Self-Supervised Learning

Dongrui Wu

School of Artificial Intelligence and Automation

Huazhong University of Science and Technology

drwu@hust.edu.cn

Outline

- Self-Supervised Learning (SSL)
- Early works
- Methods for framing SSL
- Pretext tasks
- Large Language Models
- Discussion
- Conclusions

Self-Supervised Learning (SSL)

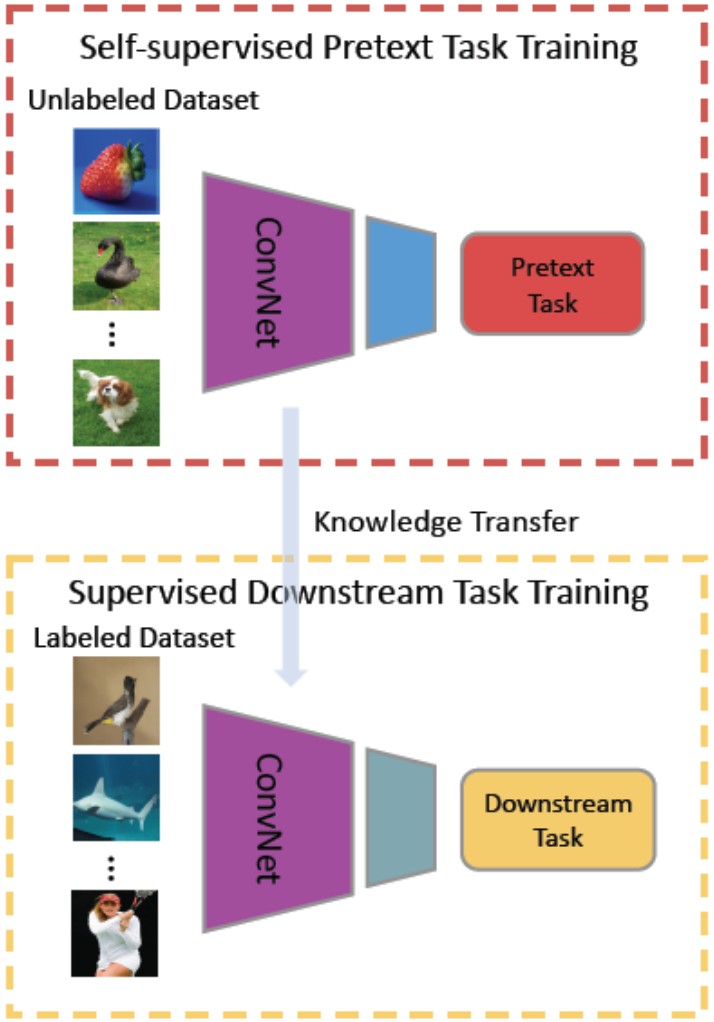
- **Goal:** Learn a good data representation from unlabeled data.
- **Motivation:** Construct supervised learning tasks out of unsupervised datasets.

- Why use self-supervised learning?
  - ✓ Data labeling is expensive, thus high-quality labeled dataset is limited.
  - ✓ Learning good representation makes it easier to transfer useful information to a variety of downstream tasks.

Self-Supervised Learning

- Self-supervised learning tasks are also known as **pretext tasks**.

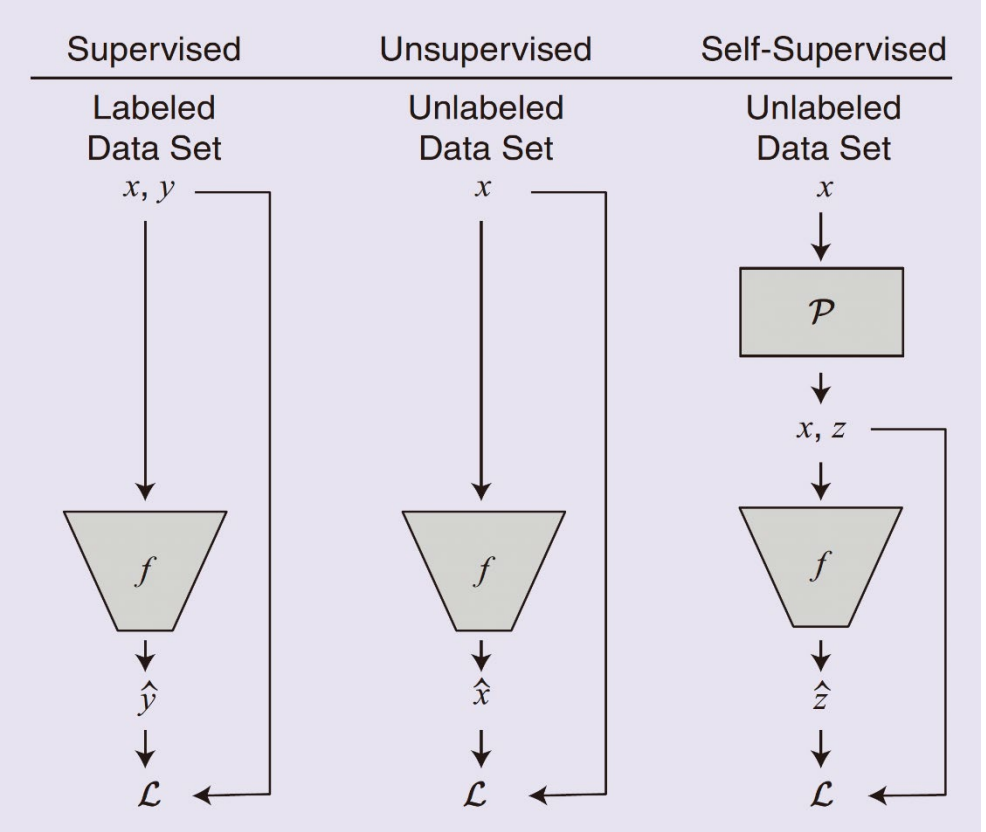
- **Self-supervised learning example:**
  - **Pretext task**: Train a model to predict the rotation degree of images (labeling is not required)
  - **Downstream task**: Use transfer learning to fine-tune the learned model from the pretext task with very few labeled examples



L. Jing, et. al., "Self-supervised visual feature learning with deep neural networks: A survey." *IEEE Trans. on Pattern Analysis and Machine Intelligence* vol. 43, no. 11, pp. 4037-4058, 2020

SSL vs Supervised / Unsupervised Learning

- **Supervised Learning :** Learning with **labeled data**
  - Collect a large dataset, manually label the data, train a model, deploy
  - Learned feature representations from large labeled datasets are often transferred via pre-trained models to smaller domain-specific datasets
- **Unsupervised Learning :** Learning with **unlabeled data**
  - Discover patterns in data via clustering, density estimation, dimensionality reduction, etc.



- **Self-supervised Learning :** Representation learning with **unlabeled data**
  - Learn useful feature representations from unlabeled data ***through pretext tasks***
  - Self-supervised: Create its own supervision (i.e., without supervision, without labels)
  - Self-supervised learning is one category of unsupervised learning

L. Ericsson, et, al., "Self-Supervised Representation Learning: Introduction, advances, and challenges," in IEEE Signal Processing Magazine, vol. 39, no. 3, pp. 42-62, 2022

SSL vs Unsupervised Learning/ Transfer Learning/ Data Augmentation

- **Self-supervised Learning versus Unsupervised Learning**
  - Self-supervised learning
    - 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

- **Self-supervised Learning versus Transfer Learning**
  - Self-supervised learning is a type of transfer learning implemented in an unsupervised manner
  - Transfer learning is often implemented in a supervised manner

- **Self-supervised Learning versus Data Augmentation**
  - For Self-supervised learning, image rotation of shifting are used for feature learning in raw unlabeled data
  - Data augmentation is often used as a regularization in **supervised** learning

What's Possible with Self-Supervised Learning?

"The birthplace of the American national anthem" [MASK]

Answering questions

"The birthplace of the American national anthem, "The Star-Spangled Banner," lies in Baltimore, Maryland."

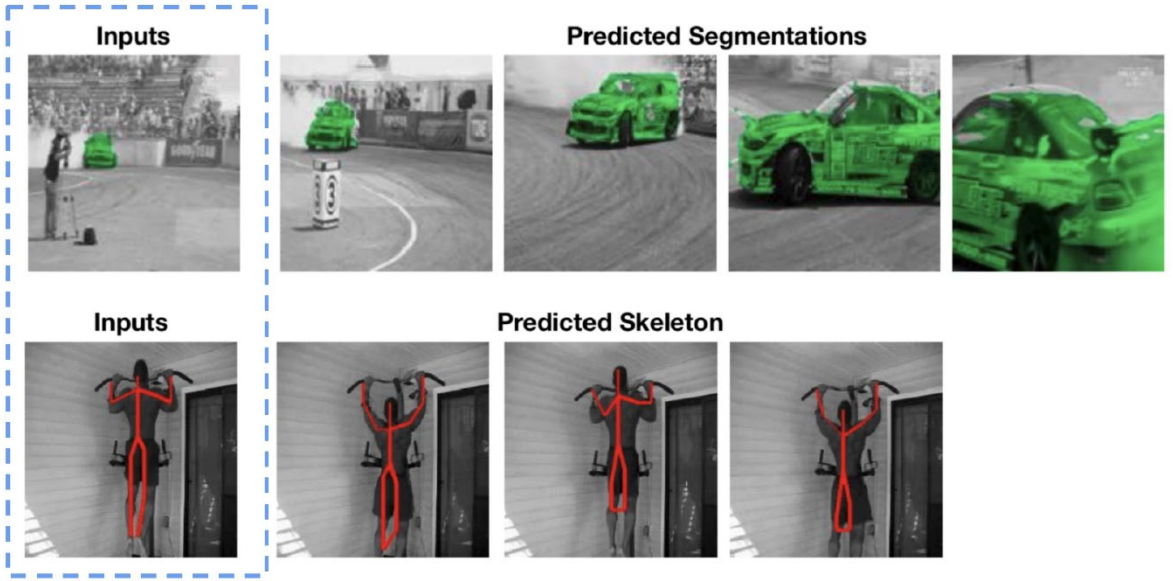
"We want to decide whether the sentiment of the review is "positive" or "negative".

Review: "While this restaurant is popular on Google, I absolutely disliked it".

The sentiment of this review is" [MASK]

Sentiment classification

"...is negative."



Video segmentation and unlabeled visual region tracking

Outpainting

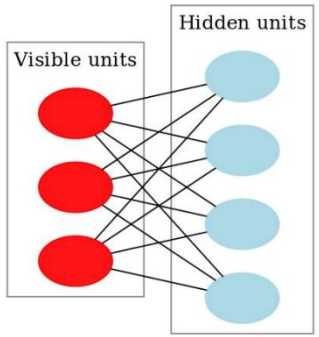
https://openai.com/blog/dall-e-introducing-outpainting/

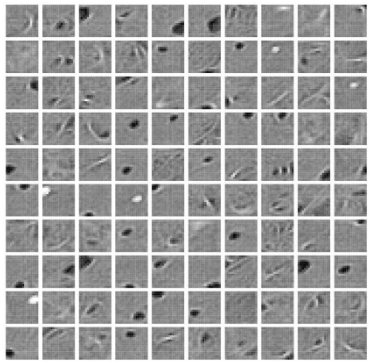
Outline

- Self-Supervised Learning (SSL)
- **Early works**
- Methods for framing SSL
- Pretext tasks
- Large Language Models
- Discussion
- Conclusions

Early Works of Self-Supervised Learning (SSL)

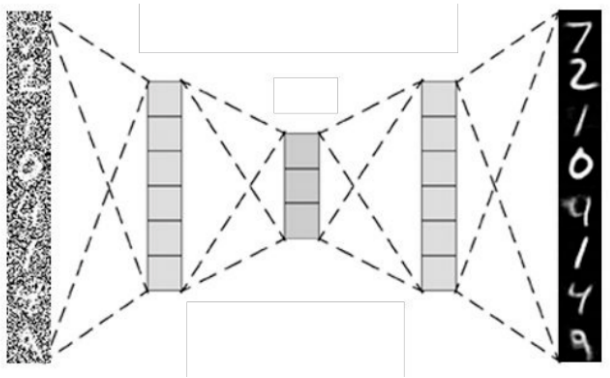
Restricted Boltzmann Machines





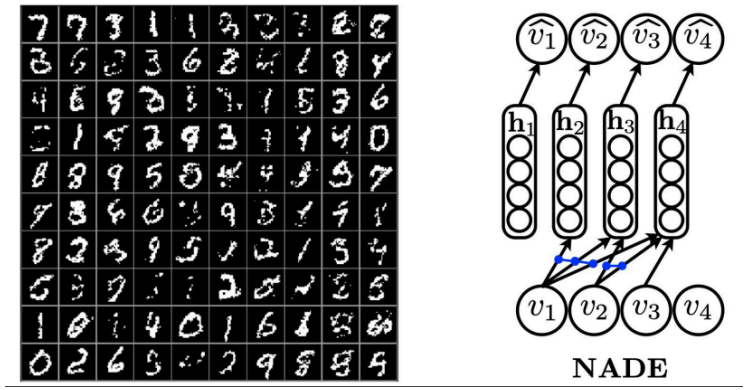
Contrastive divergence (Hinton 2000; Hinton 2002)

Autoencoders



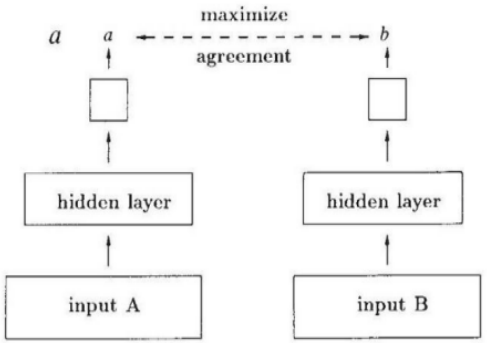
Denoising Autoencoder (Vincent et al. 2008)

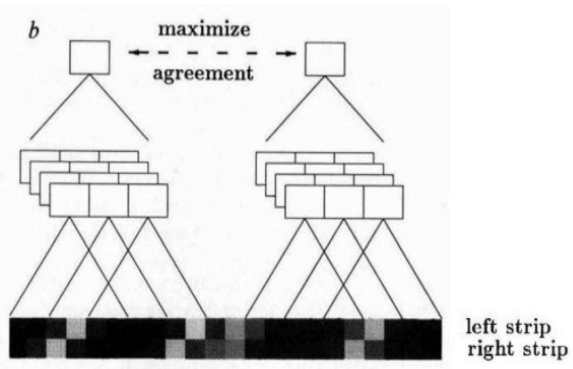
Autoregressive Modeling



Neural Autoregressive Distribution Estimator (Larochelle et al. 2011)

Siamese networks





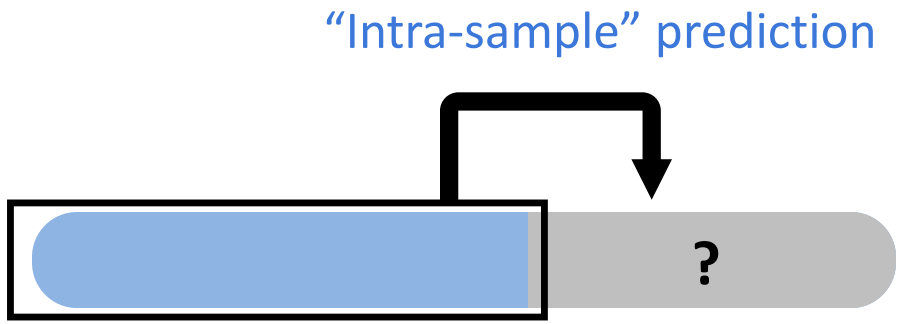
Self-organizing neural networks (Becker & Hinton 1992)

Outline

- Self-Supervised Learning (SSL)
- Early works
- **Methods for framing SSL**
  - Self-prediction
  - Contrastive learning
- Pretext tasks
- Large Language Models
- Discussion
- Conclusions

Methods for Framing SSL: Self-prediction

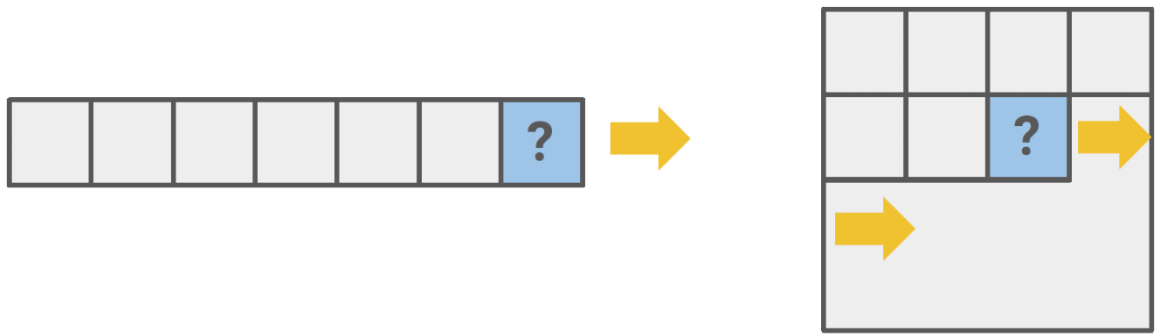
Self-prediction: Construct prediction tasks within every individual data sample, to predict a part of the data from the rest while pretending we don't know that part.



