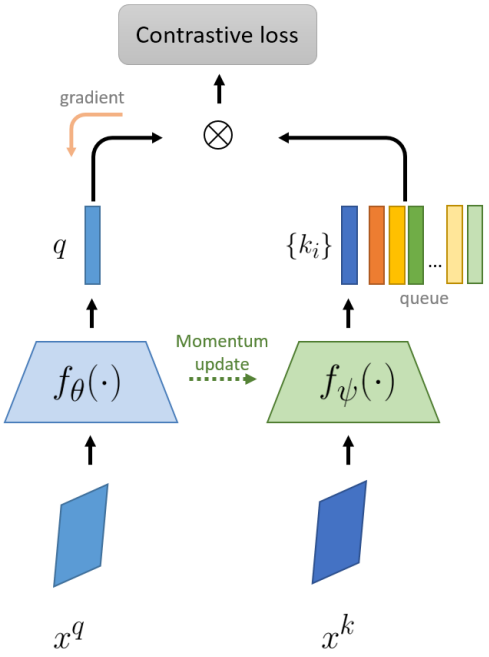
Pros

- Many negatives ($K = 65,536$) while GPU memory efficient

Cons

Z. Wu et al., Unsupervised Feature Learning via Non-Parametric Instance Disc., CVPR 2018; I. Misra et al., Self-Sup. Learning of Pretext-Invariant Representations, CVPR 2020

Momentum encoder (MoCo)

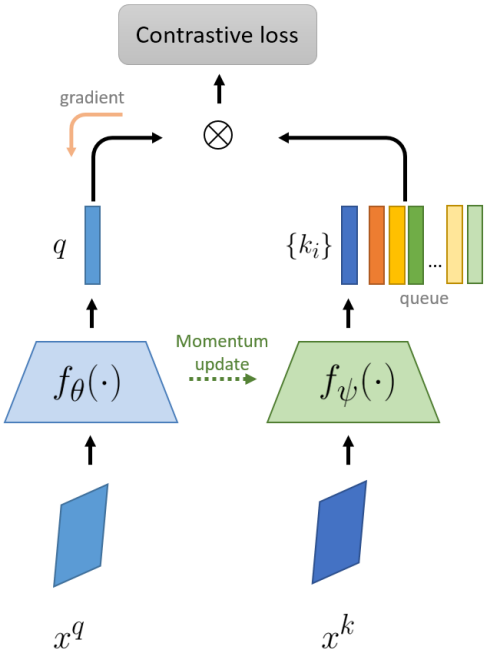


- Momentum encoder relies on a memory-bank with a different update scheme
- The encoder $f_\psi(\cdot)$ is updated instead of the keys themselves
- New samples are continuously added to the memory bank
- The momentum encoder $f_\psi(\cdot)$ is slowly pursuing $f_\theta(\cdot)$ via exponential moving average, i.e. momentum, update:

$$\psi \leftarrow m\psi + (1-m)\theta$$

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

Momentum encoder (MoCo)



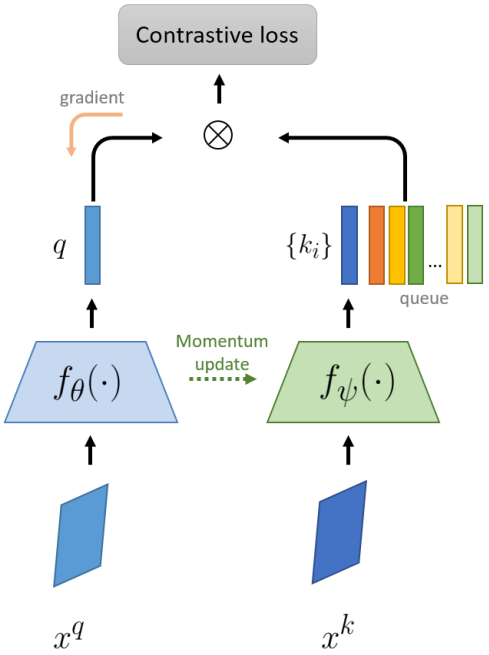
- Momentum encoder relies on a memory-bank with a different update scheme
- The encoder $f_\psi(\cdot)$ is updated instead of the keys themselves
- New samples are continuously added to the memory bank
- The momentum encoder $f_\psi(\cdot)$ is slowly pursuing $f_\theta(\cdot)$ via exponential moving average, i.e. momentum, update:

$$\psi \leftarrow m\psi + (1-m)\theta$$

```
q = f_q.forward(x_q) # queries: NxC
k = f_k.forward(x_k) # keys: NxC
k = k.detach() # no gradient to keys
...
# SGD update: query network
loss.backward()
update(f_q.params)
# momentum update: key network
f_k.params = m*f_k.params+(1-m)*f_q.params
```

