



# Intermediate Deep Learning for Engineers

---

Spring 2025, Deep Learning for Engineers  
April 10, 2025, 9<sup>th</sup> session

Amir Barati Farimani

*Associate Professor of Mechanical Engineering and Bio-Engineering  
Carnegie Mellon University*

# Lecture Outline

- Self-supervised learning
  - Motivation
  - Self-supervised learning versus other machine learning techniques
- Image-based approaches
  - Geometric transformation recognition (image rotation)
  - Patches (relative patch position, image jigsaw puzzle)
  - Generative modeling (context encoders, image colorization, cross-channel prediction, image super-resolution)
  - Automated label generation (deep clustering, synthetic imagery)
  - Contrastive learning (CPC, SimCLR, other contrastive approaches)
- Video-based approaches
  - Frame ordering, tracking moving objects, video colorization
- Approaches for natural language processing

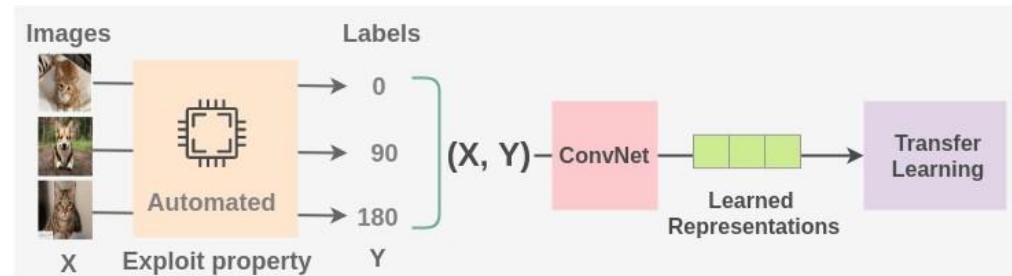
# Supervised vs Unsupervised Learning

- **Supervised learning** – learning with **labeled data**
  - Approach: collect a large dataset, manually label the data, train a model, deploy
  - It is the dominant form of ML at present
  - Learned **feature representations** on large datasets are often transferred via pre-trained models to smaller domain-specific datasets
- **Unsupervised learning** – learning with **unlabeled data**
  - Approach: discover patterns in data either via clustering similar instances, or density estimation, or dimensionality reduction ...
- **Self-supervised learning** – representation learning with **unlabeled data**
  - Learn useful **feature representations** from unlabeled data through **pretext tasks**
  - The term “self-supervised” refers to creating **its own supervision** (i.e., without supervision, without labels)
  - Self-supervised learning is one category of unsupervised learning

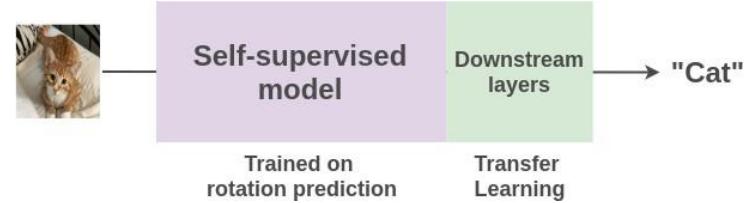
# Self-Supervised Learning

Self-supervised learning example

**Pretext task:** train a model to predict the rotation degree of rotated images with cats and dogs (we can collect million of images from internet, labeling is not required)

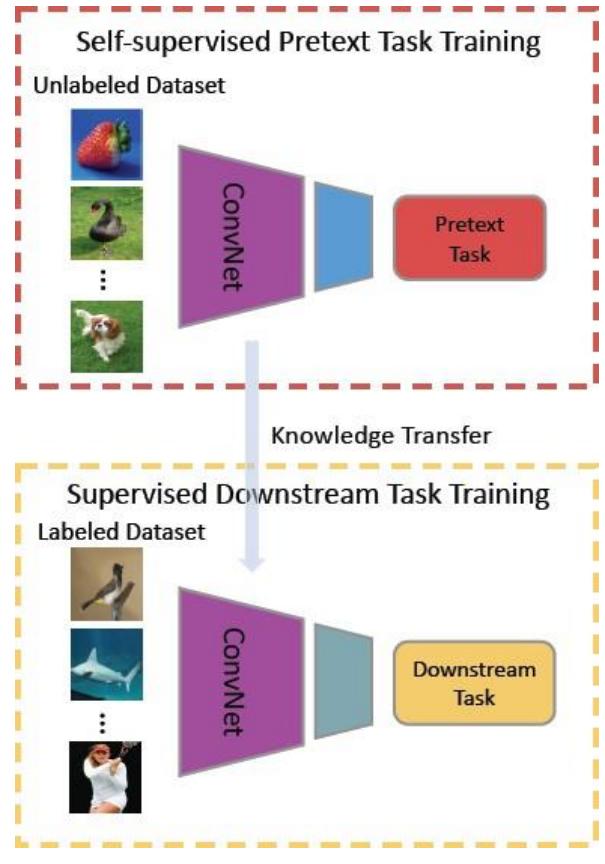


**Downstream task:** use transfer learning and fine-tune the learned model from the pretext task for **classification** of cats vs dogs with very few labeled examples



# Self-Supervised Learning

One more depiction of the general pipeline for self-supervised learning is shown in the figure



For the downstream task, reuse the trained ConvNet base model, and fine-tune the top layers on a small labeled dataset

# Self-Supervised Learning

## WHY SELF-SUPERVISED LEARNING?

- Creating labeled datasets for each task is an expensive, time-consuming, tedious task  
Requires hiring human labelers, preparing labeling manuals, creating GUIs, creating storage pipelines, etc.  
High quality annotations in certain domains can be particularly expensive (e.g., medicine)
- Self-supervised learning takes advantage of the vast amount of unlabeled data on the internet (images, videos, text)  
Rich discriminative features can be obtained by training models without actual labels
- Self-supervised learning can potentially generalize better because we learn more about the world

## CHALLENGES FOR SELF-SUPERVISED LEARNING

- How to select a suitable pretext task for an application
- There is no gold standard for comparison of learned feature representations
- Selecting a suitable loss functions, since there is no single objective as the test set accuracy in supervised learning

# Self-Supervised Learning

- Self-supervised learning versus unsupervised learning
  - **Self-supervised learning (SSL)**
    - Aims to extract useful **feature representations** from raw unlabeled data through **pretext tasks**
    - Apply the feature representation to improve the performance of **downstream tasks**
  - **Unsupervised learning**
    - Discover patterns in unlabeled data, e.g., for clustering or dimensionality reduction
    - Note also that the term “self-supervised learning” is sometimes used interchangeably with “unsupervised learning”
- Self-supervised learning versus transfer learning
  - Transfer learning is often implemented in a supervised manner
    - E.g., learn features from a labeled ImageNet, and transfer the features to a smaller dataset
  - SSL is a type of transfer learning approach implemented in an unsupervised manner
- Self-supervised learning versus data augmentation
  - Data augmentation is often used as a regularization method in supervised learning
  - In SSL, image rotation or shifting are used for feature learning in raw unlabeled data

# Image-Based Approaches

## IMAGE BASED APPROACHES

- Geometric transformation recognition
  - Image rotation
- Patches
  - Relative patch position, image jigsaw puzzle
- Generative modeling
  - Context encoders, image colorization, cross-channel prediction, image super-resolution
- Automated label generation
  - Deep clustering, synthetic imagery
- Contrastive learning
  - Contrastive predictive coding (CPC), SimCLR, and other contrastive approaches

# Image Rotation

## Geometric transformation recognition

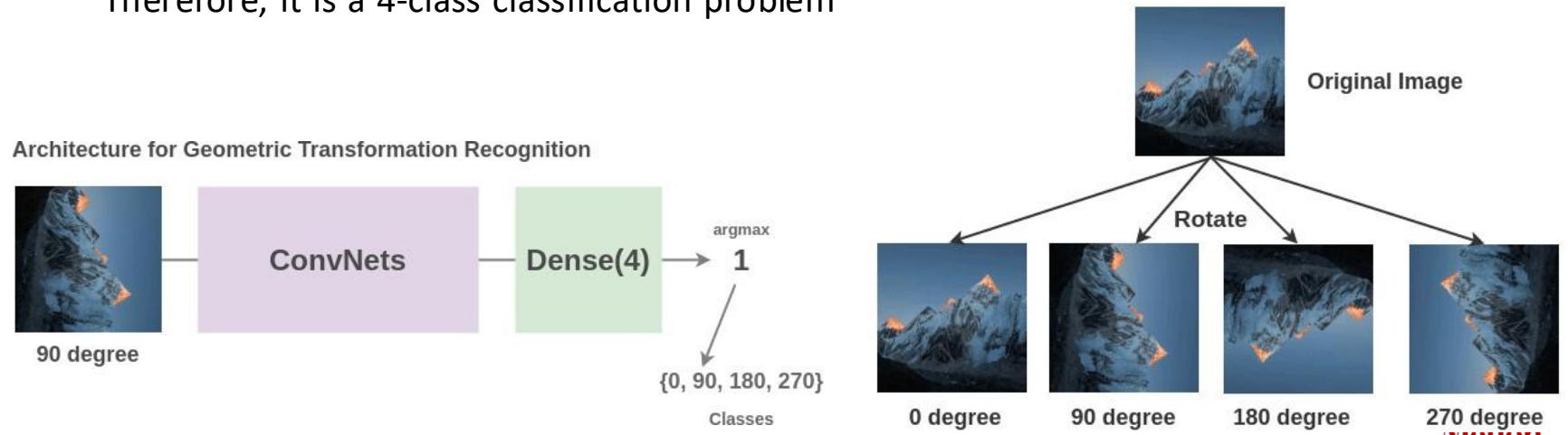
Gidaris (2018) - Unsupervised Representation Learning by Predicting Image Rotations

**Training data:** images rotated by a multiple of  $90^\circ$  at random

This corresponds to four rotated images at  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ , and  $270^\circ$

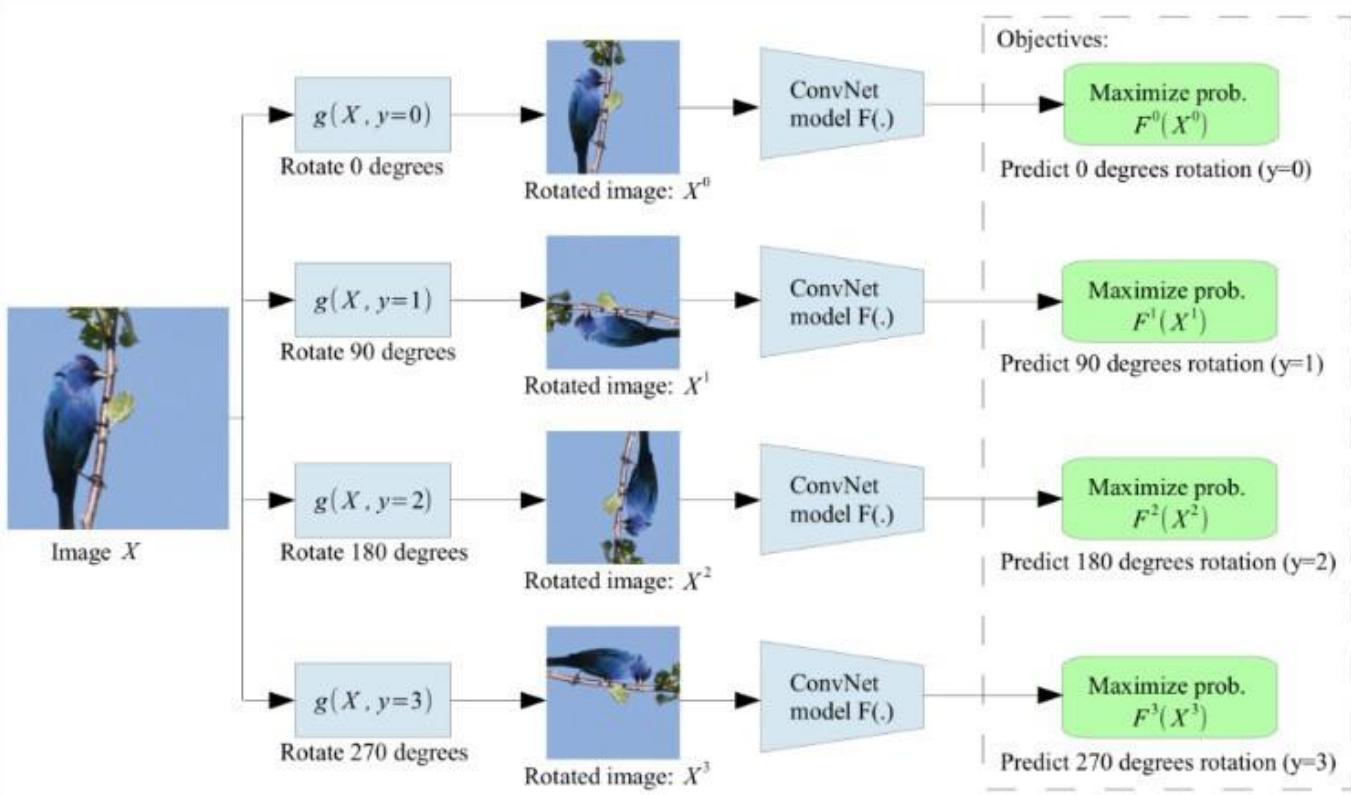
**Pretext task:** train a model to **predict the rotation degree** that was applied

Therefore, it is a 4-class classification problem



# Image Rotation

- A single ConvNet model is used to predict one of the four rotations
- The model needs to understand the location and type of the objects in images to determine the rotation degree