The part to be predicted pretends to be missing

Self-prediction: Autoregressive Generation

Autoregressive Model: Predict future behavior based on past behavior. Any data that come with an innate sequential order can be modeled with regression.

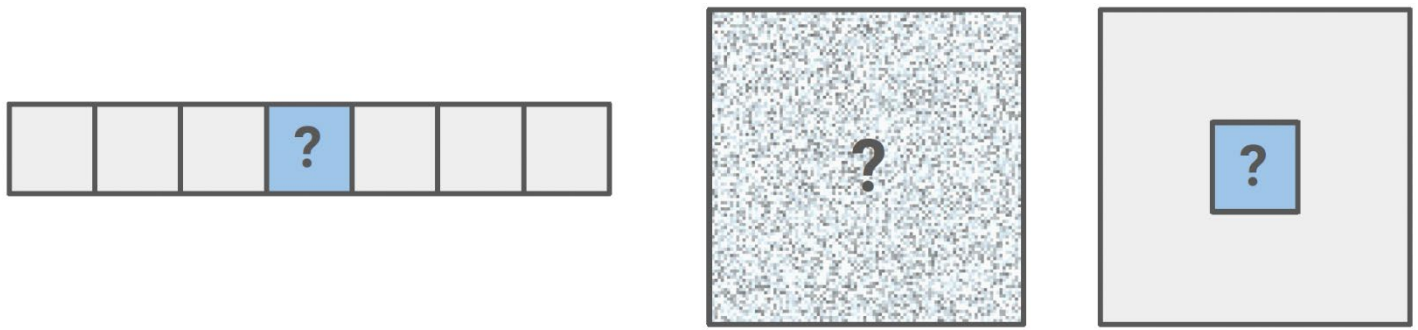


Examples:

- Audio (WaveNet, WaveRNN)
- Autoregressive language modeling (GPT, XLNet)
- Images in raster scan (PixelCNN, PixelRNN, iGPT)

Self-prediction: Masked Generation

Masked Generation: Mask a random portion of information and pretend it is missing, irrespective of the natural sequence. The model learns to predict the missing portion given other unmasked information.

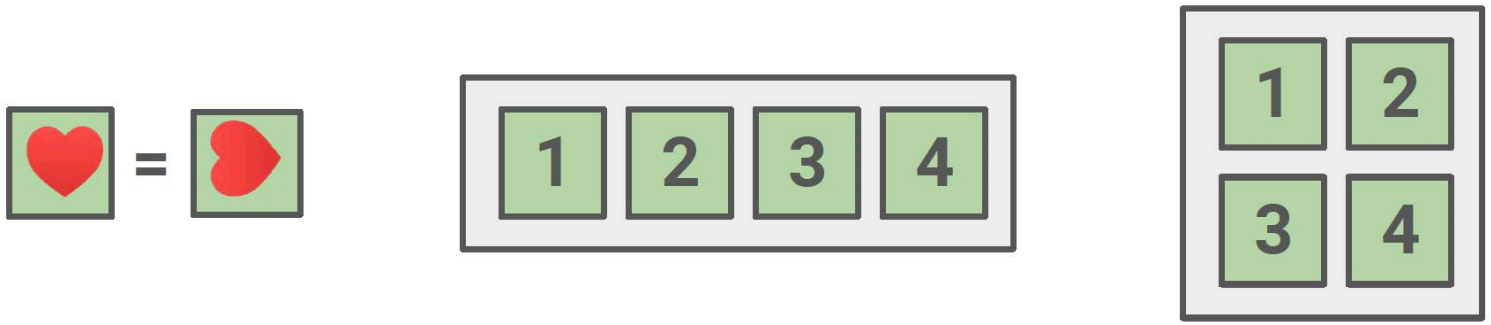


Examples:

- Masked language modeling (BERT)
- Images with masked patch (denoising autoencoder, context autoencoder, colorization)

Self-prediction: Innate Relationship Prediction

Innate Relationship Prediction: Some transformation (e.g. segmentation, rotation) of one data sample should maintain the original information or follow the desired innate logic.

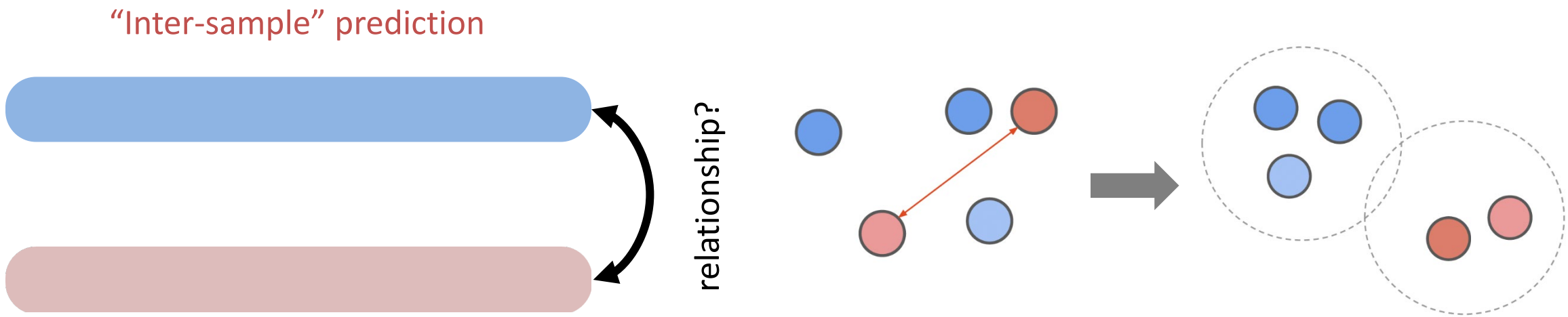


Examples:

- Order of image patches (e.g., relative position, jigsaw puzzle)
- Image rotation
- Counting features across patches

Methods for Framing SSL: Contrastive Learning

Contrastive learning : learn such an embedding space in which **similar** sample pairs stay **close** to each other while **dissimilar** ones are **far** apart.



The multiple samples can be selected from the dataset based on some known logics (e.g. the order of words / sentences), or fabricated by altering the original version.

Contrastive Learning: Inter-sample Classification

Inter-sample classification : Given both similar ("positive") and dissimilar ("negative") candidates, to identify which ones are similar to the anchor data point is a classification task.

To construct a set of data point candidates:

- The original input and its distorted version
- Data that captures the same target from different views

Common loss functions:

- Contrastive loss (Chopra et al. 2005)
- Triplet loss (Schroff et al. 2015; FaceNet)
- Lifted structured loss (Song et al. 2015)
- Multi-class n-pair loss (Sohn 2016)
- Noise contrastive estimation ("NCE"; Gutmann & Hyvarinen 2010)
- InfoNCE (van den Oord, et al. 2018)
- Soft-nearest neighbors loss (Salakhutdinov & Hinton 2007, Frosst et al. 2019)

Contrastive Learning: Common Loss Functions

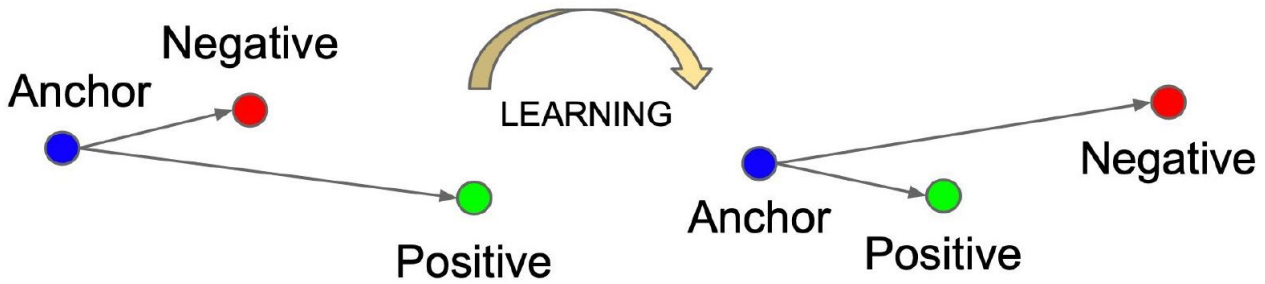
Contrastive loss: Encodes data into an embedding vector such that examples from the same class have similar embeddings and samples from different classes have different ones.

$$\mathcal{L}_{\text{cont}}(\mathbf{x}_i, \mathbf{x}_j, \theta) = \mathbb{1}[y_i = y_j] \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2^2 + \mathbb{1}[y_i \neq y_j] \max(0, \epsilon - \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2)^2$$

minimize maximize

Triplet loss: learns to minimize the distance between the anchor x and positive x+ and maximize the distance between the anchor x and negative x- at the same time.

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$



Contrastive Learning: Common Loss Functions

N-pair loss: Generalizes triplet loss to include comparison with multiple negative samples.

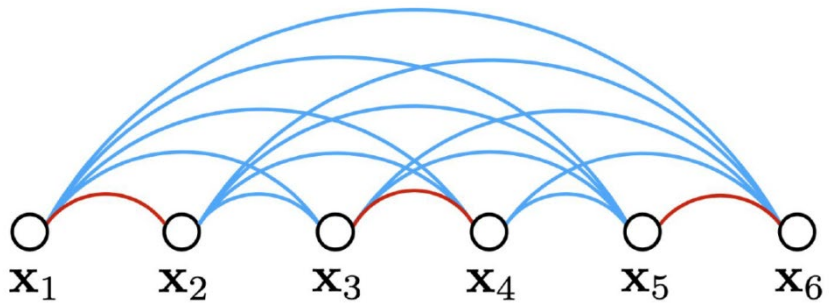
$$\mathcal{L}_{\text{N-pair}}(\mathbf{x}, \mathbf{x}^+, \{\mathbf{x}_i^-\}_{i=1}^{N-1}) = \log\left(1 + \sum_{i=1}^{N-1} \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-) - f(\mathbf{x})^\top f(\mathbf{x}^+))\right)$$

$$= -\log \frac{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+))}{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+)) + \sum_{i=1}^{N-1} \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-))}$$

Lifted structured loss: utilizes all the pairwise edges within one training batch for better computational efficiency.

$$\mathcal{L}_{\text{struct}}^{(ij)} = D_{ij} + \log\left(\sum_{(i,k)\in\mathcal{N}} \exp(\epsilon - D_{ik}) + \sum_{(j,l)\in\mathcal{N}} \exp(\epsilon - D_{jl})\right)$$

where $D_{ij} = \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2$



Contrastive Learning: Common Loss Functions

Noise Contrastive Estimation (NCE): running logistic regression to tell apart the target data from noise.

target sample distribution

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[\log\sigma(\ell_\theta(\mathbf{x}_i)) + \log(1-\sigma(\ell_\theta(\tilde{\mathbf{x}}_i)))\right], \text{ where } \ell_\theta(\mathbf{u}) = \log\frac{p_\theta(\mathbf{u})}{q(\mathbf{u})} = \log p_\theta(\mathbf{u}) - \log q(\mathbf{u})$$

noise distribution

InfoNCE: using categorical cross-entropy loss to identify the positive sample amongst a set of unrelated noise samples

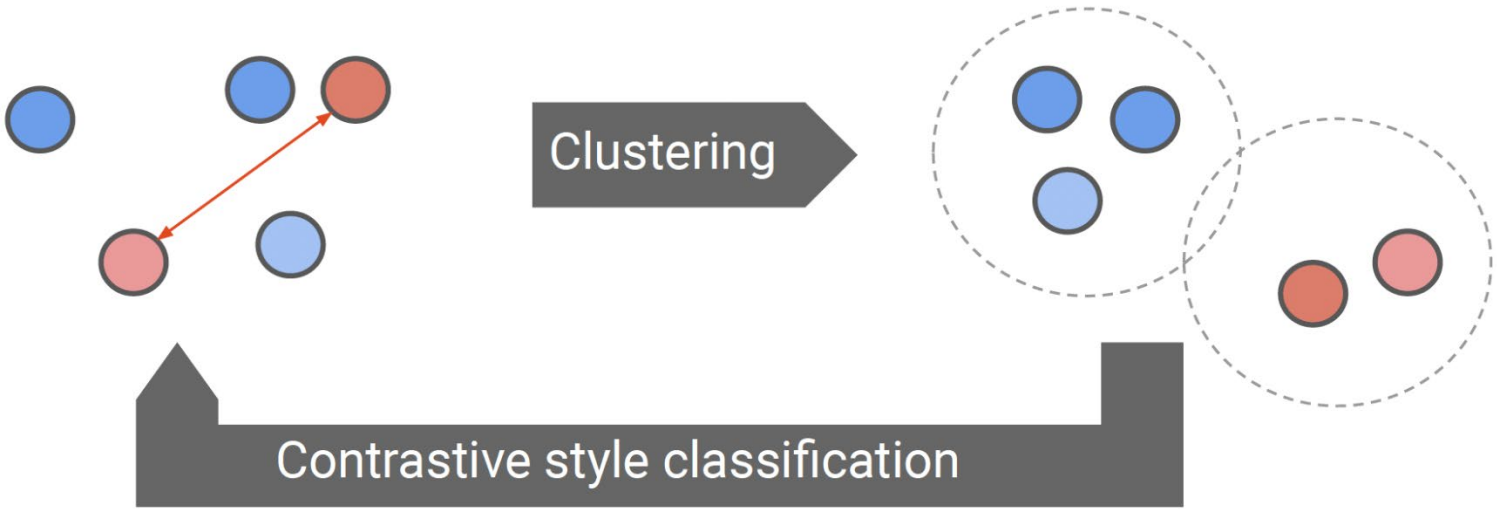
$$p(C=\text{pos}|X,\mathbf{c}) = \frac{f(\mathbf{x}_{\text{pos}},\mathbf{c})}{\sum_{j=1}^{N} f(\mathbf{x}_j,\mathbf{c})}$$, where the density function is $$f(\mathbf{x},\mathbf{c}) \propto \frac{p(\mathbf{x}|\mathbf{c})}{p(\mathbf{x})}$$

Soft-Nearest Neighbors Loss: extending the loss function to include multiple positive samples given known labels

$$\mathcal{L}_{\text{snn}} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\sum_{i\neq j, y_i=y_j, j=1,\ldots,B}\exp(-f(\mathbf{x}_i,\mathbf{x}_j)/\tau)}{\sum_{i\neq k, k=1,\ldots,B}\exp(-f(\mathbf{x}_i,\mathbf{x}_k)/\tau)}$$

Contrastive Learning: Feature Clustering

Feature Clustering : Find similar data samples by clustering them with learned features.



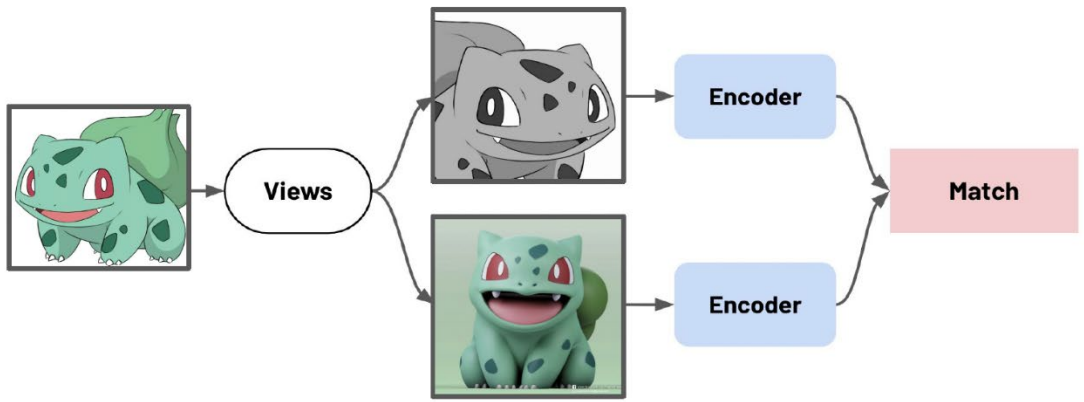
Use clustering algorithms to assign pseudo labels to samples such that we can run intra-sample contrastive learning.