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

Momentum encoder (MoCo)



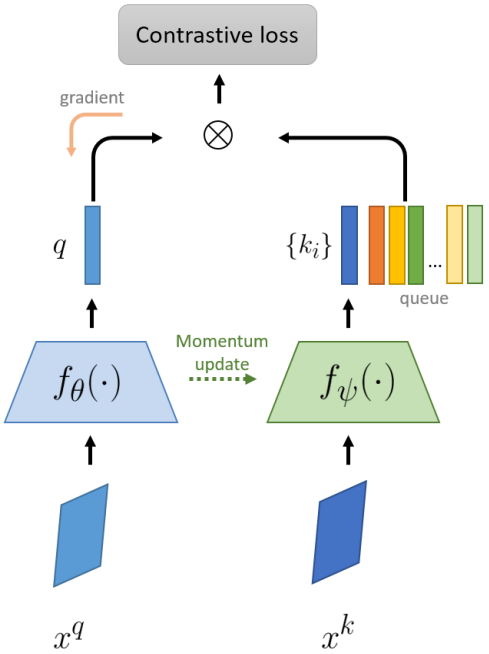
- Momentum encoder relies on a memory-bank with a different update scheme
- The encoder $f_\psi(\cdot)$ is updated instead of the keys themselves
- New samples are continuously added to the memory bank
- The momentum encoder $f_\psi(\cdot)$ is slowly pursuing $f_\theta(\cdot)$ via exponential moving average, i.e. momentum, update:

$$\psi \leftarrow m\psi + (1-m)\theta$$

```
q = f_q.forward(x_q) # queries: NxC
k = f_k.forward(x_k) # keys: NxC
k = k.detach() # no gradient to keys
...
# SGD update: query network
loss.backward()
update(f_q.params)
# momentum update: key network
f_k.params = m*f_k.params+(1-m)*f_q.params
```

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

Momentum encoder (MoCo)



Pros

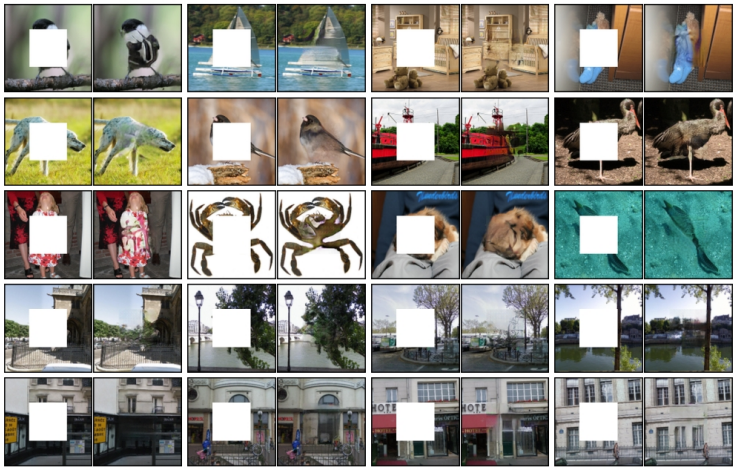
- Elegant and effective solution for large dictionaries

Cons

- Momentum requires tuning: $m \in [0.99, 0.9999]$ works well, while for $m \leq 0.9$ accuracy drops considerably.
- No backprop through the memory bank

