University of Idaho

CS 502
Directed Studies: Adversarial Machine Learning

Dr. Alex Vakanski

Lecture 16

Self-Supervised Learning

Lecture Outline

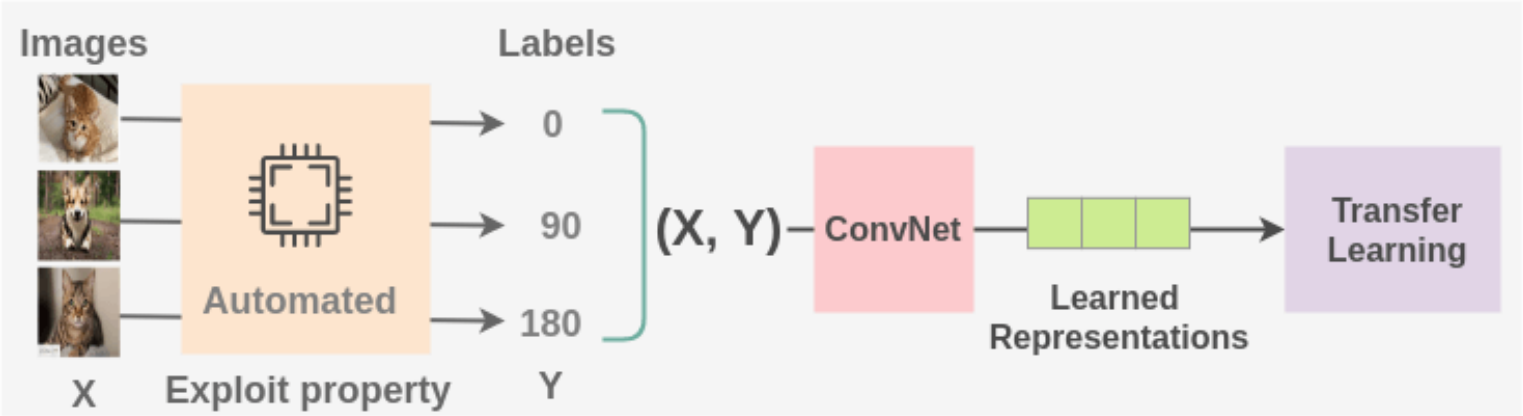
- Self-supervised learning
  - Motivation
  - Self-supervised learning versus other machine learning techniques
- Image-based approaches
  - Geometric transformation recognition (image rotation)
  - Patches (relative patch position, image jigsaw puzzle)
  - Generative modeling (context encoders, image 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

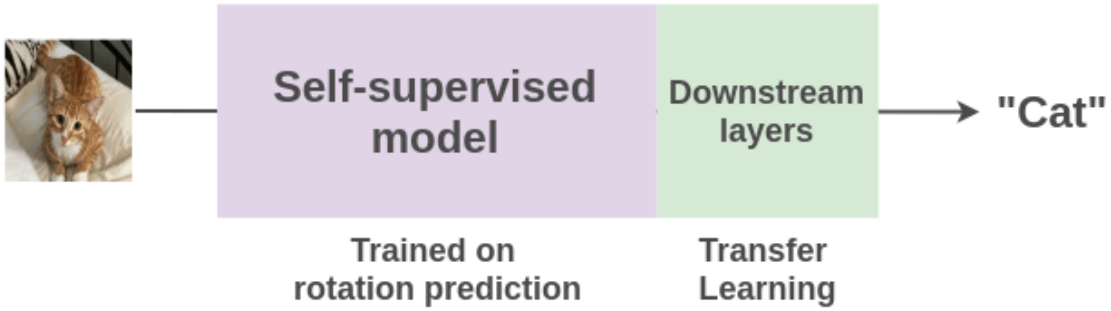
- *Supervised learning* – learning with labeled data
  - Approach: collect a large dataset, manually label the data, train a model, deploy
  - It is the dominant form of ML at present
  - Learned feature representations on large datasets are often transferred via pre-trained models to smaller domain-specific 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)



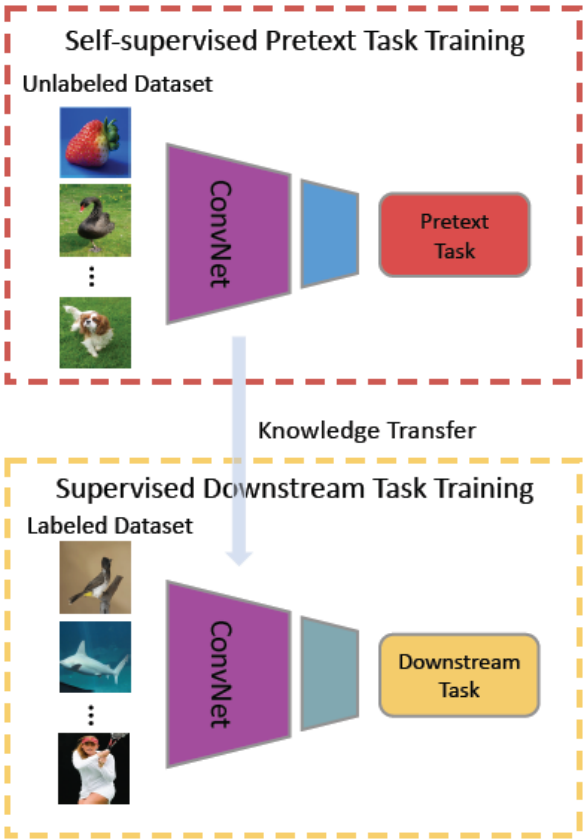
  - *Downstream task*: use transfer learning and fine-tune the learned model from the pretext task for classification of cats vs dogs with very few labeled examples



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Self-Supervised Learning

- One more depiction of the general pipeline for self-supervised learning is shown in the figure
  - For the downstream task, re-use the trained ConvNet base model, and fine-tune the top layers on a small labeled dataset



Picture from: Jing and Tian (2019) Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey

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

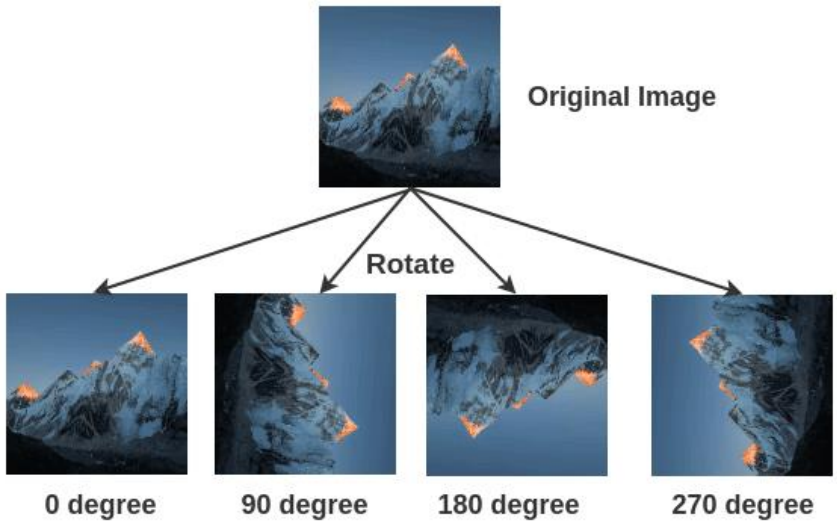
- Self-supervised learning versus unsupervised learning
  - Self-supervised learning (SSL)
    - Aims to extract useful feature representations from raw unlabeled data through ***pretext tasks***
    - Apply the feature representation to improve the performance of ***downstream tasks***
  - Unsupervised 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 of shifting are used for feature learning in raw unlabeled data

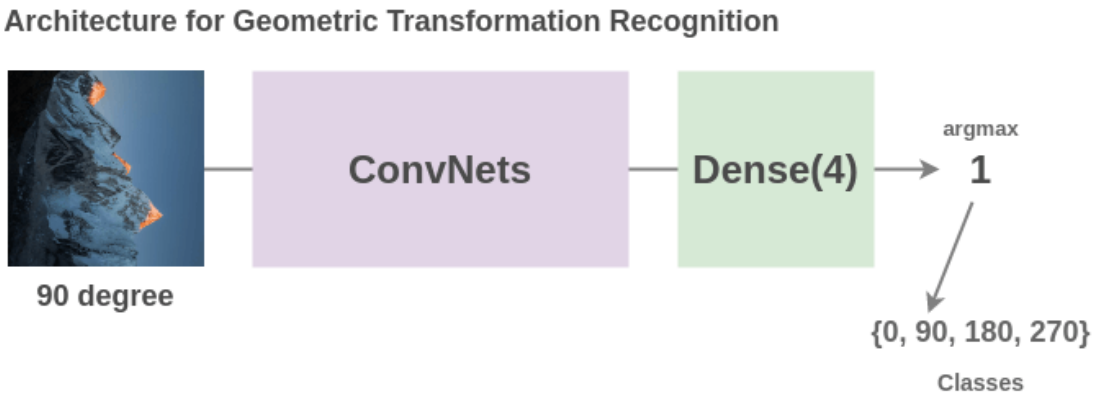
Image-Based Approaches

- *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° at random
  - This corresponds to four rotated images at 0°, 90°, 180°, and 270°
- Pretext task: train a model to predict the rotation degree that was applied
  - Therefore, it is a 4-class classification problem





Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

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

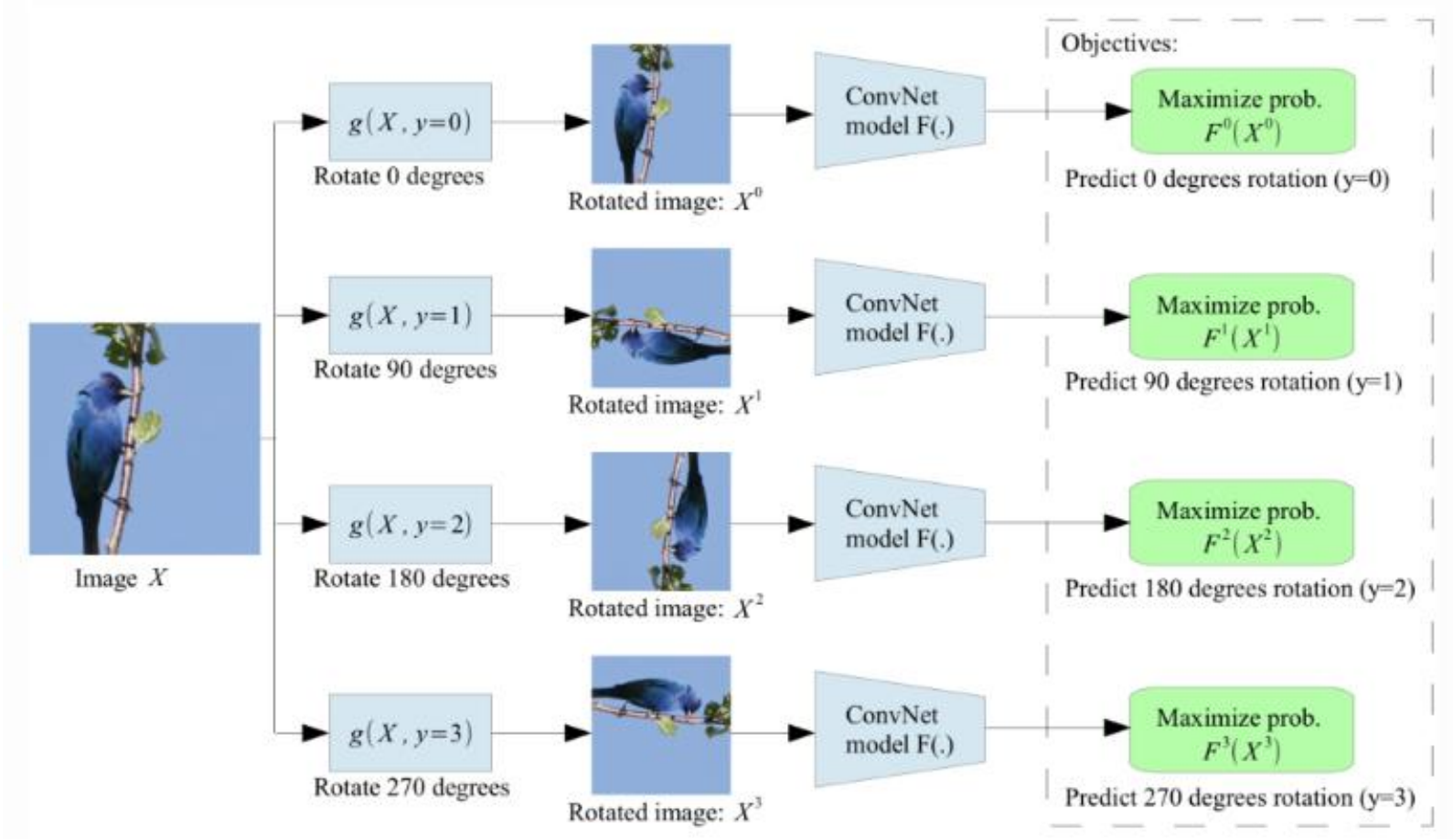


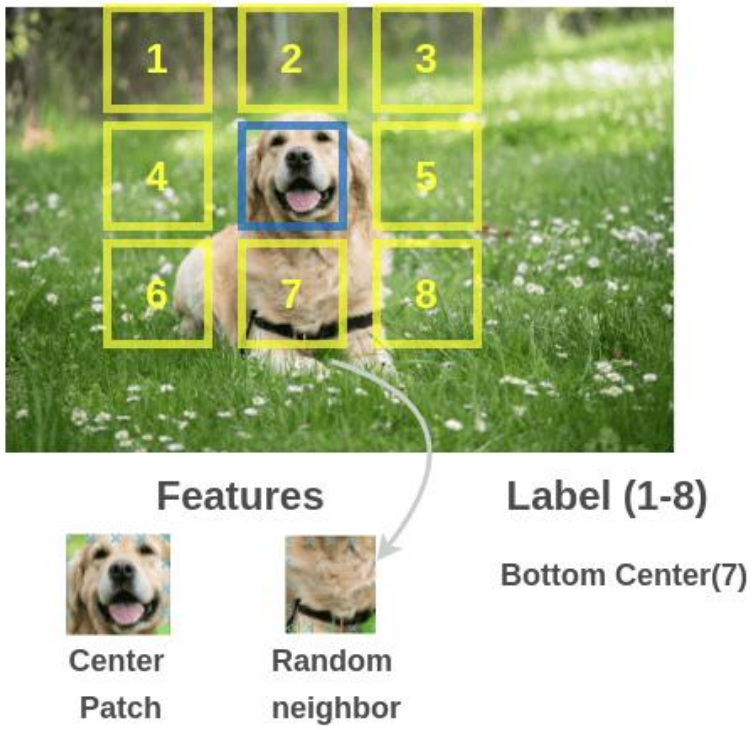
Image Rotation

- 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	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
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
Proposed self-supervised feature learning: (Ours) RotNet	**70.87**	**72.97**	**54.4**	**39.1**

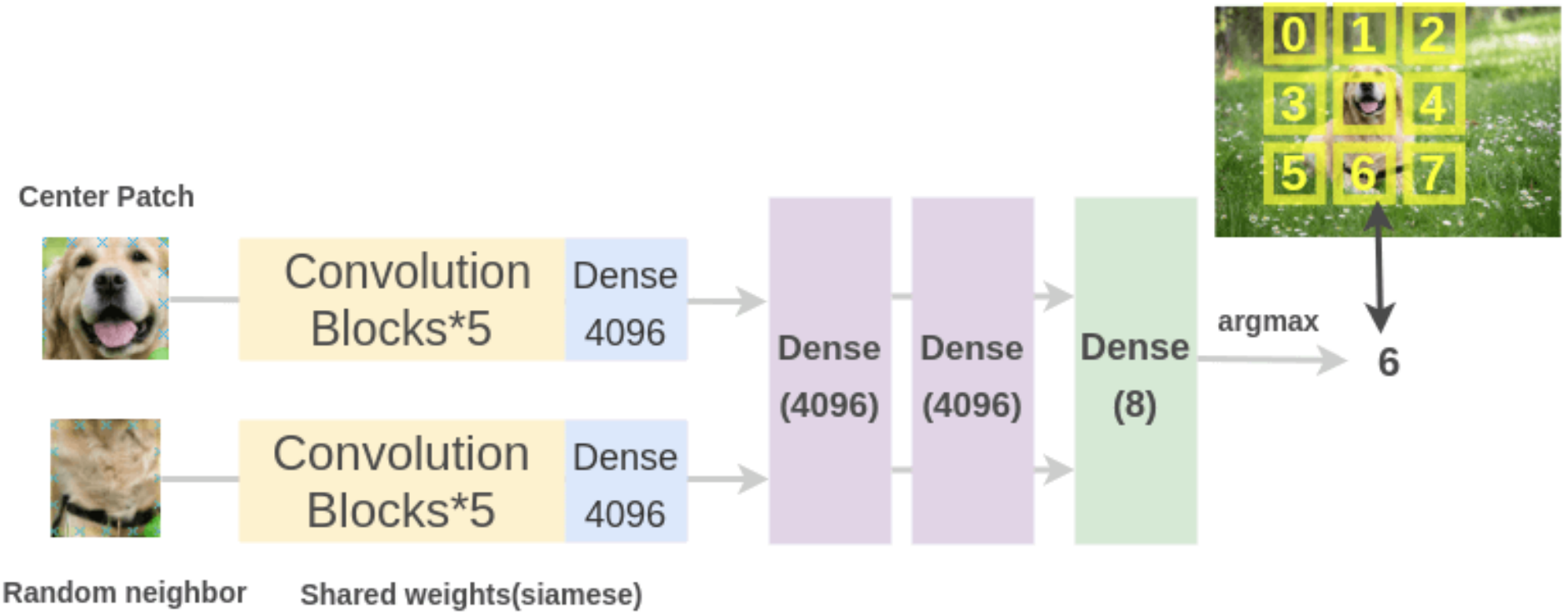
Relative Patch Position

- *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

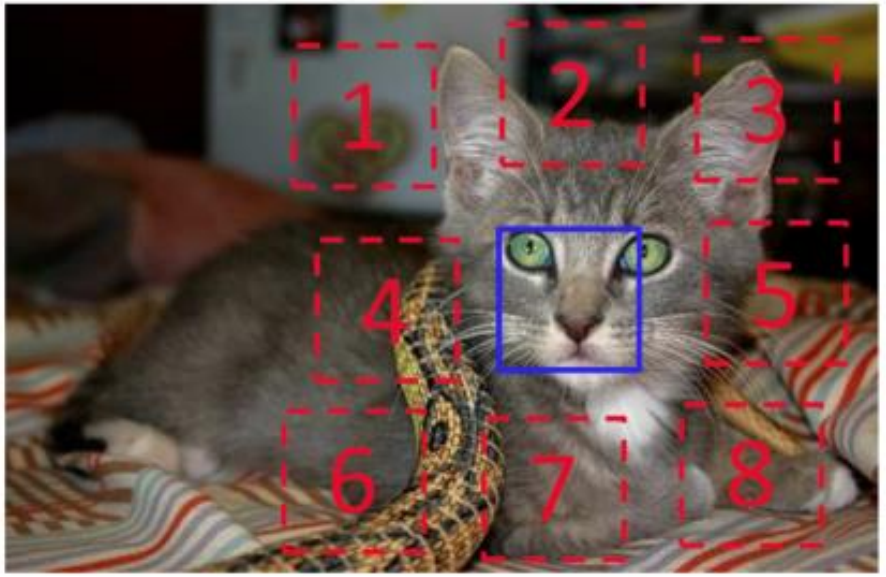
- The patches are inputted into two ConvNets with shared weights
  - The learned features by the ConvNets are concatenated
  - Classification is performed over 8 classes (denoting the 8 possible neighbor positions)
- The model needs to understand the spatial context of images, in order to predict the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