Examples:

- DeepCluster (Caron et al 2018)
- InterCLR (Xie et al 2021)

Contrastive Learning: Multiview Coding

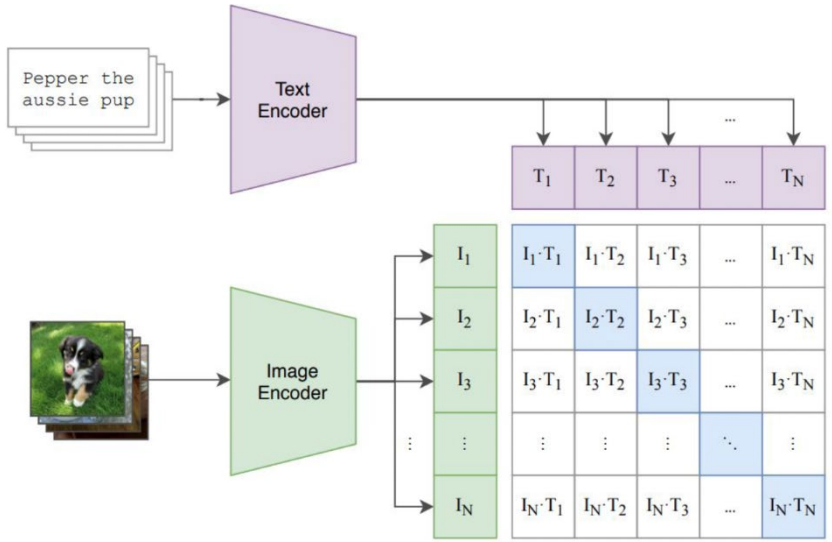
- Apply the InfoNCE objective to two or more different **views** of input data:



Examples:

- AMDIM (Bachman et al. 2019)
- Contrastive multiview coding (Tian et al. 2019)

- Apply the InfoNCE objective to two or more different **modalities** of input data:



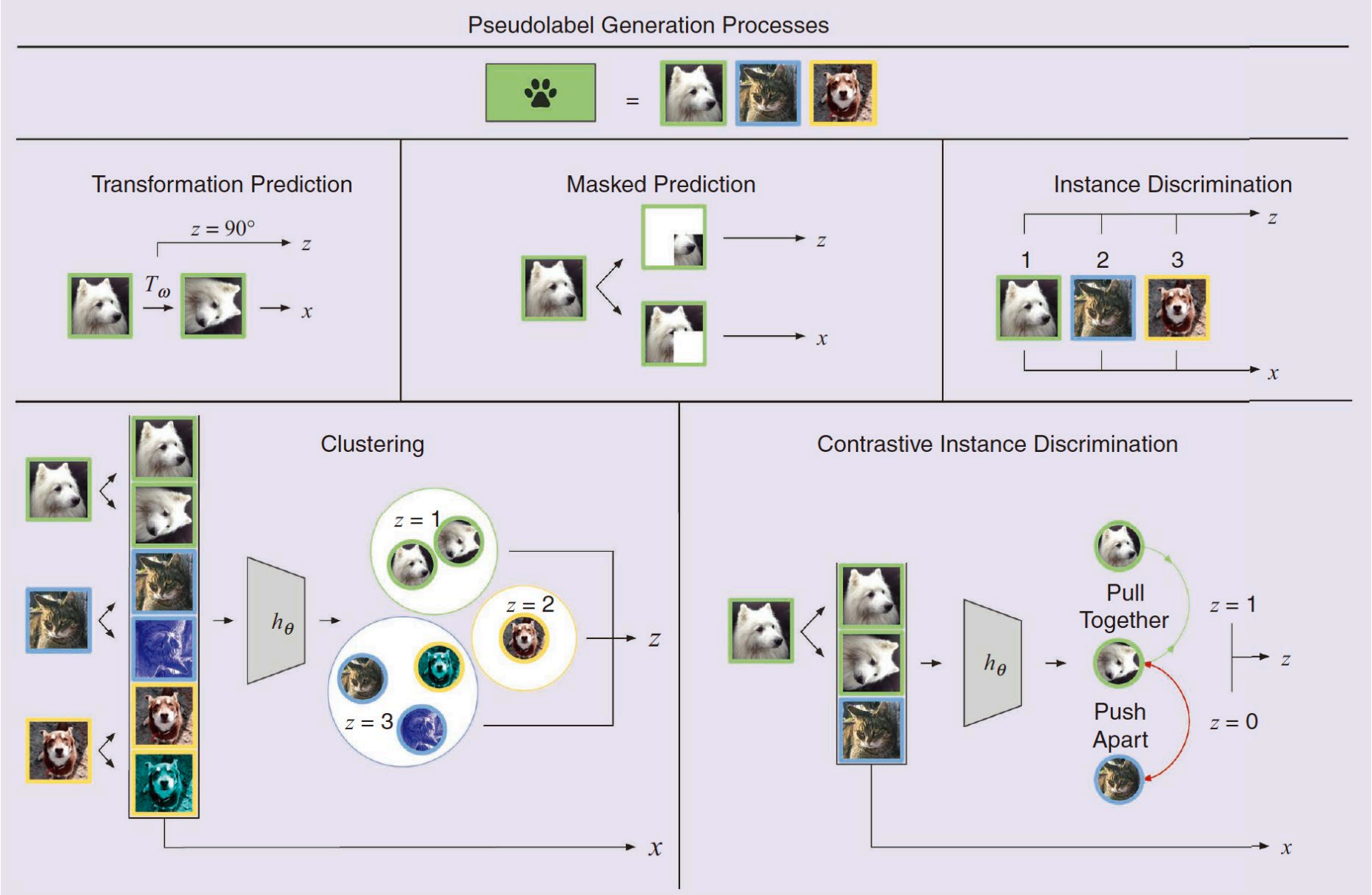
Examples:

- CLIP (Radford et al. 2021)
- ALIGN (Jia et al. 2021)

Outline

- Self-Supervised Learning (SSL)
- Early works
- Methods for framing SSL
- **Pretext tasks**
- Large Language Models
- Discussion
- Conclusions

Pretext Tasks

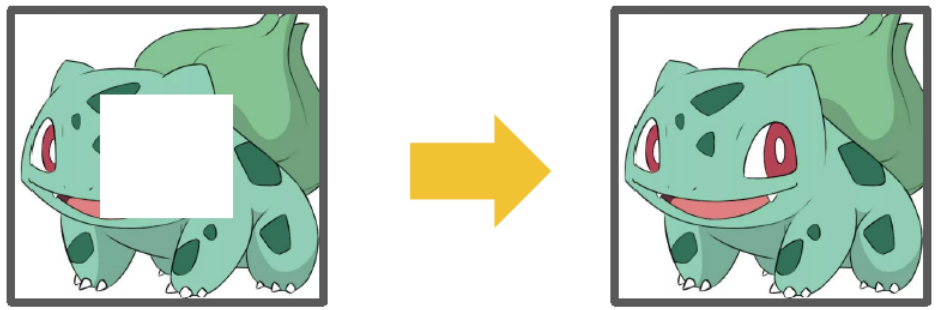


L. Ericsson, et, al., "Self-Supervised Representation Learning: Introduction, advances, and challenges," in IEEE Signal Processing Magazine, vol. 39, no. 3, pp. 42-62, 2022

Masked Prediction

I is the set of indices inside the square-mask region, the pixels in the masked region will correspond to z_i, and the pixels outside the masked region will be xi. Given x_i, z_i, the model can now be trained to minimize, e.g., the following reconstruction loss like mean square error:

$$\theta^* = \underset{\theta,\gamma}{\operatorname{argmin}} \frac{1}{|\mathcal{P}(D_s)|} \sum_{(x_i,z_i)\in\mathcal{P}(D_s)} (k_\gamma(h_\theta(x_i)) - z_i)^2$$

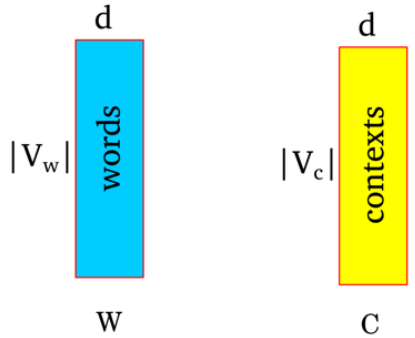


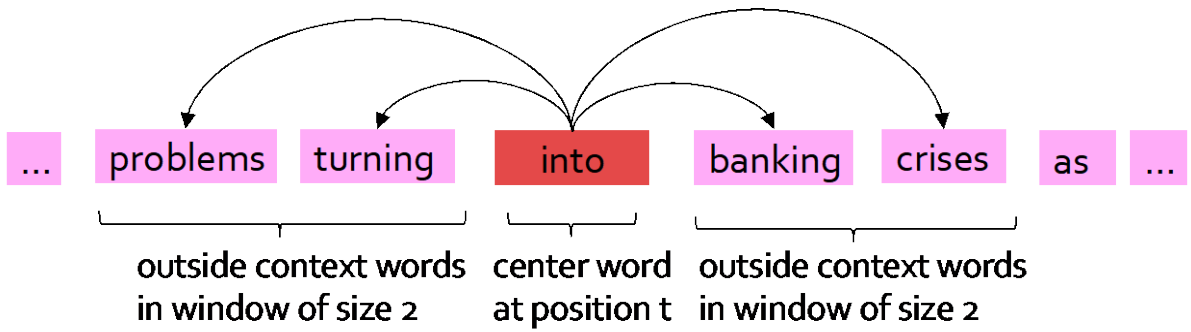
Algorithm 1. The pseudolabel generation process $\mathcal{P}$ for masked prediction.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^M$.
  for i from 1 to M **do**
    Generate indices, I, of elements to remove from $x_i^{(s)}$
    $z_i \leftarrow \{x_{i,j}^{(s)} : j \in I\}$
    $x_i \leftarrow \{x_{i,j}^{(s)} : j \notin I\}$
  end for
Output: $\{x_i, z_i\}_{i=1}^M$.

Word2Vec

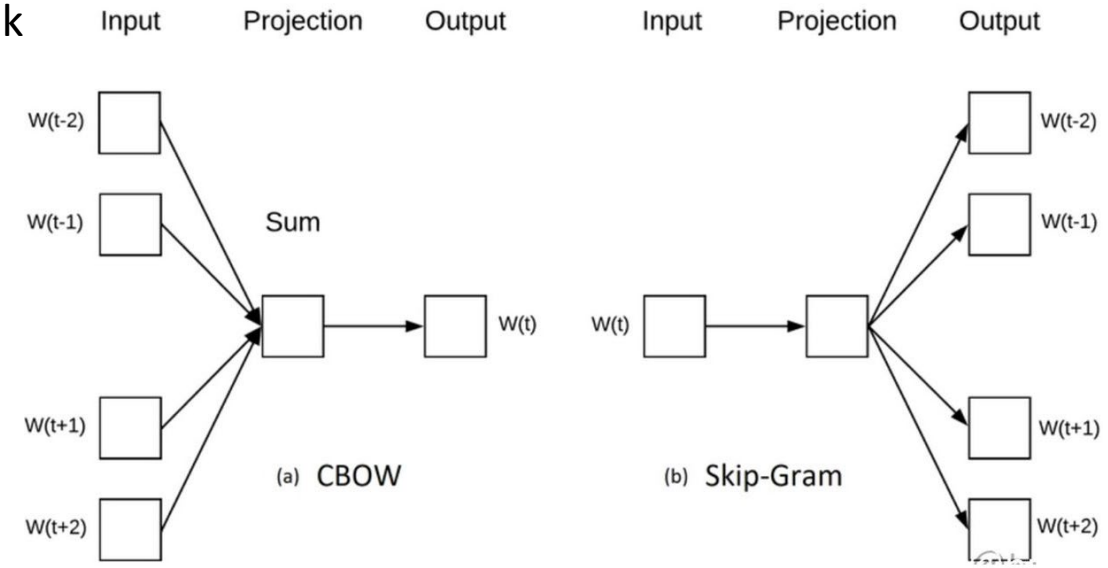
- **Word2Vec**, the most important algorithm in field of nature language processing
- Word embeddings: to map words to a ***d*** dimensional vector.
- Pretext task: find embeddings such that the words that co-occur are close.





Training word embeddings using 2 architectures:

- Continuous bag-of-words (CBOW): predict the center word
- Sk



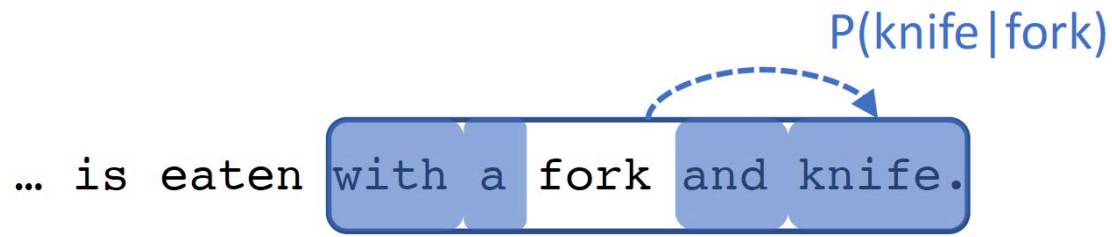
- **Loss function (skip-gram):** For a corpus with T words, minimize the negative log likelihood of the context word w_{t+j} given the center word w_t

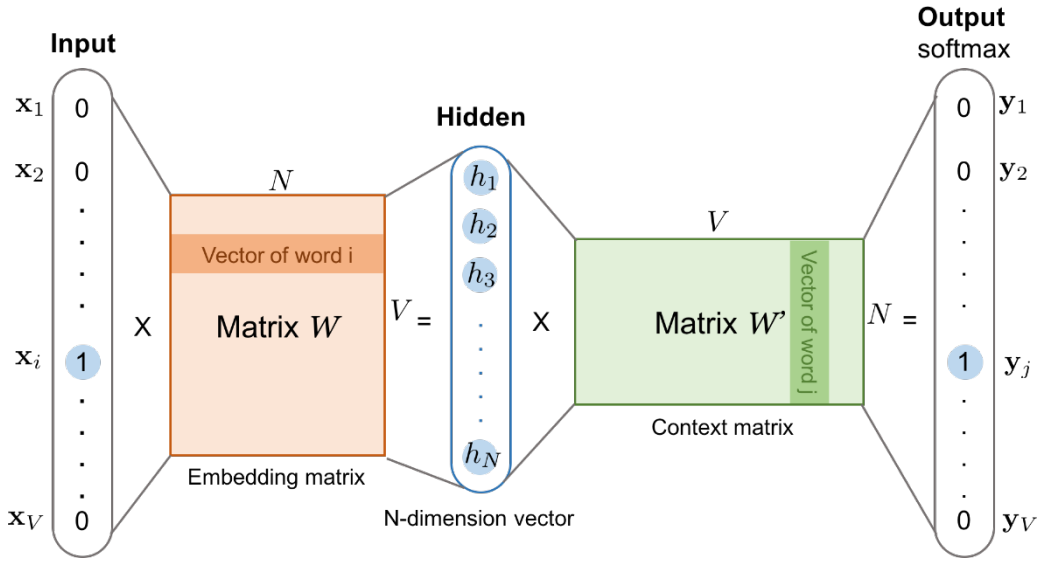
$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P\left(w_{t+j} \mid w_t;\ \theta\right)$$

Context word · Center word · Model parameters · Context window size

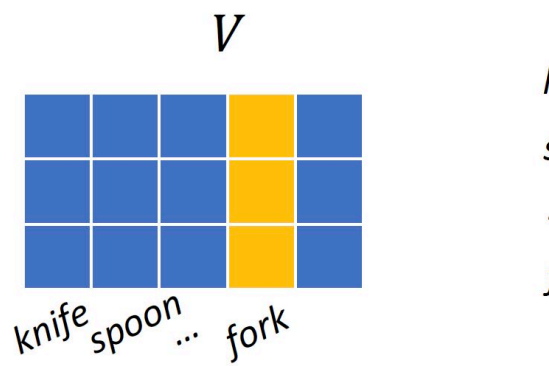
Word2Vec

Example: using the skip-gram method (predict context words), compute the probability of ”knife” given the center word “fork”.