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

Feature "reconstruction" methods



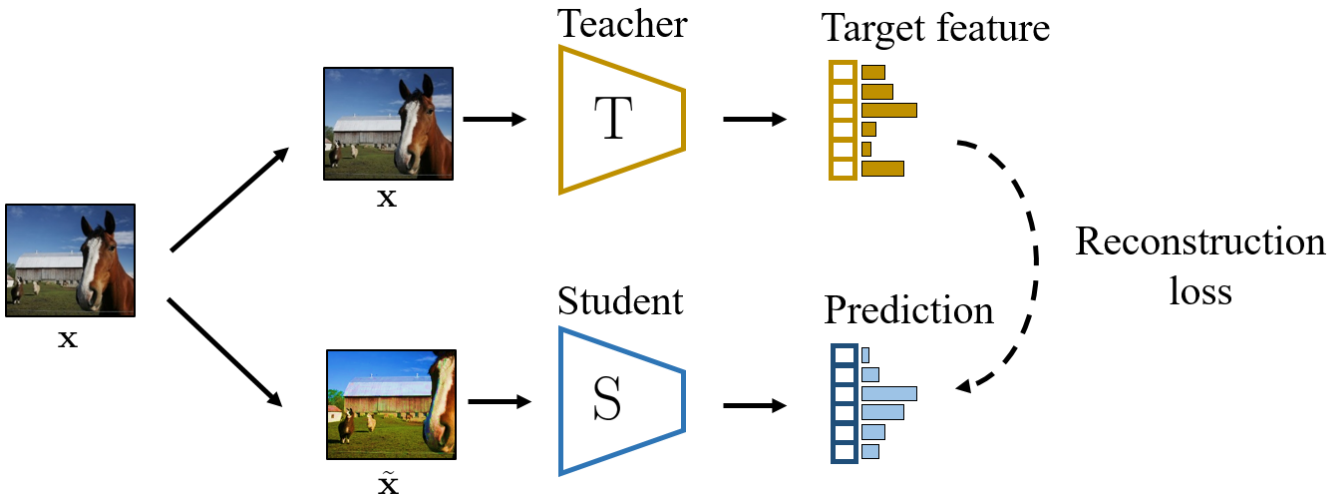
Input reconstruction is a very hard and ambiguous task.

Effort spent on *"useless"* details: exact colour, good boundary, etc., does not necessarily lead to good features.

D. Pathak, Context Encoders: Feature Learning by Inpainting, CVPR 2016

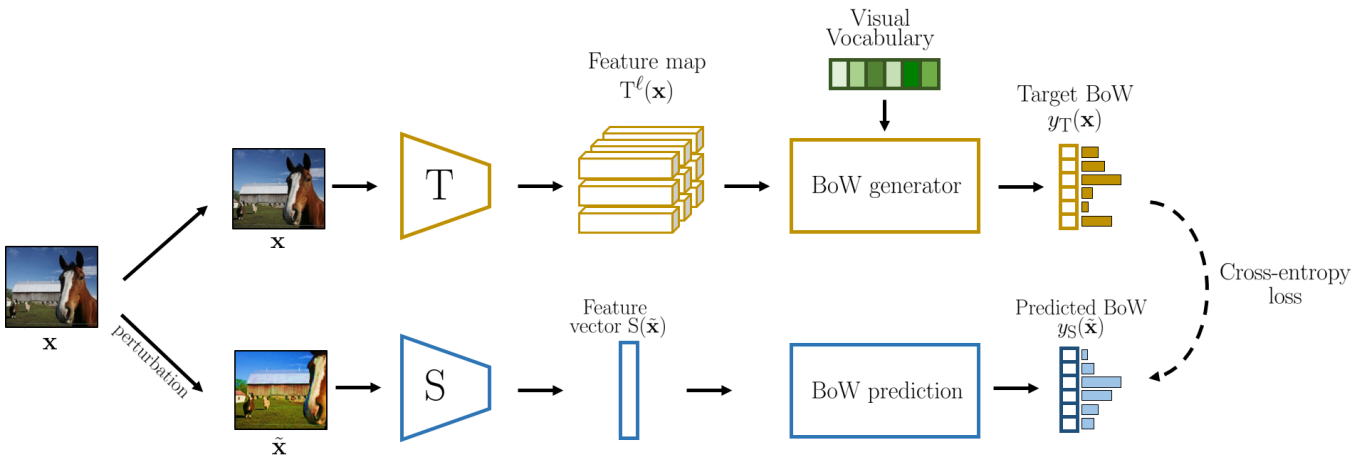
Reconstructing the image features at the output of a teacher network could enable representations, as many details are removed via feature abstraction.

Teacher-student feature "reconstruction"



- **Teacher:** generate a target feature from a given image.
- **Student:** predict this target, given as input a different random view of the same image.

Predicting bag-of-words (BoWNet)