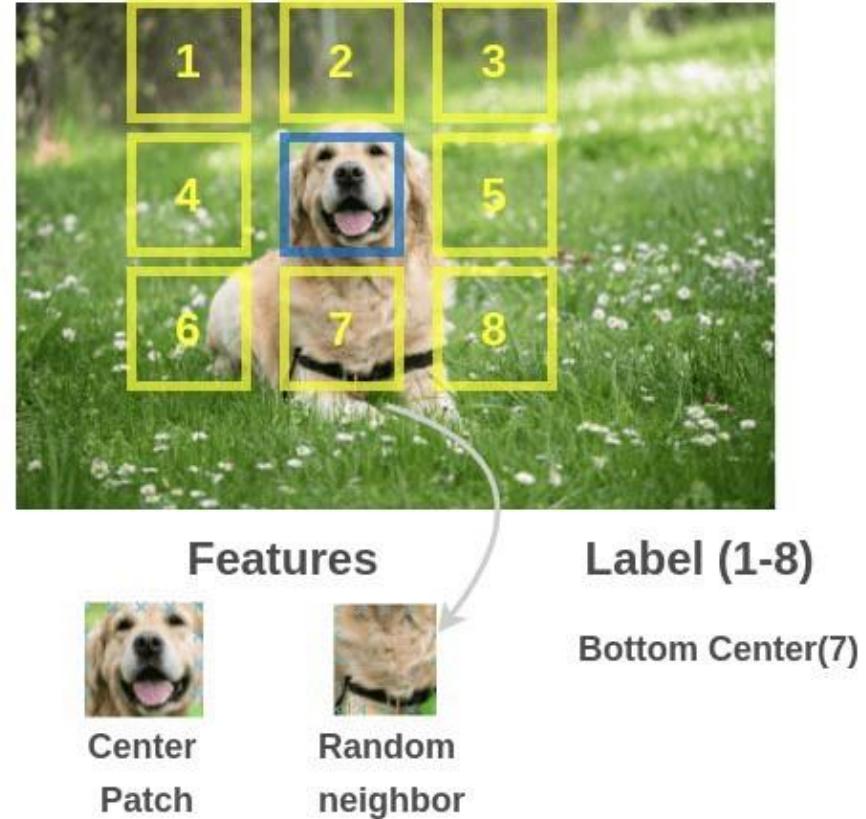
# Image Rotation

- Evaluation on the PASCAL VOC dataset for classification, detection, and segmentation tasks
  - The model (RotNet) is trained in SSL manner, and fine-tuned afterwards
  - RotNet outperformed all other SSL methods
  - The learned features are not as good as the supervised learned features based on transfer learning from ImageNet, but they demonstrate a potential

		Classification (%mAP)	Detection (%mAP)	Segmentation (%mIoU)
	Trained layers	fc6-8	all	all
Supervised feature learning	ImageNet labels	78.9	79.9	56.8 48.0
	Random		53.3	43.4 19.8
	Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6 32.6
	Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9
	Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5 29.7
	Tracking (Wang & Gupta, 2015)	55.6	63.1	47.4
	Context (Doersch et al., 2015)	55.1	65.3	51.1
	Colorization (Zhang et al., 2016a)	61.5	65.6	46.9 35.6
	BIGAN (Donahue et al., 2016)	52.3	60.1	46.9 34.9
	Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2 37.6
Proposed self-supervised feature learning	NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4
	Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7 36.0
	ColorProxy (Larsson et al., 2017)		65.9	38.4
	Counting (Noroozi et al., 2017)	-	67.7	51.4 36.6
	(Ours) RotNet	<b>70.87</b>	<b>72.97</b>	<b>54.4</b> <b>39.1</b>

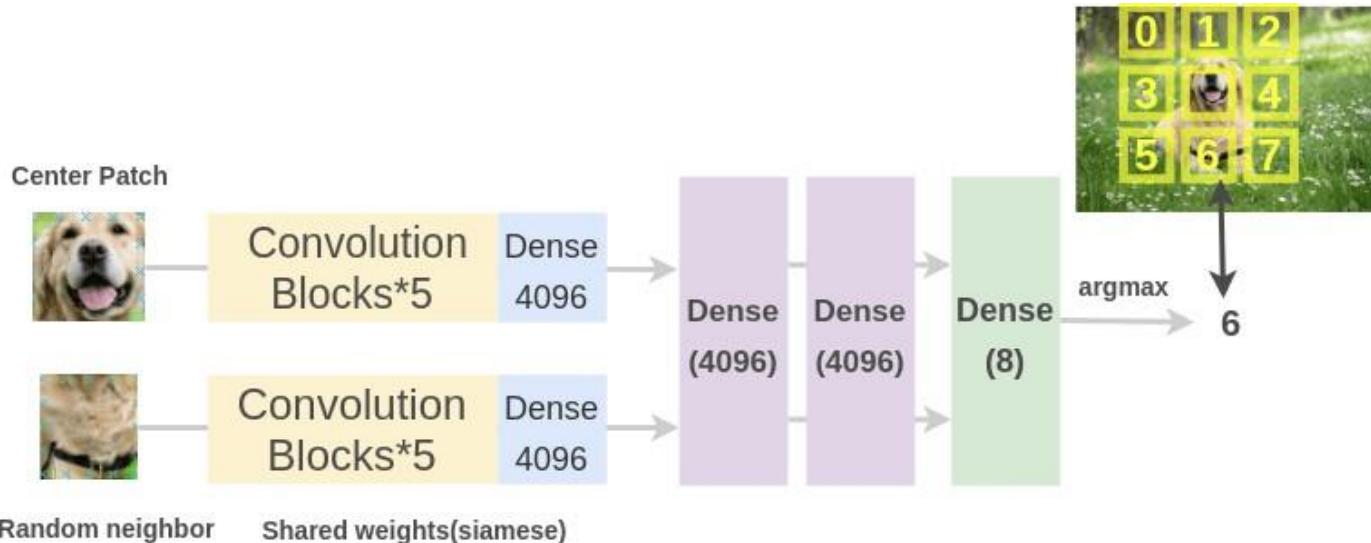
# Relative Patch Position

- **Relative patch position for context prediction**
  - Dorsch (2015) Unsupervised Visual Representation Learning by Context Prediction
- Training data: multiple **patches** extracted from images
- Pretext task: train a model to **predict the relationship between the patches**
  - E.g., predict the relative position of the selected patch below (i.e., position # 7)
    - For the center patch, there are 8 possible neighbor patches (8 possible classes)



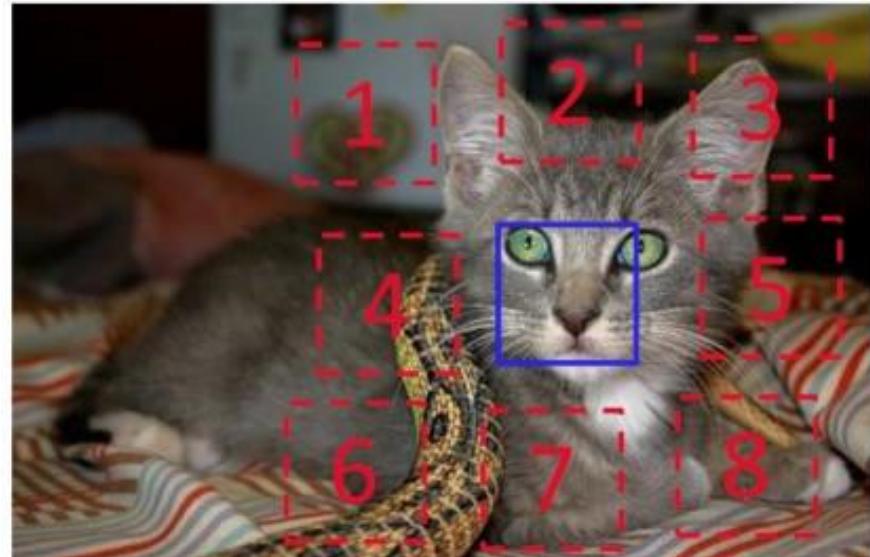
# Relative Patch Position

- The patches are inputted into two ConvNets with shared weights
  - The learned features by the ConvNets are concatenated
  - Classification is performed over 8 classes (denoting the 8 possible neighbor positions)
- The model needs to understand the spatial context of images, in order to predict the relative positions between the patches



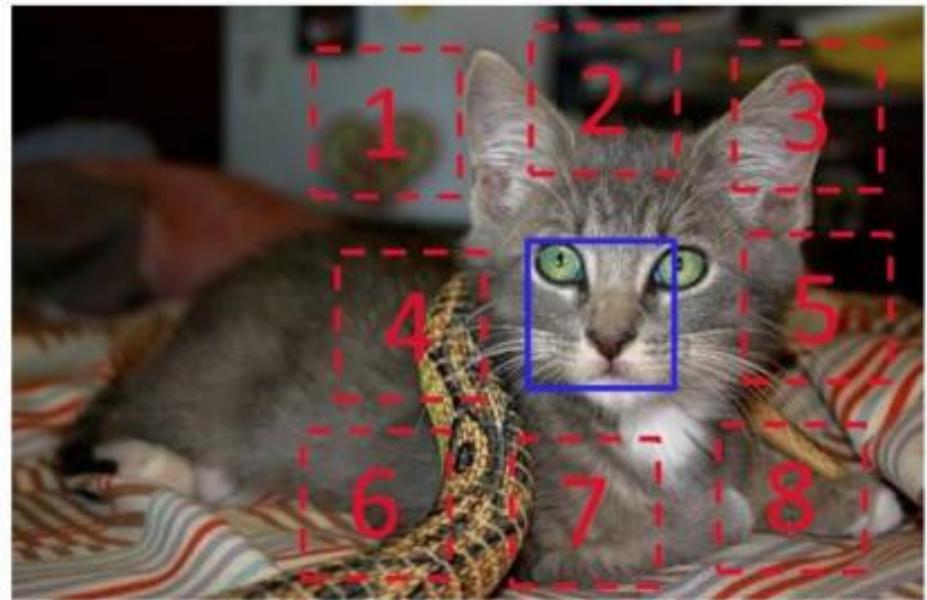
# Relative Patch Position

- The training patches are **sampled** in the following way:
  - Randomly sample the first patch, and consider it the middle of a 3x3 grid
  - Sample from 8 neighboring locations of the first central patch (blue patch)
- To avoid the model only catching low-level trivial information:
  - Add gaps between the patches
  - Add small jitters to the positions of the patches
  - Randomly downsample some patches to reduced resolution, and then upsample them
  - Randomly drop 1 or 2 color channels for some patches



# Relative Patch Position

For instance, predict the position of patch # 3 with respect to the central patch

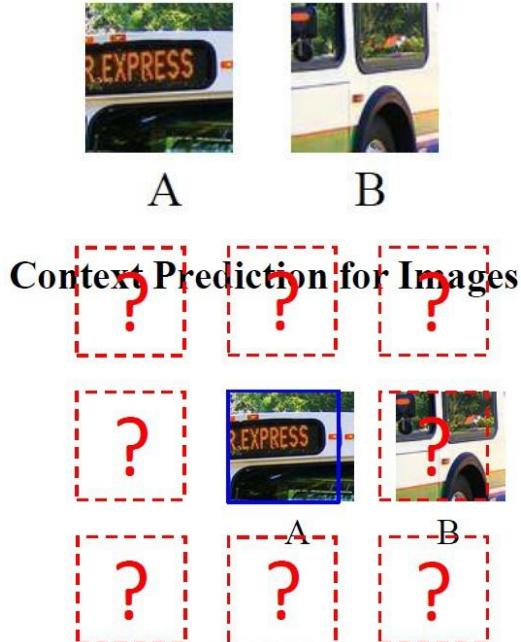


Input: two patches  
Prediction:  $Y = 3$

$$X = \left( \begin{array}{c} \text{Patch 5} \\ \text{Patch 3} \end{array} \right)$$

# Relative Patch Position

Example: predict the position of patch B with respect to patch A



# Image Jigsaw Puzzle

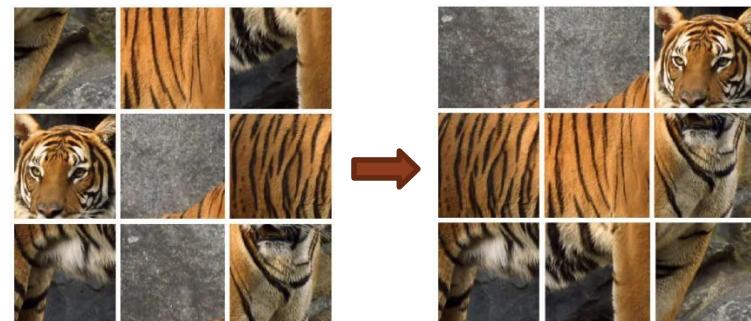
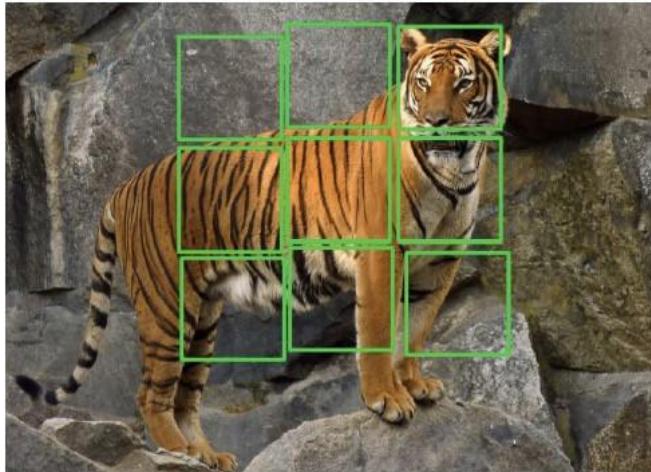
## *Predict patches position in a jigsaw puzzle*

Noroozi (2016) Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

**Training data:** 9 patches extracted in images (similar to the previous approach)

**Pretext task:** predict the **positions of all 9 patches**

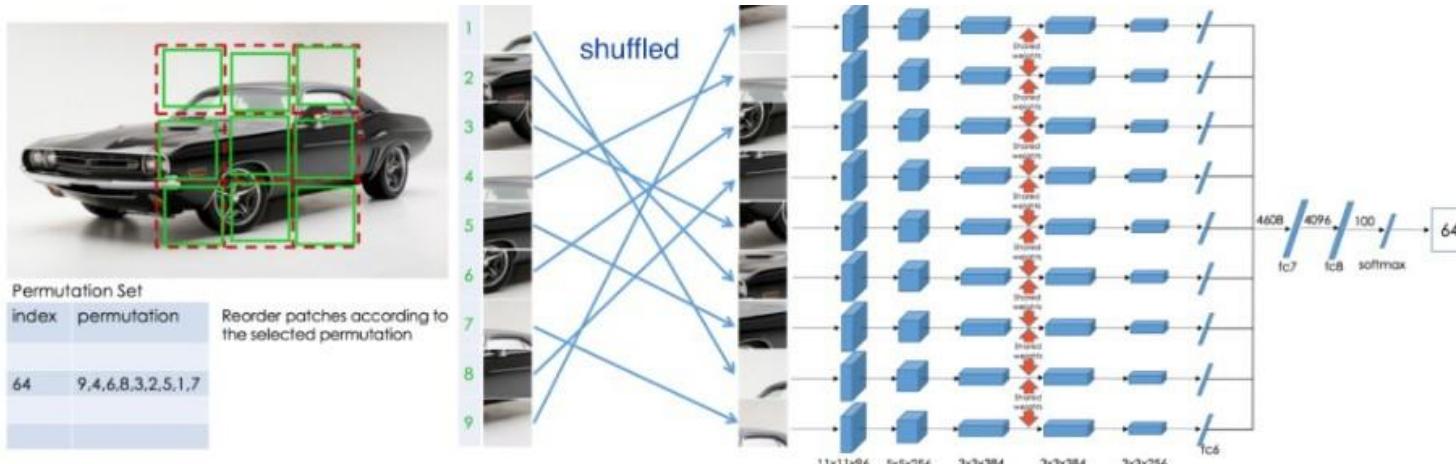
Instead of predicting the relative position of only 2 patches, this approach uses the grid of  $3 \times 3$  patches and solves a jigsaw puzzle