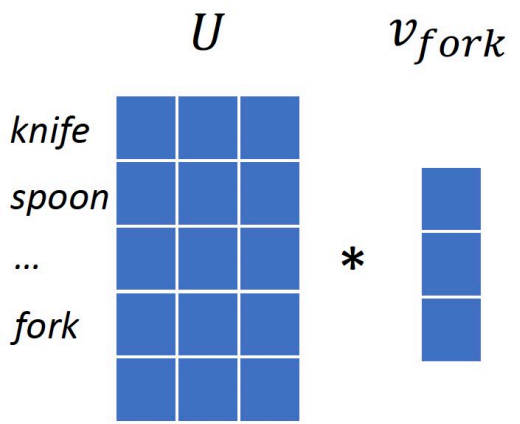




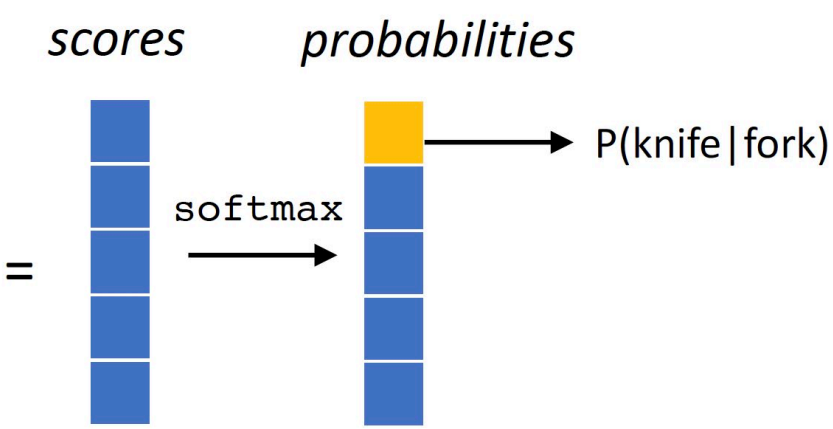
1. Get “fork” word vector v_{fork}



2. Compute scores



3. Convert to probabilities



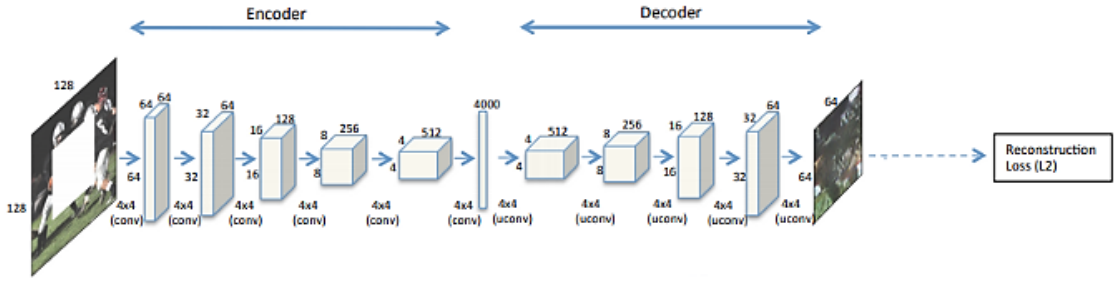
Context Encoder

- **Predict missing pieces**, also known as **context encoders**, or **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

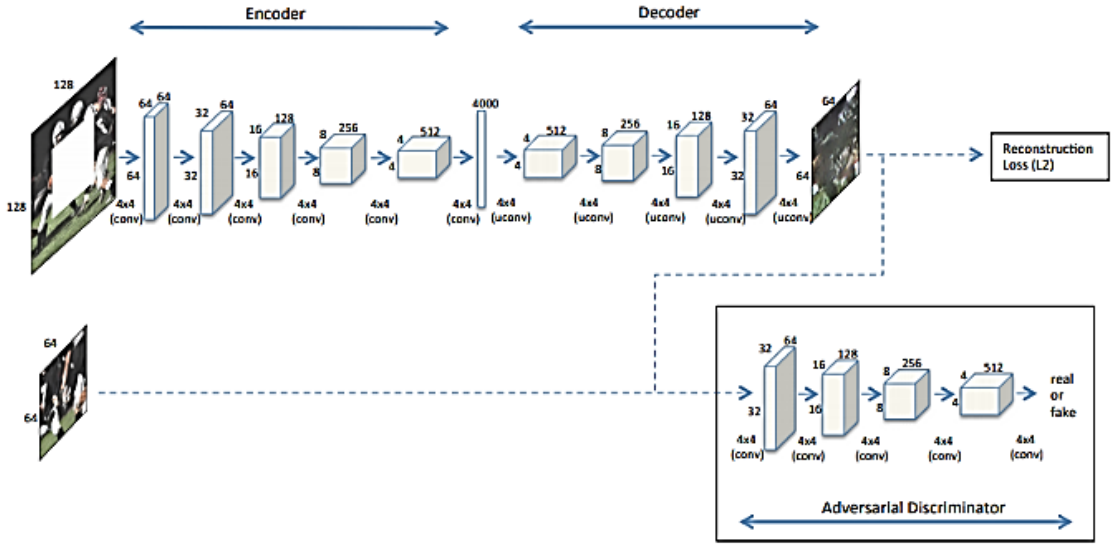
Deepak Pathak et al. Context Encoders: Feature Learning by Inpainting. CVPR 2016

Context Encoder

- 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 $\mathcal{L}_{\text{rec}}$



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

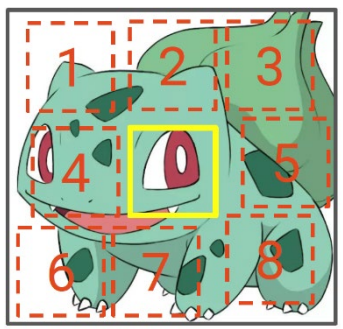


Gidaris, Spyros et al. Unsupervised Representation Learning by Predicting Image Rotations. ICLR 2018

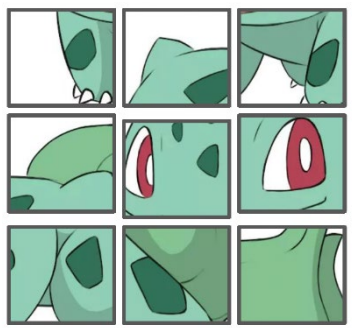
Transformation Prediction

TP methods apply a transformation that maps from canonical to alternative views and train the model to predict which transformation has been applied. Given a raw input in its canonical view, $x_i^{(s)}$, a transformation, T_ω is applied to produce $x_i = T_\omega(x_i^{(s)})$, which is fed into the model. The learning objective can be, e.g., a cross-entropy loss, in the case of categorical transformation parameters.

$$\theta^* = \underset{\theta,\gamma}{\operatorname{argmin}} \sum_{(x_i,z_i)\in\mathcal{P}(D_s)} \mathcal{L}_{CE}(k_\gamma(h_\theta(x_i)), z_i)$$



Given a patch, predict which one of 8 neighboring locations another patch is in



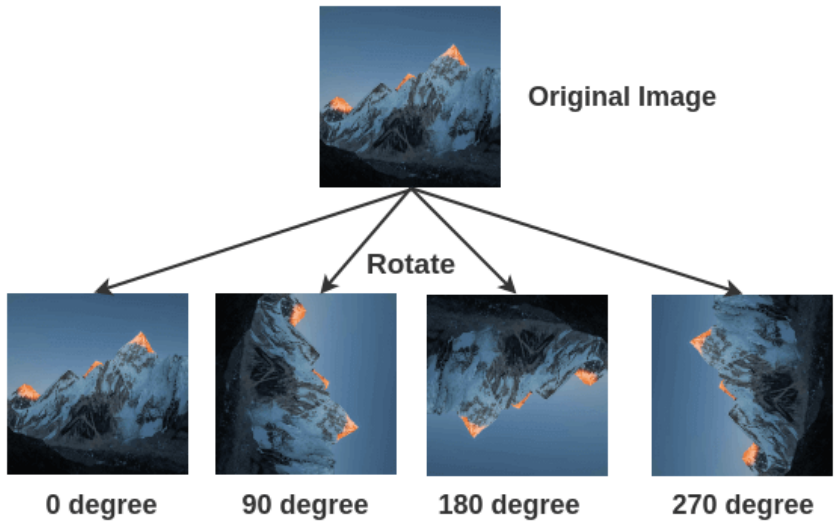
Output a probability vector per patch index out of a predefined set of permutations

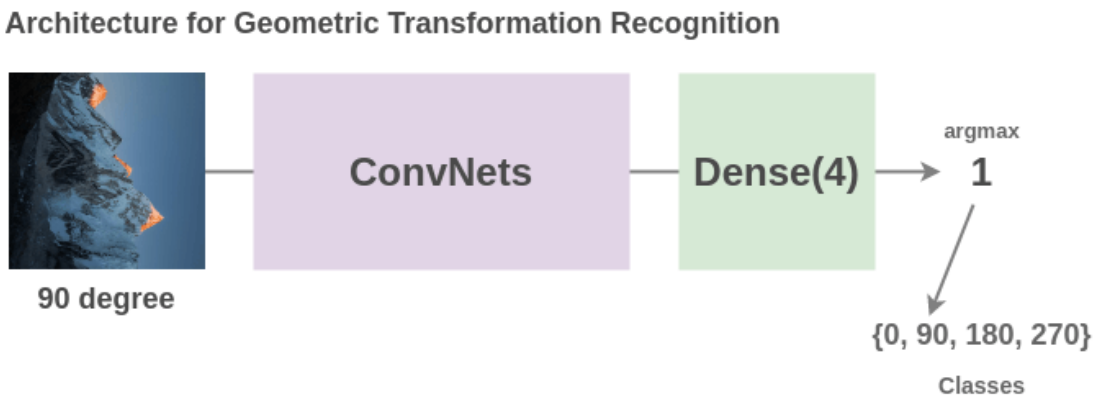
Algorithm 2. The pseudolabel generation process $\mathcal{P}$ for TP.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^M$.
  for i from 1 to M **do**
    Sample $\omega \sim \Omega$
    $x_i \leftarrow T_\omega(x_i^{(s)})$ ▷ Apply transformation to raw input
    $z_i \leftarrow \omega$
  end for
Output: $\{x_i, z_i\}_{i=1}^M$.

Image Rotation

- **Geometric transformation recognition**
- 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

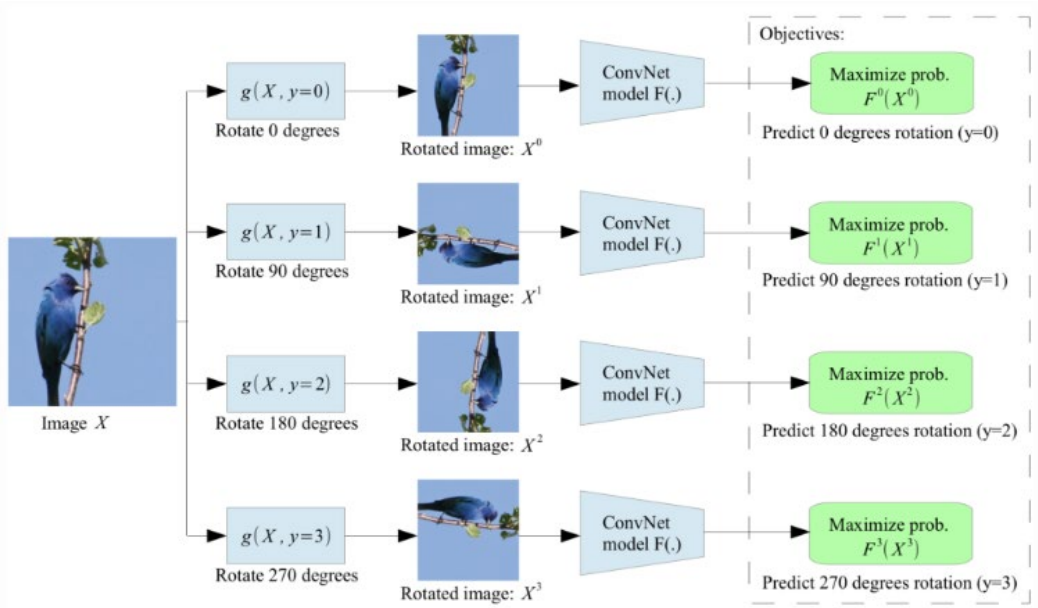




Gidaris, Spyros et al. Unsupervised Representation Learning by Predicting Image Rotations. ICLR 2018

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



- 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

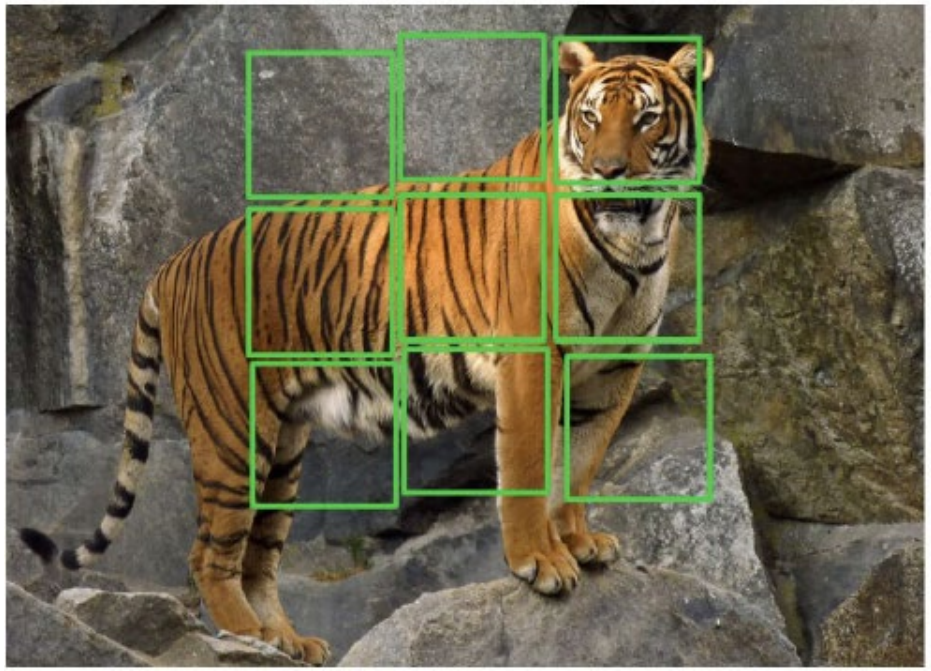
	Classification (%mAP)		Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all	all
ImageNet labels	78.9	79.9	56.8	48.0
Random		53.3	43.4	19.8
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6	32.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9	
Context Encoders (Pathak et al., 2016b)	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
(Ours) RotNet	**70.87**	**72.97**	**54.4**	**39.1**

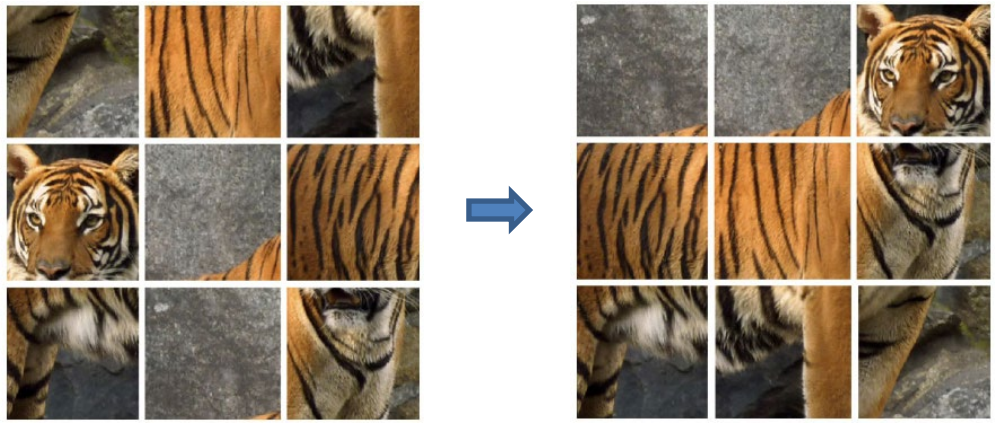
Supervised feature learning

Proposed self-supervised feature learning

Image Jigsaw Puzzle

- **Predict patches position in a jigsaw puzzle**
- 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