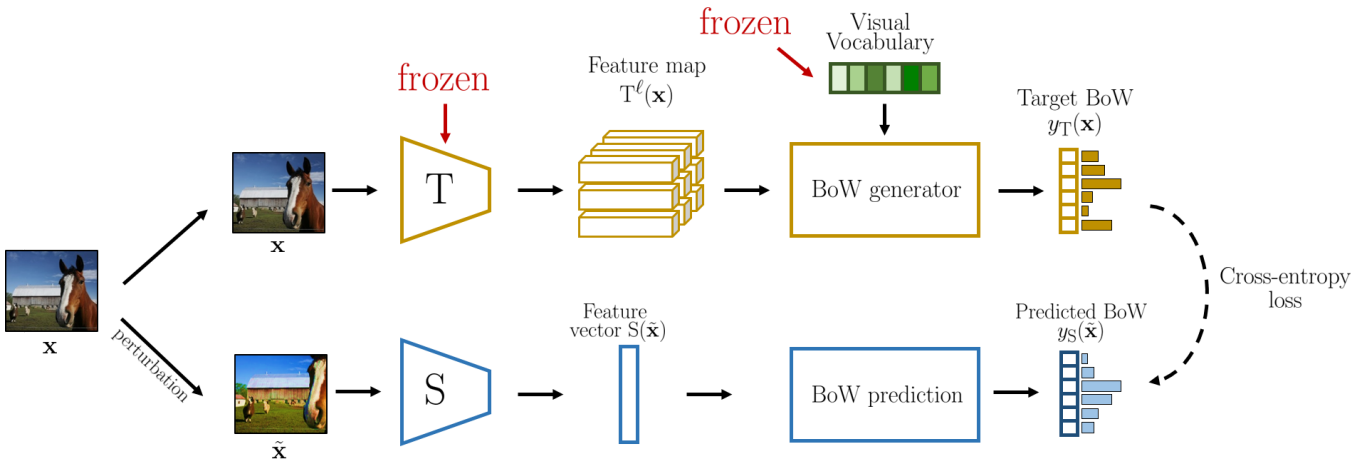


Feature reconstruction method defined over high-level discrete visual words:

- **Teacher:** extract feature maps + convert them to Bag-of-Words (BoW) vectors
- **Student:** must predict the BoW of an image, given as input a perturbed version

S. Gidaris et al., Learning Representations by Predicting Bags of Visual Words, CVPR 2020

Limitation of BoWNet



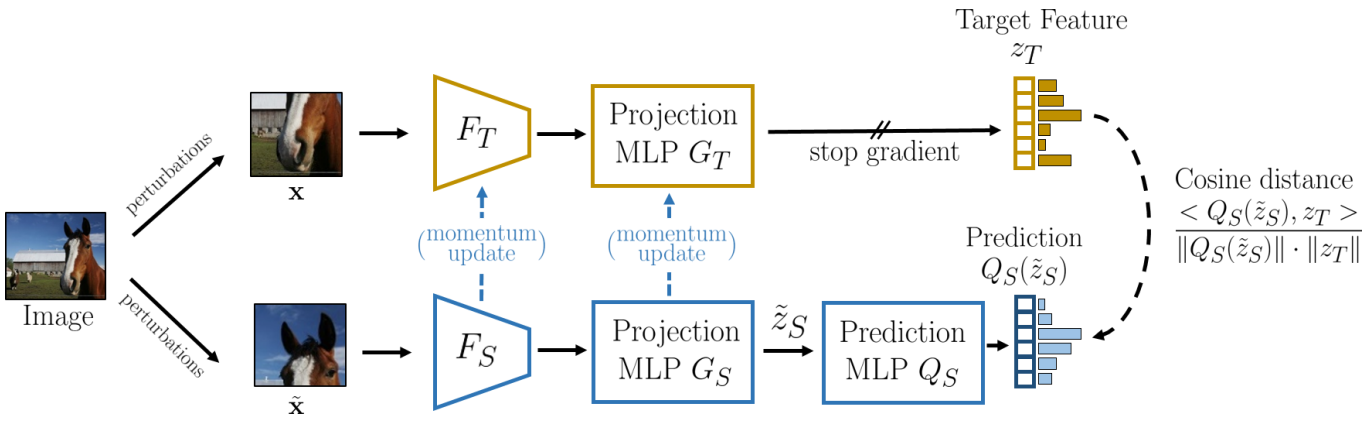
Uses a pre-trained and frozen teacher network

- Requires pre-training with another self-supervised method
- Frozen teacher → suboptimal supervisory signal for the student training

S. Gidaris et al., Learning Representations by Predicting Bags of Visual Words, CVPR 2020

Dynamic teacher-student feature “reconstruction” methods

Bootstrap Your Own Latent (BYOL)