# Image Jigsaw Puzzle

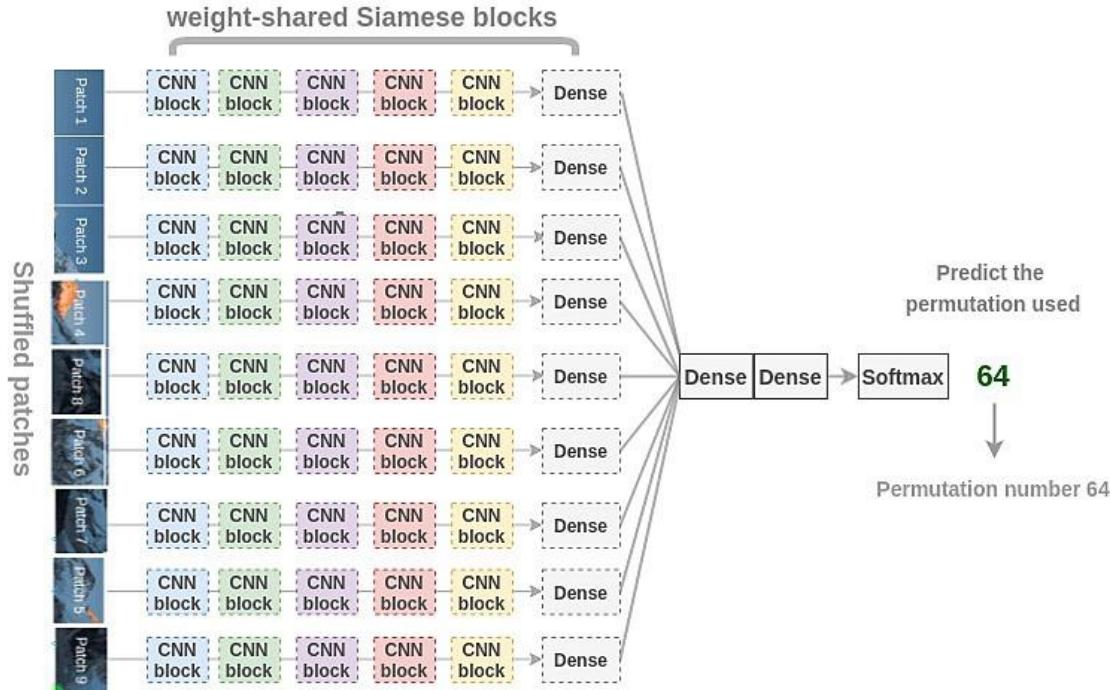
A ConvNet model passes the individual patches through the same Conv layers that have shared weights

- The features are combined and passed through fully-connected layers
- Output is the positions of the patches (i.e., the shuffling permutation of the patches)
- The patches are shuffled according to a set of 64 predefined permutations
- Namely, for 9 patches, in total there are 362,880 possible puzzles
- The authors used a small set of 64 shuffling permutations with the highest hamming distance



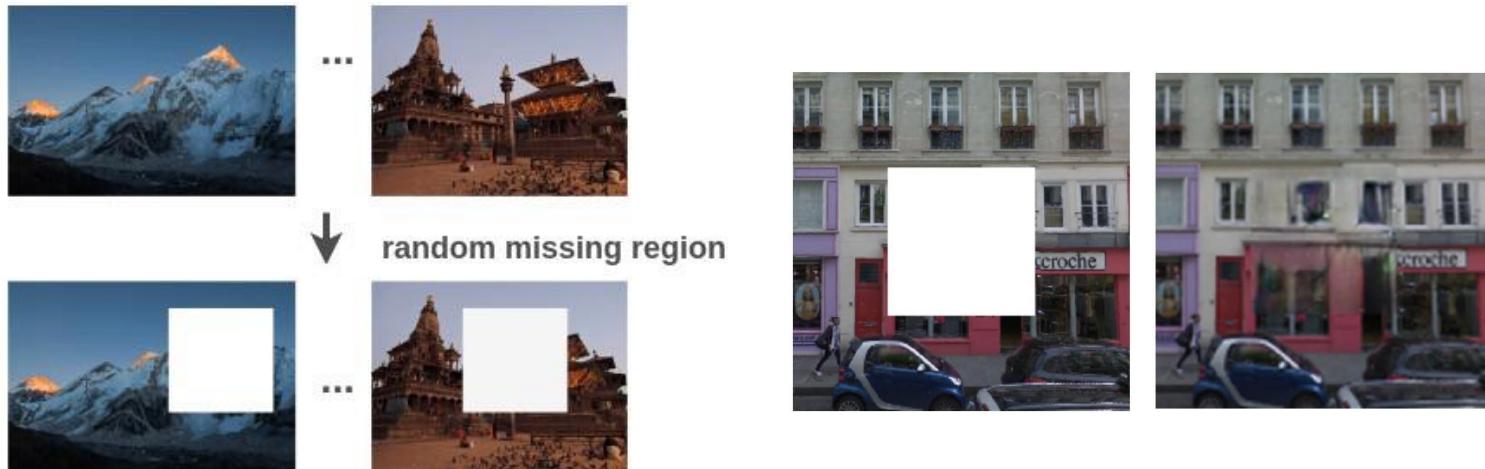
# Image Jigsaw Puzzle

- The model needs to learn to identify how parts are assembled in an object, relative positions of different parts of objects, and shape of objects
- The learned representations are useful for downstream tasks in classification and object detection



# Context Encoders

- **Predict missing pieces**, also known as **context encoders**, or **inpainting**
  - Pathak (2016) Context Encoders: Feature Learning by Inpainting
- **Training data**: remove a random region in images
- **Pretext task**: fill in a **missing piece** in the image
  - The model needs to understand the content of the entire image, and produce a plausible replacement for the missing piece



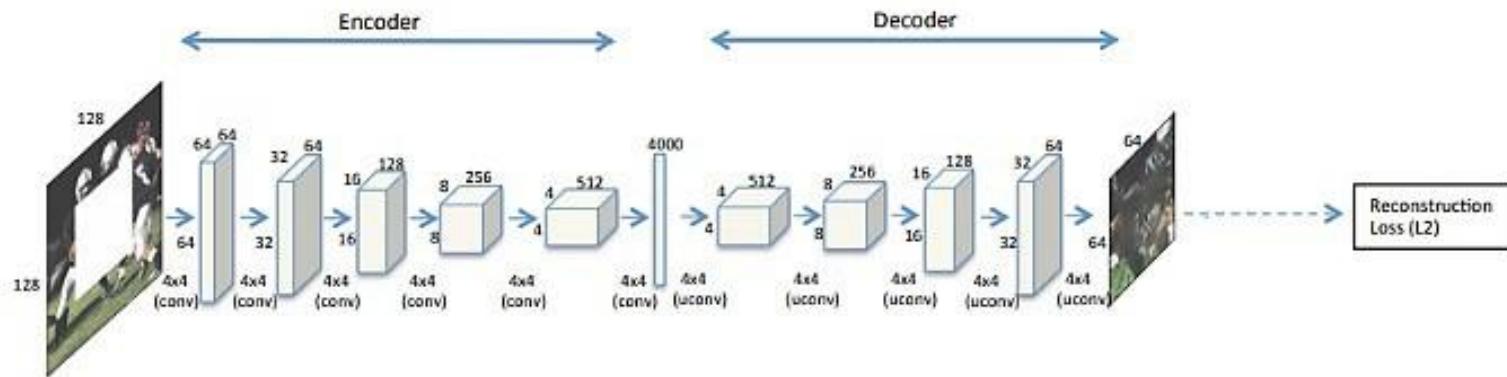
# Context Encoders

The initially considered model uses an **encoder-decoder** architecture

The encoder and decoder have multiple Conv layers, and a shared central fully-connected layer

The output of the decoder is the reconstructed input image

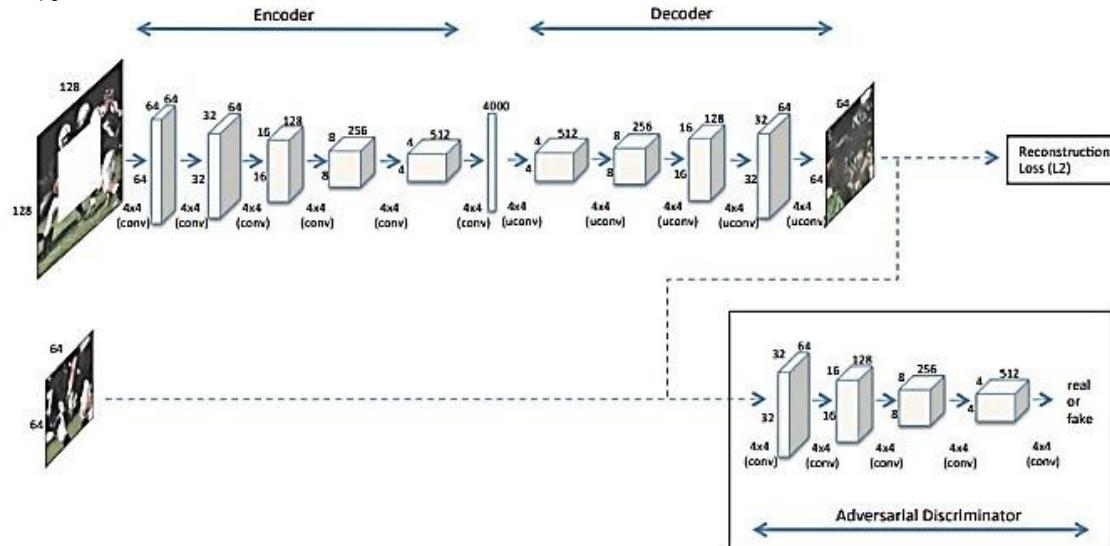
A Euclidean  $\ell_1$  distance is used as the reconstruction loss function  $\mathcal{L}^{\text{recon}}$



# Context Encoders

- The authors found that the reconstruction loss alone couldn't capture fine details in the missing region
- Improvement was achieved by adding a GAN branch, where the generator of the GAN learns to reconstruct the missing piece
  - The overall loss function is a weighted combination of the reconstruction and the GAN losses, i.e.,

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{GAN}}$$



# Context Encoders

The joint loss function  $\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2$  resulted in improved prediction of the missing piece



Input  
image

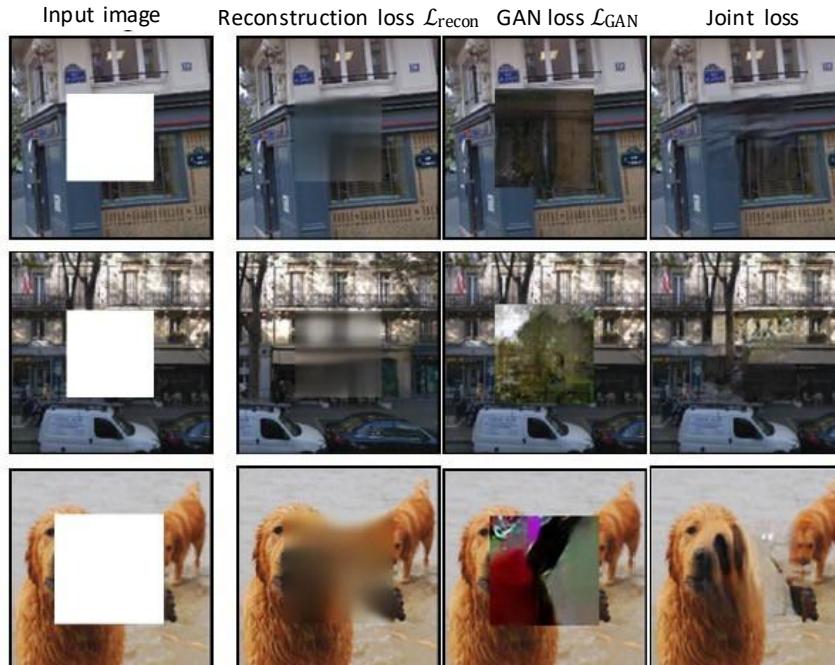
Encoder-  
decoder with  
reconstructio  
n loss  $\mathcal{L}_1$

GAN with loss  
 $\mathcal{L}_2$

Joint loss  
 $\mathcal{L} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2$

# Context Encoders

- Additional examples comparing the used loss functions
  - The joint loss produces the most realistic images



# Context Encoders

- The removed regions can have different shapes
  - Random region and random block masks outperformed the central region features

Central region



Random block



Random region



# Context Encoders

- Evaluation on PASCAL VOC for several downstream tasks
  - The learned features by the context encoder are not as good as supervised features
  - But are comparable to other unsupervised methods, and perform better than randomly initialized models
    - E.g., over 10% improvement for segmentation over random initialization

Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	<b>78.2%</b>	<b>56.8%</b>	<b>48.0%</b>
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal <i>et al.</i> [1]	egomotion	10 hours	52.9%	41.8%	-
Doersch <i>et al.</i> [7]	context	4 weeks	55.3%	<b>46.6%</b>	-
Wang <i>et al.</i> [39]	motion	1 week	<b>58.4%</b>	44.0%	-
Ours	context	14 hours	56.5%	44.5%	<b>29.7%</b>

# Image Colorization

## *Image colorization*

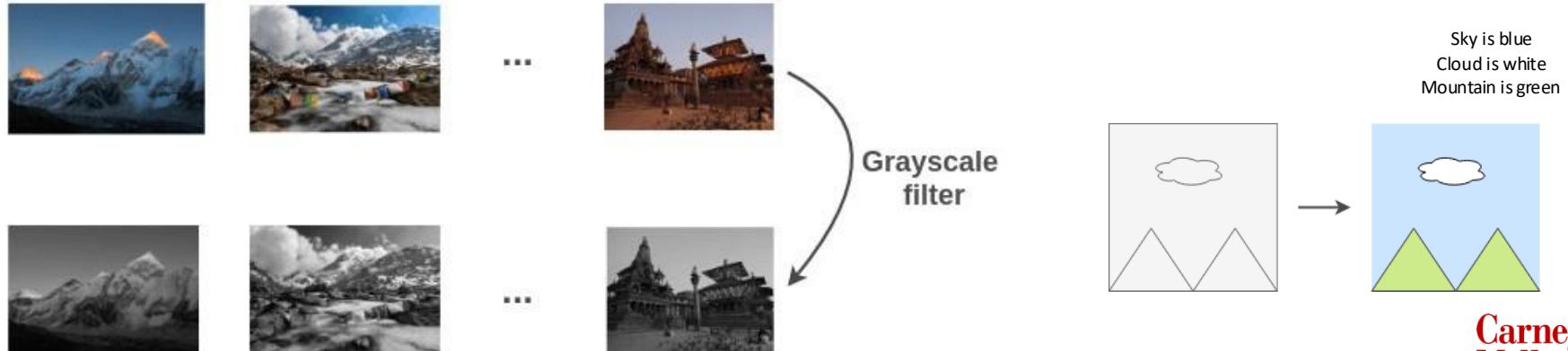
### Zhang (2016) Colorful Image Colorization