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

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

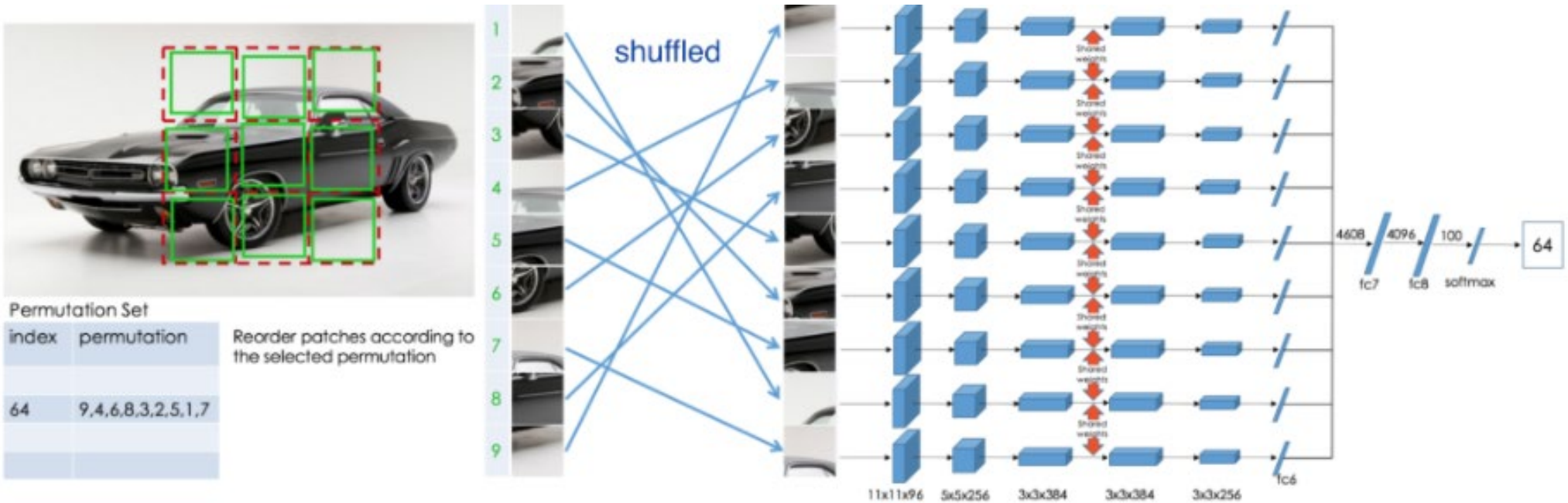
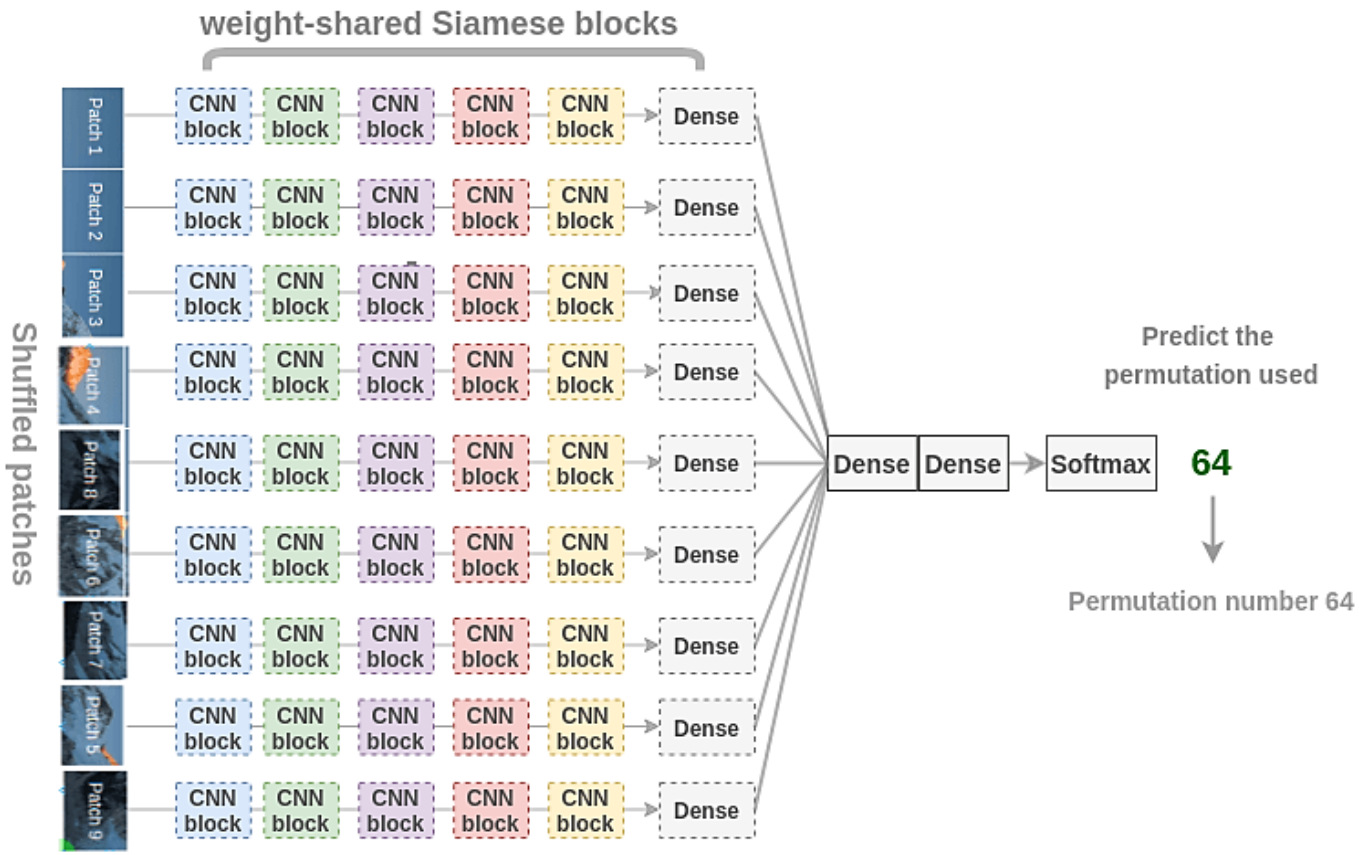


Image Jigsaw Puzzle

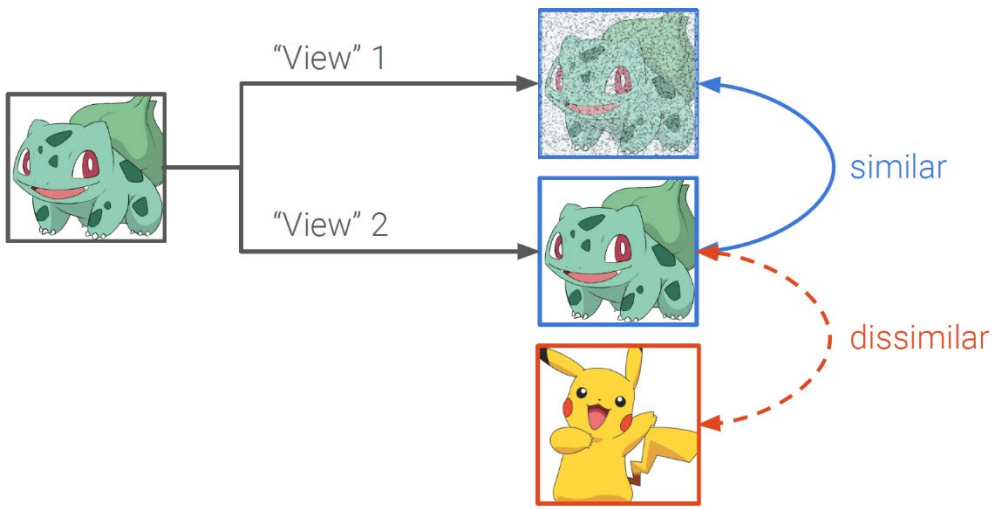
- The model needs to learn to identify how parts are assembled in an object, relative positions of different parts of objects, and shape of objects
  - The learned representations are useful for downstream tasks in classification and object detection



Instance Discrimination

One input, $x^a \sim T(x_i^{(s)})$ is chosen to be the anchor and is compared with a positive sample, $x^+ \sim T(x_i^{(s)})$, which is another view or transform of the same input. The anchor is also contrasted with a negative sample, which is a view of a different image, $x_j^- \sim T(x_j^{(s)})$. The samples are then encoded by the feature extractor to obtain their representations, $r^a = h_\theta(x^a), r^+ = h_\theta(x^+), r_j^- = h_\theta(x_j^-)$. A similarity function Φ is used to measure the similarity between positive (the anchor with a positive sample) and negative pairs (the anchor with a negative sample). The system is then trained to pull positive pairs closer and push negative pairs apart. A general formulation of the contrastive loss used in many works is

$$\mathcal{L}_{\text{con}} = -\mathbb{E}\left[\log \frac{\Phi(r^a, r^+)}{\Phi(r^a, r^+) + \sum_{j=1}^{k} \Phi(r^a, r_j^-)}\right]$$

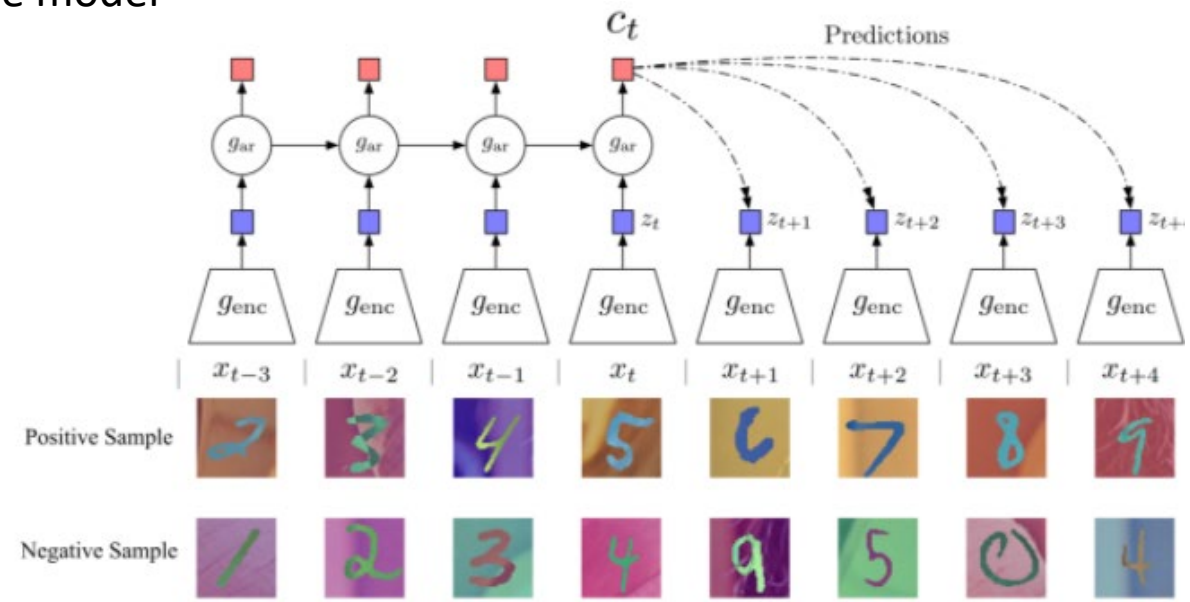


Algorithm 3. The pseudolabel generation process $\mathcal{P}$ for contrastive-instance discrimination.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^M$.
  for i from 1 to M **do**
    Sample $x^a \sim T(x_i^{(s)})$
    Sample $x^+ \sim T(x_i^{(s)})$
    for k from 1 to K **do**
      Sample $j \sim \mathcal{U}(1, M)$ ▷ Pick another raw input.
      Sample $x_k^- \sim T(x_j^{(s)})$ ▷ Get a random transform
    end for
    $x_i \leftarrow \{(x^a, x^+), (x^a, x_1^-), \ldots, (x^a, x_K^-)\}$.
    $z_i \leftarrow \{1, 0, \ldots, 0\}$.
  end for
Output: $\{x_i, z_i\}_{i=1}^M$.

Contrastive Predictive Coding

- **Contrastive Predictive Coding (CPC),** 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

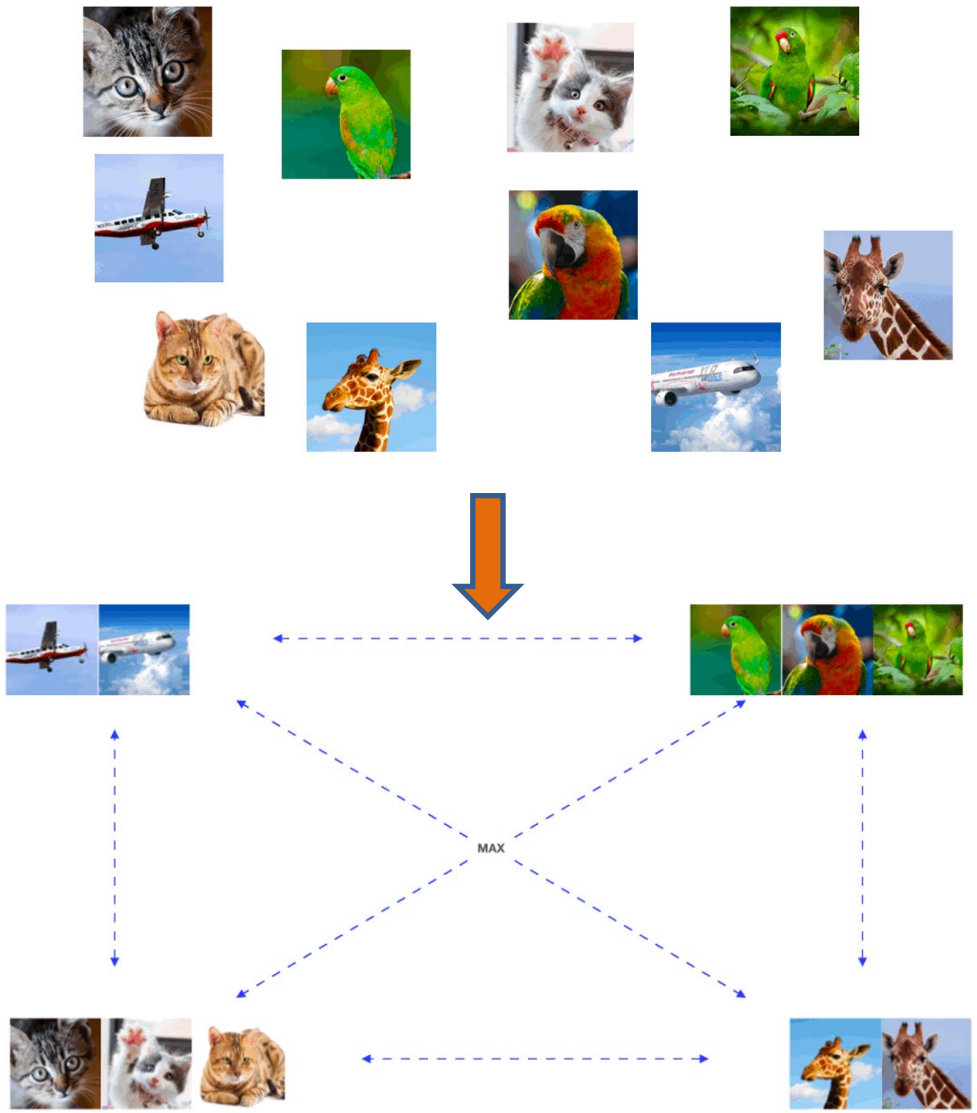


An example with images from MNIST, where a positive sequence contains sorted numbers, and a negative sequence contains random numbers

Oord, Aäron van den et al. Representation Learning with Contrastive Predictive Coding. ArXiv abs/1807.03748 (2018)

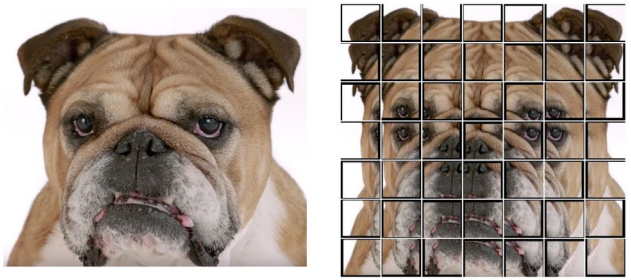
Contrastive Predictive Coding

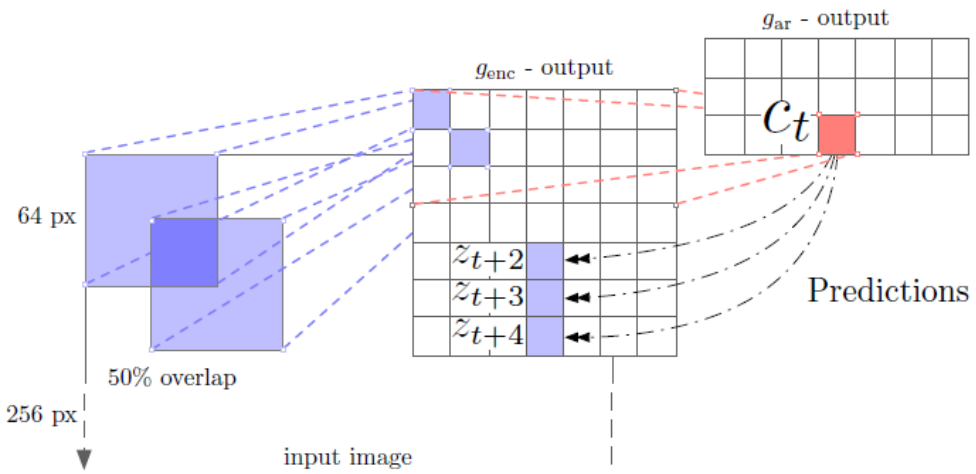
- **Contrastive Predictive Coding (CPC)**
- Training data: extracted patches from input images
- Pretext task: predict the order for a sequence of patches using contrastive learning
  - E.g., how to predict the next (future) patches based on encoded information of previous (past) patches in the image
- The approach was implemented in different domains: speech audio, images, natural language, and reinforcement learning
- Contrastive 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 256×256 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

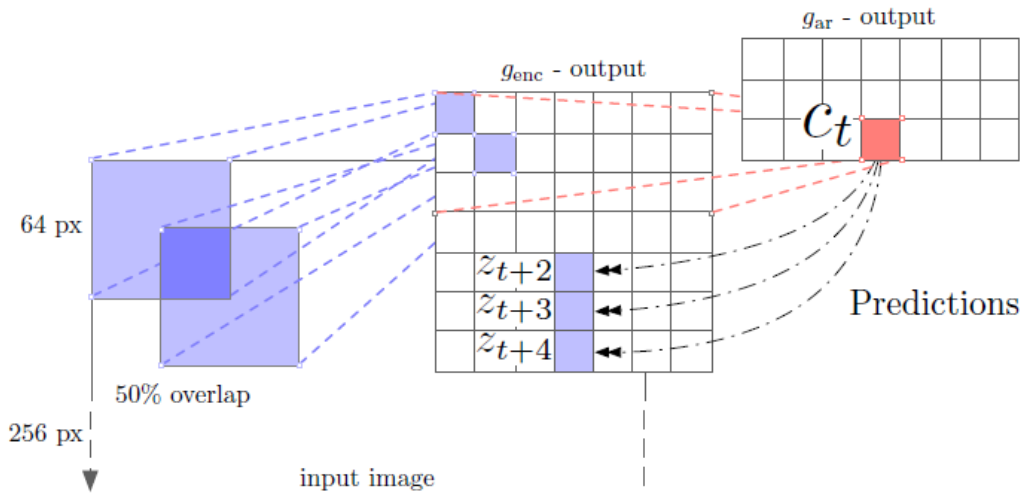




- An encoder g_{enc} is used to project each patch into a low-dimensional latent space
  - The latent representation obtained by encoders is often referred to as code (or context)
- E.g., the leftmost portion of the image depicts extracting patches of 64×64 pixels size with 50% overlap between the patches
  - 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., $z_1, z_2, \ldots, z_{49}$)