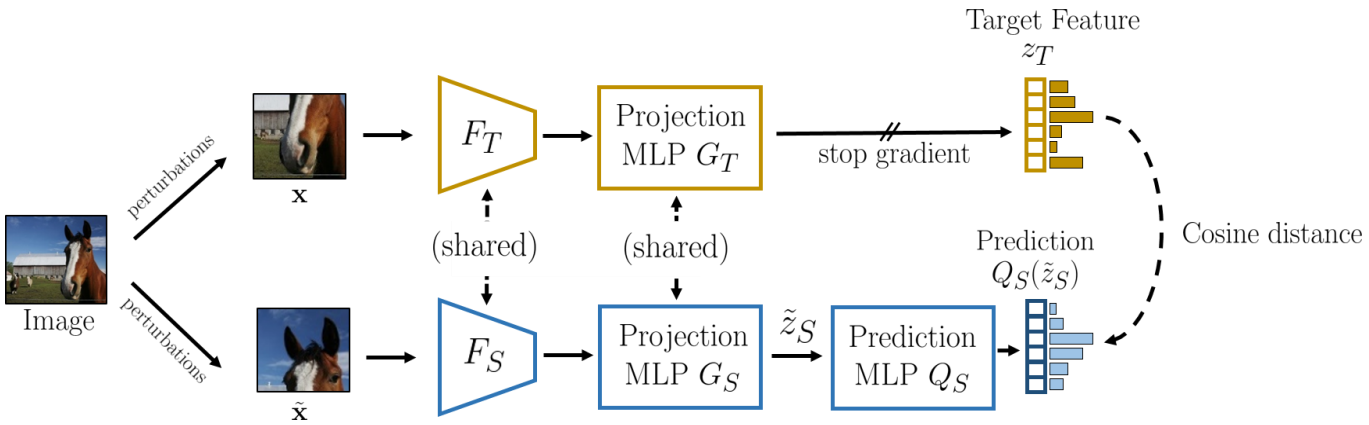


Feature reconstruction method:

- **Teacher:** extract a target feature vector from a random view of an image
- **Student:** predict this target, given as input a different random view of the same image

J.B. Grill et al., Bootstrap your own latent: A new approach to self-supervised Learning, NeurIPS 2020

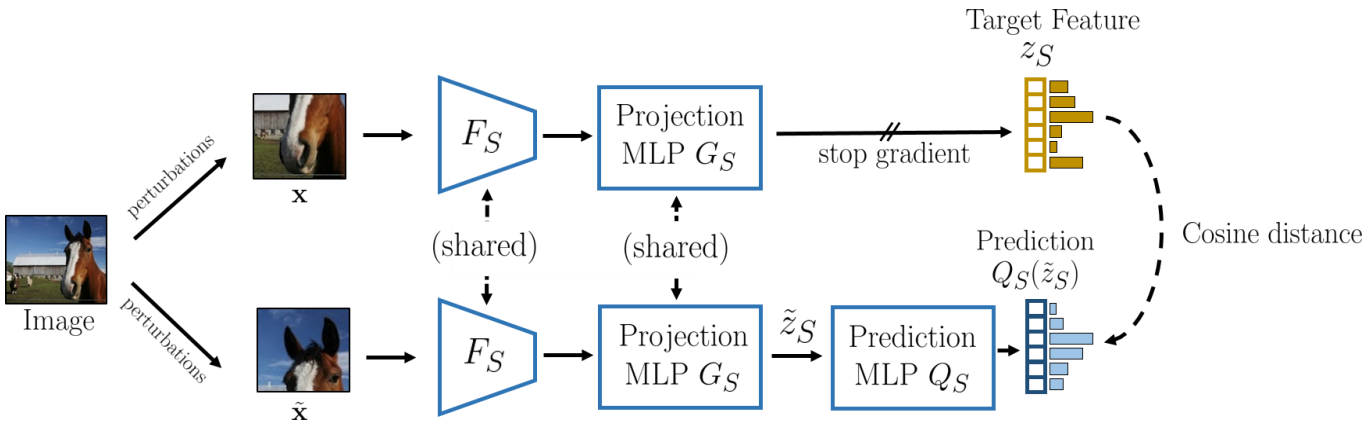
SimSiam



SimSiam: BYOL without the momentum teacher (the teacher is identical to the student)

X. Chen et al., Exploring Simple Siamese Representation Learning, CVPR 2021

SimSiam



SimSiam: BYOL without the momentum teacher (the teacher is identical to the student)

X. Chen et al., Exploring Simple Siamese Representation Learning, CVPR 2020

SSL in the age of Transformers

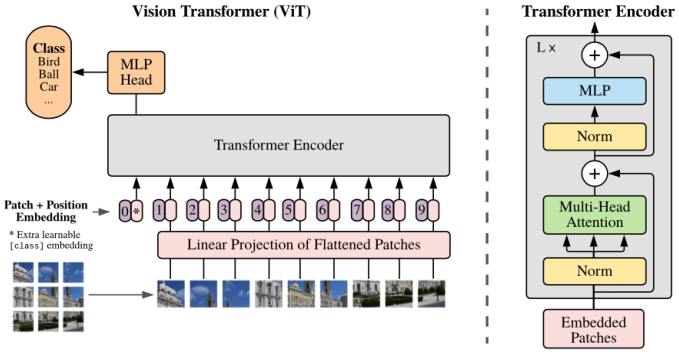
In the last years, the emergence of Vision Transformers (ViT), has led to a new line of SSL approaches strongly related with practices from NLP.