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

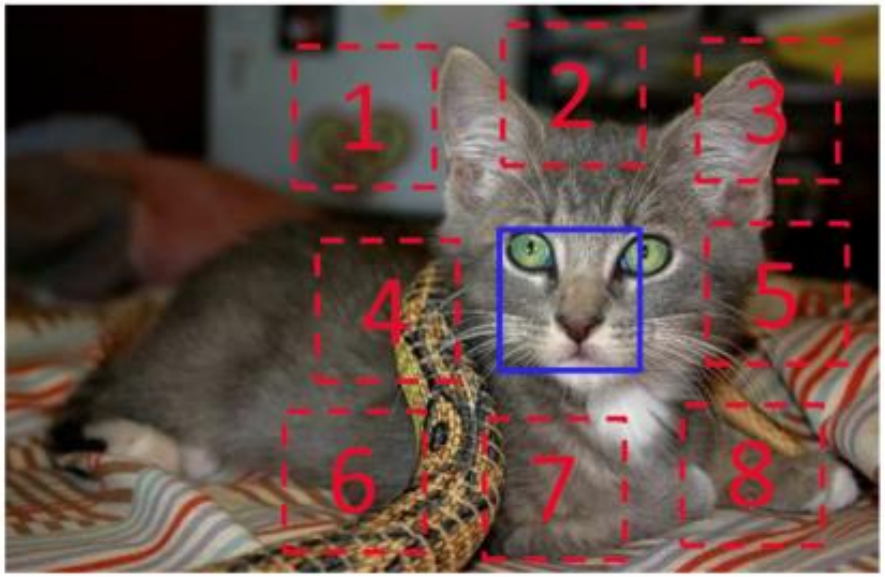
Relative Patch Position

- 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

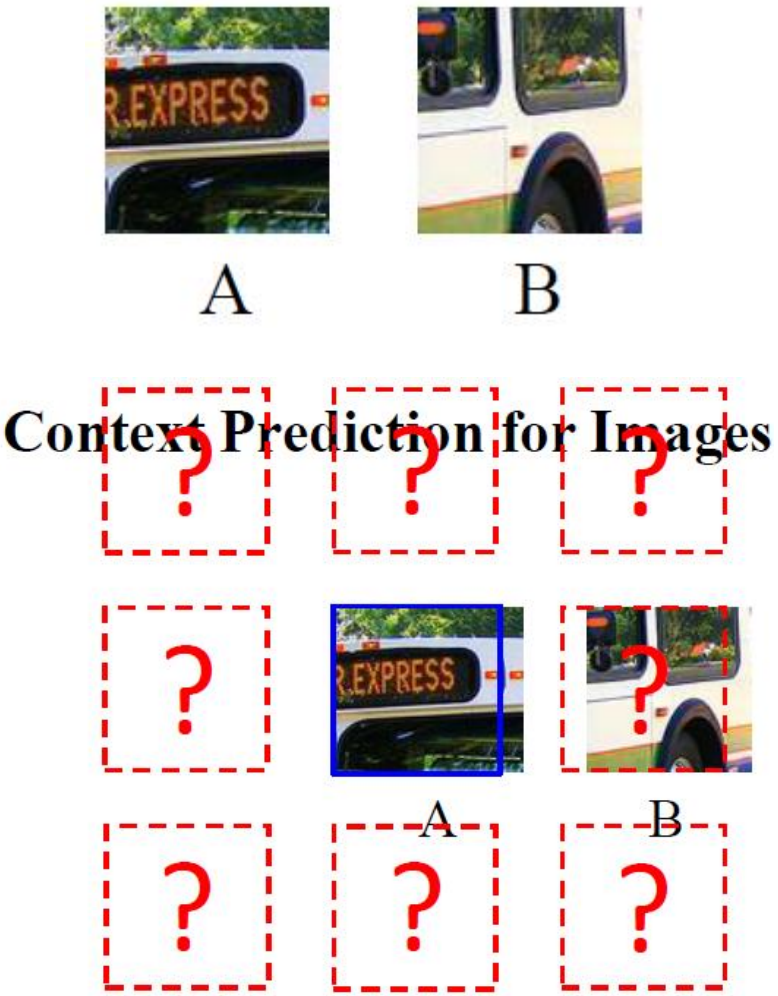
- For instance, predict the position of patch # 3 with respect to the central patch



- Input: two patches $X = ($  $)$
- Prediction: **Y = 3**

Relative Patch Position

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



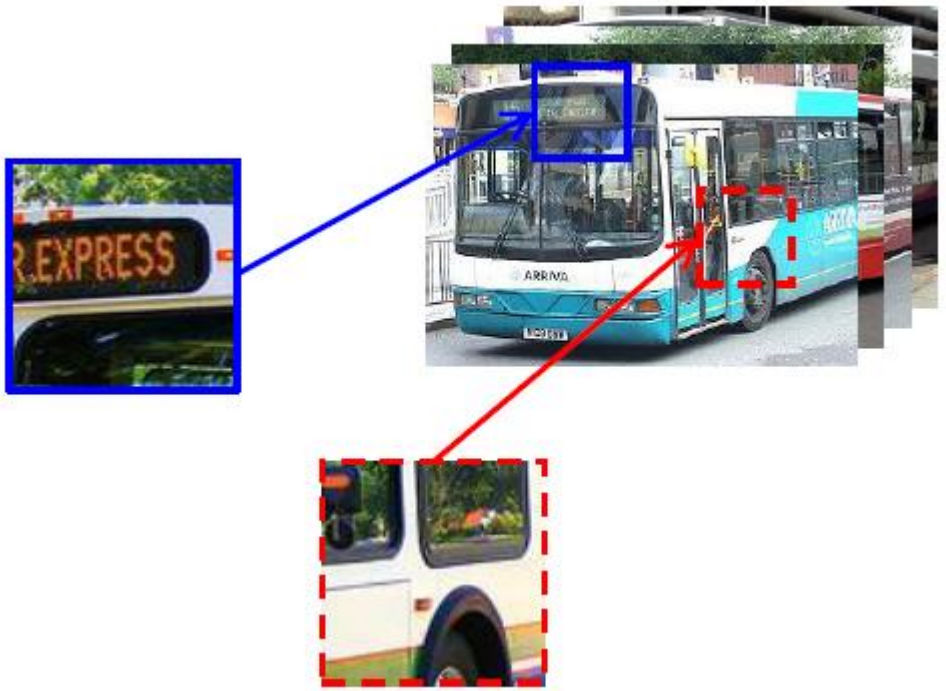
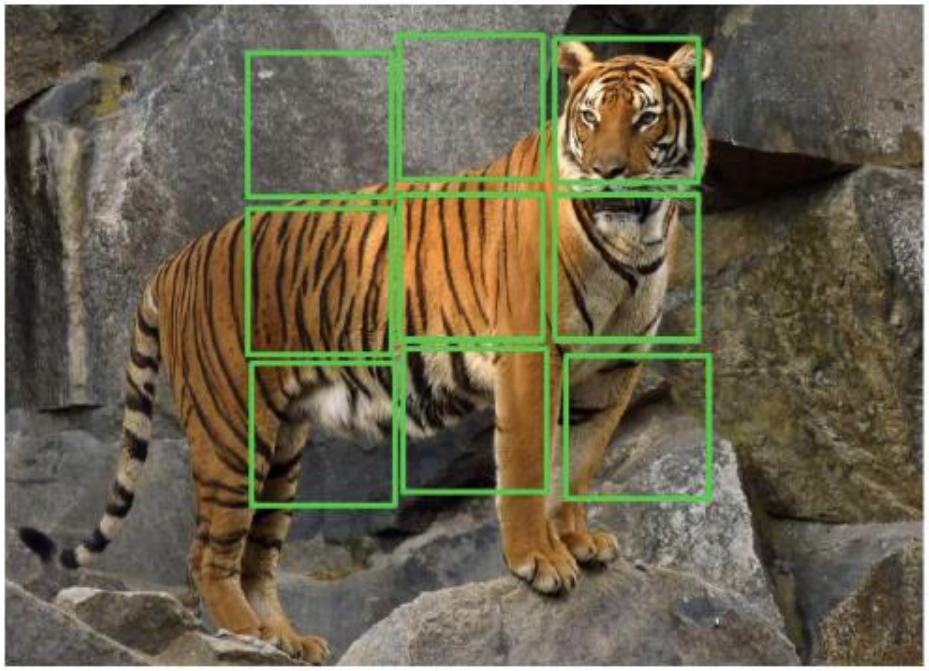


Image Jigsaw Puzzle

- *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 3×3 patches and solves a jigsaw puzzle



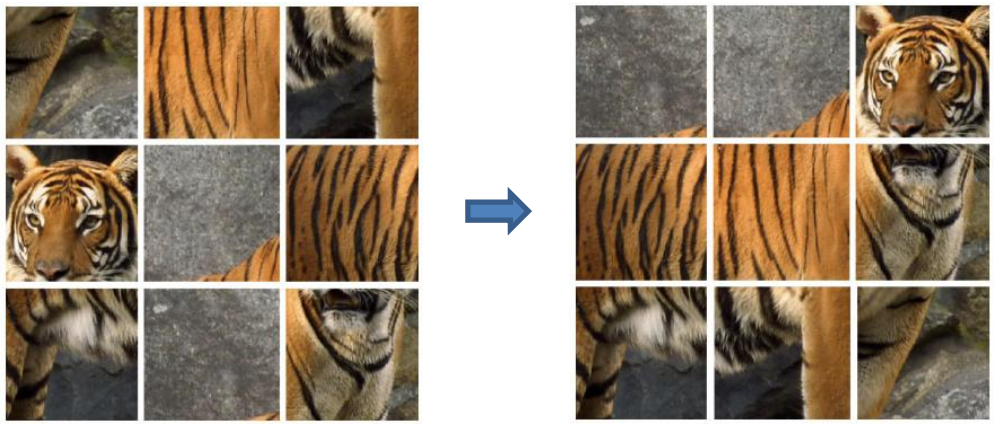


Image Jigsaw Puzzle

- 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

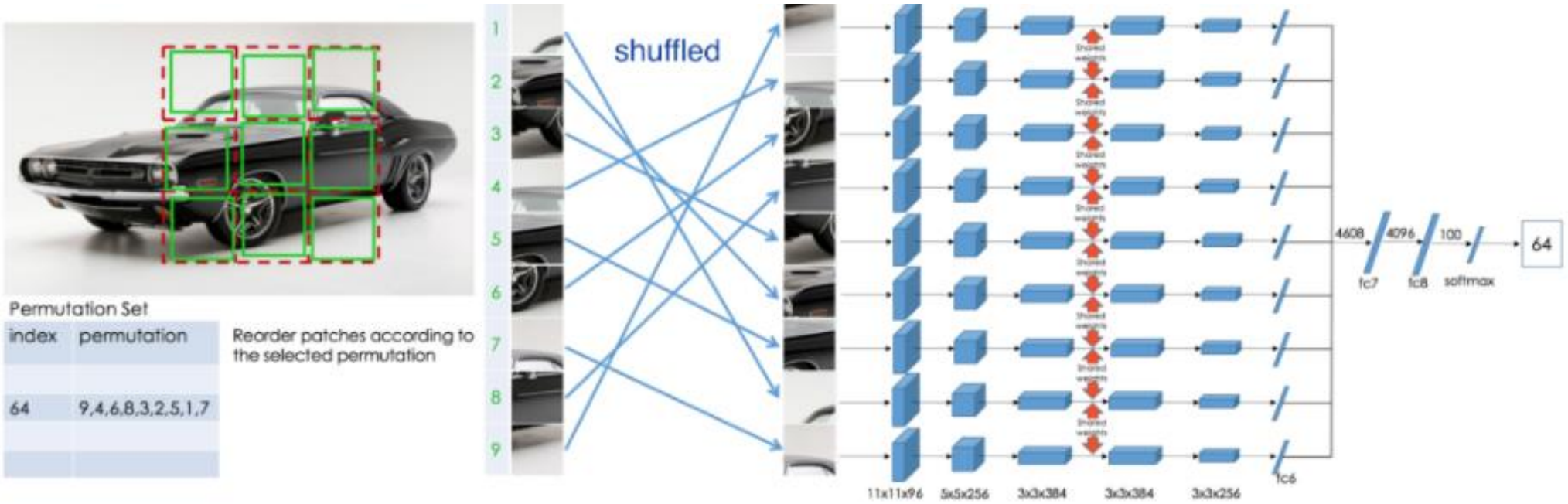
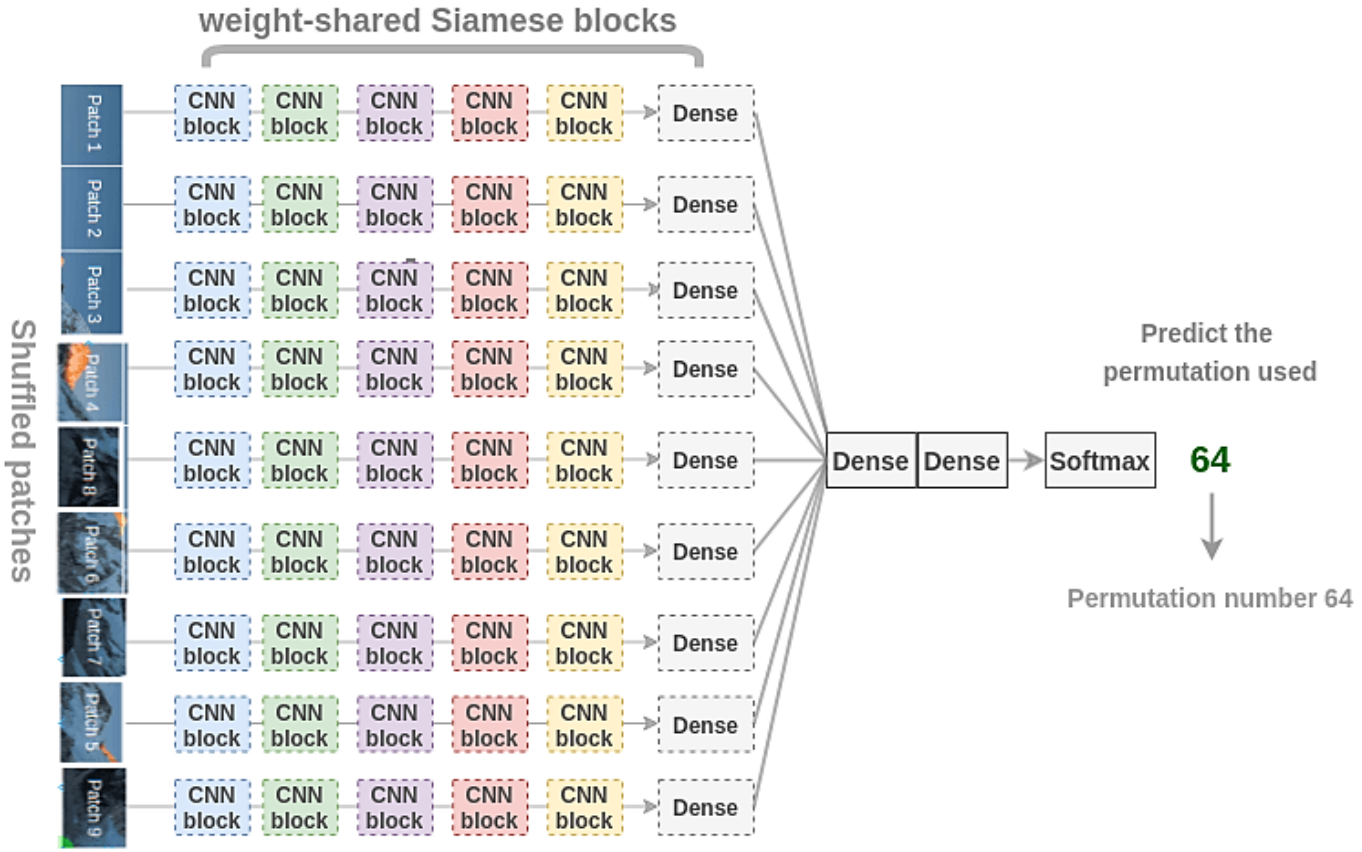