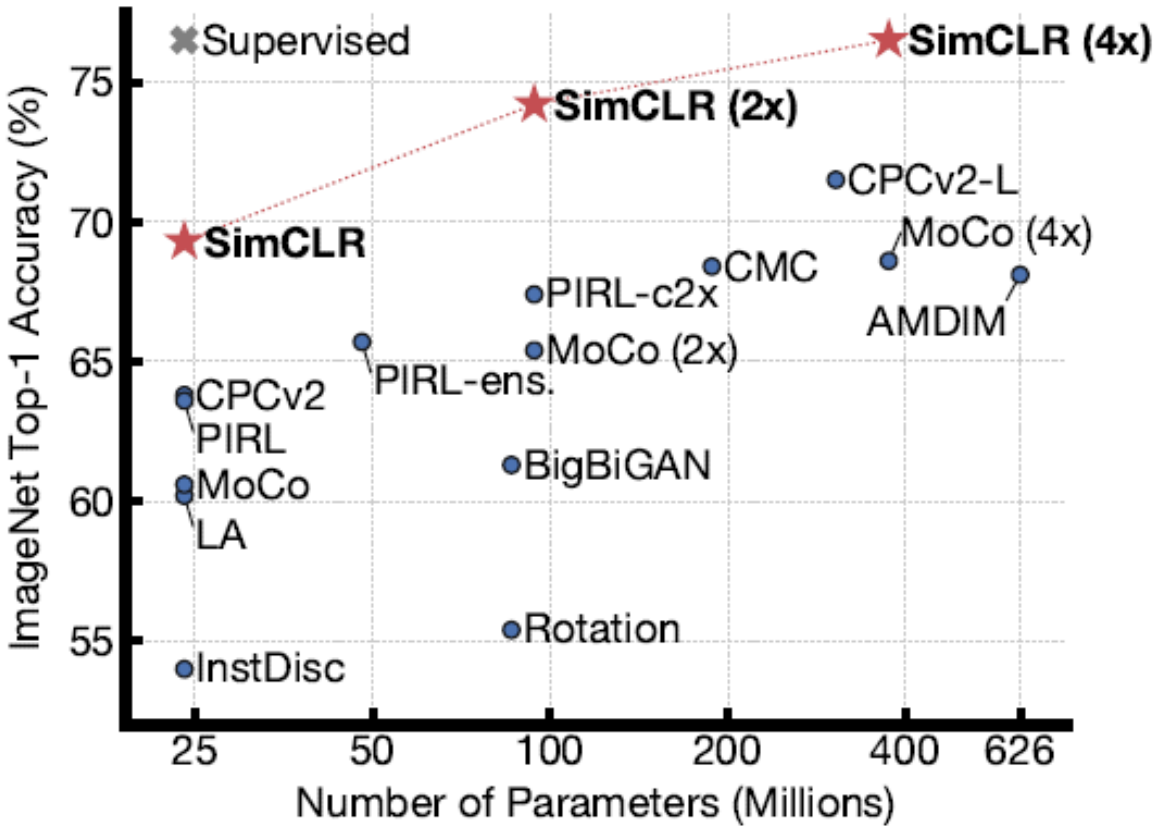
Contrastive Predictive Coding

- 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



SimCLR

- **SimCLR, a Simple framework for Contrastive Learning of 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



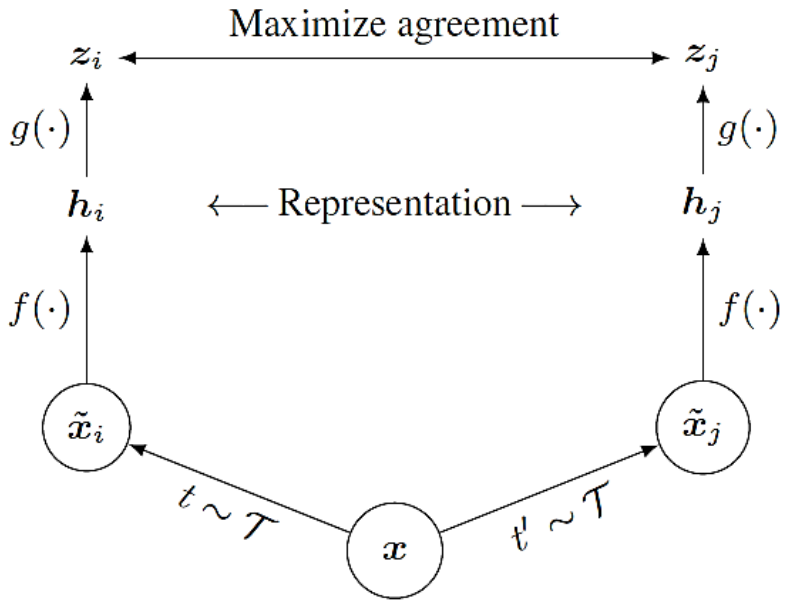
C. Ting et al. "A Simple Framework for Contrastive Learning of Visual Representations." ArXiv abs/2002.05709 (2020)

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:

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

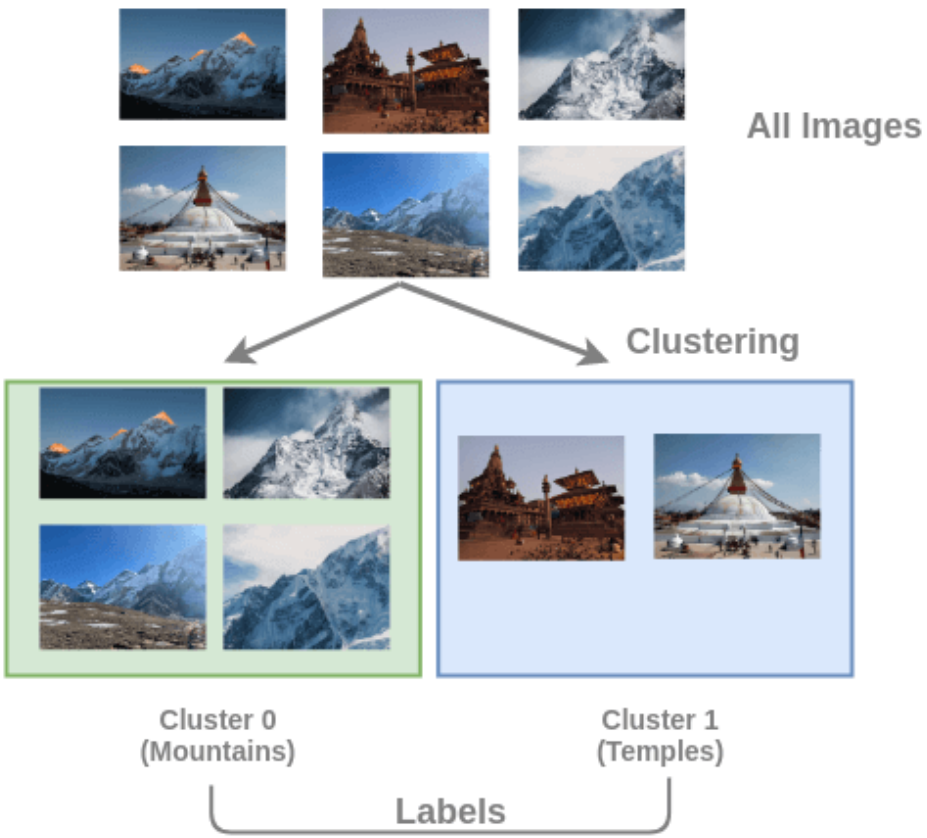
  - $\mathbf{1}_{[k \neq i]}$ has a value of 1 if $k \neq i$ and 0 otherwise, τ is a temperature hyperparameter
- The overall loss $\sum_{i,j} \mathcal{L}_{i,j}$ is calculated across all positive pairs $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$ in a mini-batch
- For downstream tasks, the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

	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

SimCLR outperformed supervised models on most datasets

Clustering

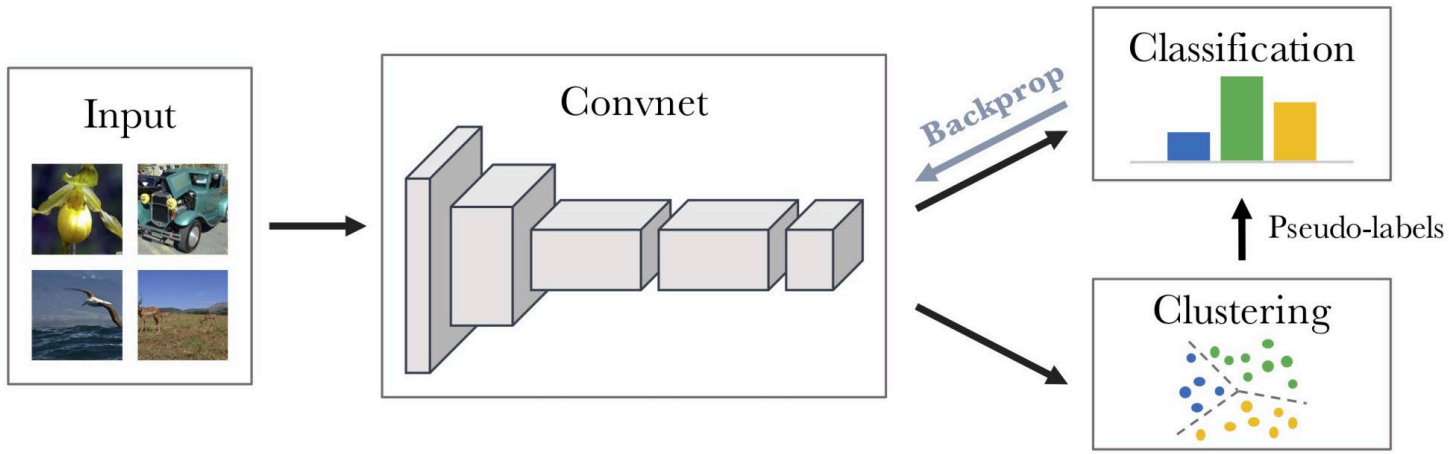
- **Deep clustering of images**
- Training data: clusters of images based on the content
  - E.g., clusters on mountains, temples, etc.
- Pretext task: predict the cluster to which an image belongs



M. Caron, et al., "Deep Clustering for Unsupervised Learning of Visual Features", ECCV 2018

Clustering

A common approach to self-supervised clustering is by alternating two steps:
1) optimizing the clustering objective by assigning data points into clusters based on their representations
2) optimizing the model by using the cluster assignments as the pseudo-labels in updates.

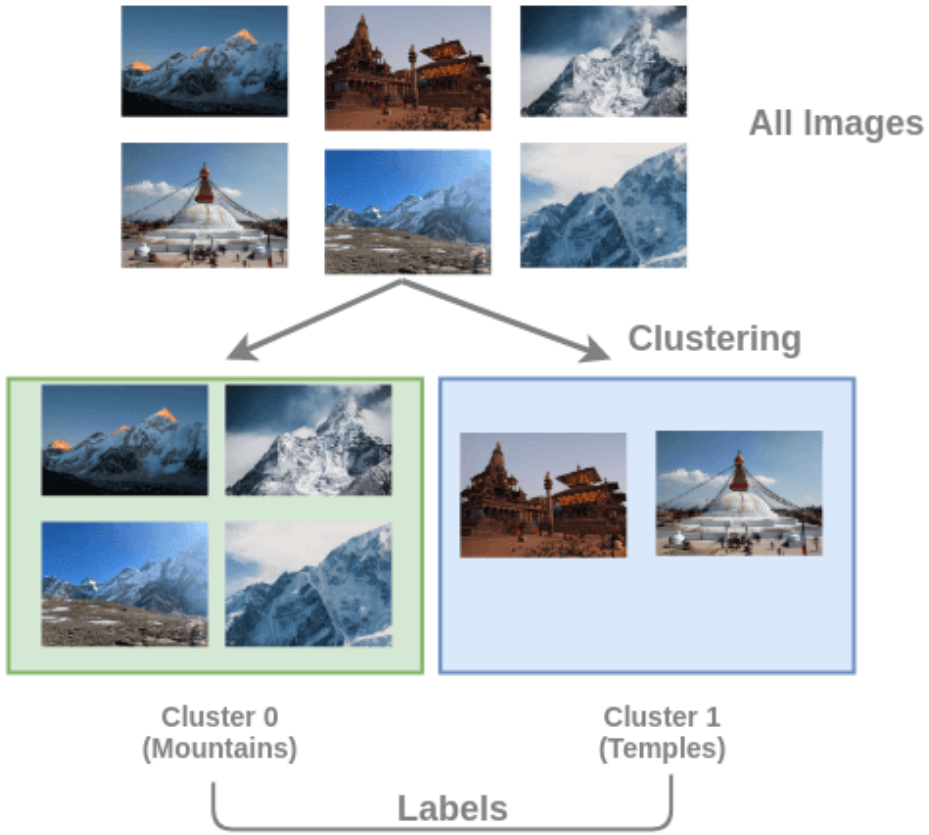


Algorithm 4. The pseudolabel generation process $\mathcal{P}$ for clustering.

Input: Unlabeled data set $D_s = \{x_i^{(s)}\}_{i=1}^{M}$.
Input: Representations $\{r_i\}_{i=1}^{M}$, where $r_i \leftarrow h_\theta(x_i^{(s)})$
Input: Cluster centers $\{c_j\}_{j=1}^{k}$, via clustering on $\{r_i\}_{i=1}^{M}$.
  for i from 1 to M **do**
    Sample $x_i \sim T(x_i^{(s)})$
    $z_i \leftarrow \operatorname{argmin}_{j \in [k]} \| c_j - r_i \|$
  end for
Output: $\{x_i, z_i\}_{i=1}^{M}$.