The Vision Transformer (ViT)

- patch trick: generate a sequence from images by cropping non-overlapping patches
- spatial information given by position embeddings
- Transformer layers are identical with one from NLP
- Outperforms ResNet-50 only above 20M training samples

A. Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2021

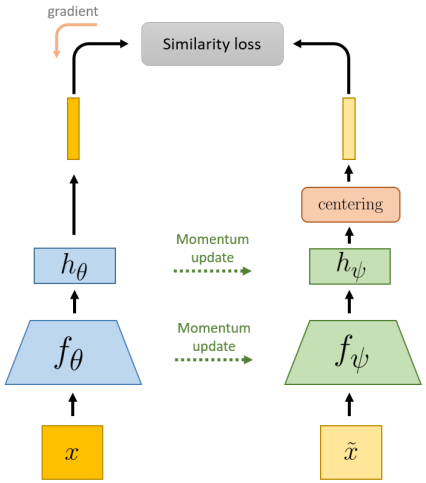
The Vision Transformer (ViT)



- patch trick: generate a sequence from images by cropping non-overlapping patches
- spatial information given by position embeddings
- Transformer layers are identical with one from NLP
- Outperforms ResNet-50 only above 20M training samples

A. Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR 2021

DINO



Main idea: No prediction head; post-processing of teacher outputs to avoid feature collapse

- Centering by subtracting the mean feature: prevents collapsing to constant 1-hot targets
- Sharpening by using low softmax temperature: prevents collapsing to a uniform target vector
- Cross-entropy loss

f_θ, f_ψ: encoder (ViT, ResNet-50);

h_θ, h_ψ: projection (MLP).

Caron et al., Emerging Properties in Self-Supervised Vision Transformers, ICCV 2021

Remember BERT?

Input: The man went to the $[MASK]_1$. He bought a $[MASK]_2$ of milk .
Labels: $[MASK]_1$ = store; $[MASK]_2$ = gallon

Missing word prediction task.

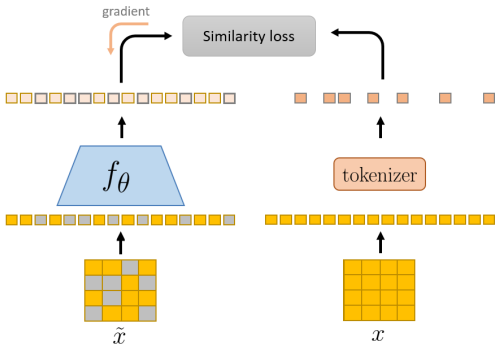
Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

Next sentence prediction task.

T. Mikolov et al., Efficient estimation of word representations in vector space, ArXiv 2013
T. Mikolov et al., Distributed representations of words and phrases and their compositionality, NeurIPS 2013
J. Devlin, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, ArXiv 2018

Masked Image Modelling: BEiT



Main idea: pre-train ViTs by learning to predict tokens of masked patches

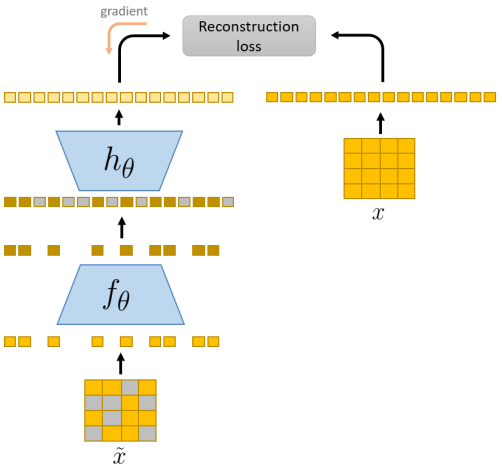
- Mimicking practices from large language models (BERT)
- Learn to produce discrete visual tokens from masked input images
- Use learnable mask-token for masked patches
- Trained with cross-entropy loss over masked tokens

f_θ: encoder (ViT);

tokenizer: pretrained autoencoder (DALL-E).