Image Jigsaw Puzzle

- 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



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

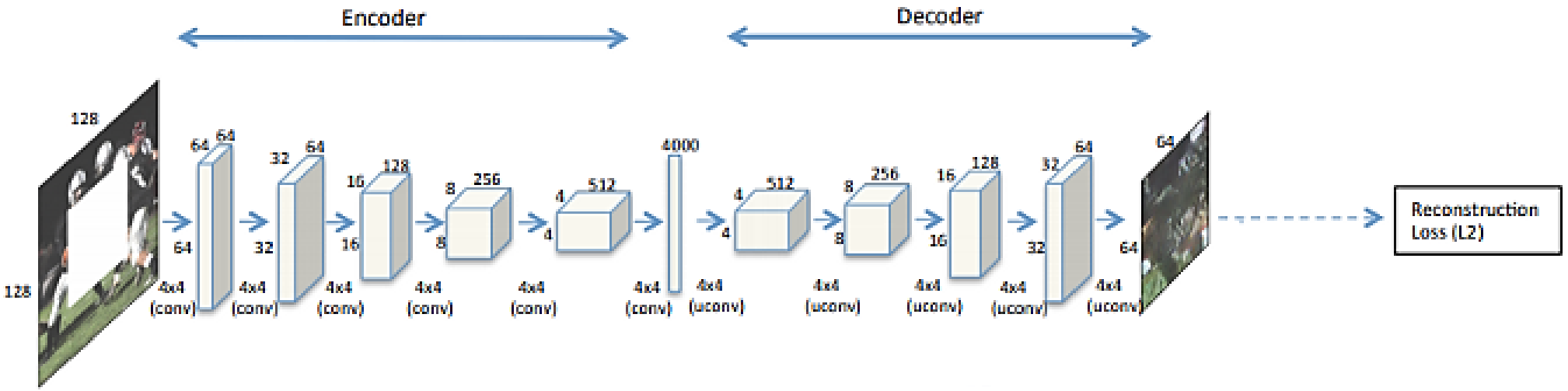
Context Encoders

- *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

Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

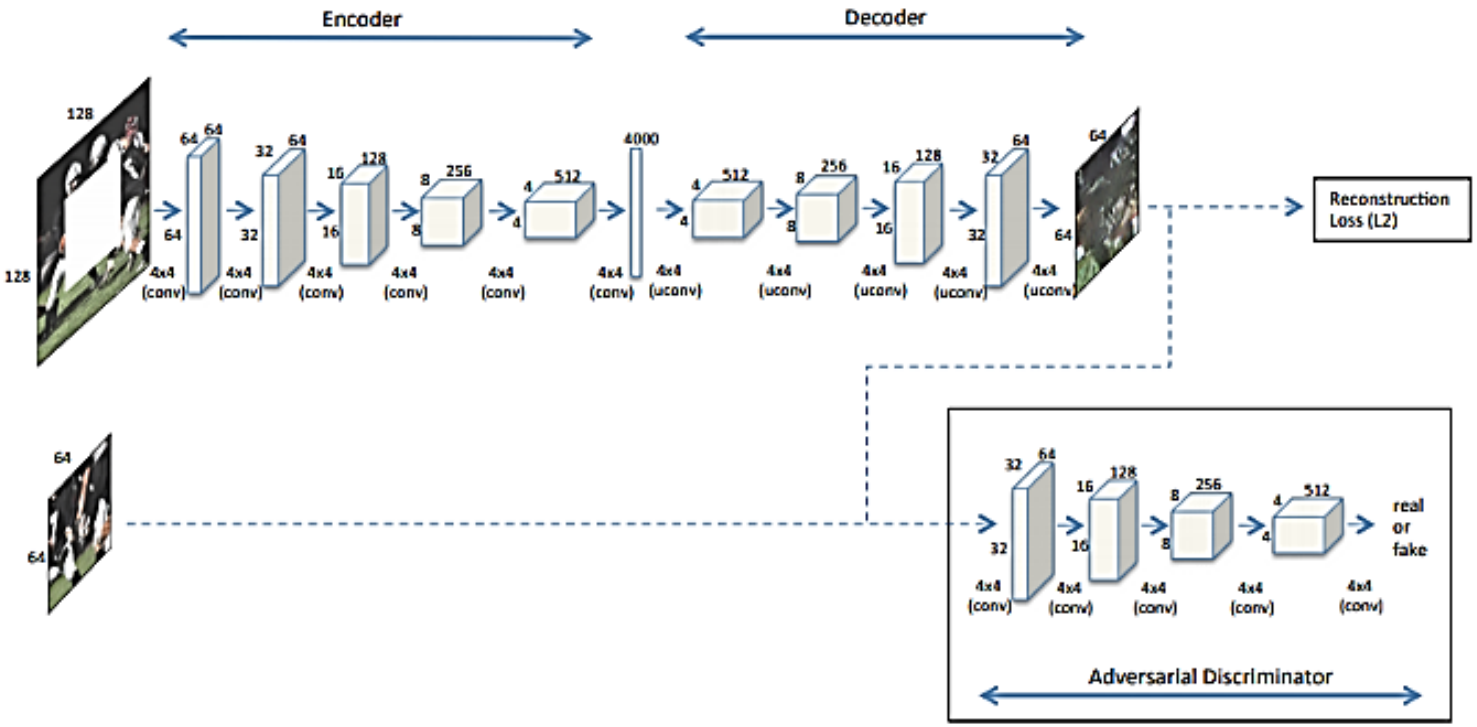
Context Encoders

- 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 ℓ_2 distance is used as the reconstruction loss function $\mathcal{L}_{\text{rec}}$



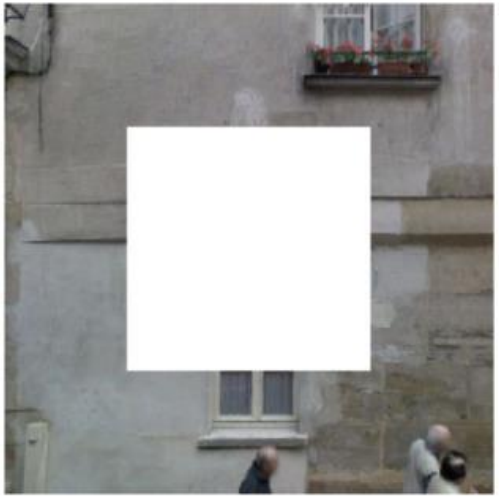
Context Encoders

- 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., $\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{gan}}\mathcal{L}_{\text{gan}}$



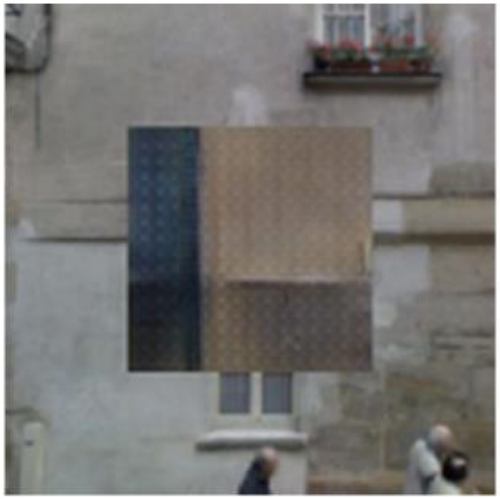
Context Encoders

- The joint loss function $\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}+\lambda_{\text{gan}}\mathcal{L}_{\text{gan}}$ resulted in improved prediction of the missing piece



Input image

Encoder-decoder with reconstruction loss $\mathcal{L}_{\text{rec}}$

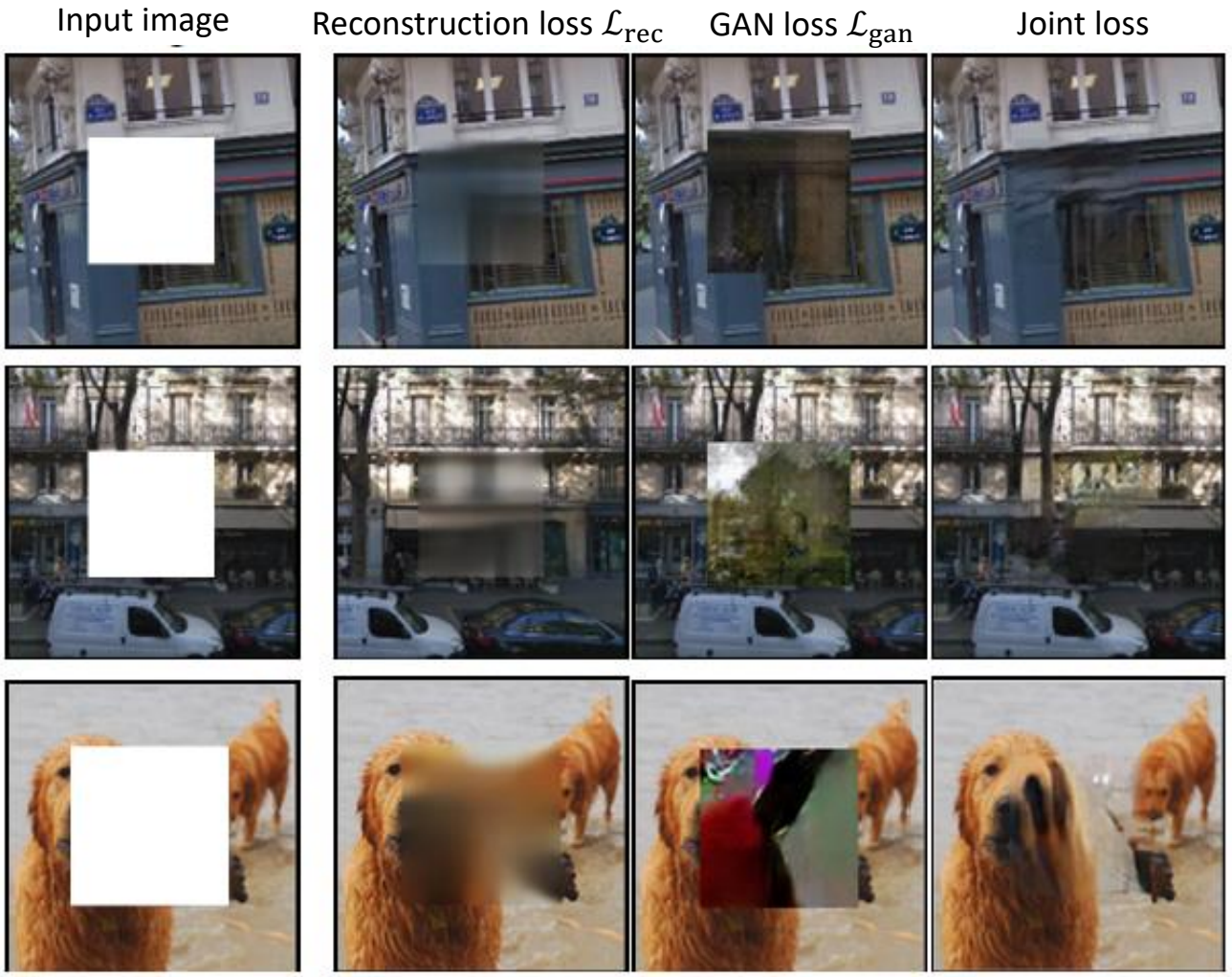


GAN with loss $\mathcal{L}_{\text{gan}}$

Joint loss $\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}+ \lambda_{\text{gan}}\mathcal{L}_{\text{gan}}$

Context Encoders

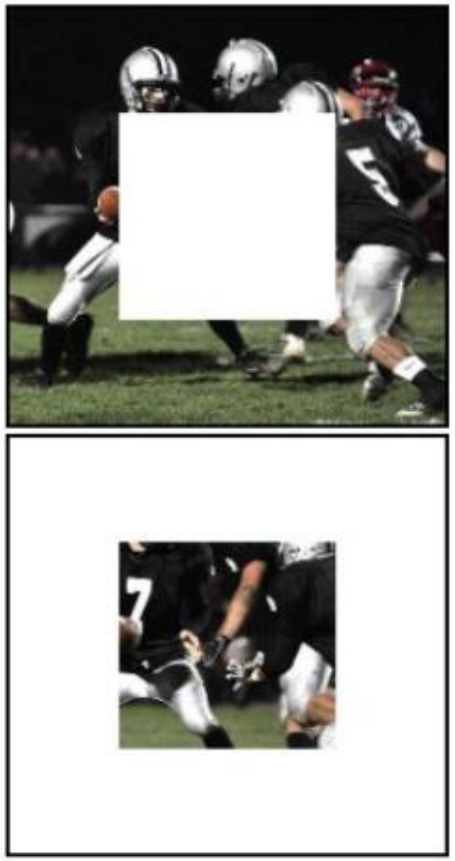
- Additional examples comparing the used loss functions
  - The joint loss produces the most realistic images



Context Encoders

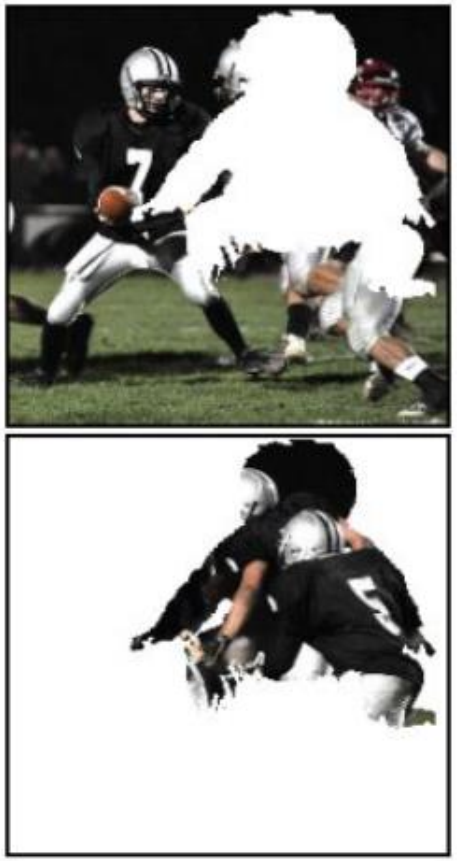
- The removed regions can have different shapes
  - Random region and random block masks outperformed the central region features

Central region



Random block

Random region



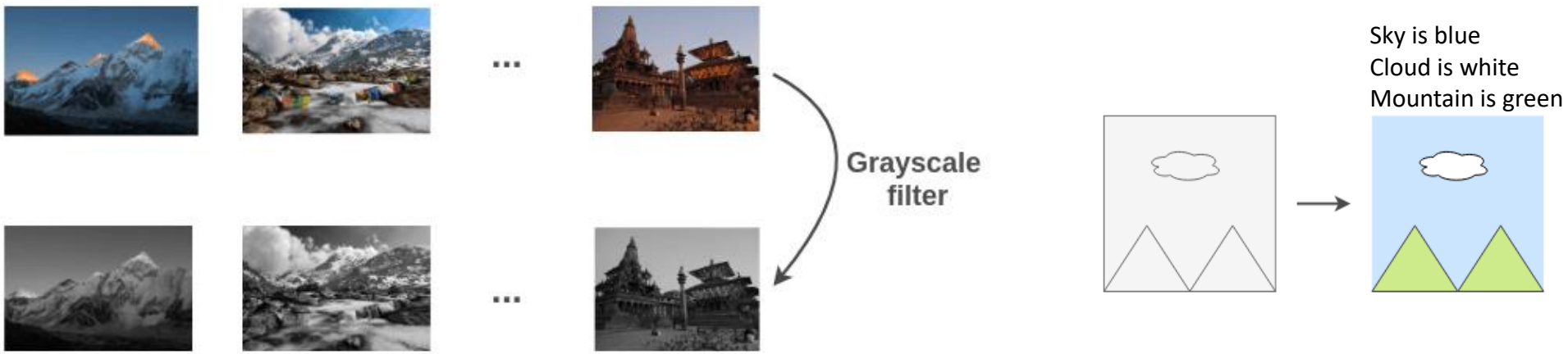
Context Encoders

- 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	**78.2%**	**56.8%**	**48.0%**
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal *et al.* [1]	egomotion	10 hours	52.9%	41.8%	-
Doersch *et al.* [7]	context	4 weeks	55.3%	**46.6%**	-
Wang *et al.* [39]	motion	1 week	**58.4%**	44.0%	-
Ours	context	14 hours	56.5%	44.5%	**29.7%**

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



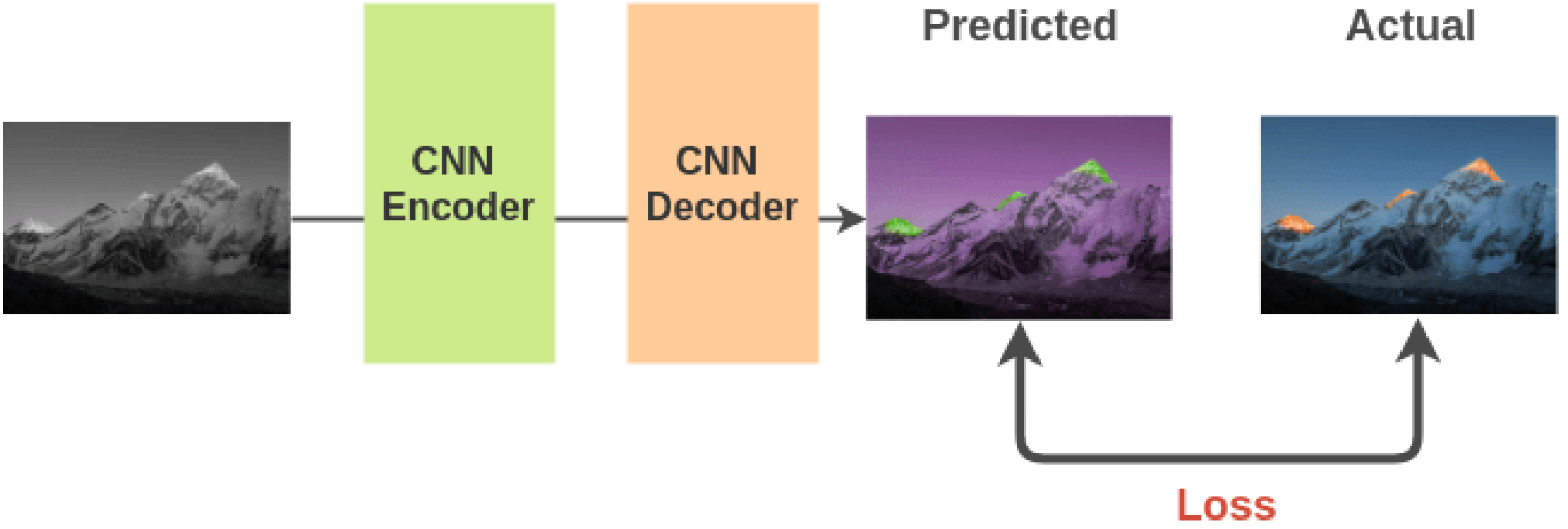
Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Image Colorization

- Input examples

Image Colorization

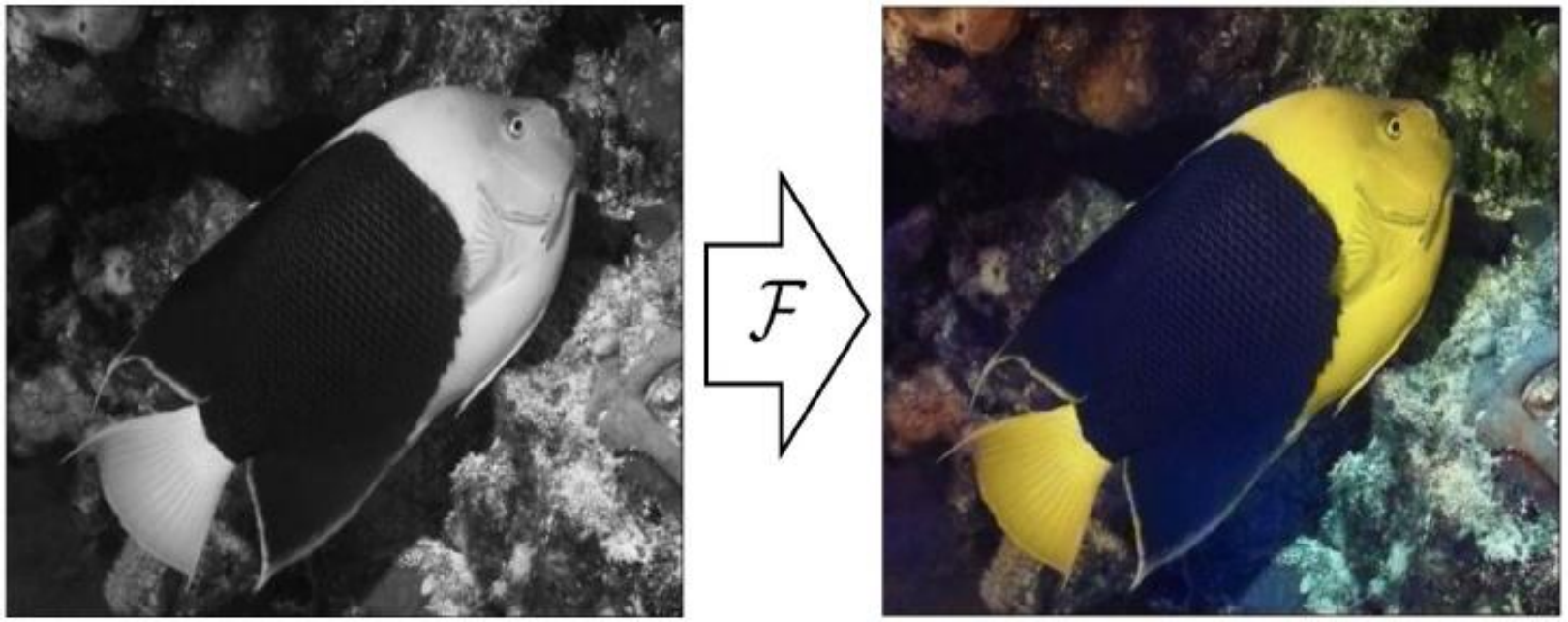
- An encoder-decoder architecture with convolutional layers is used
  - ℓ_2 loss between the actual color image and the predicted colorized image is used for training