Deep Clustering

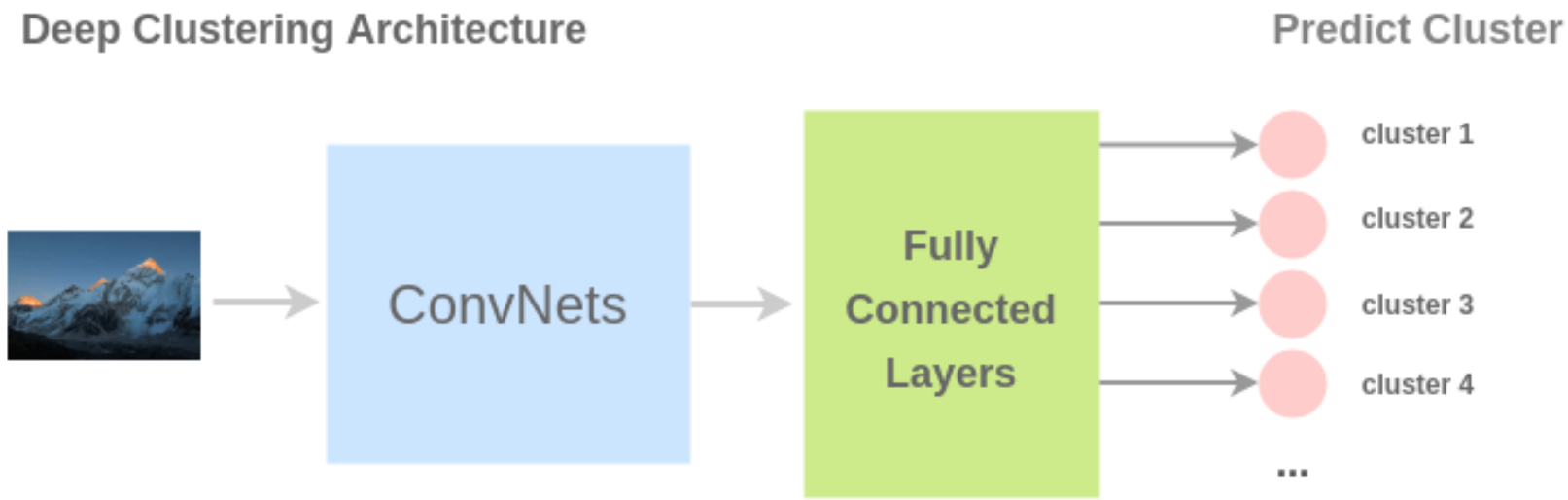
- **Deep clustering of images**
- Training data: clusters of images based on the content
  - E.g., clusters on mountains, temples, etc.
- Pretext task: predict the cluster to which an image belongs



M. Caron, et al., "Deep Clustering for Unsupervised Learning of Visual Features", ECCV 2018

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



Outline

- Self-Supervised Learning (SSL)
- Early works
- Methods for framing SSL
- Pretext tasks
- **Large Language Models**
- Discussion
- Conclusions

Natural Language Processing

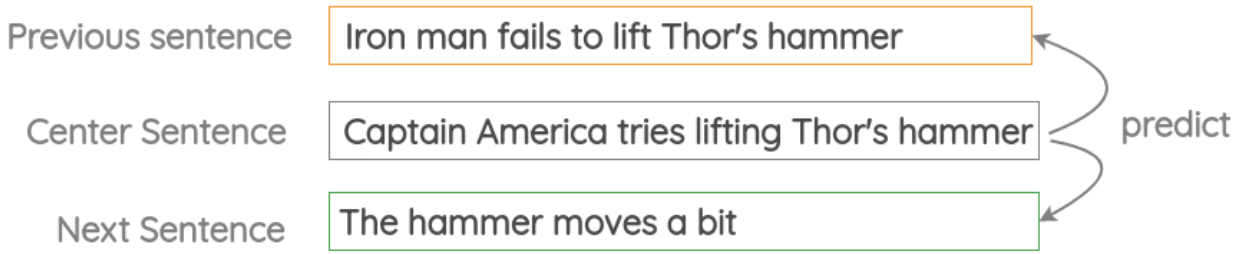
- Self-supervised learning has driven the recent progress in the ***Natural Language Processing*** (NLP) field
  - Models like ELMO, BERT, RoBERTa, ALBERT, Turing NLG, GPT-3 have demonstrated immense potential for automated NLP
- Employing various pretext tasks for leaning from raw text produced rich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

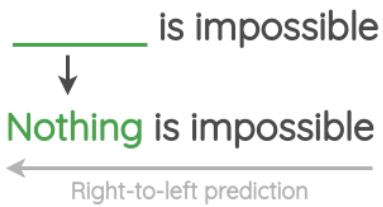
A. Chaudhary. "Self Supervised Representation Learning in NLP," 2020

Natural Language Processing

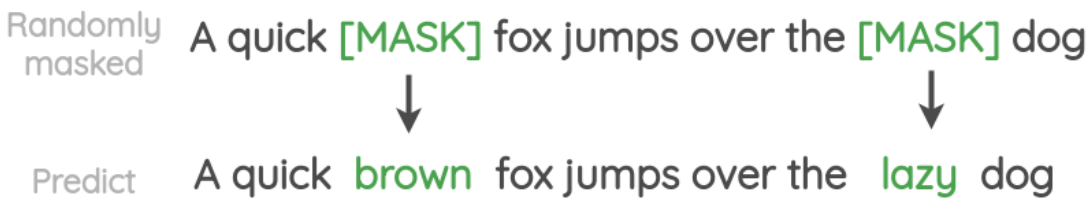
- Pretext tasks in NLP:
  - From three consecutive sentences, predict the previous and the next sentence, given the center sentence



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



  - Predict randomly masked words in sentences



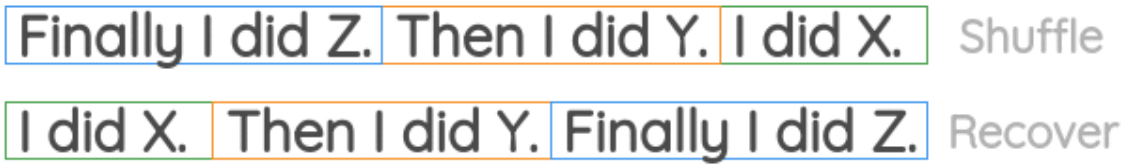
A. Chaudhary. "Self Supervised Representation Learning in NLP," 2020

Natural Language Processing

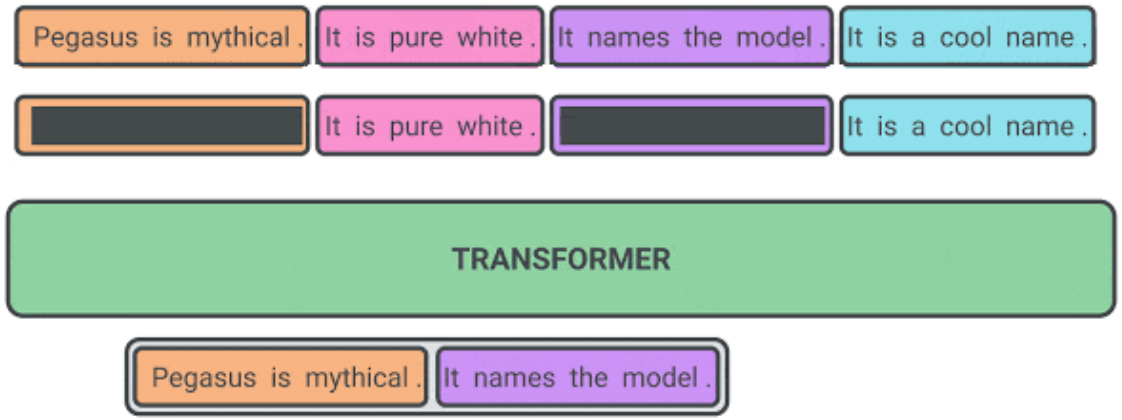
- Pretext tasks in NLP:
  - Predict if the ordering of two sentences is correct

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

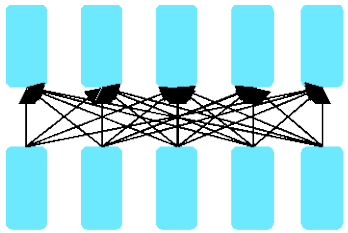


  - Predict masked sentences in a document



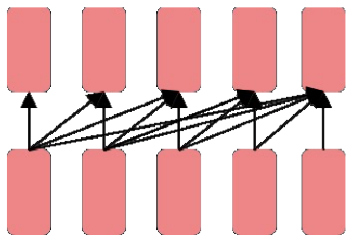
A. Chaudhary. "Self Supervised Representation Learning in NLP," 2020

Large Language Model with Transformer



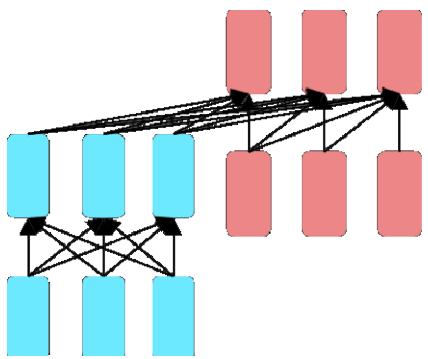
Encoders

- Examples: BERT, RoBERTa, SciBERT.
- Captures bidirectional context. Wait, how do we pretrain them?



Decoders

- Examples: GPT-2, GPT-3, LaMDA
- Other name: causal or auto-regressive language model
- Nice to generate from; can't condition on future words



Encoder-Decoders

- Examples: Transformer, T5, Meena
- What's the best way to pretrain them?

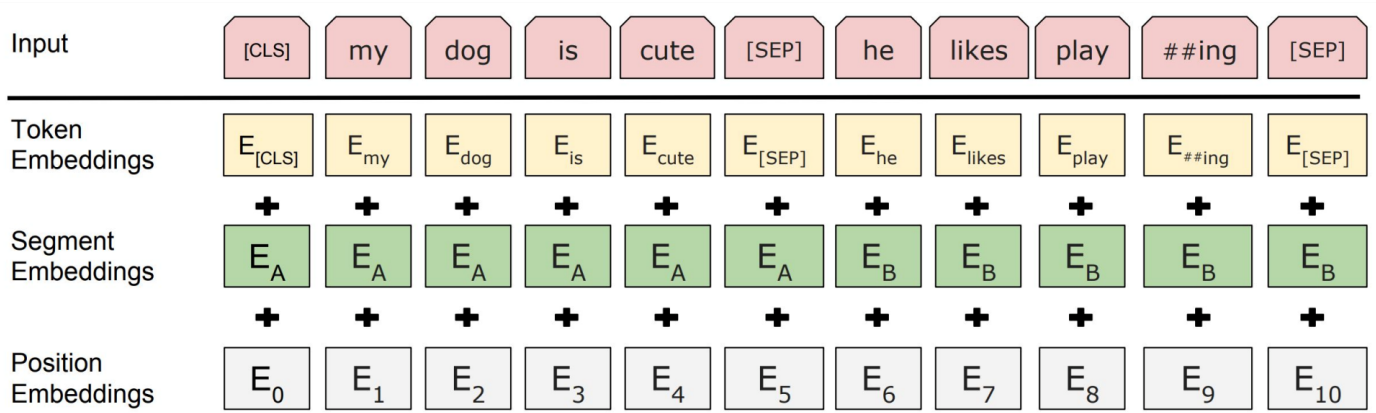
Large Language Model: BERT

BERT: Transformer based network to learn representations of language

Improvements:

1. Bi-directional LSTM -> Self-attention
2. Massive data
3. Masked-LM objective

BERT: Input Representation

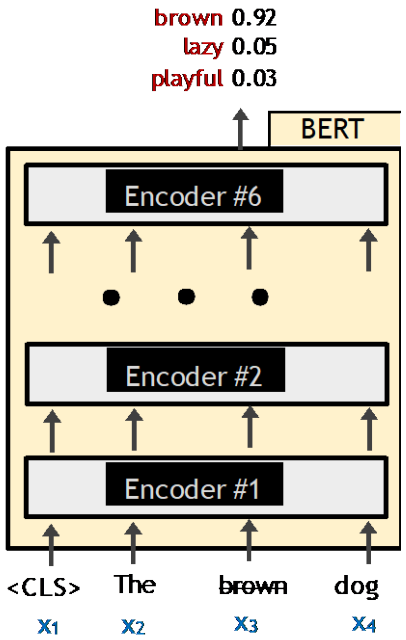


- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings
  - Addition to transformer encoder: sentence embedding

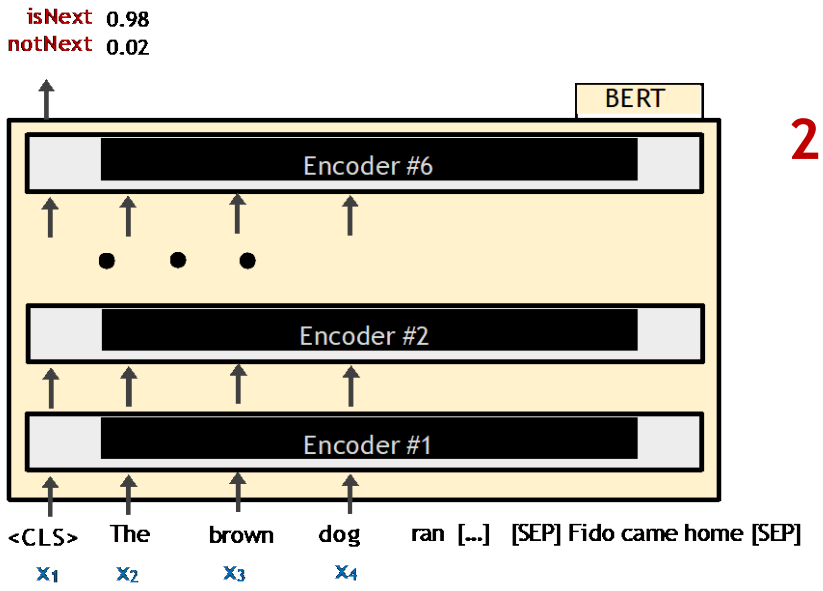
J. Devlin, et, al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding ," 2018, arXiv preprint arXiv:1810.04805. https://github.com/google-research/bert#fine-tuning-with-bert

Large Language Model: BERT

BERT: Pre-training



Predict the masked word



Sentence ordering

2 training objectives:

1. Predict the Masked word

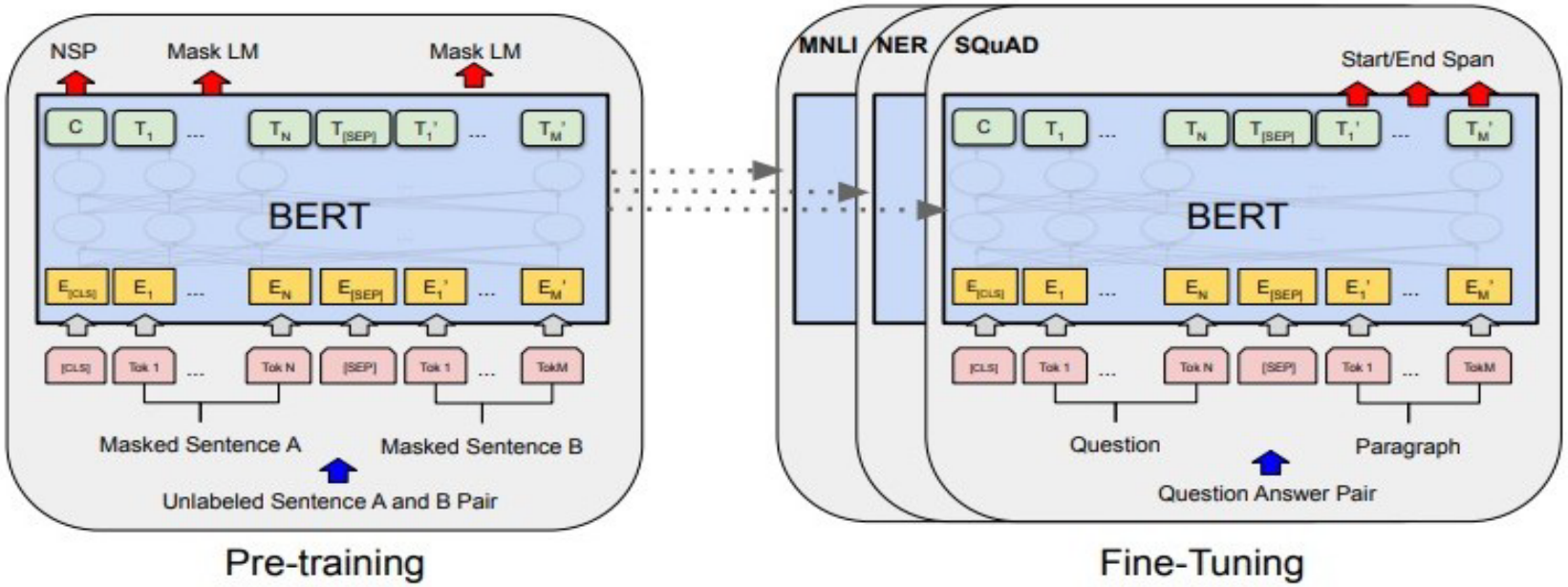
15% of all input words are randomly masked.

2. Sentence ordering

Two sentences are fed in at a time. Predict the if the second sentence of input truly follows the first one or not.

Large Language Model: BERT

BERT: Fine-tunning



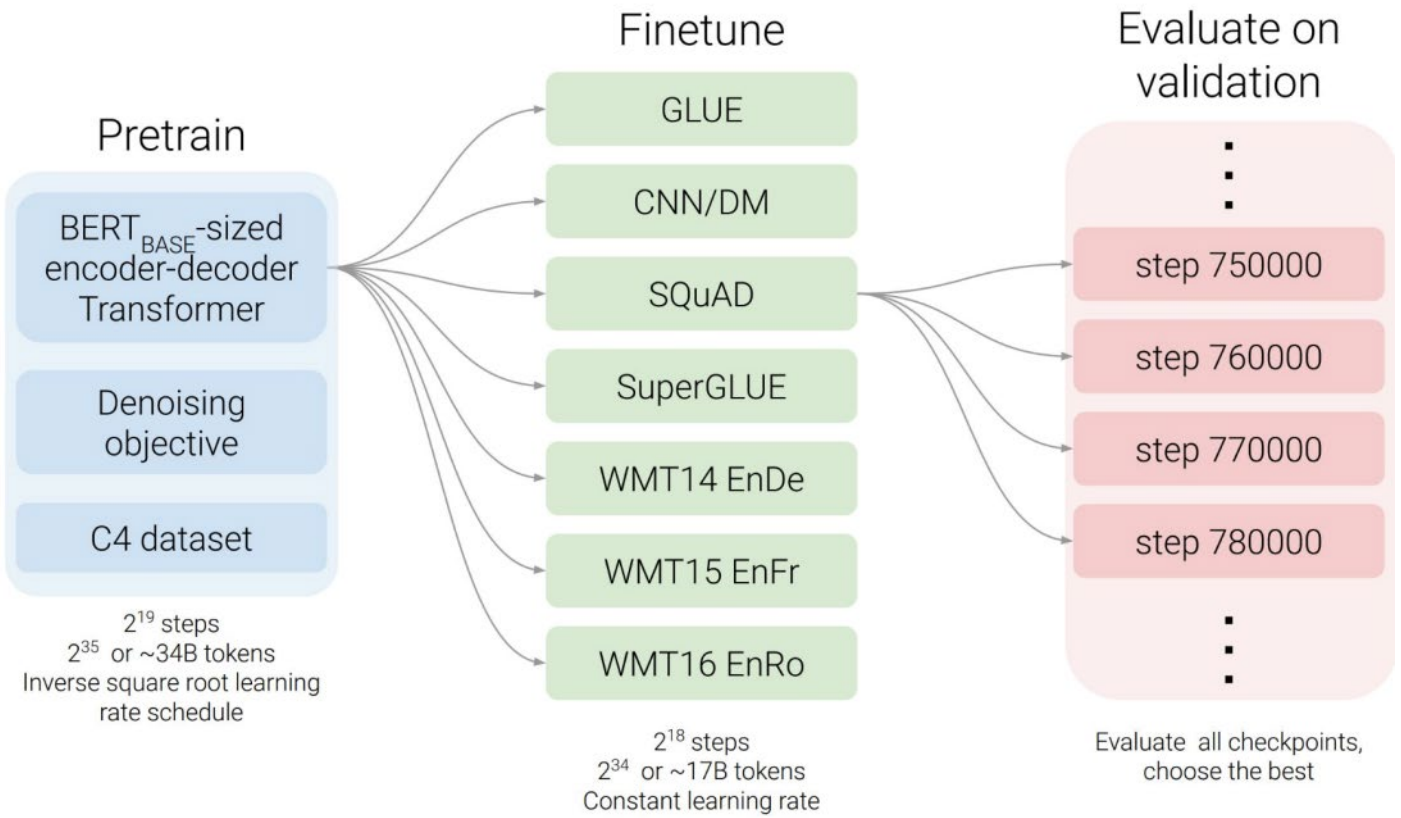
Idea: Make pre-trained model **usable** in **downstream tasks**

Initialized with pre-trained model parameters

Fine-tune model parameters using labeled data from downstream tasks

Large Language Model: BART/T5

Text-To-Text Transfer Transformer (T5): Understand the first order effect of each design choice by altering it while keeping other choices fixed.