Bao et al., BEiT: BERT Pre-Training of Image Transformers, ICLR 2022

Masked Image Modelling: MAE



Main idea: learn to reconstruct masked pixels

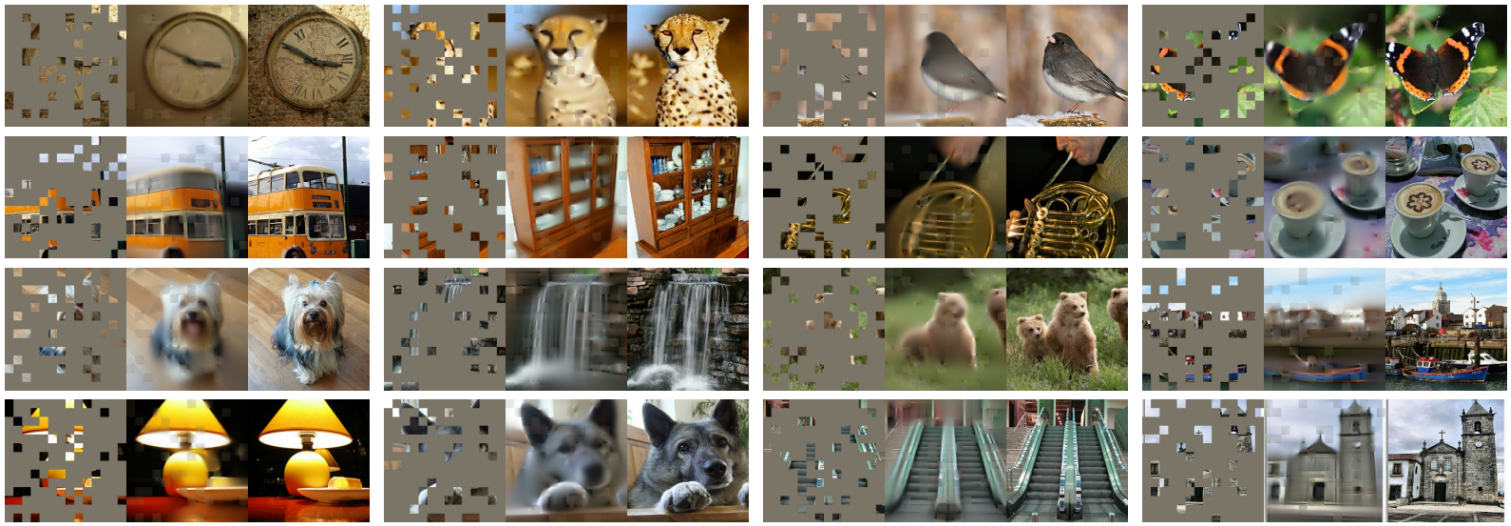
- Simplified MIM pipeline without pre-trained tokenizer nor data augmentation
- Encoder operates only on visible patches without mask tokens
- Lightweight ViT decoder (removed after pre-training)
- Aggressive masking (up to 75% of patches)
- Shines when fine-tuned on the downstream task

f_θ: encoder (ViT);

h_θ: decoder (ViT).

He et al., Masked Autoencoders Are Scalable Vision Learners, CVPR 2022

Masked Image Modelling: MAE



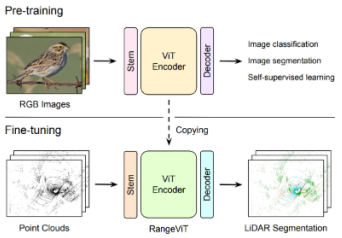
Example results on ImageNet validation images. For each triplet, we show the masked image (left), our MAE reconstruction(middle), and the ground-truth (right). The masking ratio is 80%, leaving only 39 out of 196 patches

He et al., Masked Autoencoders Are Scalable Vision Learners, CVPR 2022

Foundation models such as DINOv2 have significantly changed the way we use pretrained models and they types of "stage 2" applications

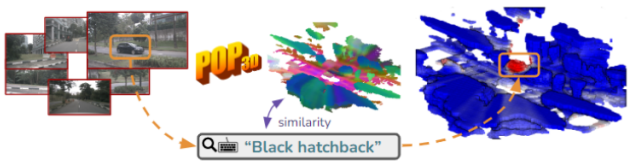
Stage 2: Can also be distillation, data mining, active learning, model init. …

reuse pretrained backbones on other modalities



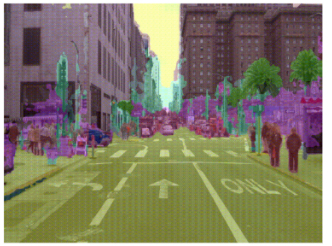
RangeViT [CVPR'23]