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

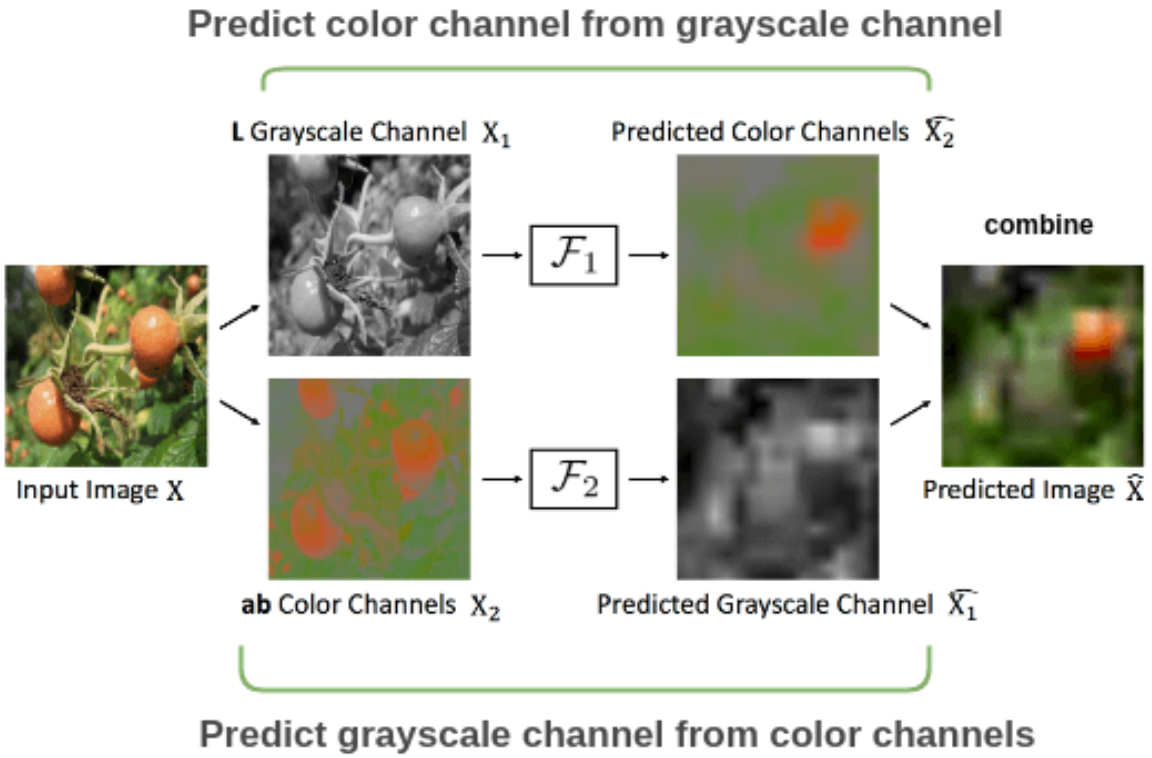
Cross-channel Prediction

- *Split-brain autoencoder* or *cross-channel prediction*
  - Zhang (2017) Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction
- Training data: remove some of the image color channels
- Pretext task: predict the missing channel from the other image channels
  - E.g., use the grayscale channel to predict the color channels in the image



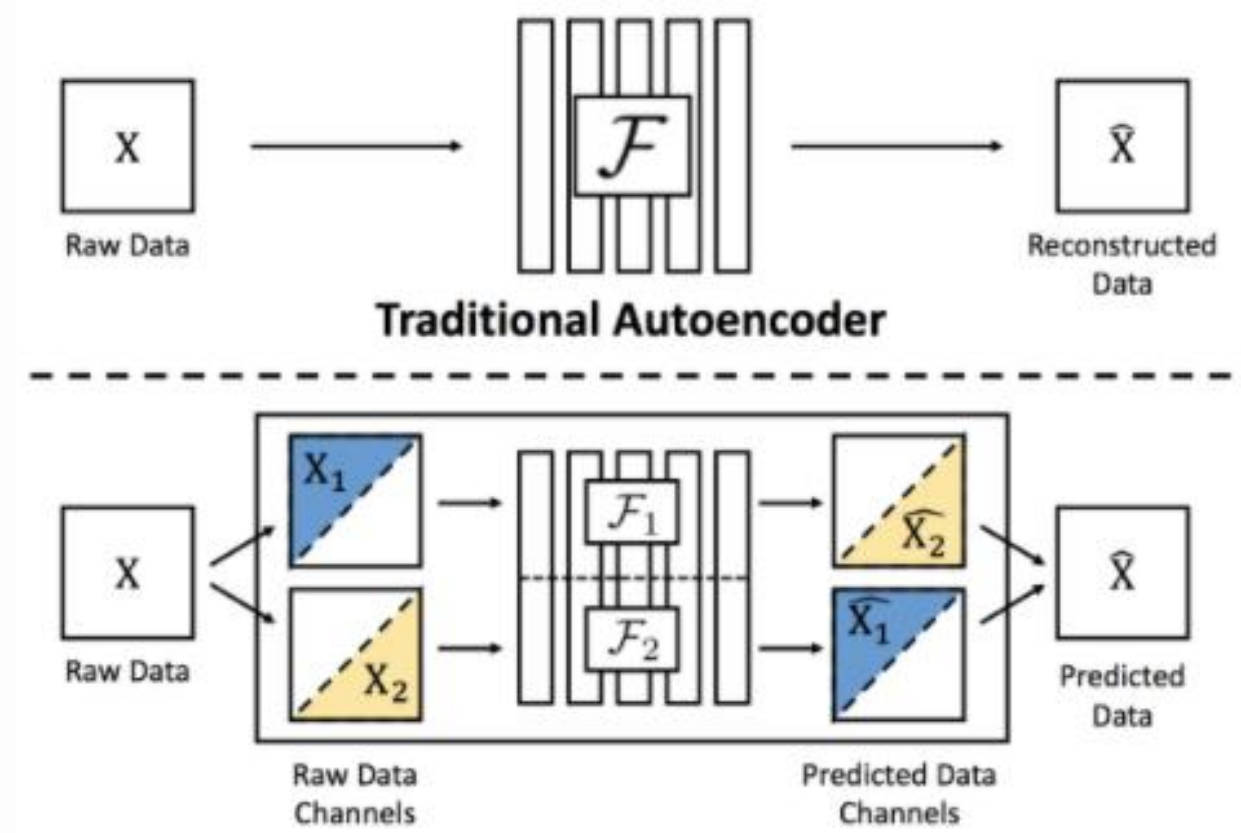
Cross-channel Prediction

- The input image (e.g., tomato) is split into grayscale and color channels
  - Two encoder-decoders are trained: F_1 predicts the color channels from the gray channel, and F_2 predicts the gray channel from the color channels
  - The two predicted images are combined to reconstruct the original image
    - A loss function (e.g., cross-entropy) is selected to minimize the distance between the original and reconstructed image



Cross-channel Prediction

- The general model is shown below, in comparison to a traditional autoencoder
  - Any combinations of image channels X_1 and X_2 can be used with this SSL approach for training and reconstruction
  - The reconstructed images are denoted with a "hat" notation $(\hat{X}_1, \hat{X}_2)$



Cross-channel Prediction

- It is also possible the split-brain autoencoder to predict HHA depth channels in images
  - The HHA format encodes three channels for each pixel, including the horizontal disparity, height above ground, and angle with gravity
- The two autoencoders predict depth from color and color from depth, and combine their outputs into a single representation

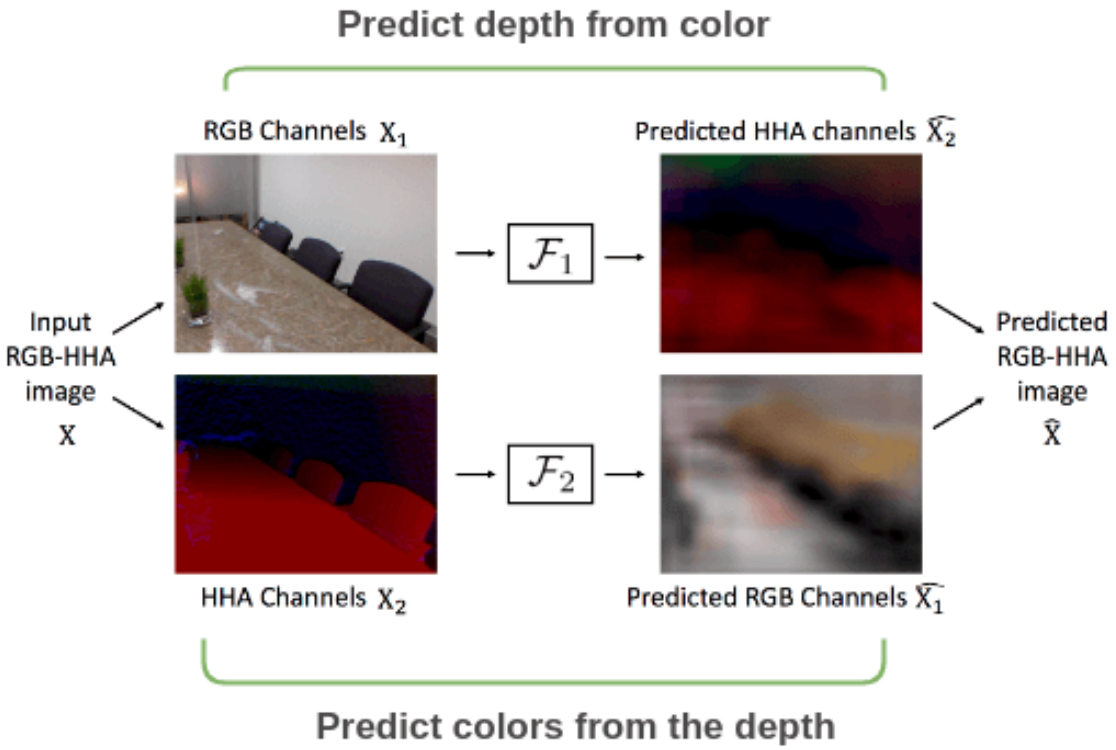
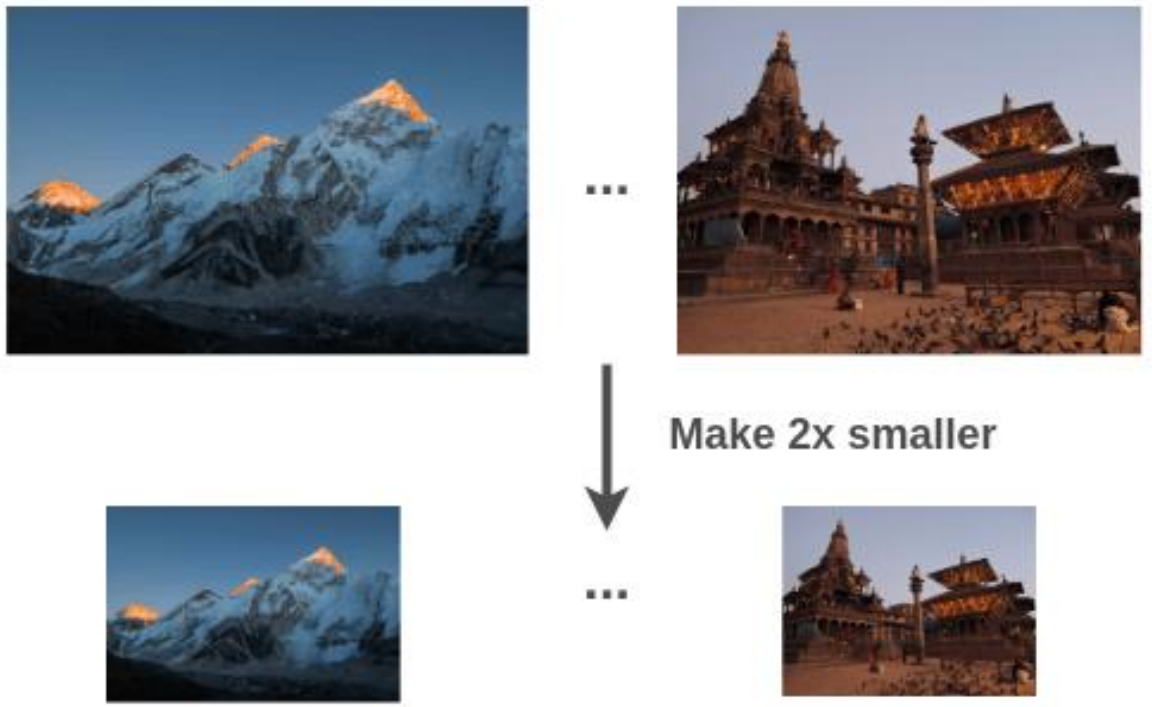


Image Super-Resolution

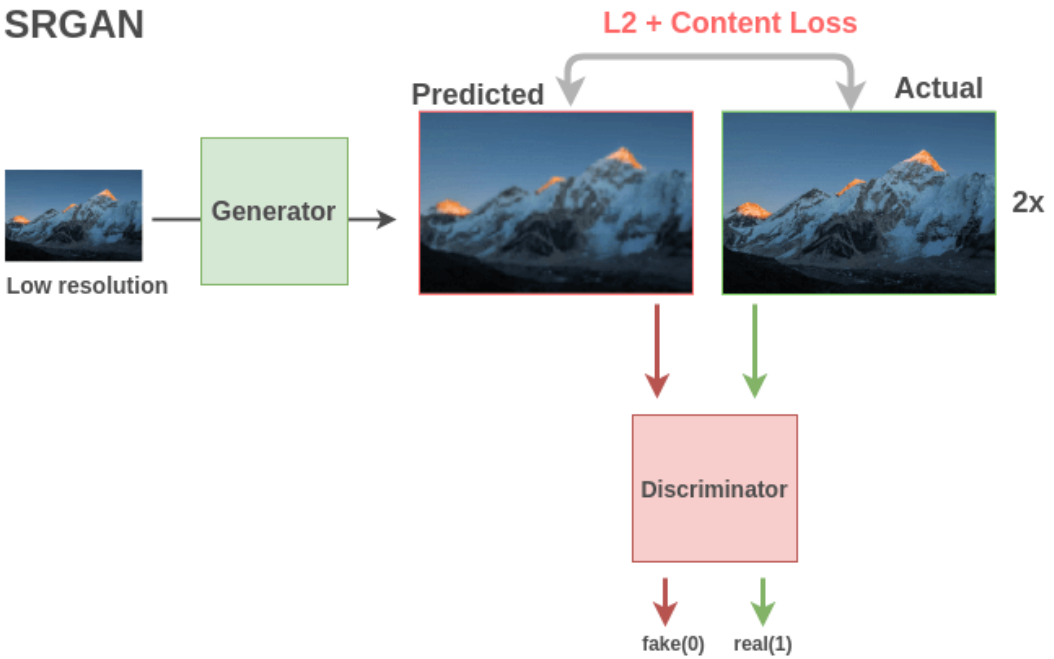
- *Image Super-Resolution*
  - Ledig (2017) Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network
- Training data: pairs of regular and downsampled low-resolution images
- Pretext task: predict a high-resolution image that corresponds to a downsampled low-resolution image



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Image Super-Resolution

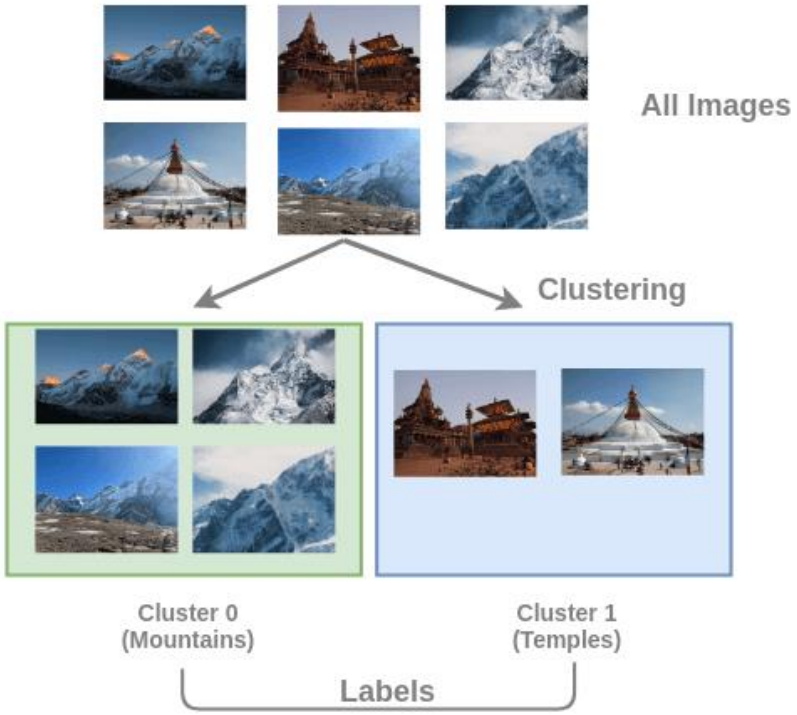
- SRGAN (Super-Resolution GAN) is a variant of GAN for producing super-resolution images
  - The generator takes a low-resolution image and outputs a high-resolution image using a fully convolutional network
  - The discriminator uses a loss function that combines L_2 and content loss to distinguish between the actual (real) and generated (fake) super-resolution images
    - **Content loss** compares the feature content between the actual and predicted images