C. Raffel, et,al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", Journal of Machine Learning Research, vol. 21, pp. 1-67, 2020

Large Language Model: BART/T5

Text-To-Text Transfer Transformer (T5): Understand the first order effect of each design choice by altering it while keeping other choices fixed.

- Prefix language modeling
  - **Input:** Thank you for inviting
  - **Output:** me to your party last week
- BERT-style denoising
  - **Input:** Thank you <M> <M> me to your party apple week
  - **Output:** Thank you for inviting me to your party last week
- Deshuffling
  - **Input:** party me for your to. last fun you inviting week Thanks.
  - **Output:** Thank you for inviting me to your party last week

- IID noise, replace spans
  - **Input:** Thank you <X> me to your party <X> week
  - **Output:** <X> for inviting <Y> last <Z>
- IID noise, drop tokens
  - **Input:** Thank you me to your party week .
  - **Output:** for inviting last

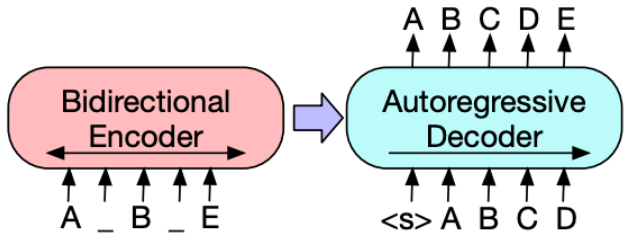
Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

- All the variants perform similarly
- "Replace corrupted spans" and "Drop corrupted tokens" are more appealing because target sequences are shorter, speeding up training.

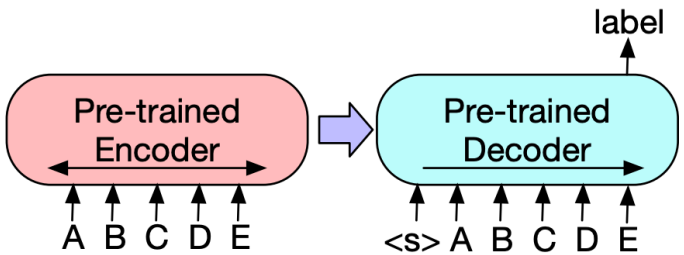
Large Language Model: BART/T5

BART: Similar Architecture as T5.

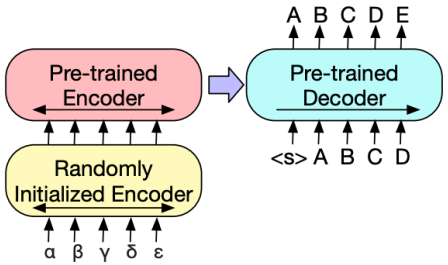
- Performs competitive to RoBERTa and XLNet on discriminative/classification tasks.
- Outperformed existing methods on generative tasks (question answering, and summarization).
- Improved results on machine translation with fine-tuning on target language.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.



(a) To use BART for classification problems, the same input is fed into the encoder and decoder, and the representation from the final output is used.



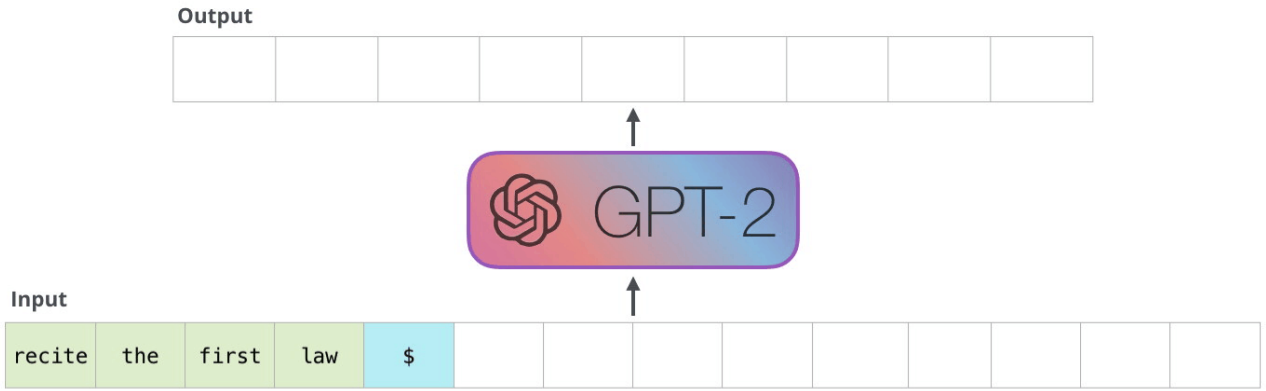
(b) For machine translation, we learn a small additional encoder that replaces the word embeddings in BART. The new encoder can use a disjoint vocabulary.

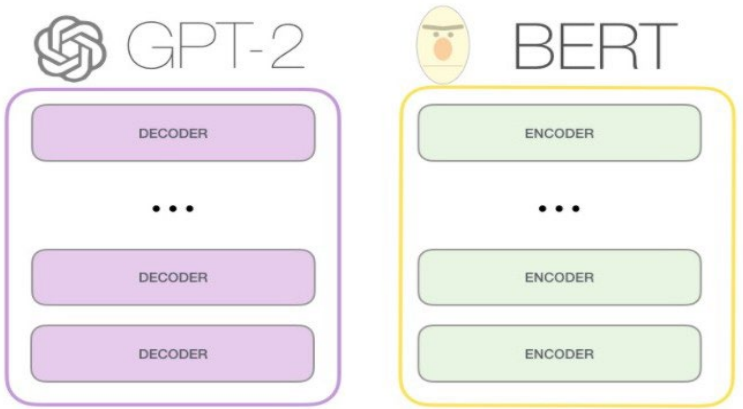
Figure 3: Fine tuning BART for classification and translation.

M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," ,2019, arXiv preprint arXiv: 1910.13461.

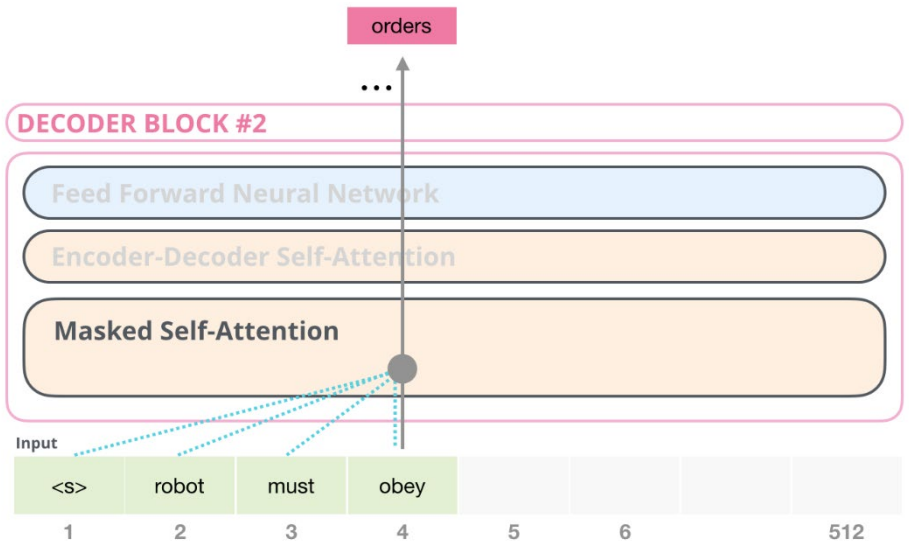
Large Language Model: GPT-2

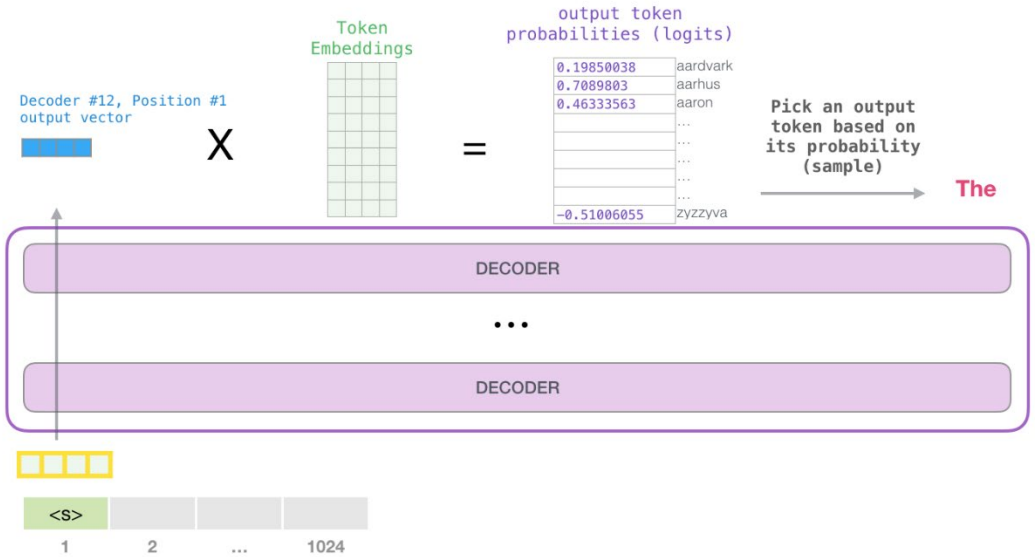
GPT-2 uses only Transformer Decoders (no Encoders) to generate new sequences from scratch or from a starting sequence.





As it processes each subword, it masks the "future" words and conditions on and attends to the previous words





http://jalammar.github.io/illustrated-gpt2/

Large Language Model: GPT-3

- **GPT-3** stands for ***Generative Pre-trained Transformer***
- 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

- BERT-Base model has 12 transformer blocks, 12 attention heads,
  - 110M parameters!
- BERT-Large model has 24 transformer blocks, 16 attention heads,
  - 340M parameters!
- GPT-2 is trained on 40GB of text data (8M webpages)!
  - 1.5B parameters!
- GPT-3 is an even bigger version of GPT-2, but isn't open-source
  - 175B parameters!

https://beta.openai.com/playground

Outline

- Self-Supervised Learning (SSL)
- Early works
- Methods for framing SSL
- Pretext tasks
- Large Language Models
- **Discussion**
- Conclusions

Discussion

- **Pretraining cost:** Although there is also tremendous research activity in developing more efficient pretraining algorithms, the net cost of pretraining is trending upward due to the fact that bigger data sets and larger network architectures have systematically led to better performance.
- **Data requirement and curation:** For benchmarking purposes (especially in vision and audio and graphs, but less so in text), methods are often actually trained on curated data while ignoring the labels. It is not clear how much existing algorithm design is overfitted to these curated data sets, and whether the relative performance of different approaches is maintained when real, uncurated data are used instead.
- **Architecture choice and deployment costs:** the trend has been that bigger architectures lead to better representation performance. This is welcome from the perspective of near- "automatic" performance improvement as data sets and computation capabilities grow. However, it does pose a concern for deployment of the resulting models on resource-constrained or embedded devices with limited memory and/or computation capability, which may limit the benefit of this line of improvement for such applications.

L. Ericsson, et, al., "Self-Supervised Representation Learning: Introduction, advances, and challenges," in IEEE Signal Processing Magazine, vol. 39, no. 3, pp. 42-62, 2022

Discussion

- **Transferability:** The current state of the graph modality is that transferability is good to unseen nodes within the same graph and to unseen graphs within the same data set, e.g., protein-protein interactions. However, there is little information to suggest transfer across graph types, like chemical-to-biological or citation-to-social, currently has any benefit.

- **Choosing the right pretext task:** Since self-supervised pretexts rely on exploiting the structure of data, which in turn differs significantly across modalities, their efficacy can vary substantially across modalities.

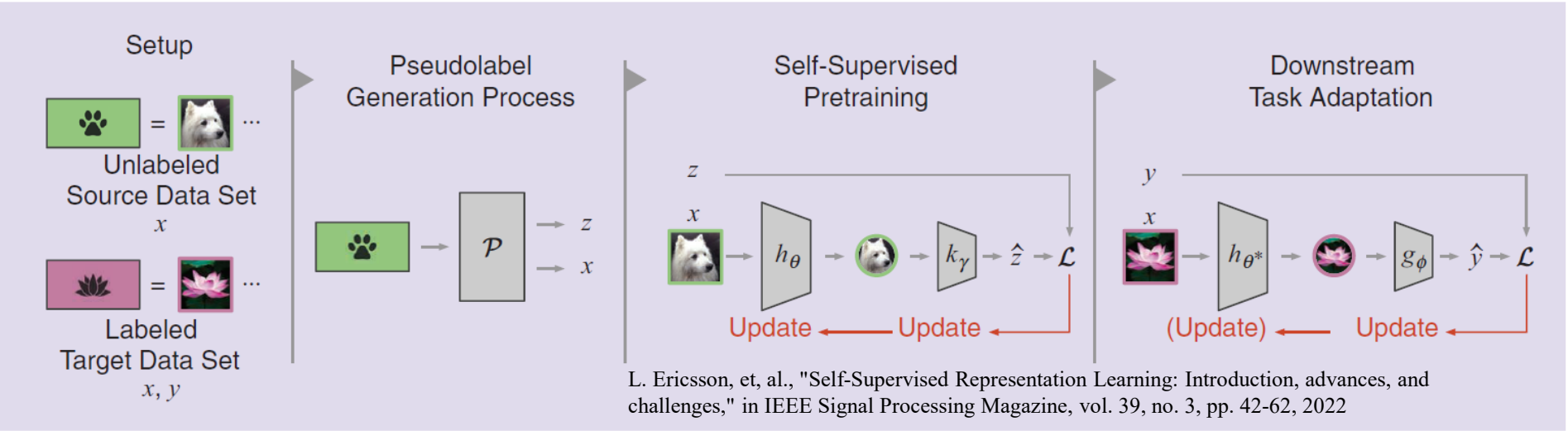
- **Self-supervised versus semi-supervised:** As both families of methods are making rapid progress and there have been few direct comparisons, it is not yet clear if/when one family should be preferred.

Outline

- Self-Supervised Learning (SSL)
- Early works
- Methods for framing SSL
- Pretext tasks
- Large Language Models
- Discussion
- **Conclusions**

Conclusions

Goal of self-supervised learning: Enable representation learning on easily obtained, uncurated data.



L. Ericsson, et, al., "Self-Supervised Representation Learning: Introduction, advances, and challenges," in IEEE Signal Processing Magazine, vol. 39, no. 3, pp. 42-62, 2022

FIGURE 2. The self-supervised workflow starts with an unlabeled source data set and a labeled target data set. As defined by the pretext task, pseudolabels are programmatically generated from the unlabeled set. The resulting inputs, x and pseudolabels z, are used to pretrain the model $k_\gamma(h_\theta(\cdot))$—composed of feature extractor h_θ and output k_γ modules—to solve the pretext task. After pretraining is complete, the learned weights θ^* of the feature extractor h_{θ^*} are transferred and used together with a new output module g_ϕ to solve the downstream target task.

Step 1: Pre-train a model for a pretext task

Step 2: Transfer to downstream task applications

Thank you!