Training data: pairs of color and grayscale images

Pretext task: predict the colors of the objects in grayscale images

The model needs to understand the objects in images and paint them with a suitable color

Right image: learn that the sky is blue, cloud is white, mountain is green



# Image Colorization

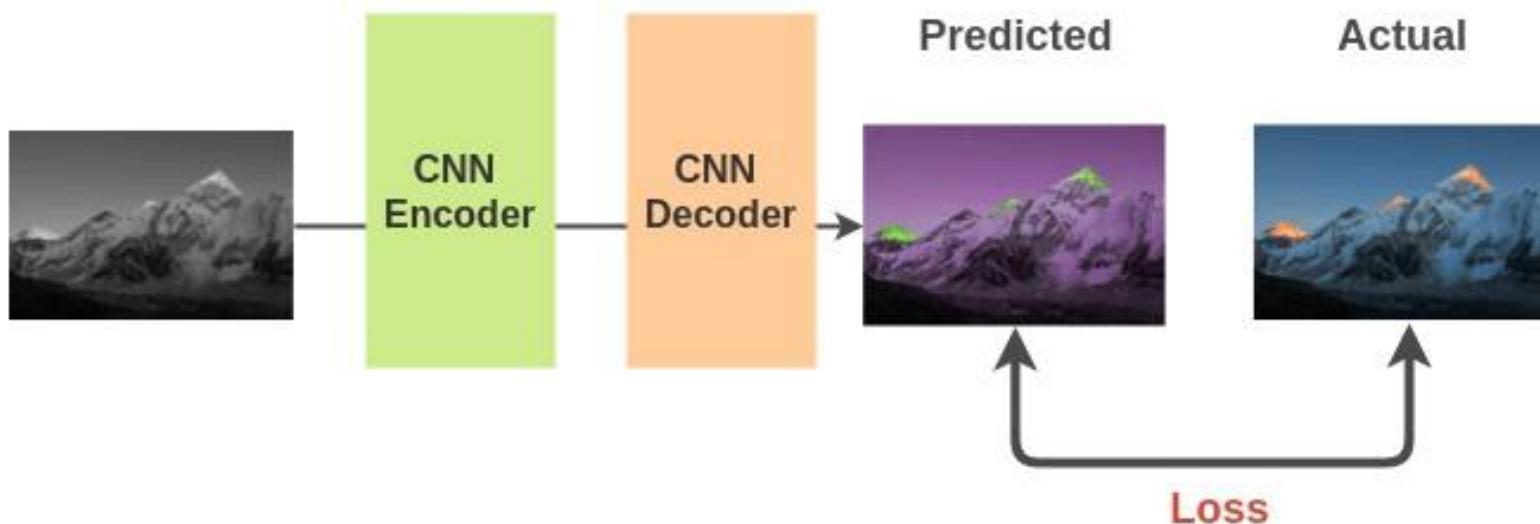
## Input examples



# Image Colorization

An encoder-decoder architecture with convolutional layers is used

- loss between the actual color image and the predicted colorized image is used for training



# Contrastive Predictive Coding

## Contrastive Predictive Coding (CPC)

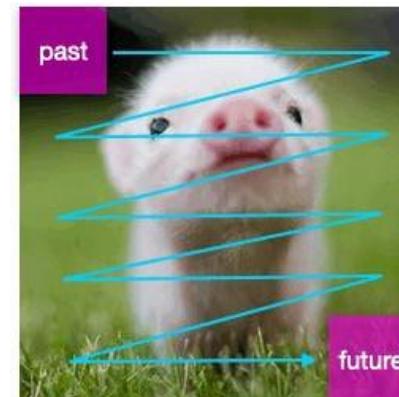
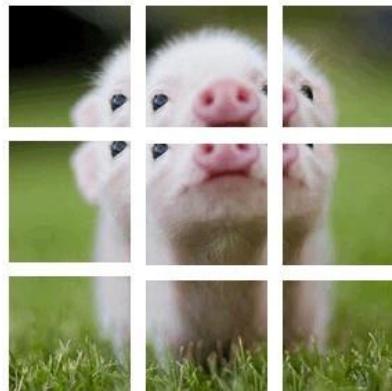
Van der Oord (2018) Representation Learning with Contrastive Predictive Coding

**Training data:** extracted patches from input images

**Pretext task:** predict the order for a sequence of patches using contrastive learning

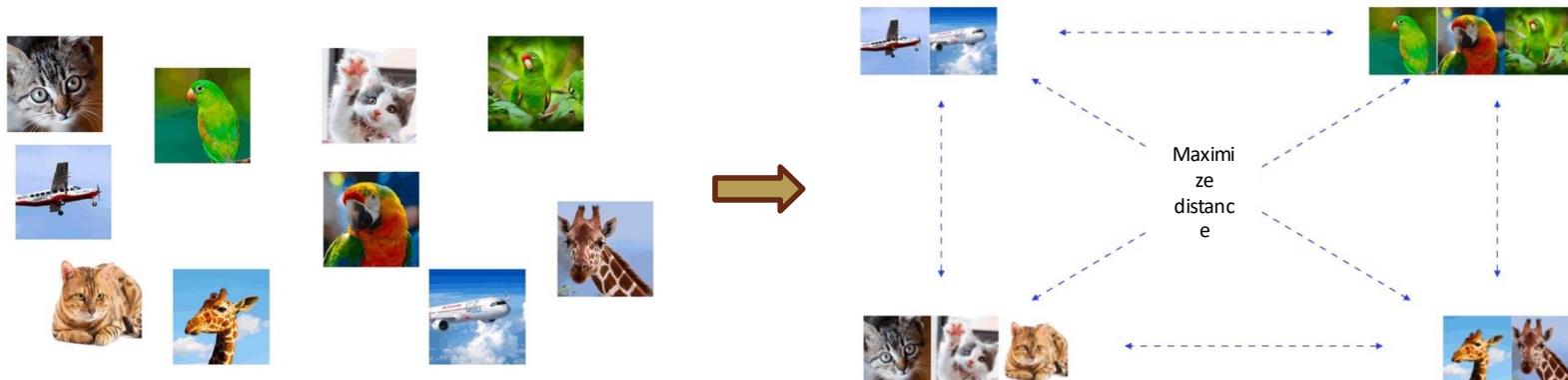
E.g., how to predict the next (future) patches based on encoded information of previous (past) patches in the image

The approach was implemented in different domains: speech audio, images, natural language, and reinforcement learning



# Contrastive Predictive Coding

- Contrastive learning is based on grouping similar examples together
  - E.g., cluster the shown images into groups of similar images
- Noise-Contrastive Estimation (NCE) loss is commonly used in contrastive learning
  - The NCE loss minimizes the distance between similar images (positive examples) and maximizes the distance to dissimilar images (negative examples)
  - Other used terms are InfoNCE loss, or contrastive cross-entropy loss
  - (A forthcoming slide explains the NCE loss in more details)



# Contrastive Predictive Coding

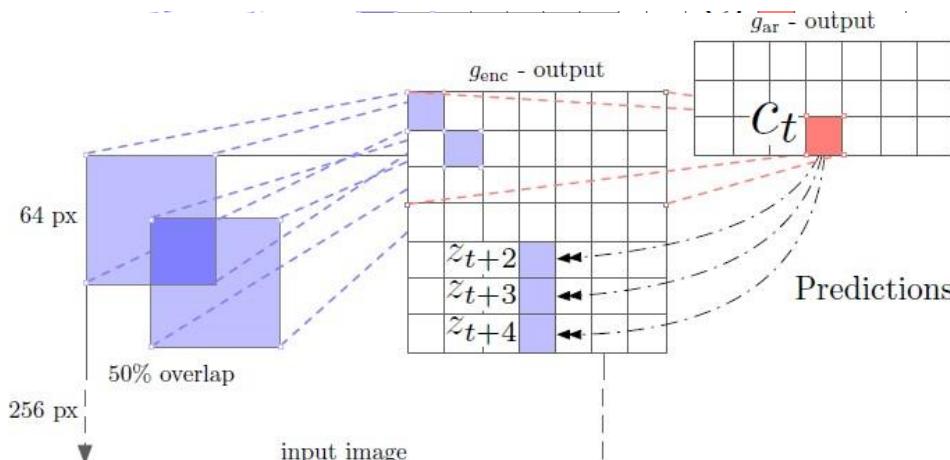
For an input image resized to  $256 \times 256$  pixels, the authors extracted a grid of  $7 \times 7$  patches of size  $64 \times 64$  pixels with 50% overlap between the patches

Therefore, there are 49 overlapping patches in total for each image



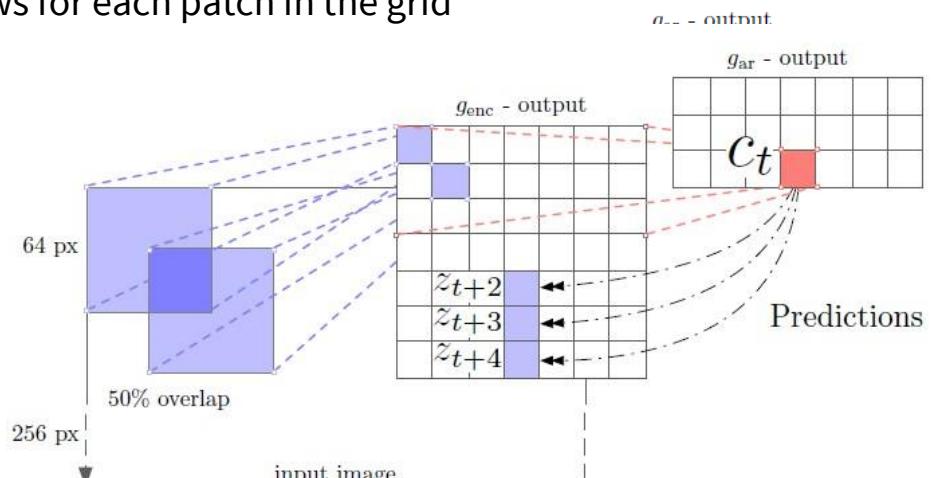
# Contrastive Predictive Coding

- An **encoder  $g_{\text{enc}}$**  is used to project each patch into a low-dimensional latent space
  - The latent representation obtained by encoders is often referred to as **code** (or **context**)
- E.g., the leftmost portion of the image depicts extracting patches of  $64 \times 64$  pixels size with 50% overlap between the patches
  - A **ResNet-101 encoder** is used for projecting the **patch  $\eta$**  into a **code representation &**
  - The middle image shows the outputs of the encoder  $g_{\text{enc}}$  for each patch,  $\&_t = [z_t, z_{t+1}, z_{t+2}, z_{t+3}, z_{t+4}]$
  - For the 49 patches ( $7 \times 7$  grid), the outputs are  $7 \times 7 \times 1,024$  tensors (i.e.,  $\&_t, \&_{t+1}, \dots, \&_{t+6}$ )



# Contrastive Predictive Coding

- CPC considers the patches as an ordered sequence (e.g., like video frames)
  - An **autoregressive model**  $g_{ar}$  is used to predict the **future patches** in the sequence
    - The output of the autoregressive model for the shown **red patch**  $*_t$  (in row 3 and column 4) is the sum of all vectors  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$  for the previous patches in the sequence (e.g., all patches in the above rows and right columns of the red patch)
    - The code representation of the patch  $*_t$  is used to predict the blue patches in the next rows and the same column as the red patch, denoted  $\&_{t+1}, \&_{t+2}$  and  $\&_{t+3}$
    - The authors predicted up to five rows for each patch in the grid



# Contrastive Predictive Coding

The **NCE loss** is used for training the model on each patch in the image

The patch at position  $t$  is considered a positive sample, and all other patches are considered negative samples  
For a set of  $N$  random patches  $X = \{x_1, \dots, x_N\}$ , containing one positive sample  $x_t$  and  $N - 1$  negative samples, the NCE loss is:

$$\mathcal{L}_t = -\mathbb{E} \left[ \log \frac{f_{\theta}(x_t, c_t)}{\sum_{i \neq t} f_{\theta}(x_i, c_i)} \right]$$

$c_t$  is the code representation of the patch at position  $t$ , i.e.,  $c_t = g_{\theta}(z_t)$

$x_{t+k}$  is a predicted patch in the sequence at position  $t + k$

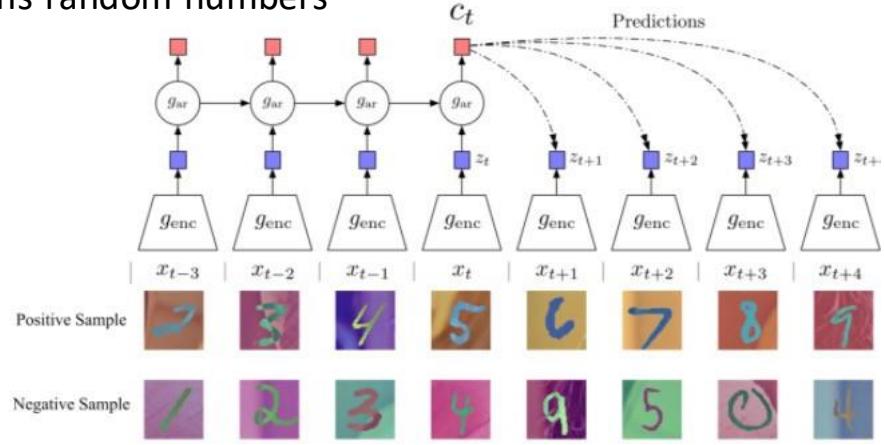
$f_{\theta}(x_{t+k}, c_t)$  is a density function that approximates the probability  $p_{\theta}(x_{t+k}|c_t)$  of estimating the patch  $x_{t+k}$  for a given code representation  $c_t$

The authors used a log-bilinear model  $f_{\theta}(x_{t+k}, c_t) = \exp(z_{t+k}^T W_{\theta} c_t)$ , where  $W_{\theta}$  is the matrix of weights for the prediction step  $k$