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Deep Clustering

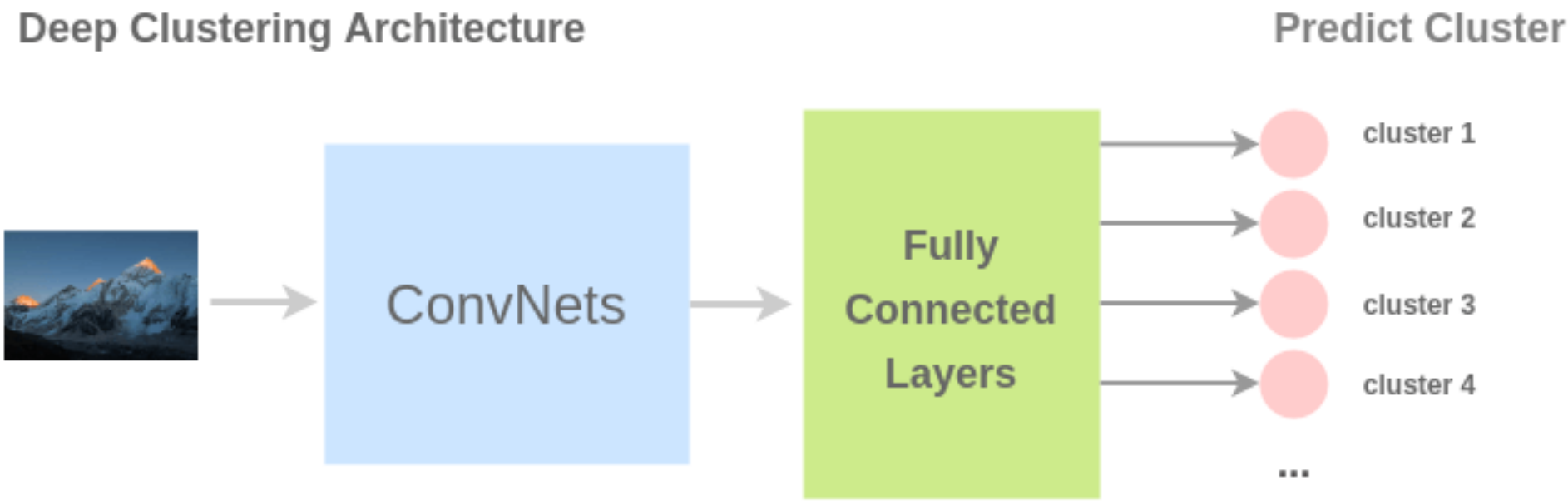
- *Deep clustering of images*
  - Caron (2019) Deep Clustering for Unsupervised Learning of Visual Features
- Training data: clusters of images based on the content
  - E.g., clusters on mountains, temples, etc.
- Pretext task: predict the cluster to which an image belongs



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Deep Clustering

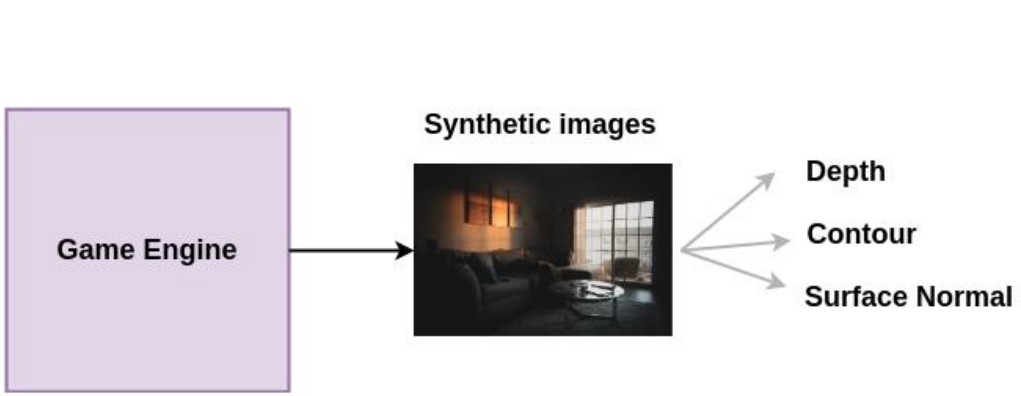
- The architecture for SSL is called deep clustering
  - The model treats each cluster as a separate class
  - The output is the number of the cluster (i.e., cluster label) for an input image
  - The authors used *k*-means for clustering the extracted feature maps
- The model needs to learn the content in the images in order to assign them to the corresponding cluster

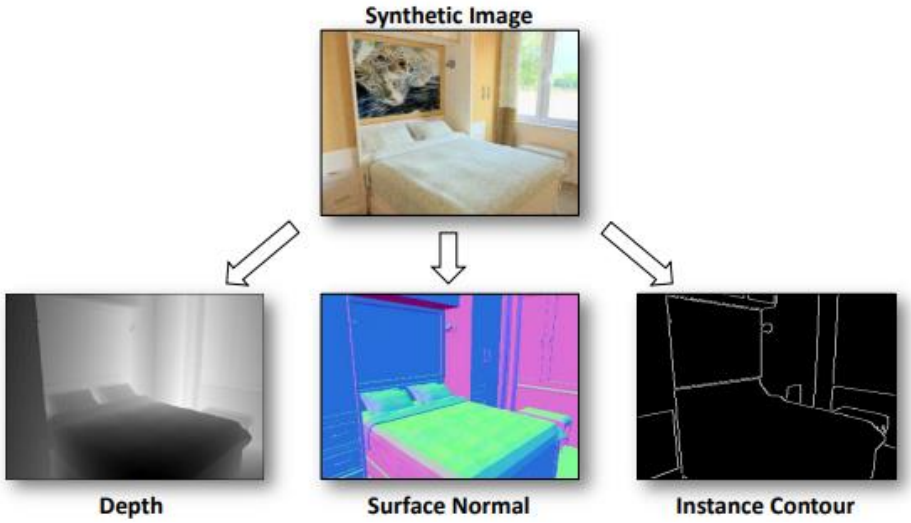


Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Synthetic Imagery

- *Synthetic imagery*
  - Ren (2017) Cross-Domain Self-supervised Multi-task Feature Learning using Synthetic Imagery
- Training data: synthetic images generated by game engines and real images
  - Graphics engines in games can produce realistically looking synthetic images
- Pretext task: predict whether an input image is synthetic or real, based on predicted depth, surface normal, and instance contour maps
  - The learned features are useful for segmentation and classification downstream tasks

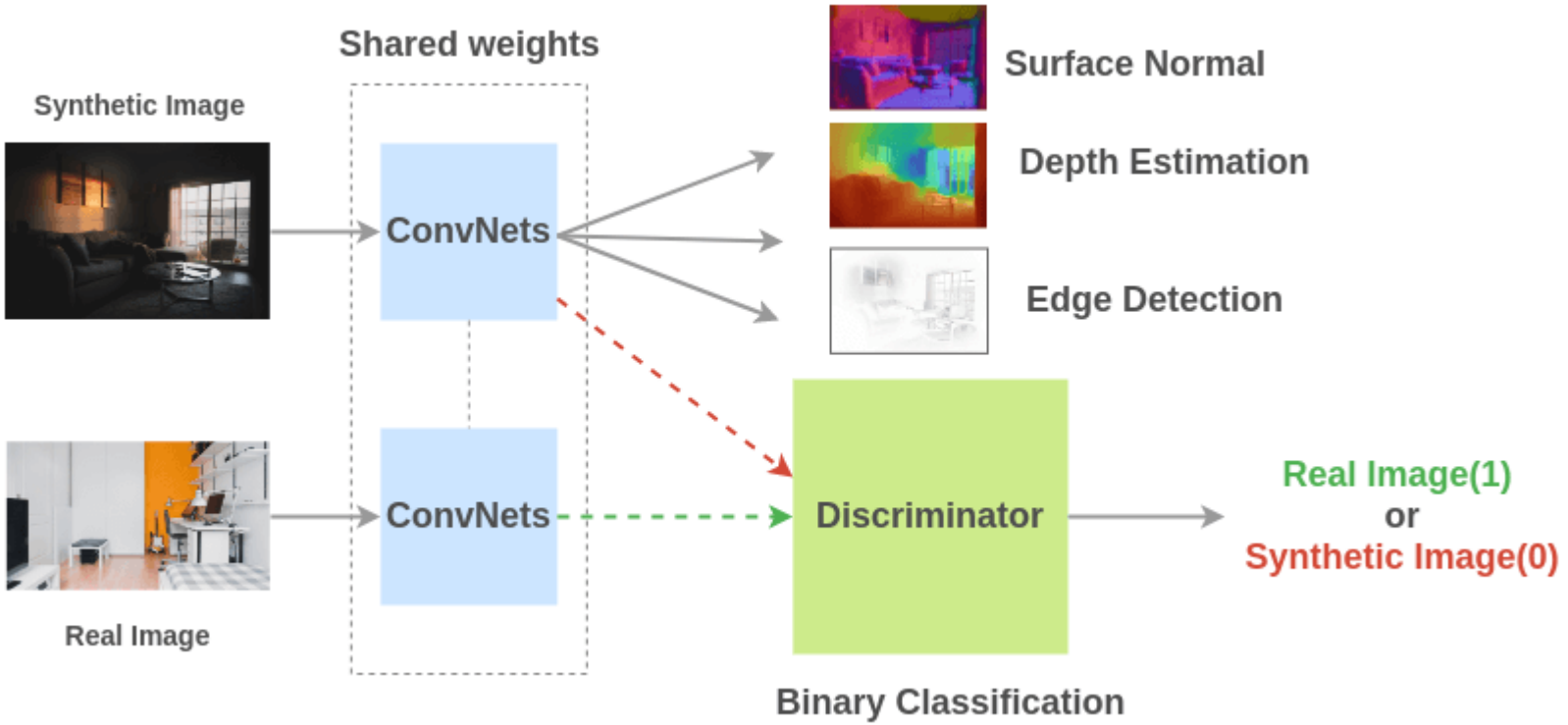




Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Synthetic Imagery

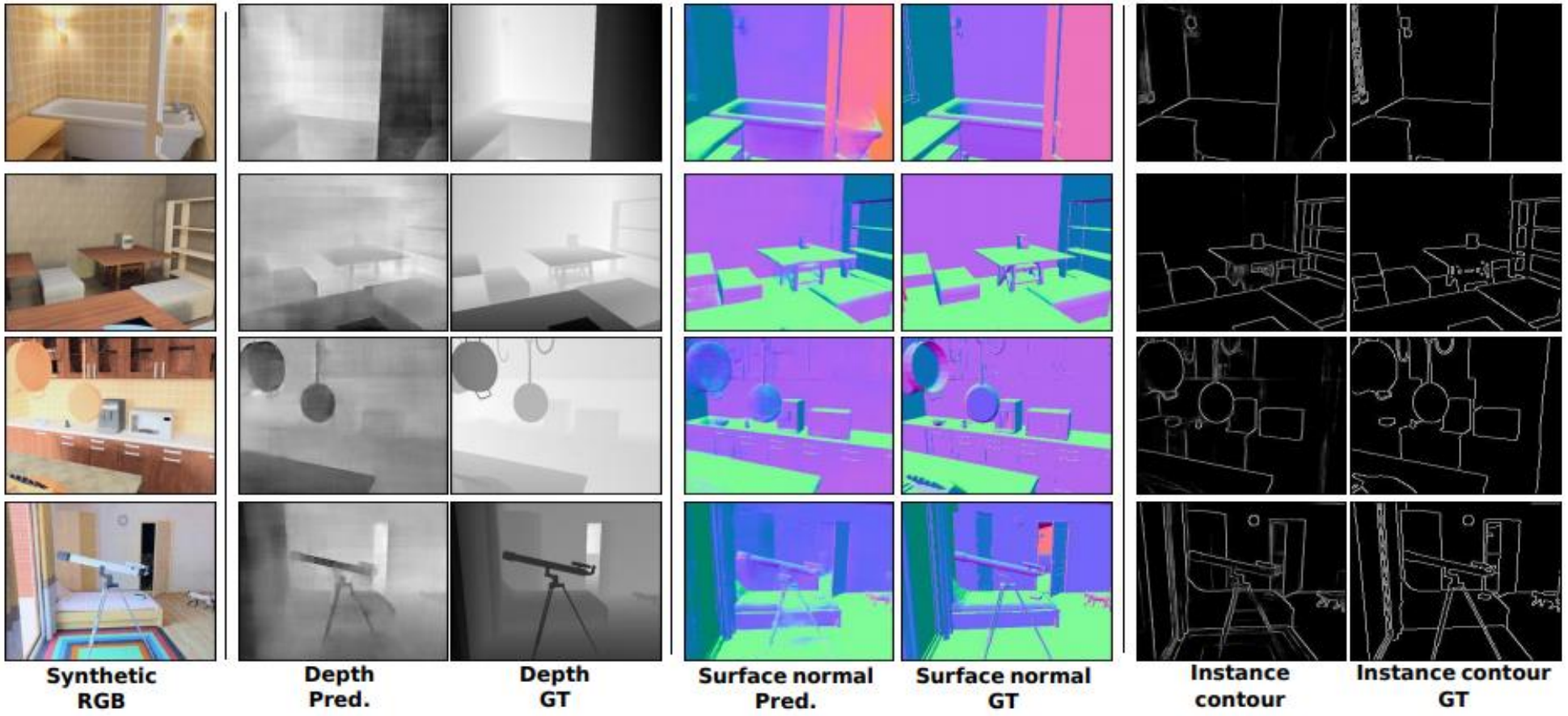
- The model has weight-shared ConvNets that are trained on both real and synthetic images
  - The discriminator learns to distinguish real from synthetic images, based on the surface normal, depth, and edge maps
  - The model is trained in an adversarial manner (as in GANs), by simultaneously improving both the generator and discriminator



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

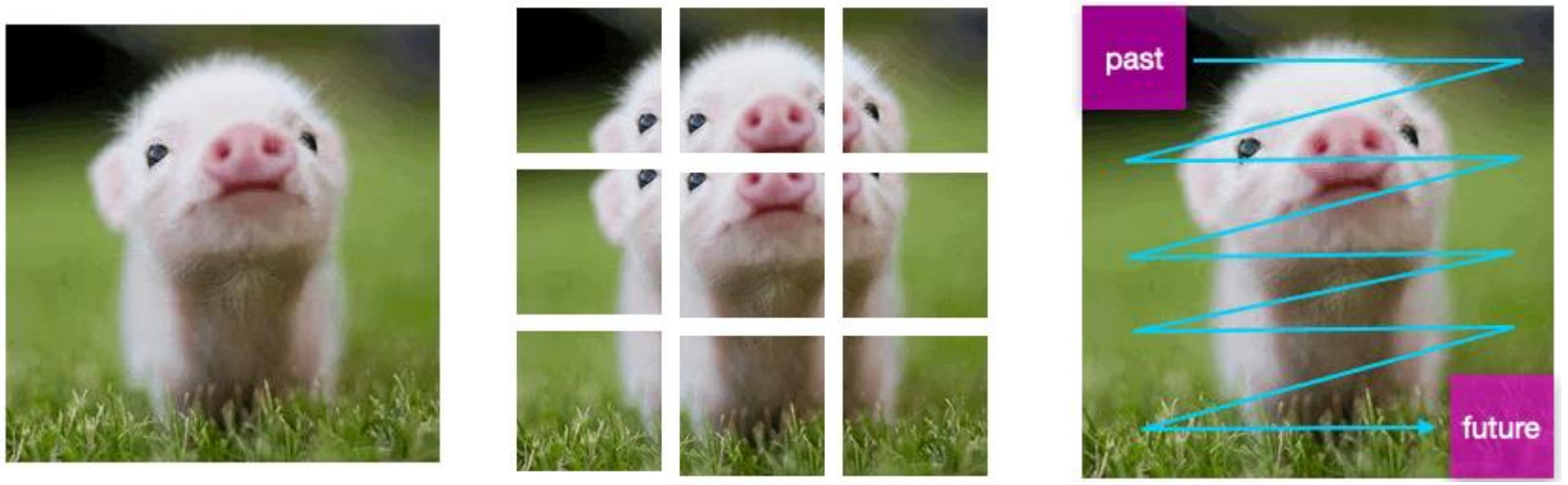
Synthetic Imagery

- Learned depth, surface normal, and instance contour maps, and the corresponding ground truth in synthetic images



Contrastive Predictive Coding

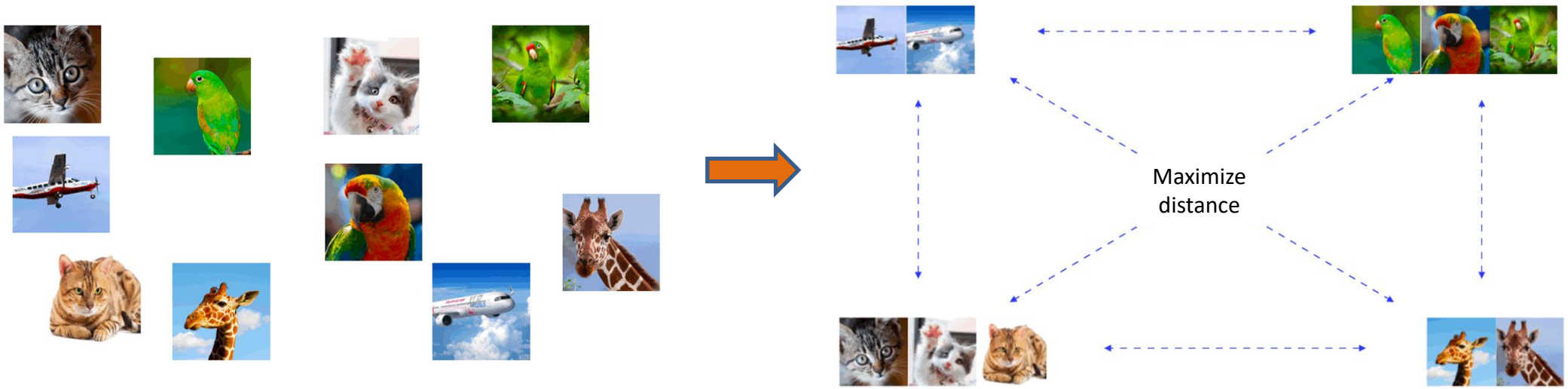
- *Contrastive Predictive Coding (CPC)*
  - Van der Oord (2018) Representation Learning with Contrastive Predictive Coding
- Training data: extracted patches from input images
- Pretext task: predict the order for a sequence of patches using contrastive learning
  - E.g., how to predict the next (future) 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



Picture from: William Falcon – A Framework for Contrastive Self-Supervised Learning and Designing a New Approach