text-driven 3D retrieval from cameras



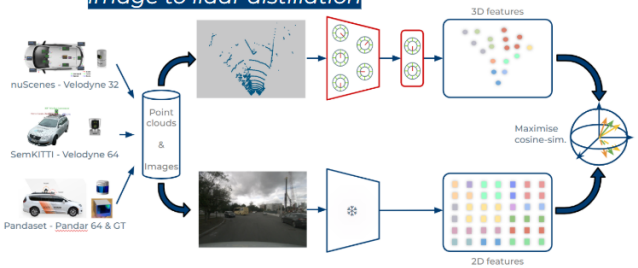
POP-3D [NeurIPS'23]

unsupervised semantic segmentation



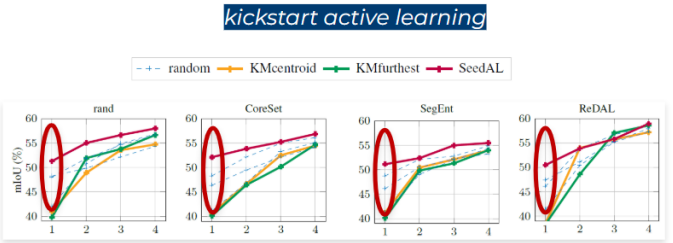
Drive&Segment [ECCV'22]

image to lidar distillation



ScaLR [CVPR'24]

kickstart active learning



SeedAL [ICCV'23]

DINOv2 is the default image encoder in computer vision nowadays

Backbone	Fine-tune Method	Trainable Params*	mIoU Citys	BDD	Map	Avg.
CLIP [63] (ViT-Large)	Full	304.15M	51.3	47.6	54.3	51.1
	Freeze	0.00M	53.7	48.7	55.0	52.4
	Rein	2.99M	**57.1**	**54.7**	**60.5**	**57.4**
MAE [27] (Large)	Full	330.94M	53.7	**50.8**	58.1	54.2
	Freeze	0.00M	43.3	37.8	48.0	43.0
	Rein	2.99M	**55.0**	49.3	**58.6**	**54.3**
SAM [42] (Huge)	Full	632.18M	57.6	51.7	61.5	56.9
	Freeze	0.00M	57.0	47.1	58.4	54.2
	Rein	4.51M	**59.6**	**52.0**	**62.1**	**57.9**
EVA02 [18, 19] (Large)	Full	304.24M	62.1	56.2	64.6	60.9
	Freeze	0.00M	56.5	53.6	58.6	56.2
	Rein	2.99M	**65.3**	**60.5**	**64.9**	**63.6**
DINOV2 [58] (Large)	Full	304.20M	63.7	57.4	64.2	61.7
	Freeze	0.00M	63.3	56.1	63.9	61.1
	Rein	2.99M	**66.4**	**60.4**	**66.1**	**64.3**

ReIN

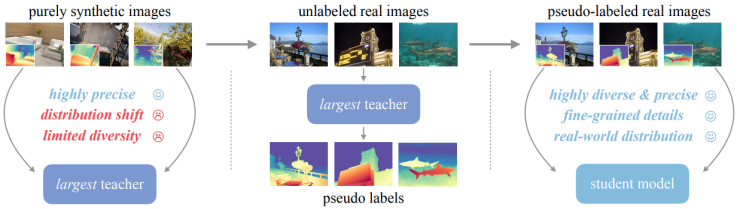


Figure 7: Depth Anything V2. We first train the most capable teacher on precise synthetic images. Then, to mitigate the distribution shift and limited diversity of synthetic data, we annotate unlabeled real images with the teacher. Finally, we train student models on high-quality pseudo-labeled images.

DepthAnythingV2

Method	Ref.	BRAVO↑	Semantic↑	OOD↑
DINOv2-OOD	Sec. 2.2	**77.9**	**69.8**	**88.1**
PixOOD w/ ResNet-101 DeepLabv3 [26]	Sec. 2.3	61.2	58.7	64.0
Ensemble	Sec. 2.4	61.1	64.3	58.2
PhyFea [1]	Sec. 2.5	33.6	66.3	22.5
Baseline: SegFormer-B5 [30]	-	47.1	45.3	49.2
Baseline: ObsNet-ResNet101 [2]	-	45.3	51.5	40.5
Baseline: RbA Swin-B [16]	-	37.7	27.7	59.2

BRAVO Challenge results

Z. Wei et al., Stronger, Fewer, & Superior: Harnessing Vision Foundation Models for Domain Generalized Semantic Segmentation, CVPR 2024 | T. Vu et al., The BRAVO Semantic Segmentation Challenge Results in UNCV2024, ECCVW 2024 | L. Yang et al., Depth Anything V2, NeurIPS 2024

Today

Transfer learning

Off-the shelf networks

Fine-tuning

(Task) transfer learning

Multi-task learning

Domain adaptation

Self-supervised learning

The end.