The numerator of the NCE loss represents the distance to the positive sample, and the denominator represents the distance to the negative samples

# Contrastive Predictive Coding

- The name of the approach is based on the following:
  - **Contrastive**: representations are learned by contrasting positive and negative examples, which is implemented with the NCE loss
  - **Predictive**: the model needs to predict future patches in the sequences of overlapping patches for a given position in the sequence
  - **Coding**: the model performs the prediction in the latent space, i.e., using code representations from an encoder and an auto-regressive model
- Here is one more example with images from MNIST, where a positive sequence contains sorted numbers, and a negative sequence contains random numbers



# CPC v2

## Contrastive Predictive Coding v2

Henaff (2019) Data-efficient Image Recognition with Contrastive Predictive Coding

This is an extension of the initial CPC work by the same authors

The approach **surpasses** supervised ML methods for image classification on the ImageNet dataset, achieving an increase in Top-5 accuracy by 1.3% (right table)

It also surpasses supervised approaches for object detection on PASCAL VOC by 2%

Method	Architecture	Top-5 accuracy				
		1%	5%	10%	50%	100%
<sup>†</sup> Supervised baseline	ResNet-200	44.1	75.2*	83.9	93.1	95.2#
<i>Methods using label-propagation:</i>						
Pseudolabeling [63]	ResNet-50	51.6	-	82.4	-	-
VAT + Entropy Minimization [63]	ResNet-50	47.0	-	83.4	-	-
Unsup. Data Augmentation [61]	ResNet-50	-	-	88.5	-	-
Rotation + VAT + Ent. Min. [63]	ResNet-50 $\times 4$	-	-	91.2	-	95.0
<i>Methods using representation learning only:</i>						
Instance Discrimination [60]	ResNet-50	39.2	-	77.4	-	-
Rotation [63]	ResNet-152 $\times 2$	57.5	-	86.4	-	-
ResNet on BigBiGAN (fixed)	RevNet-50 $\times 4$	55.2	73.7	78.8	85.5	87.0
ResNet on AMDIM (fixed)	Custom-103	67.4	81.8	85.8	91.0	92.2
ResNet on CPC v2 (fixed)	ResNet-161	77.1	87.5	90.5	95.0	96.2
ResNet on CPC v2 (fine-tuned)	ResNet-161	77.9*	88.6	91.2	95.6#	96.5

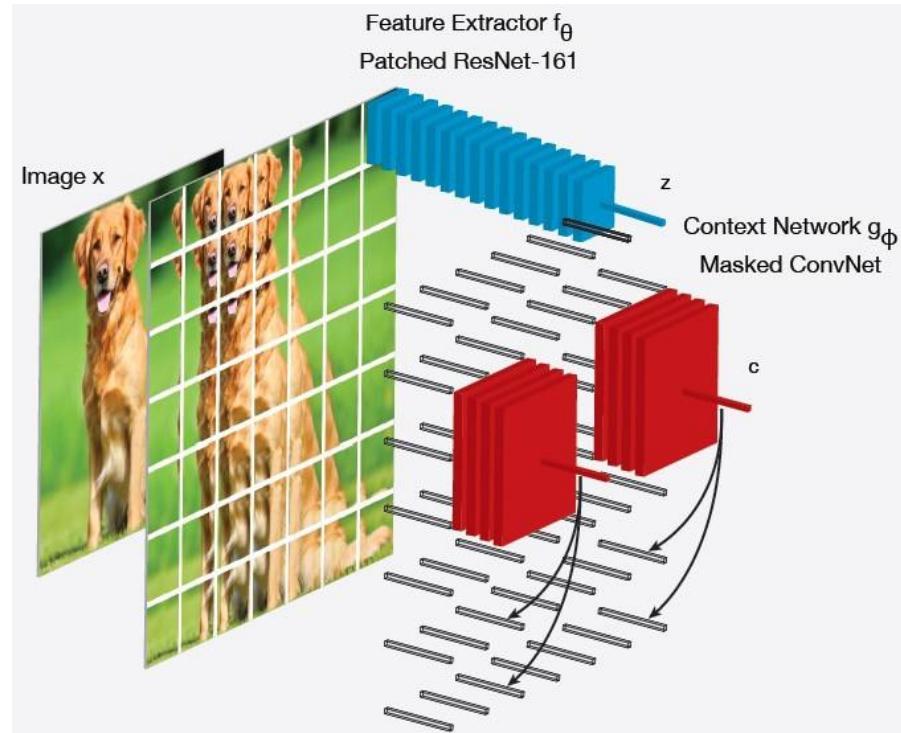
Method	Architecture	mAP
<i>Transfer from labeled data:</i>		
Supervised baseline	ResNet-152	74.7
<i>Transfer from unlabeled data:</i>		
Exemplar [17] by [13]	ResNet-101	60.9
Motion Segmentation [47] by [13]	ResNet-101	61.1
Colorization [64] by [13]	ResNet-101	65.5
Relative Position [14] by [13]	ResNet-101	66.8
Multi-task [13]	ResNet-101	70.5
Instance Discrimination [60]	ResNet-50	65.4
Deep Cluster [7]	VGG-16	65.9
Deeper Cluster [8]	VGG-16	67.8
Local Aggregation [66]	ResNet-50	69.1
Momentum Contrast [25]	ResNet-50	74.9
Faster-RCNN trained on CPC v2	ResNet-161	76.6

# CPC v2

- The differences in CPC v2 versus the initial CPC approach include:
  - Use ResNet-161 instead of ResNet-101 to increase the model capacity
  - Apply layer normalization (i.e., normalize the inputs across the features)
  - For predicting, use patches not only from above the current patch position, but also from below, left, and right to the patch
  - Data augmentation by randomly dropping two or three color channels in each patch, and applying random shearing, rotation, and elastic transformations
- The new architecture of CPC v2 delivered improved results over the initial CPC

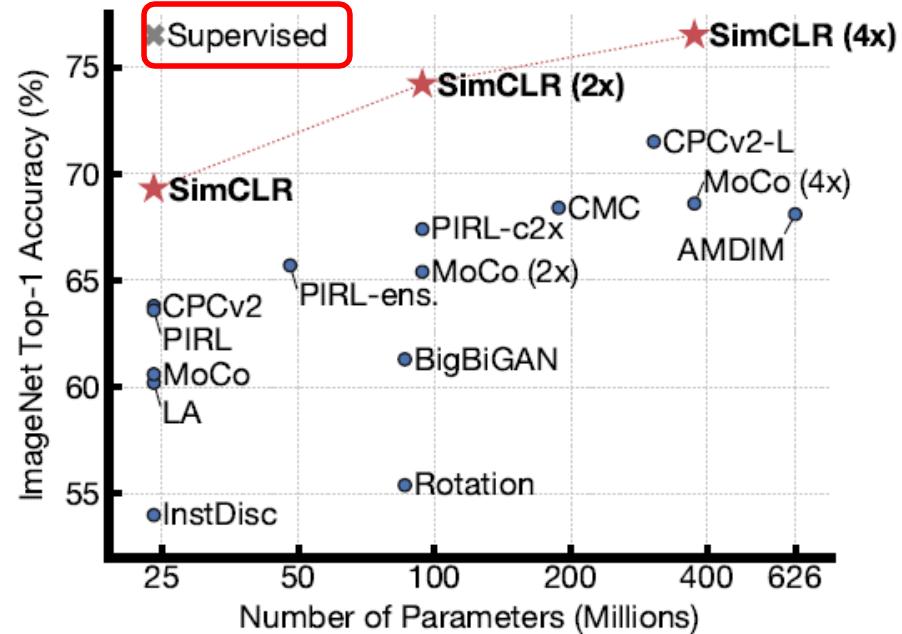
# CPC v2

- Each patch is encoded with a **feature extractor**  $f_A$  (blue) followed by average-pooling (vector  $z$ )
  - The **autoregressive context network**  $\mathcal{g}_B$  (red) aggregates the feature vectors  $z$  into a context vector  $c$ , which is used to predict the future patches



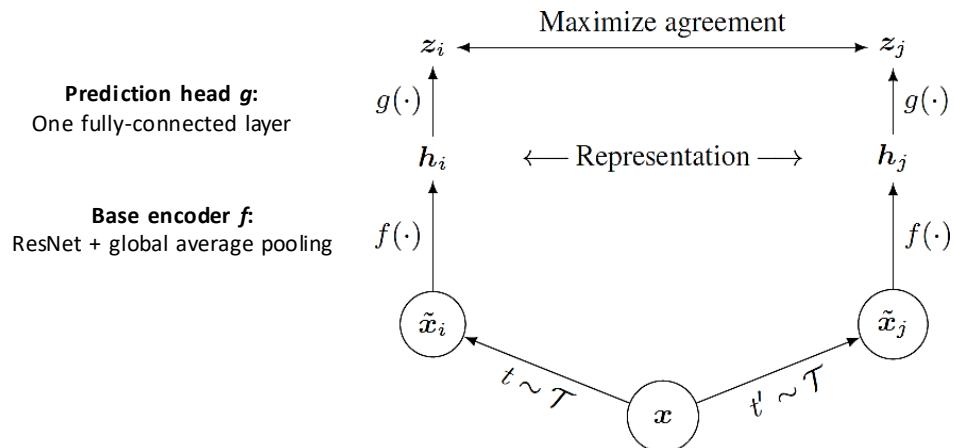
# SimCLR

- **SimCLR**, a **Simple** framework for **Contrastive Learning of Representations**
  - [Chen \(2020\) A Simple Framework for Contrastive Learning of Visual Representations](#)
- SimCLR is an approach for contrastive learning, similar to CPC
- It achieved state-of-the-art in SSL, surpassing the Top-1 accuracy by a supervised ResNet-50 on ImageNet



# SimCLR

- Approach:
  - Randomly sample a **mini-batch** of  $n$  inputs  $x$ , and apply two different **data augmentation** operations " $\cdot$ " and " $\cdot'$ ", resulting in  $2n$  samples  $\tilde{x}_i = x(\cdot)$  and  $\tilde{x}_j = x'(\cdot)$ 
    - Data augmentation includes random crop, resize with random flip, color distortions, and Gaussian blur (data augmentation is crucial for contrastive learning)
  - Apply a **base encoder**  $f(\cdot)$  to  $\tilde{x}_i$  and  $\tilde{x}_j$  to obtain the code representations  $z_i = f(\tilde{x}_i)$  and  $z_j = f(\tilde{x}_j)$
  - Apply another **prediction head encoder**  $g(\cdot)$  (one fully-connected layer) to  $z_i$  and  $z_j$  to obtain the code representations  $\hat{z}_i = g(z_i)$  and  $\hat{z}_j = g(z_j)$



# SimCLR

For one **positive pair** of samples  $\mathbf{z}_i$  and  $\mathbf{z}_j$  and for the remaining  $2(n - 1)$  samples treated as **negative**, a cosine similarity is calculated as

$$\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^T \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}$$

The **contrastive prediction task** aims for a given sample  $\tilde{\mathbf{x}}_i$  to identify a positive pairing sample  $\tilde{\mathbf{x}}_j$ .  
The **NCE loss** for the instances  $\tilde{\mathbf{x}}_i$  and  $\tilde{\mathbf{x}}_j$  is calculated as:

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2n} \mathbf{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

$\mathbf{1}_{[k \neq i]}$  has a value of 1 if  $k \neq i$  and 0 otherwise,  $\tau$  is a temperature hyperparameter  
The overall loss  $\sum_{i,j} \mathcal{L}_{i,j}$  is calculated across all positive pairs  $\tilde{\mathbf{x}}_i$  and  $\tilde{\mathbf{x}}_j$  in a mini-batch  
For downstream tasks, the head  $g(\cdot)$  is discarded and only the representation  $\mathbf{h}_i$  is used

# SimCLR

## Data augmentation



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate  $\{90^\circ, 180^\circ, 270^\circ\}$



(g) Cutout



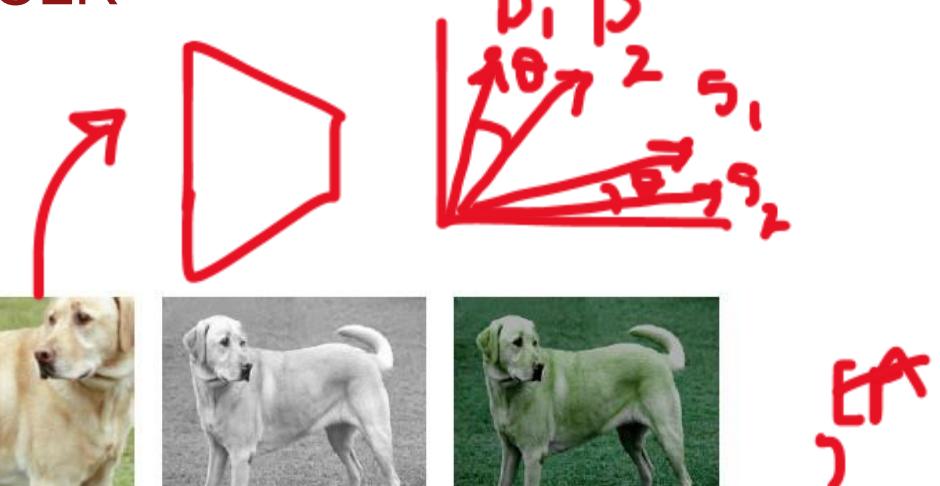
(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