Contrastive Predictive Coding

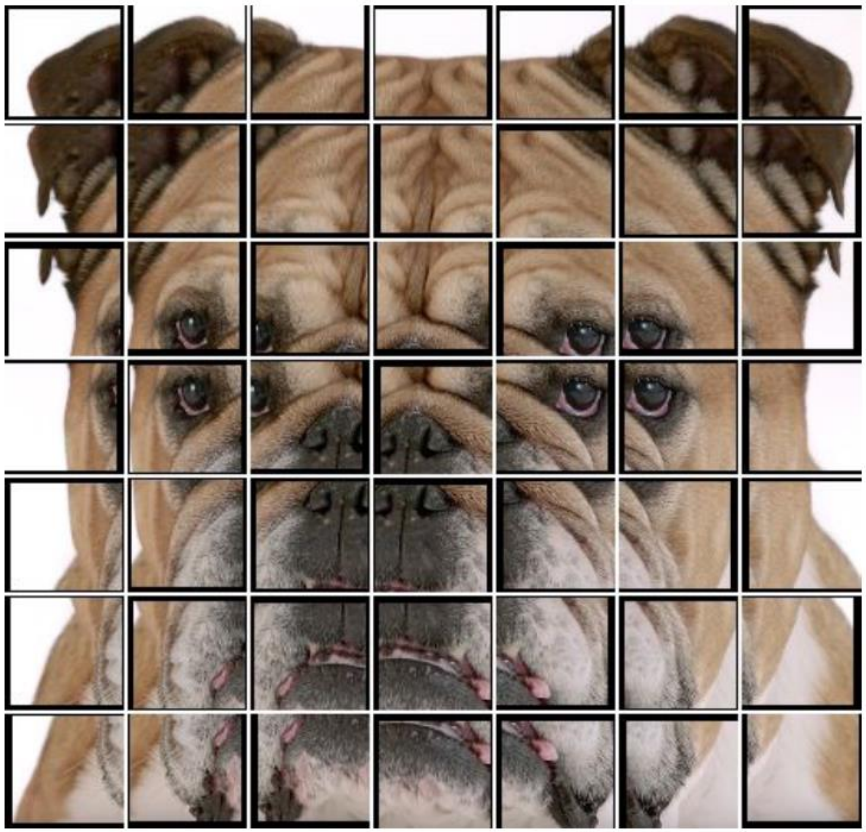
- Contrastive learning is based on grouping similar examples together
  - E.g., cluster the shown images into groups of similar images
- Noise-Contrastive Estimation (NCE) loss is commonly used in contrastive learning
  - The NCE loss minimizes the distance between similar images (positive examples) 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)



Picture from: William Falcon – A Framework for Contrastive Self-Supervised Learning and Designing a New Approach

Contrastive Predictive Coding

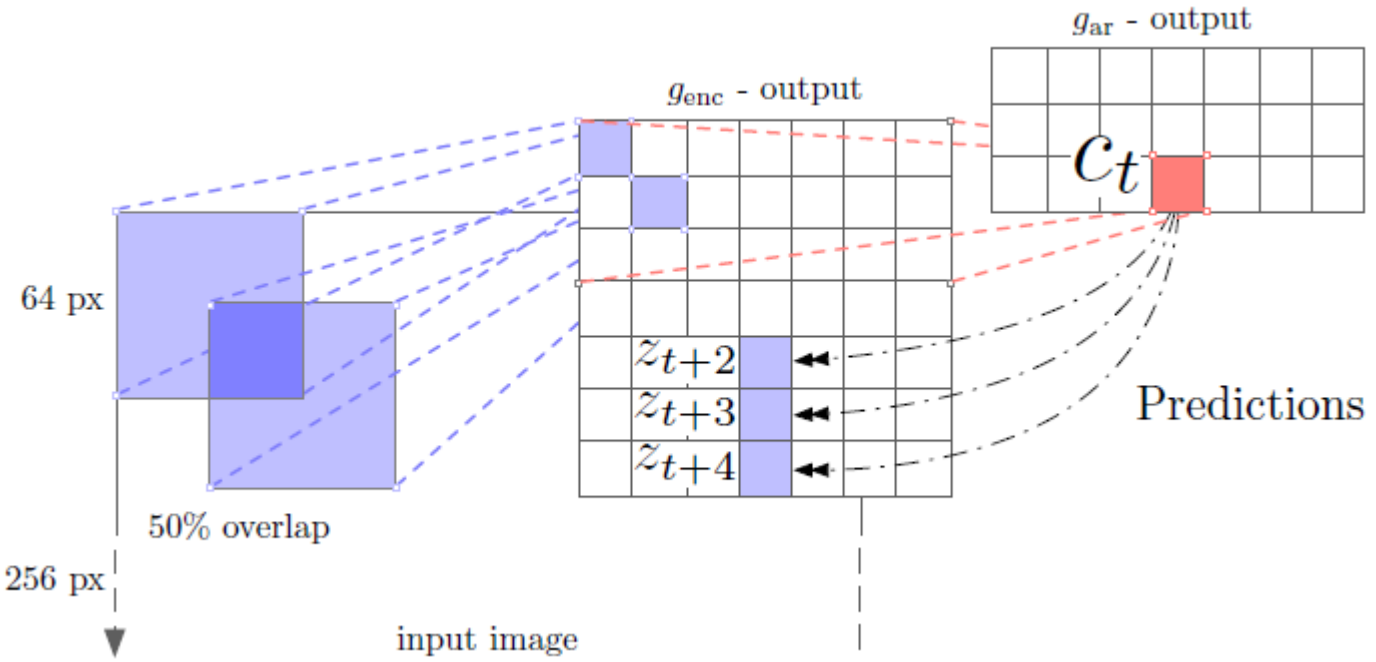
- For an input image resized to 256×256 pixels, the authors extracted a grid of 7×7 patches of size 64×64 pixels with 50% overlap between the patches
  - Therefore, there are 49 overlapping patches in total for each image



Picture from: Pieter Abbeel, Lecture 7 – Self-Supervised Learning

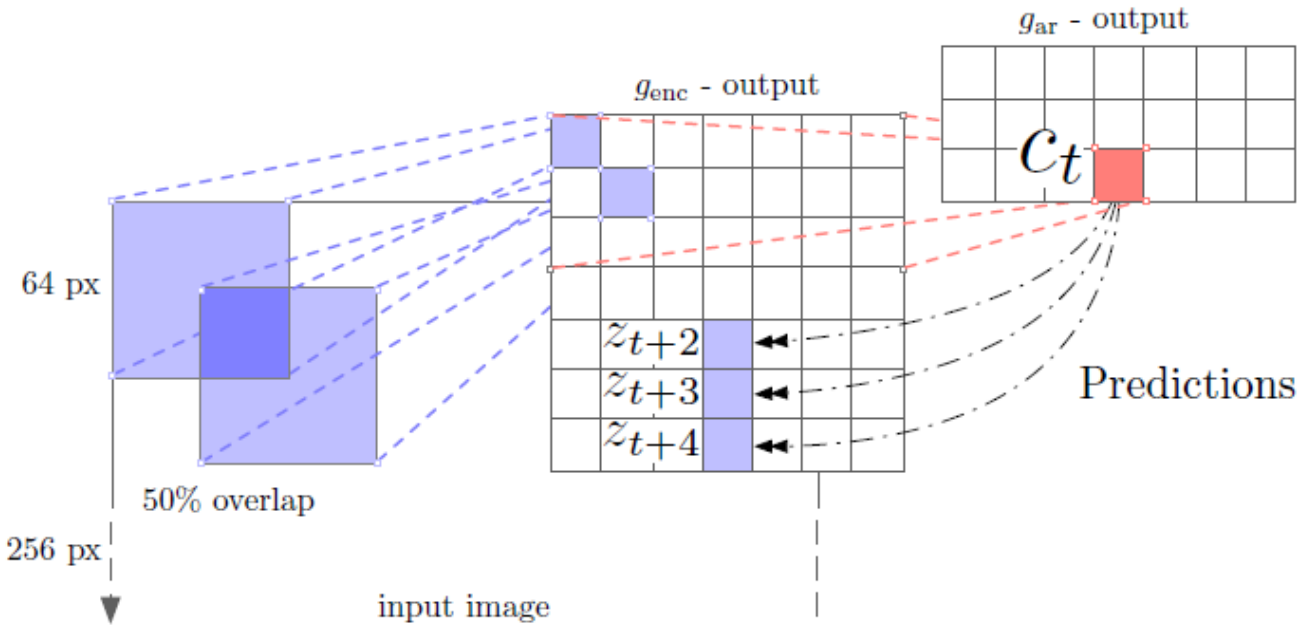
Contrastive Predictive Coding

- An encoder g_{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 64×64 pixels size with 50% overlap between the patches
  - A ResNet-101 encoder is used for projecting the patch x_t into a code representation z_t
  - The middle image shows the outputs of the encoder g_{enc} for each patch, $z_t = g_{\text{enc}}(x_t)$
  - For the 49 patches (7×7 grid), the outputs are 7×7×1,024 tensors (i.e., $z_1, z_2, \ldots, z_{49}$)



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 c_t (in row 3 and column 4) is the sum of all vectors $g_{\text{ar}}(z_{\leq t})$ 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 c_t is used to predict the blue patches in the next rows and the same column as the red patch, denoted z_{t+2}, z_{t+3} and z_{t+4}
  - The authors predicted up to five rows for each patch in the grid



Contrastive Predictive Coding

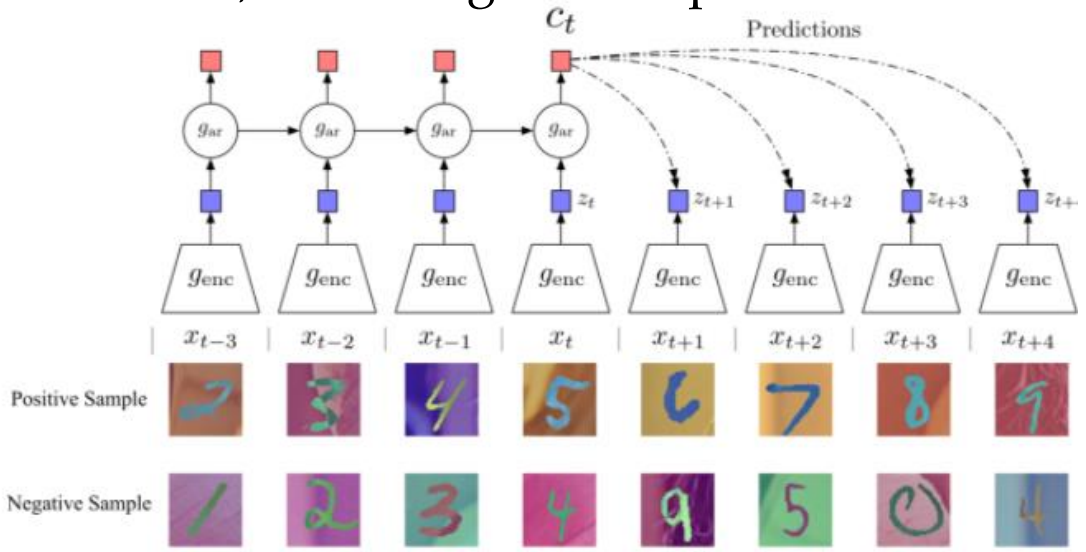
- The NCE loss is used for training the model on each patch in the image
  - The patch at position t is considered a positive sample, and all other patches are considered negative samples
- For a set of N random patches $\mathrm{X} = \{x_1, \ldots, x_N\}$, containing one positive sample x_t and $N - 1$ negative samples, the NCE loss is:

$$\mathcal{L}_N = -\mathbb{E}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k\left(x_j, c_t\right)}\right]$$

- c_t is the code representation of the patch at position t, i.e., $c_t = g_{\mathrm{ar}}(z_{\leq t})$
- x_{t+k} is a predicted patch in the sequence at position $t + k$
- $f_k(x_{t+k}, c_t)$ is a density function that approximates the probability $p_k(x_{t+k}|c_t)$ of estimating the patch x_{t+k} for a given code representation c_t
  - The authors used a log-bilinear model $f_k(x_{t+k}, c_t) = exp\left(z_{t+k}^T W_k c_t\right)$, where W_k 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



Picture from: David Tellez – Contrastive predictive coding

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				
Labeled data		1%	5%	10%	50%	100%
†Supervised baseline	ResNet-200	44.1	75.2*	83.9	93.1	95.2#
Methods using label-propagation:						
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 ×4	-	-	**91.2**	-	95.0
Methods using representation learning only:						
Instance Discrimination [60]	ResNet-50	39.2	-	77.4	-	-
Rotation [63]	ResNet-152 ×2	57.5	-	86.4	-	-
ResNet on BigBiGAN (fixed)	RevNet-50 ×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**

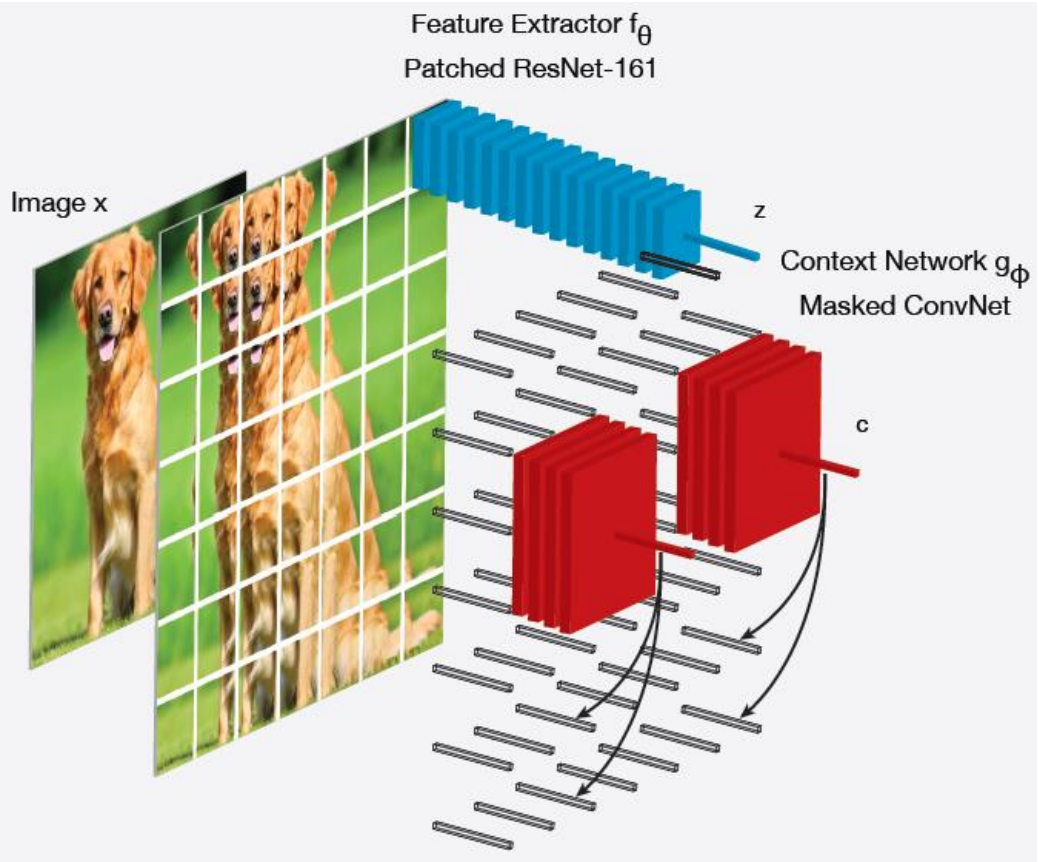
Method	Architecture	mAP
Transfer from labeled data:		
Supervised baseline	ResNet-152	74.7
Transfer from unlabeled data:		
Exemplar [17] by [13]	ResNet-101	60.9
Motion Segmentation [47] by [13]	ResNet-101	61.1
Colorization [64] by [13]	ResNet-101	65.5
Relative Position [14] by [13]	ResNet-101	66.8
Multi-task [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

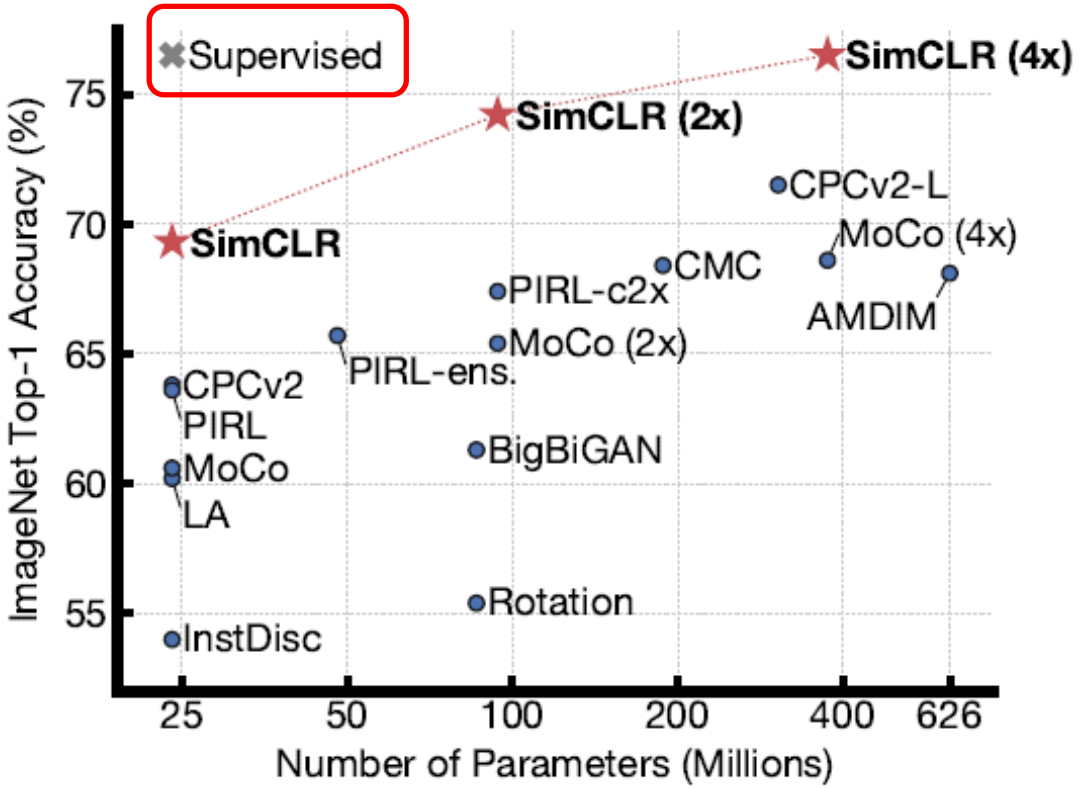
CPC v2

- Each patch is encoded with a feature extractor f_θ (blue) followed by average-pooling (vector z)
  - The autoregressive context network g_φ (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