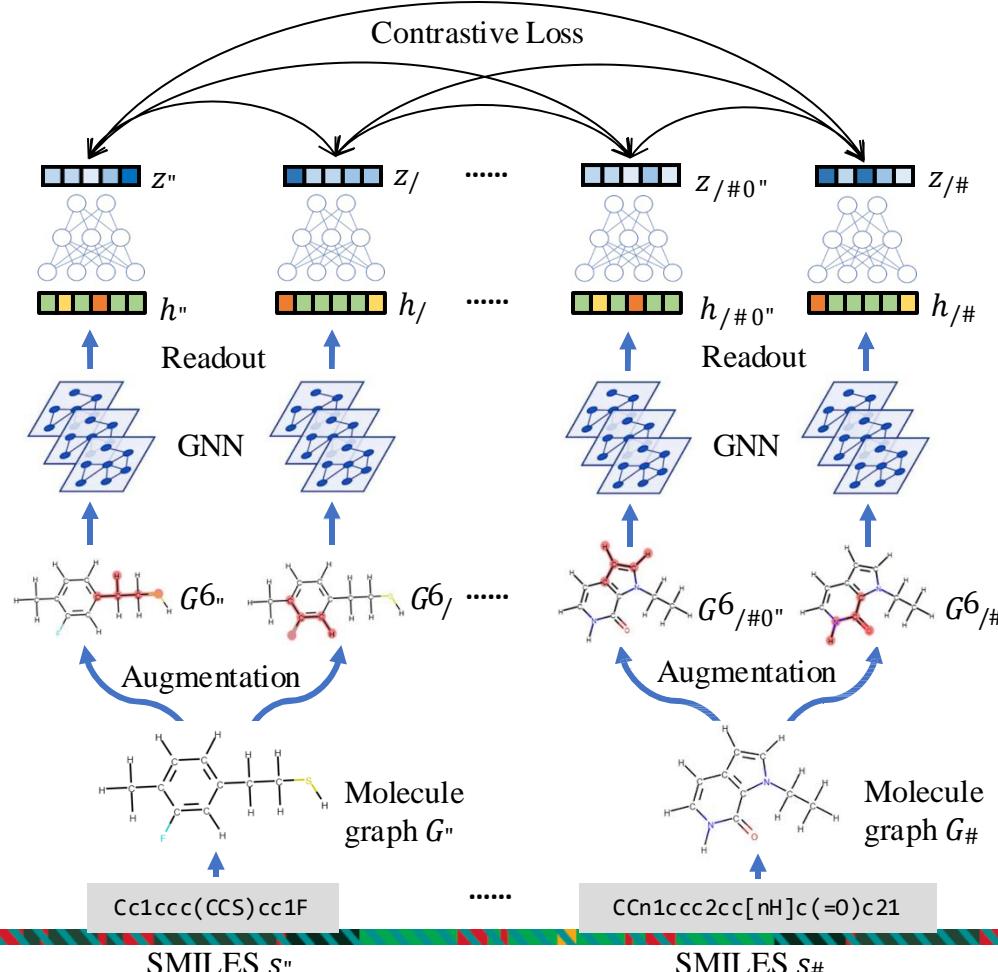


# SimCLR

- Experimental results on 10 image datasets
  - SimCLR outperformed supervised models on most datasets

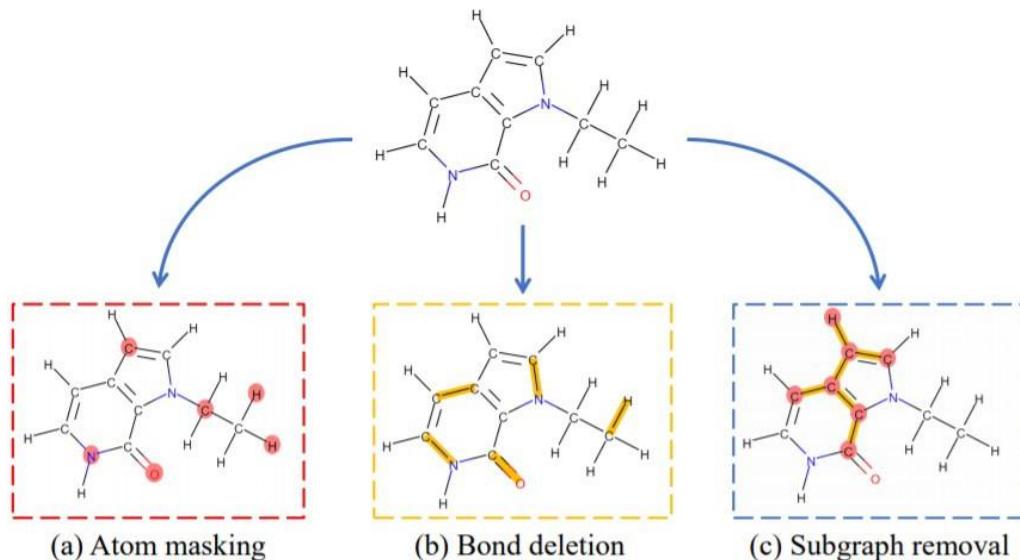
	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
<i>Linear evaluation:</i>												
SimCLR (ours)	<b>76.9</b>	<b>95.3</b>	80.2	48.4	<b>65.9</b>	60.0	61.2	<b>84.2</b>	<b>78.9</b>	89.2	93.9	<b>95.0</b>
Supervised	75.2	<b>95.7</b>	<b>81.2</b>	<b>56.4</b>	64.9	<b>68.8</b>	<b>63.8</b>	83.8	<b>78.7</b>	<b>92.3</b>	<b>94.1</b>	94.2
<i>Fine-tuned:</i>												
SimCLR (ours)	<b>89.4</b>	<b>98.6</b>	<b>89.0</b>	<b>78.2</b>	<b>68.1</b>	<b>92.1</b>	<b>87.0</b>	<b>86.6</b>	<b>77.8</b>	92.1	<b>94.1</b>	97.6
Supervised	88.7	98.3	<b>88.7</b>	<b>77.8</b>	67.0	91.4	<b>88.0</b>	86.5	<b>78.8</b>	<b>93.2</b>	<b>94.2</b>	<b>98.0</b>
Random init	88.3	96.0	81.9	<b>77.0</b>	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

# MoICLR

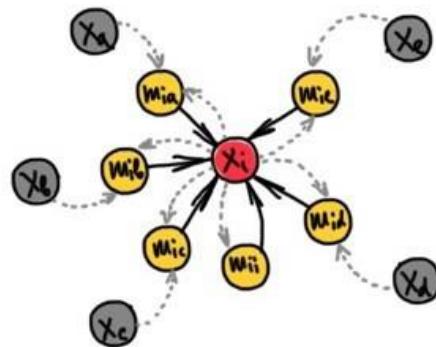


# Molecular Graph Augmentations & Encoding

## Molecular Graph Augmentation



## Graph Neural Network



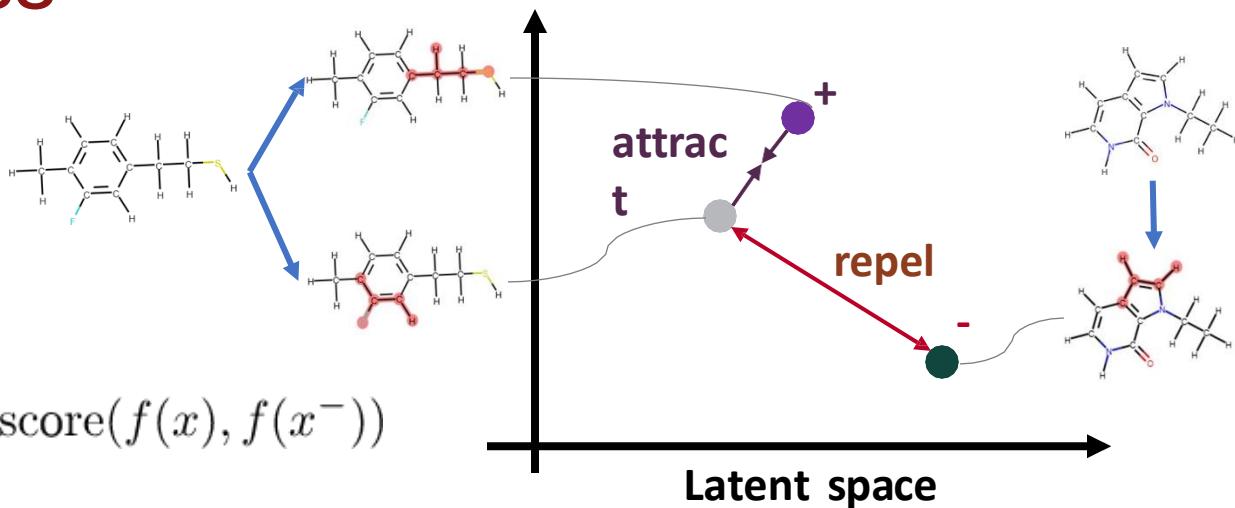
$$\mathbf{m}_i^{l+1} = \sum_{j \in \mathcal{N}_i} f_m(\mathbf{h}_i^l, \mathbf{h}_j^l, e_{ij})$$

$$\mathbf{h}_i^{l+1} = f_u(\mathbf{h}_i^l, \mathbf{m}_i^{l+1})$$

J. Gilmer et al., *ICML 2017*  
M. Bronstein, *The Gradient 2022*

# Contrastive Loss

$$\text{score}(f(x), f(x^+)) \gg \text{score}(f(x), f(x^-))$$

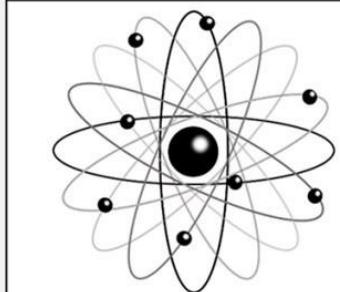


Normalized temperature-scaled cross entropy loss (NT-Xent):

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\cos(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(\cos(z_i, z_k)/\tau)}$$

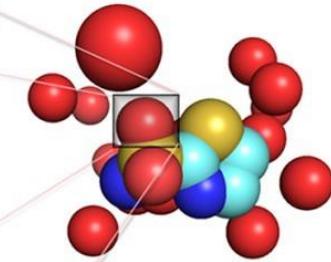
Pre-train on ~10M molecules from PubChem.

# MoleculeNet Benchmarks



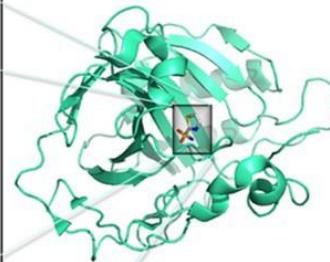
## Quantum Mechanics

- QM7
- QM8
- QM7b
- QM9



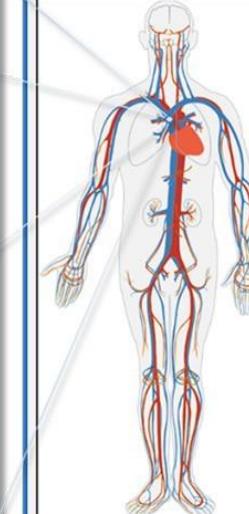
## Physical Chemistry

- ESOL
- Lipophilicity
- FreeSolv



## Biophysics

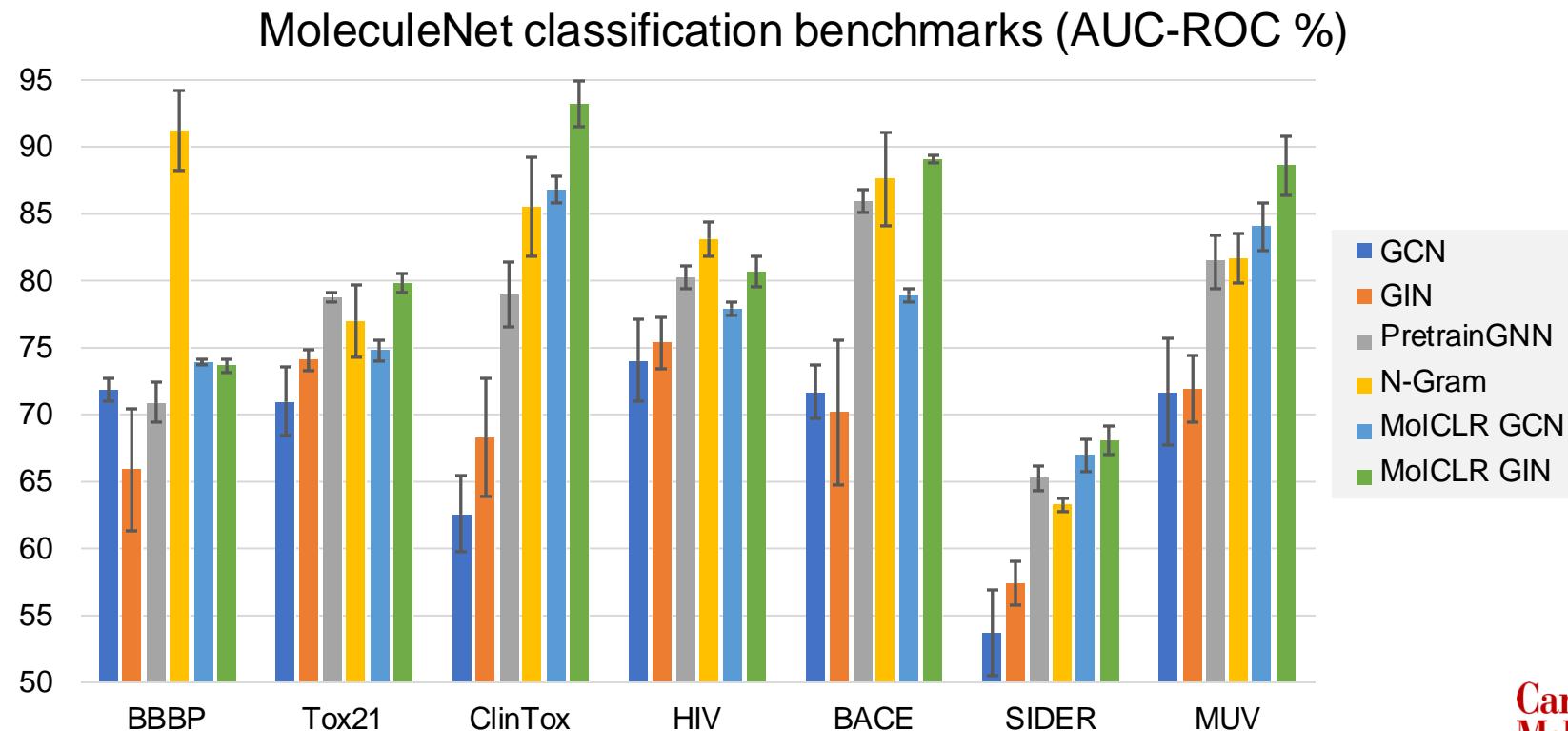
- HIV
- PDBbind
- BACE
- PCBA
- MUV



## Physiology

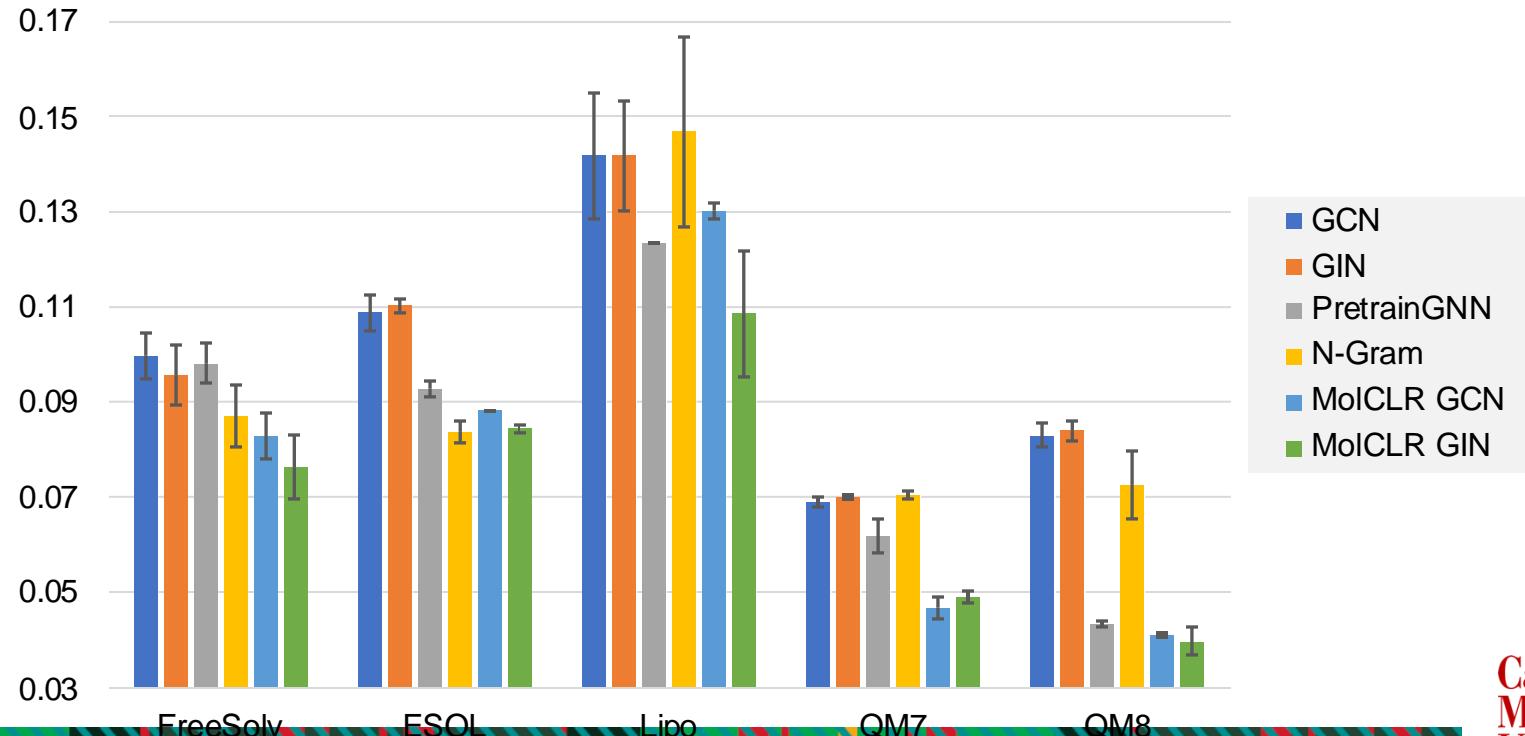
- BBBP
- Tox21
- ToxCast
- SIDER
- ClinTox

# Property Prediction: Classification

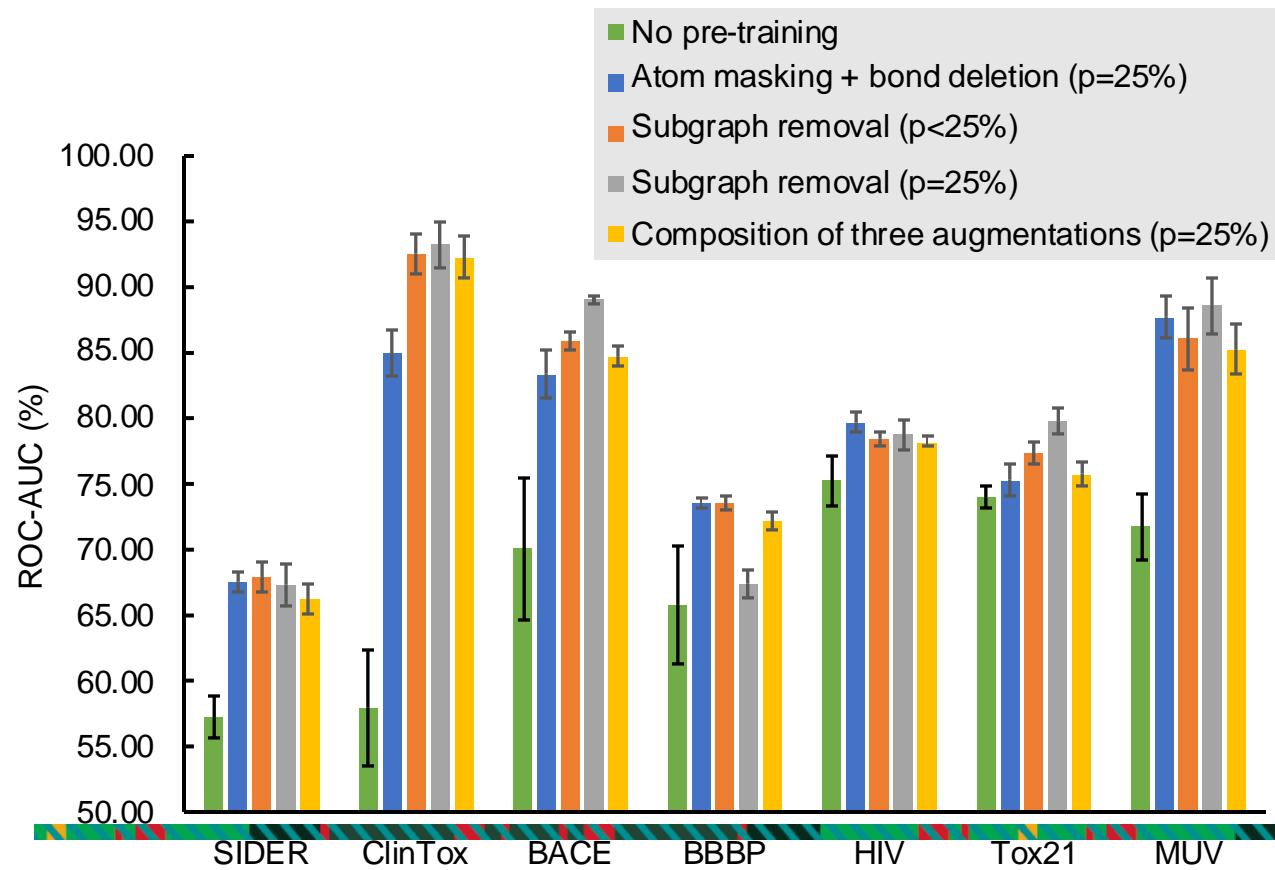


# Property Prediction: Regression

MoleculeNet regression benchmarks (scaled error)



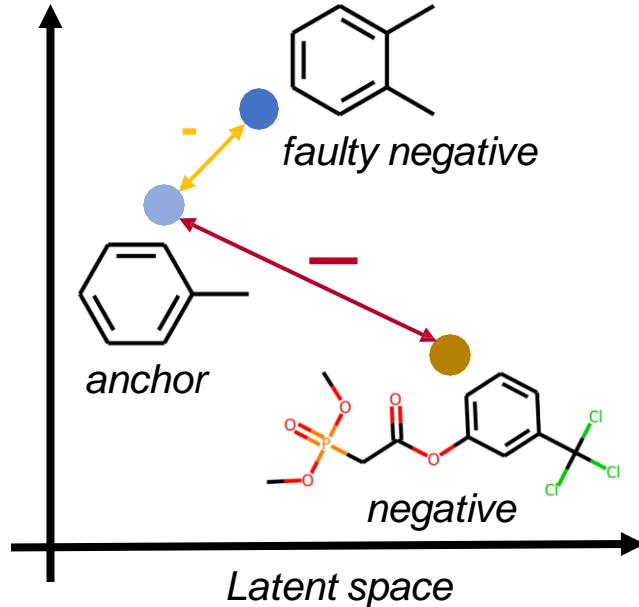
# Ablation Study: Augmentation



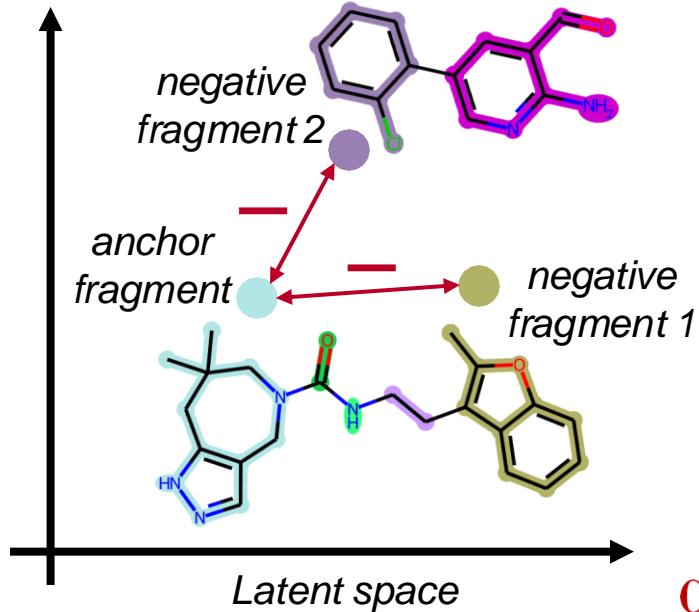
Test ROC-AUC (%) of pre-trained MolCLR model with different compositions of molecular graph augmentation strategies.

# iMoCLR: improved MolCLR

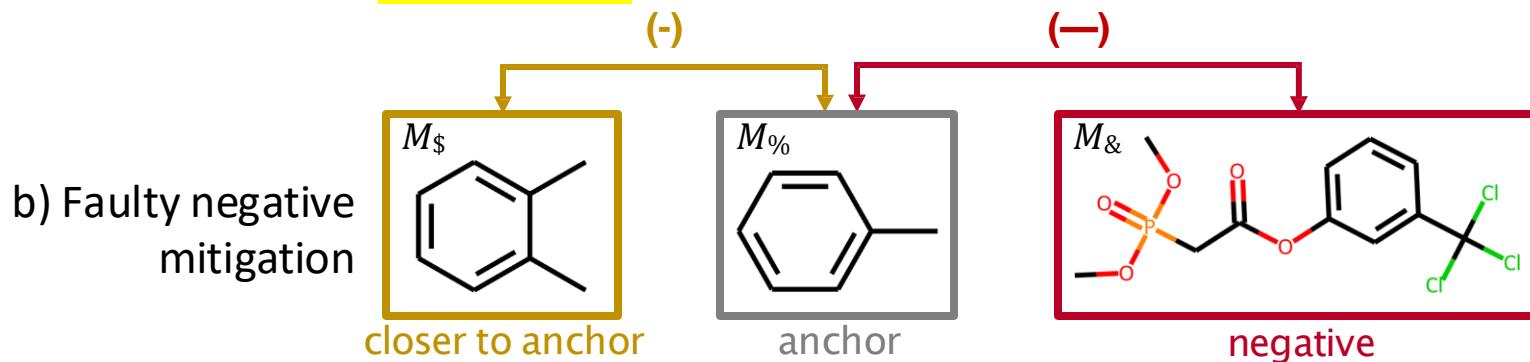
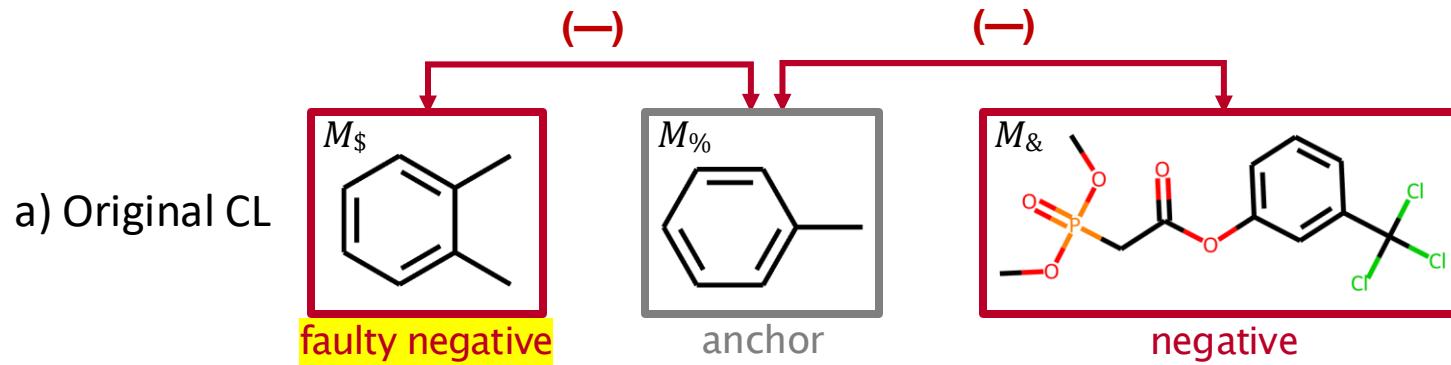
Faulty negative mitigation



Fragment contrast



# Faulty Negative Mitigation



Weighted NT-Xent

$$\mathcal{L}_{i,j}^w = - \log \frac{\exp(\cos(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(w_{ik} \cos(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

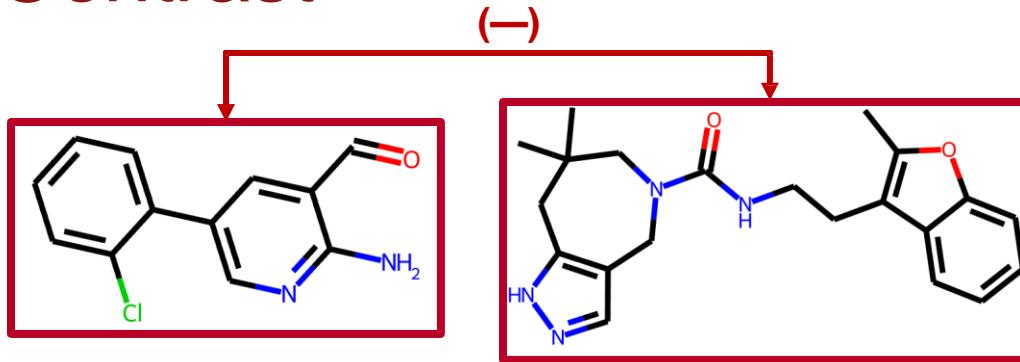
$$w_{ik} = 1 - \lambda_1 \text{FPSim}(M_i, M_k) \text{gie} \text{ Mellon}$$

(ECFP + Tanimoto Similarity)