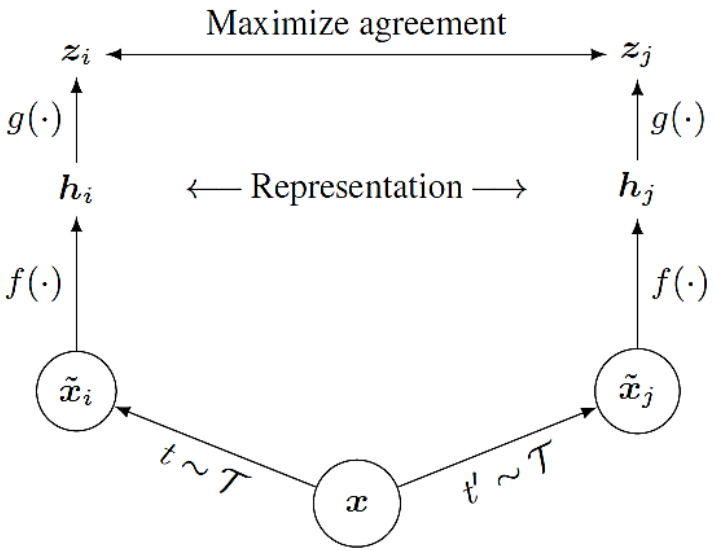


SimCLR

- Approach:
  - Randomly sample a mini-batch of n inputs $\boldsymbol{x}$, and apply two different data augmentation operations t and t', resulting in $2n$ samples $\tilde{\boldsymbol{x}}_i = t(\boldsymbol{x})$ and $\tilde{\boldsymbol{x}}_j = t'(\boldsymbol{x})$
    - 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 $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ to obtain the code representations $\boldsymbol{h}_i = f(\tilde{\boldsymbol{x}}_i)$ and $\boldsymbol{h}_j = f(\tilde{\boldsymbol{x}}_j)$
  - Apply another prediction head encoder $g(\cdot)$ (one fully-connected layer) to $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ to obtain the code representations $\boldsymbol{z}_i = g(\boldsymbol{h}_i)$ and $\boldsymbol{z}_j = g(\boldsymbol{h}_j)$

Prediction head *g*:
One fully-connected layer

Base encoder *f*:
ResNet + global average pooling



SimCLR

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

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

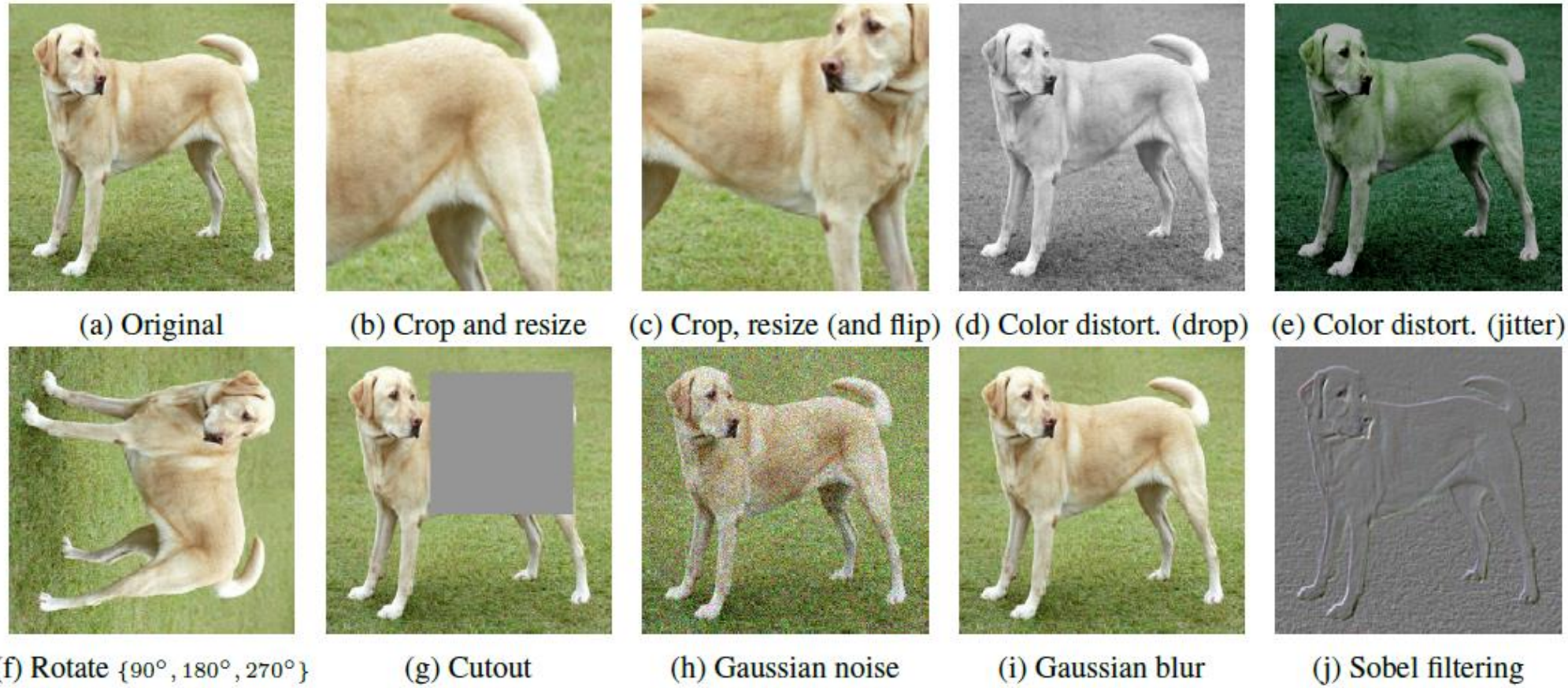
- The contrastive prediction task aims for a given sample $\tilde{\boldsymbol{x}}_i$ to identify a positive pairing sample $\tilde{\boldsymbol{x}}_j$
- The NCE loss for the instances $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ is calculated as:

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

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

SimCLR

- Data augmentation



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
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**78.8**	**93.2**	**94.2**	**98.0**
Random init	88.3	96.0	81.9	**77.0**	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

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

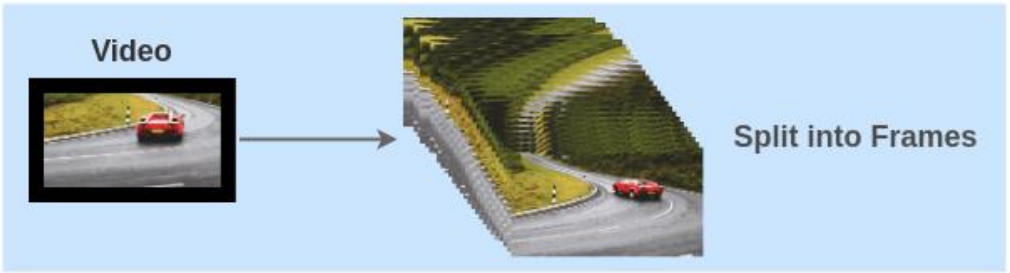
Dataset	GPUs	Hours	Equivalent AWS cost
CIFAR-10	2 V100 (32GB)	24	$188
STL-10	8 V100 (32GB)	72	$2,246
ImageNet	256 V100 (32 GB)	336	$335,462

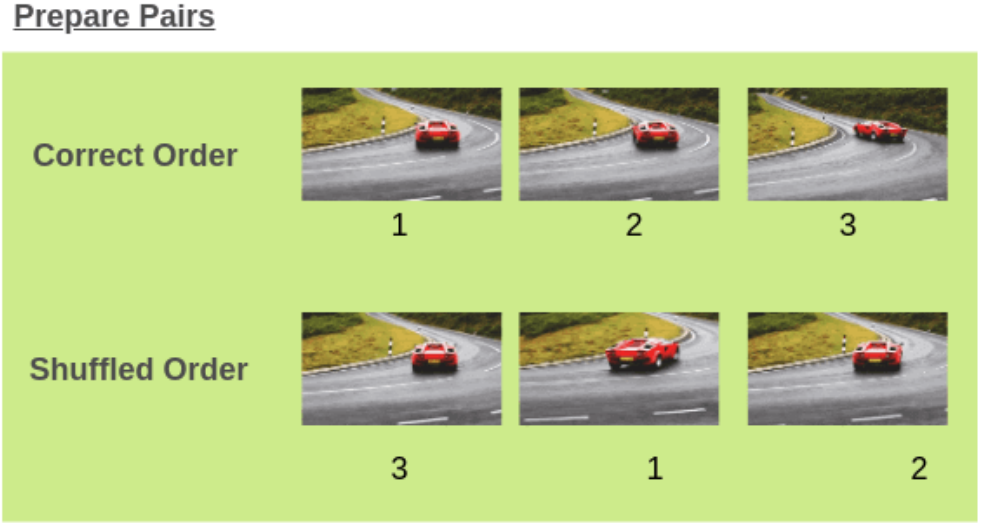
Video-based Approaches

- *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

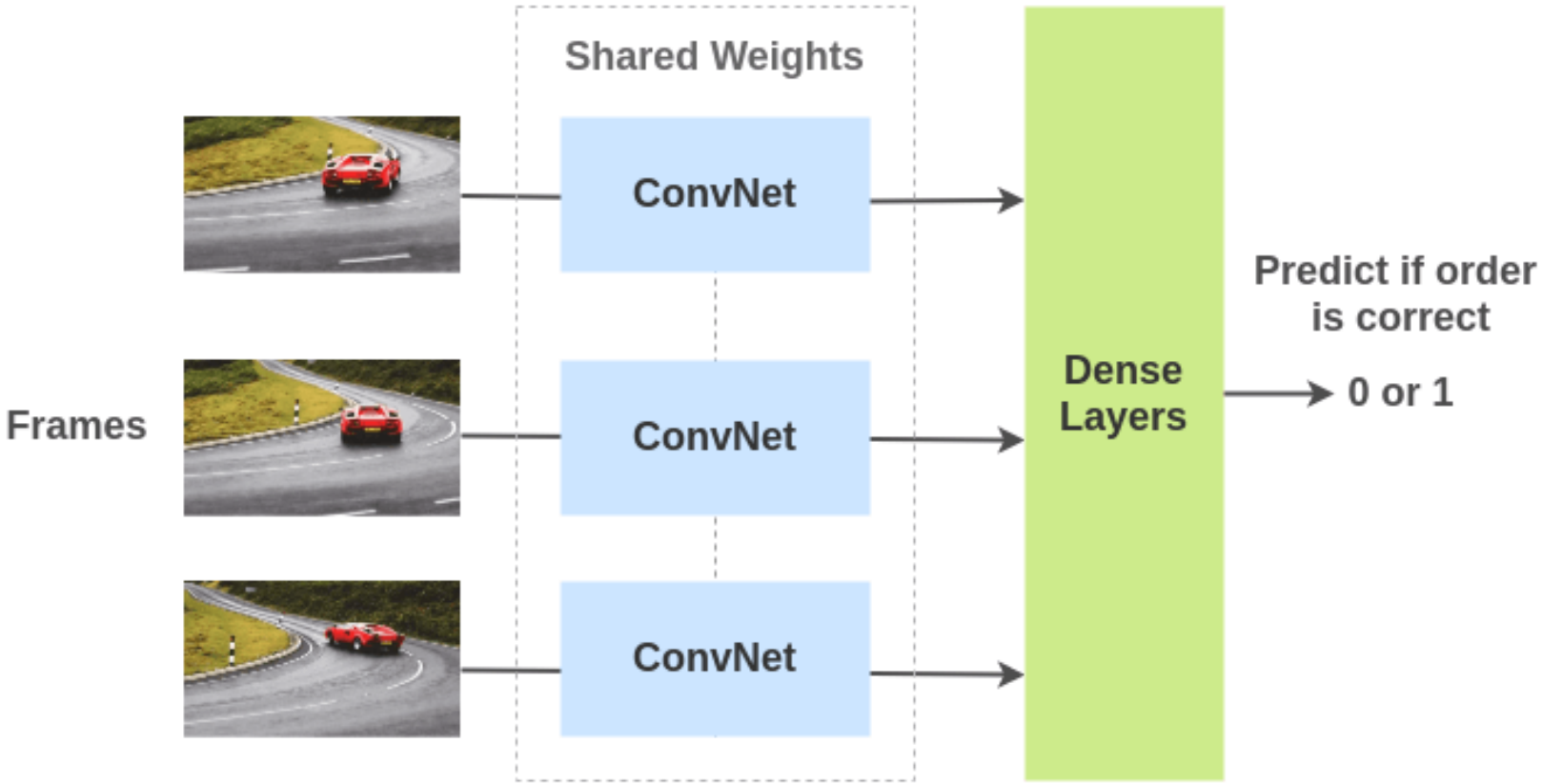




Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Frame Ordering

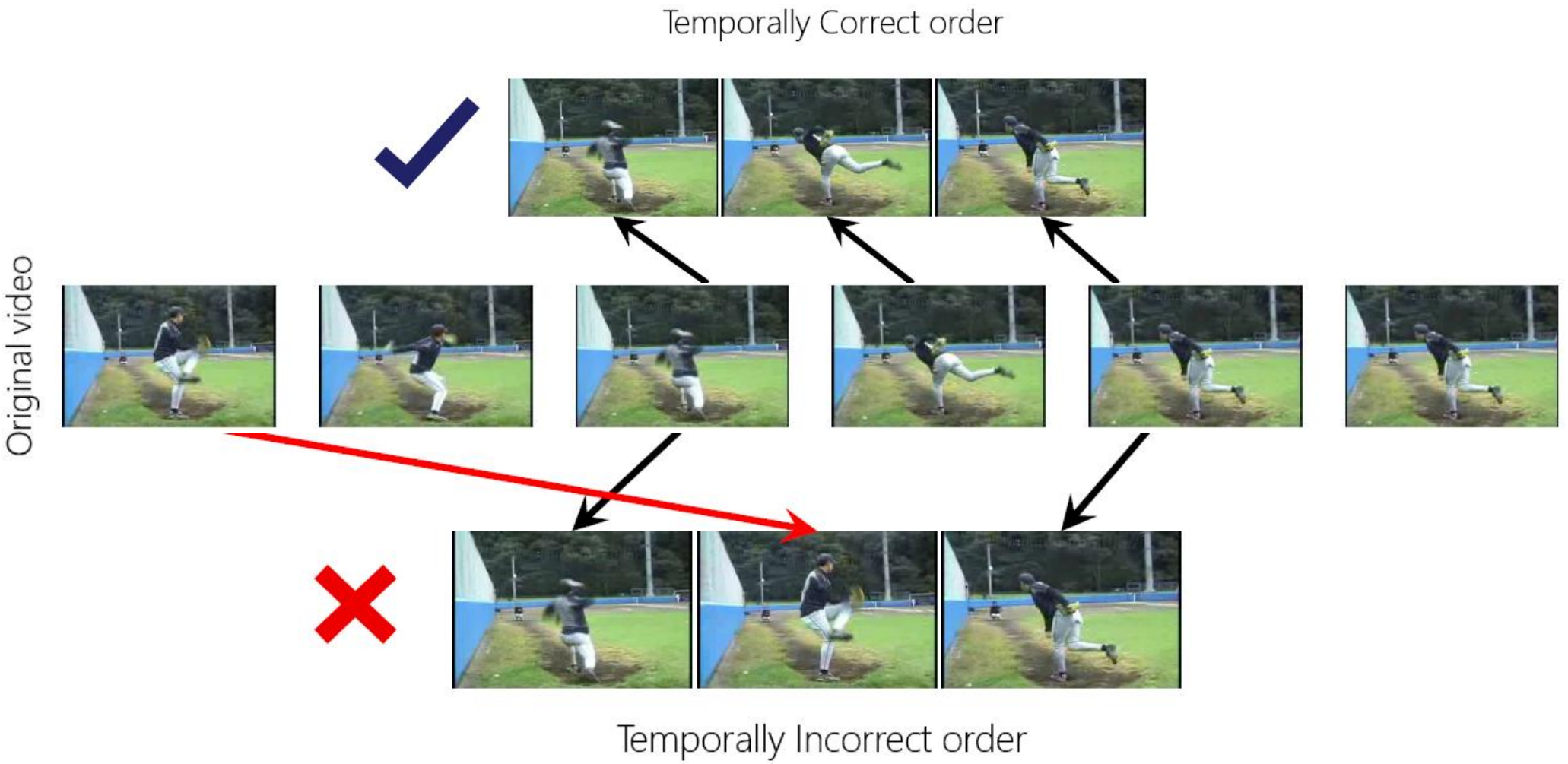
- The model employs ConvNets with shared weights
  - The output is a binary prediction on whether the frames are in the correct order or not



Picture from: Amit Chaudhary – The Illustrated Self-Supervised Learning

Frame Ordering

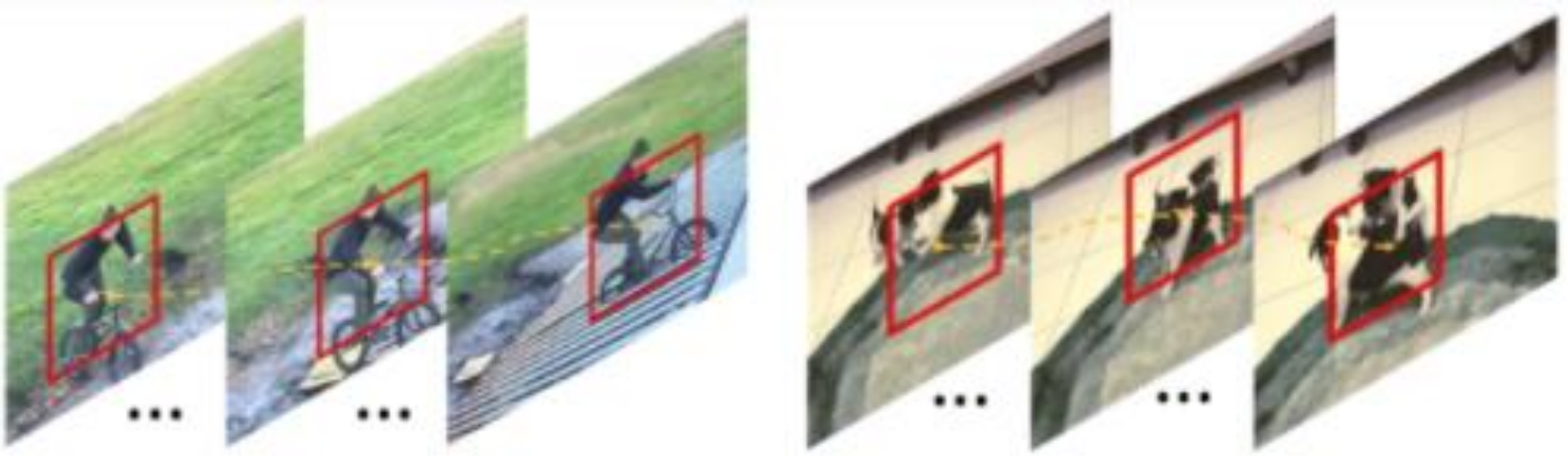
- Example



Picture from: Andrew Zisserman – Self-Supervised Learning

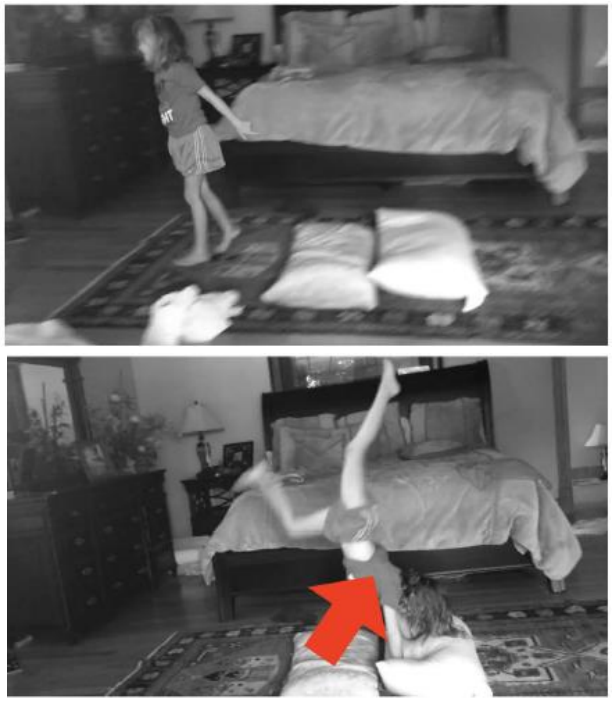
Tracking Moving Objects

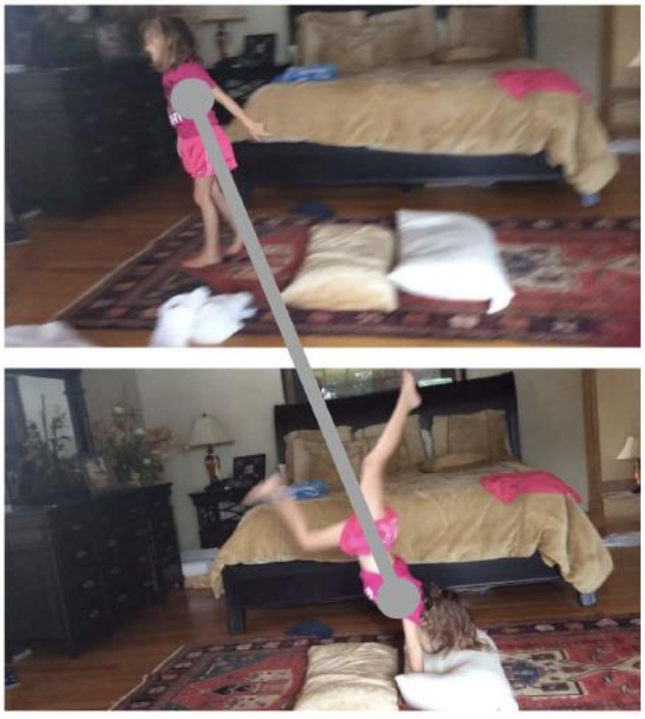
- *Tracking moving objects*
  - Wang (2015) Unsupervised Learning of Visual Representations using Videos
- Training data: videos of moving objects
- Pretext task: predict the location of a patch with a moving object across frames
  - An optical flow approach based on a SURF feature extractor is used for matching feature points across video frames
  - A ConvNet model is designed for predicting the patch location in the next frames
  - The model learns representations by minimizing the distance (in the latent space) to the tracked patch across the frames



Video Colorization

- *Video colorization* or *temporal coherence of color*
  - Vondrick (2018) Tracking Emerges by Colorizing Videos
- Training data: pairs of color and grayscale videos of moving objects
- Pretext task: predict the color of moving objects in other frames
  - The learned representations are useful for downstream segmentation, object tracking, and human pose estimation tasks

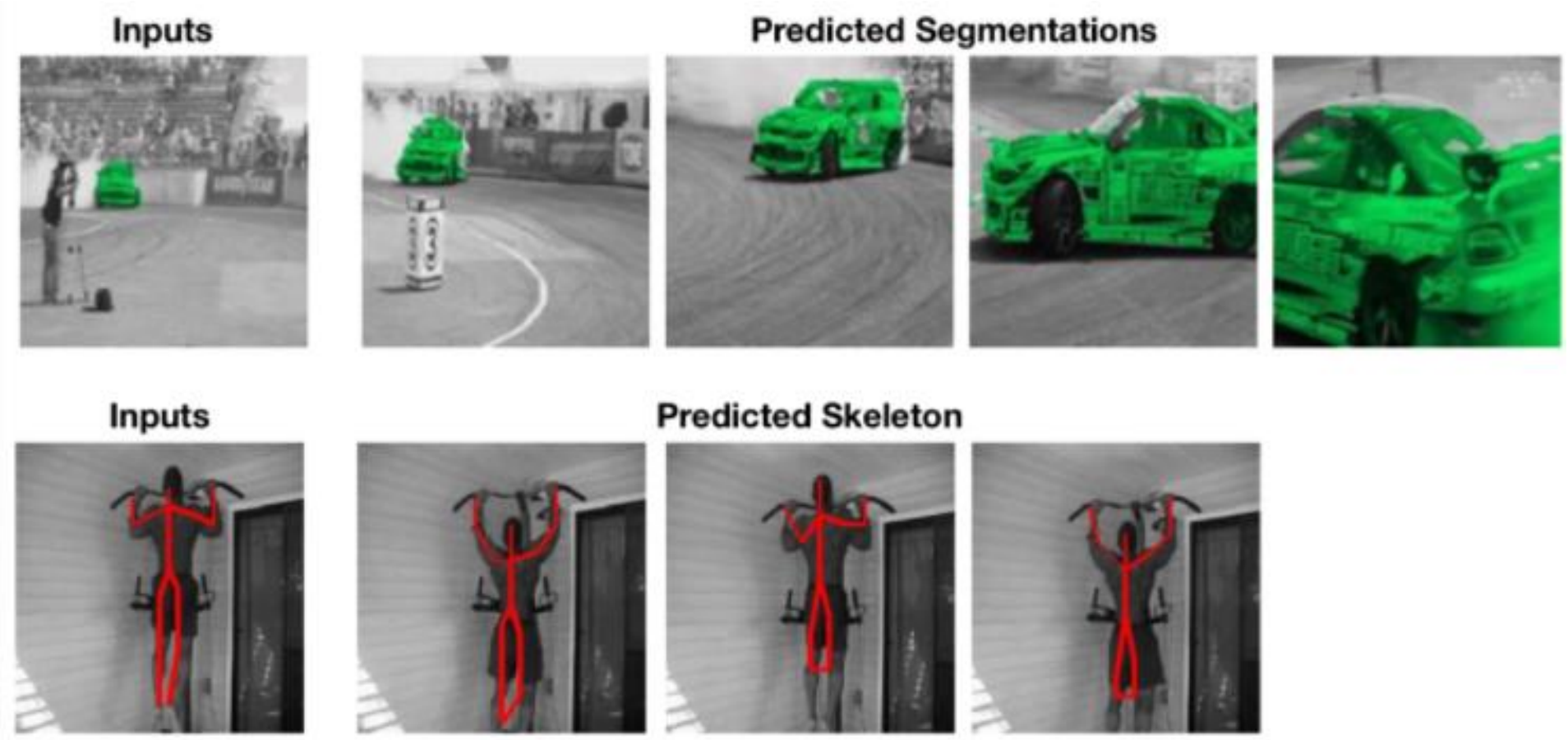




Picture from: Andrew Zisserman – Self-Supervised Learning

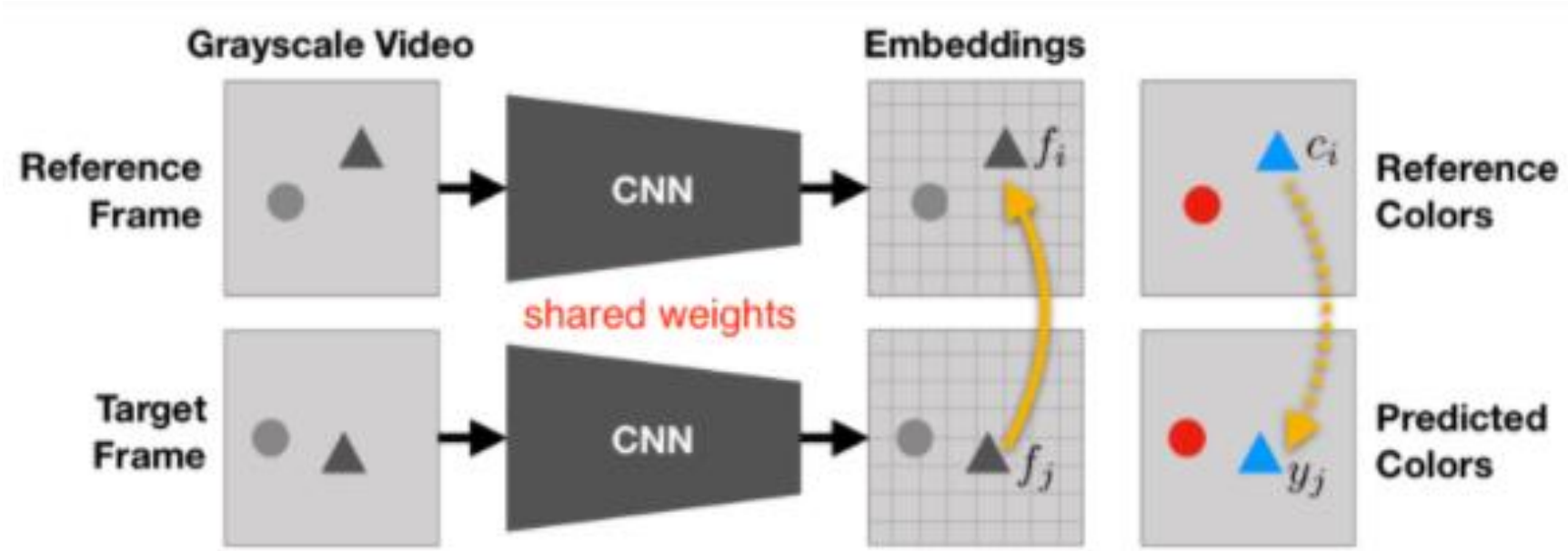
Video Colorization

- Inputs and predicted outputs for video segmentation and human skeleton pose prediction in videos



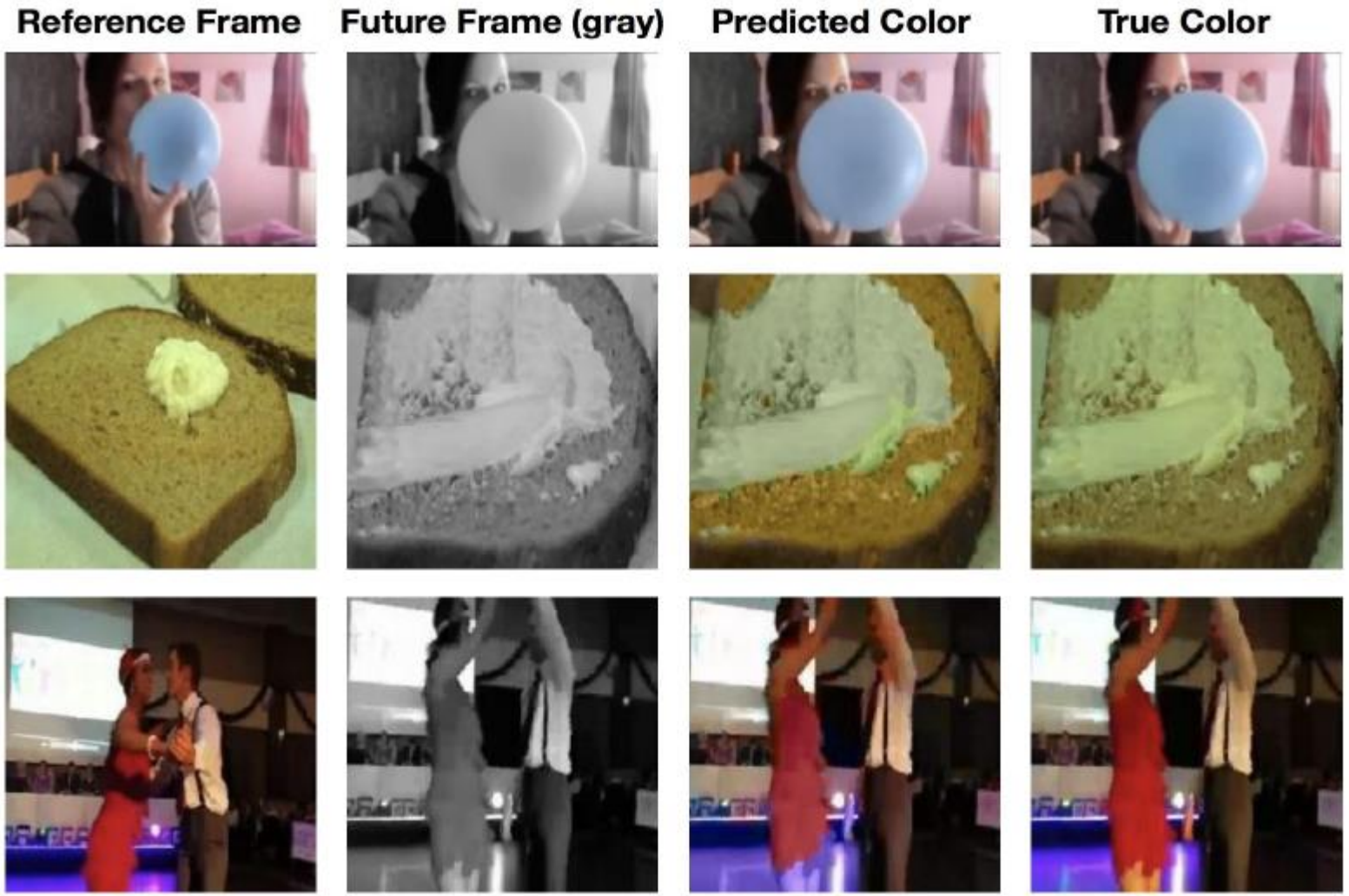
Video Colorization

- The goal is to copy colors from a reference frame in color to another target frame in grayscale
- The model needs to employ the temporal consistency of the objects across frames in order to learn how to apply colors to the grayscale frames
  - This includes tracking correlated pixels in different frames
  - The reference and target frames should not be too far apart in time



Video Colorization

- Video colorization examples



NLP

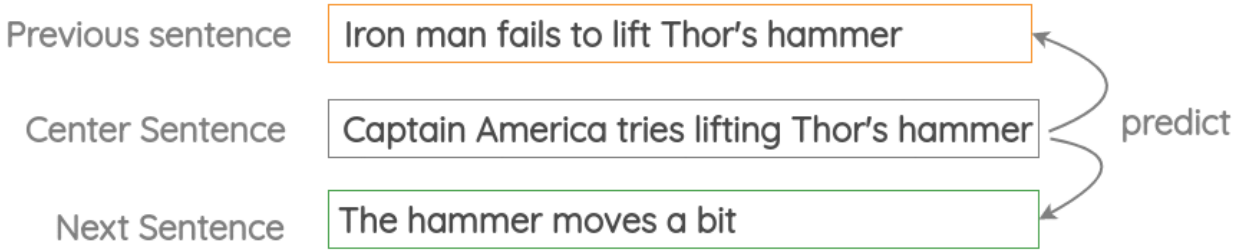
- Self-supervised learning has driven the recent progress in the ***Natural Language Processing*** (NLP) field
  - Models like ELMO, BERT, RoBERTa, ALBERT, Turing NLG, GPT-3 have demonstrated immense potential for automated NLP
- Employing various pretext tasks for leaning from raw text produced rich 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

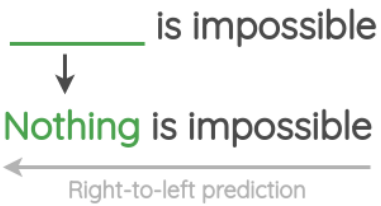
Picture from: Amit Chaudhary – Self-Supervised Representation Learning in NLP

NLP

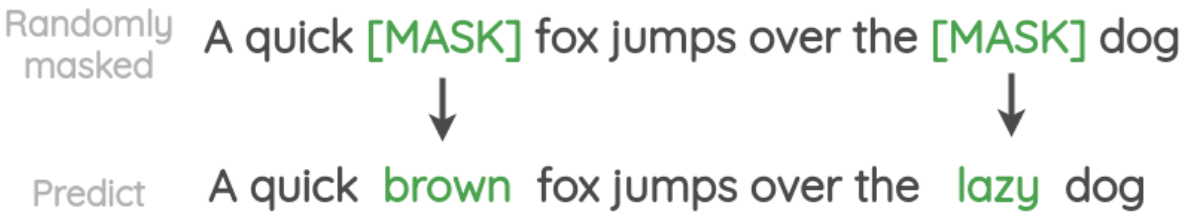
- Pretext tasks in NLP:
  - From three consecutive sentences, predict the previous and the next sentence, given the center sentence



  - Predict the previous or the next word, given surrounding words



  - Predict randomly masked words in sentences



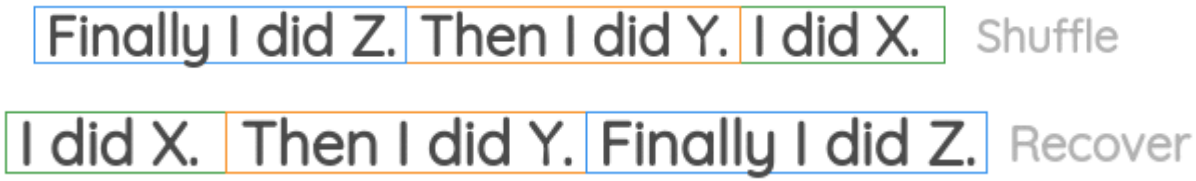
Picture from: Amit Chaudhary – Self-Supervised Representation Learning in NLP

NLP

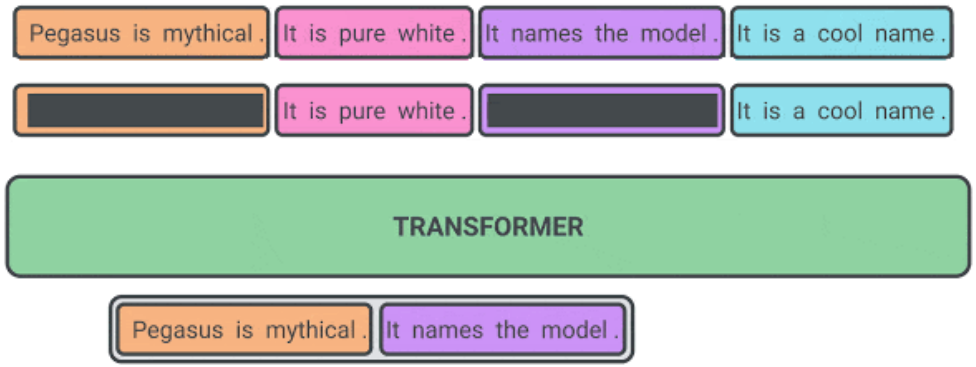
- Pretext tasks in NLP:
  - Predict if the ordering of two sentences is correct

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



  - Predict masked sentences in a document



Picture from: Amit Chaudhary – Self-Supervised Representation Learning in NLP

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, link: Lil'Log
2. Pieter Abbeel, UC Berkley, CS294-158 Deep Unsupervised Learning, Lecture 7 – Self-Supervised Learning
3. Amit Chaudhary – The Illustrated Self-Supervised Learning, link
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, link
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, link