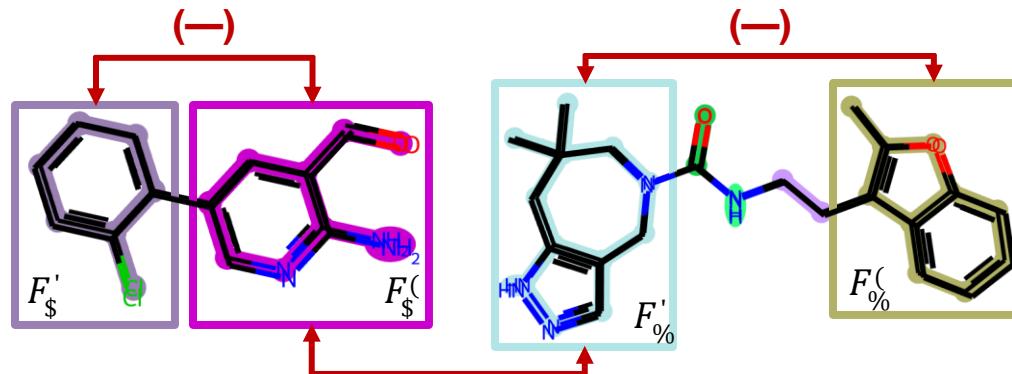
University

# Fragment Contrast

a) Original CL



b) Fragment contrast



Fragment NT-Xent

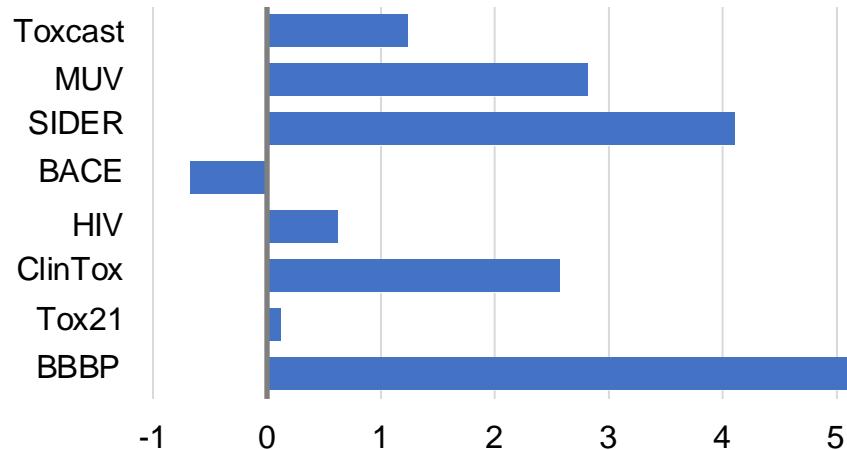
$$\mathcal{L}_{i,j}^{\text{frag}} = - \log \frac{\exp(\cos(\mathbf{z}_i^{\text{frag}}, \mathbf{z}_j^{\text{frag}})/\tau)}{\sum_{k=1}^{2M} \mathbb{1}_{k \neq i} \exp(\cos(\mathbf{z}_i^{\text{frag}}, \mathbf{z}_k^{\text{frag}})/\tau)}$$

(BRICS decomposition)

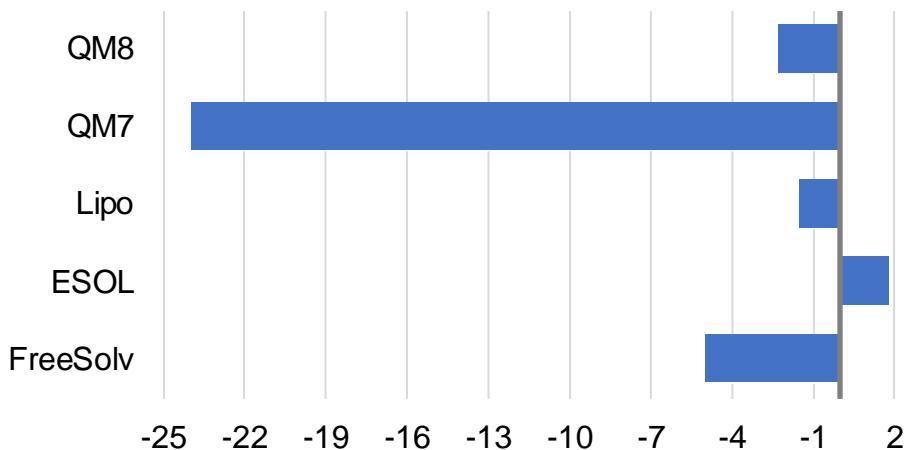
J. Degen et al., *ChemMedChem* 2008

# iMoICLR vs. MoICLR

iMoICLR vs. MoICLR on Classification (%)



iMoICLR vs. MoICLR on Regression (%)



	BBBP	Tox21	ClinTox	HIV	BACE	SIDER	MUV	Toxcast	FreeSolv	ESOL	Lipo	QM7	QM8
p-value	<u>.00</u>	0.23	<u>.04</u>	0.76	0.81	<u>.05</u>	<u>.01</u>	<u>.00</u>	<u>.01</u>	0.25	<u>.03</u>	<u>.00</u>	<u>.00</u>

# Other Contrastive SSL Approaches

Other recent self-supervised approaches based on **contrastive learning** include:

***Augmented Multiscale Deep InfoMax*** or ***AMDIM***

Bachman (2019) Learning Representations by Maximizing Mutual Information Across Views

***Momentum Contrast*** or ***MoCo***

He (2019) Momentum Contrast for Unsupervised Visual Representation Learning

***Bootstrap Your Own Latent*** or ***BYOL***

Grill (2020) Bootstrap your own latent: A new approach to self-supervised Learning

***Swapping Assignments between multiple Views of the same image*** or ***SwAV***

Caron (2020) Unsupervised Learning of Visual Features by Contrasting Cluster Assignments

***Yet Another DIM*** or ***YADIM***

Falcon (2020) A Framework for Contrastive Self-Supervised Learning and Designing a New Approach

# Contrastive SSL Approaches

- The contrastive SSL approaches are computationally expensive
  - Estimated costs for some of these approaches as reported in this [blog post](#)
  - The costs are based on 23dn.24xlarge AWS instance at \$31.212 per hour

AMDIM

Dataset	GPUs	Hours	Equivalent AWS cost
CIFAR-10	2 V100 (32GB)	24	\$188
STL-10	8 V100 (32GB)	72	\$2,246
ImageNet	96 V100 (32 GB)	72	\$26,956

CPC

Dataset	GPUs	Hours	Equivalent AWS cost
CIFAR-10	2 V100 (32GB)	24	\$188
STL-10	4 V100 (32 GB)	144	\$2,246
ImageNet	32 V100 (32 GB)	504+	\$62,899

YADIM

Dataset	GPUs	Hours	Equivalent AWS cost
CIFAR-10	2 V100 (32GB)	24	\$188
STL-10	8 V100 (32GB)	72	\$2,246
ImageNet	256 V100 (32 GB)	336	\$335,462

# Video-based Approaches

- ***Video-based approaches***
- SSL methods are often used for learning useful feature representations in videos
- Videos provide richer visual information than images
  - The consistency of spatial and temporal information across video frames lend them suitable for learning from raw videos without labels
  - Models based on recurrent NNs in combination with ConvNets are naturally more often encountered in SSL for videos, due to the temporal character
- The following video-based approaches are briefly reviewed next
  - Frame ordering, tracking moving objects, video colorization
- A detailed overview of video-based SSL approaches can be found in the review paper by Jing and Tian (see References)

# Frame Ordering

**Frame ordering** also known as **shuffle and learn**

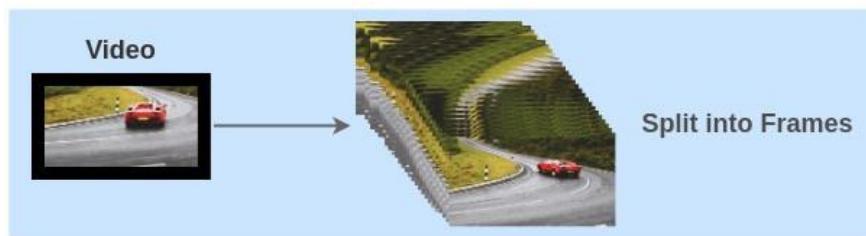
Misra (2016) Shuffle and Learn: Unsupervised Learning using Temporal Order Verification

**Training data:** videos of objects in motion with shuffled order of the frames

**Pretext task:** predict if the frames are in the correct **temporal order**

The frames are shuffled, and pairs of videos with correct and shuffled order are used for training the model

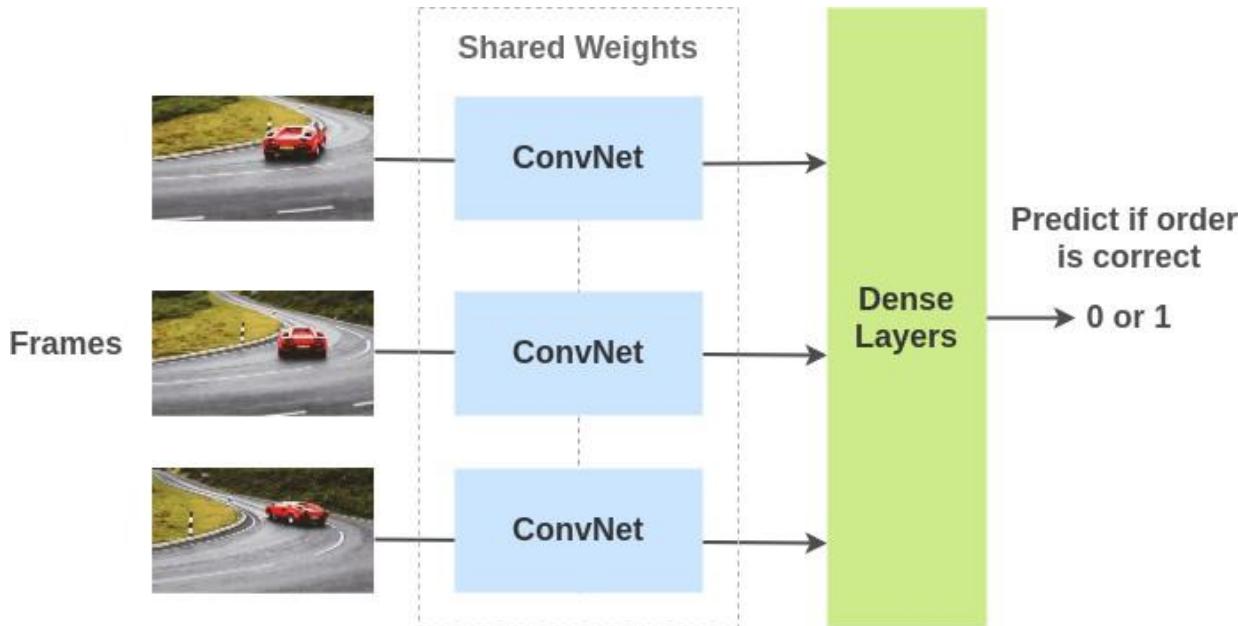
The model needs to learn the object classes, as well as it needs to learn the temporal ordering of the objects' positions across the frames



# Frame Ordering

The model employs ConvNets with shared weights

The output is a binary prediction on whether the frames are in the correct order or not



# Frame Ordering

Example

Original video



Temporally Correct order



Temporally Incorrect order



# NLP

- Self-supervised learning has driven the recent progress in the *Natural Language Processing* (NLP) field
  - Models like ELMO, BERT, RoBERTa, ALBERT, Turing NLG, GPT-3 have demonstrated immense potential for automated NLP
- Employing various pretext tasks for leaning from raw text produced rich feature representations, useful for different downstream tasks
- **Pretext tasks** in NLP:
  - Predict the center word given a window of surrounding words
    - The word highlighted with green color needs to be predicted



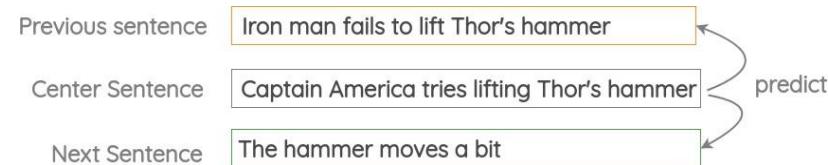
- Predict the surrounding words given the center word

A **quick** brown **fox** jumps over the lazy dog

# NLP

## Pretext tasks in NLP:

From three consecutive sentences, predict the previous and the next sentence, given the center sentence



Predict the previous or the next word, given surrounding words



Predict randomly masked words in sentences



# NLP

## Pretext tasks in NLP:

Predict if the ordering of two sentences is correct

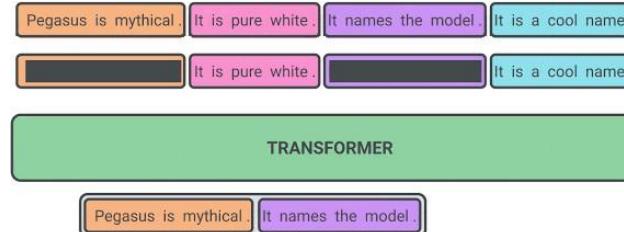
Sentence 1	Sentence 2	Next Sentence
I am going outside	I will be back in the evening	yes
I am going outside	You know nothing John Snow	no

Predict the order of words in a randomly shuffled sentence

Finally I did Z. Then I did Y. I did X. Shuffle

I did X. Then I did Y. Finally I did Z. Recover

Predict masked sentences in a document



# GPT-3

- **GPT-3** stands for *Generative Pre-trained Transformer*
  - It was created by [OpenAI](#), and introduced in May 2020
- **Transformers** are currently the most common model architecture for NLP tasks
  - They employ attention blocks for discovering correlation in text
- GPT-3 generates text based on initial input prompt from the end-user
  - It is trained using next word prediction on huge amount of raw text from the internet
  - The quality of text generated is often undistinguishable from human-written text
  - GPT-3 can also be used for other tasks, such as answering questions, summarizing text, automated code generation, and many others
- It is probably the largest NN model at the present, having 175 billion parameters
  - The cost for training GPT-3 reportedly is \$ 12 million
  - For comparison, Microsoft's Turing NLG (Natural Language Generation) model has 17 billion parameters
- Currently, OpenAI allows access to GPT-3 only to selected applicants
- Controversies: GPT-3 just memorizes text from other sources, risk of abuse by certain actors

# Additional References

1. Lilian Weng – Self-Supervised Representation Learning
2. Pieter Abbeel, UC Berkley, CS294-158 Deep Unsupervised Learning, Lecture 7 – Self-Supervised Learning
3. Amit Chaudhary – The Illustrated Self-Supervised Learning,
4. Jing and Tian (2019) Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey
5. William Falcon – A Framework for Contrastive Self-Supervised Learning and Designing a New Approach,
6. Andrew Zisserman – Self-Supervised Learning, slides from: Carl Doersch, Ishan Misra, Andrew Owens, Carl Vondrick, Richard Zhang
7. Amit Chaudhary –Self-Supervised Representation Learning in NLP,