Self-supervised learning Part 2

Self-supervised learning

- BERT vs GPT
- Multi-modal
- Contrastive learning

Flavors of supervision

Supervised

All labels known

Semi-supervised

Some labels known

Unsupervised

No labels

Representation learning

Transfer learning

Self-supervised learning

Surrogate task from pseudo labels

Reinforcement learning

Agent gathers data, reward function known

Self-supervised learning

Self-supervised learning

Unsupervised learning trained using supervised learning techniques

Cleverly exploit property of the data to create pseudo labels

Mostly used for representation learning

Need small supervised data to map to useful task

Yann LeCun
April 30, 2019 ·

I now call it "self-supervised learning", because "unsupervised" is both a loaded and confusing term.

In self-supervised learning, the system learns to predict part of its input from other parts of it input. In other words a portion of the input is used as a supervisory signal to a predictor fed with the remaining portion of the input.

Self-supervised learning uses way more supervisory signals than supervised learning, and enormously more than reinforcement learning. That's why calling it "unsupervised" is totally misleading. That's also why more knowledge about the structure of the world can be learned through self-supervised learning than from the other two paradigms: the data is unlimited, and amount of feedback provided by each example is huge.

Self-supervised learning has been enormously successful in natural language processing. For example, the BERT model and similar techniques produce excellent representations of text.
BERT is a prototypical example of self-supervised learning: show it a sequence of words on input, mask out 15% of the words, and ask the system to predict the missing words (or a distribution of words). This an example of masked auto-encoder, itself a special case of denoising auto-encoder, itself an example of self-supervised learning based on reconstruction or prediction. But text is a discrete space in which probability distributions are easy to represent.

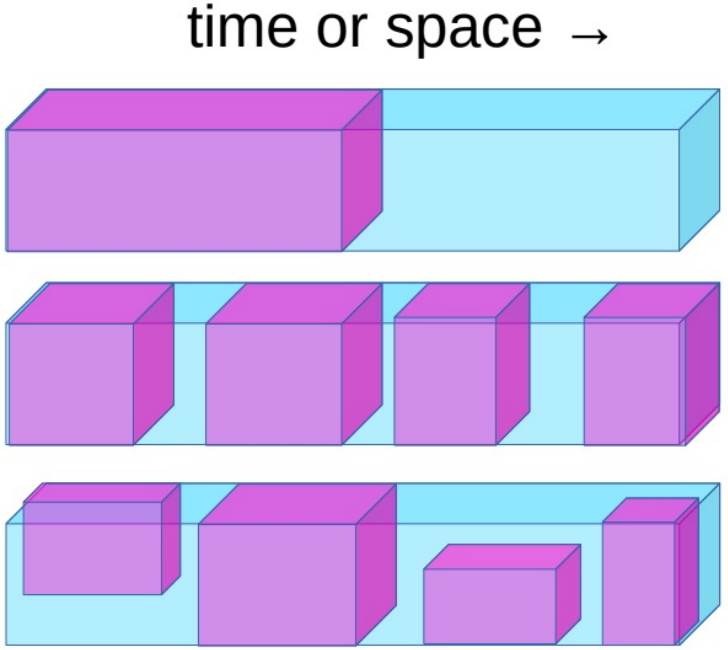
So far, similar approaches haven't worked quite as well for images or videos because of the difficulty of representing distributions over high-dimensional continuous spaces.

Doing this properly and reliably is the greatest challenge in ML and AI of the next few years in my opinion.

Self-Supervised Learning = Filling in the Blanks

Y. LeCun

- **Predict any part of the input from any other part.**
- **Predict the future from the past.**
- **Predict the masked from the visible.**
- **Predict the any occluded part from all available parts.**



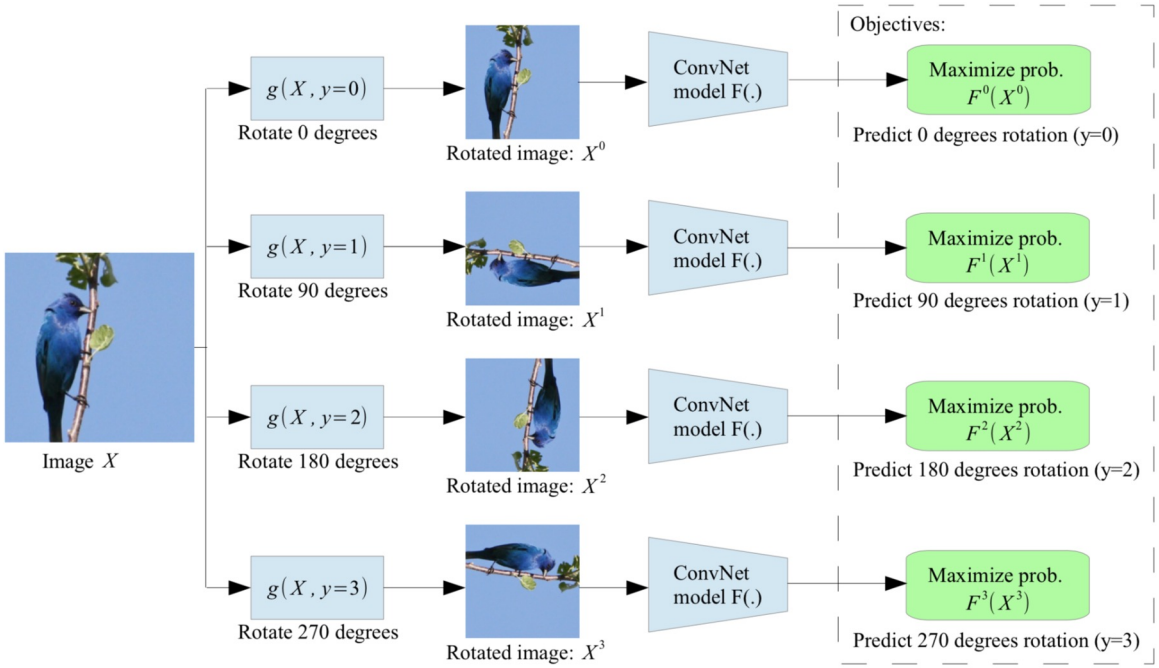
- **Pretend there is a part of the input you don't know and predict that.**
- **Reconstruction = SSL when any part could be known or unknown**

https://twitter.com/ylecun/status/1226838002787344391?lang=en

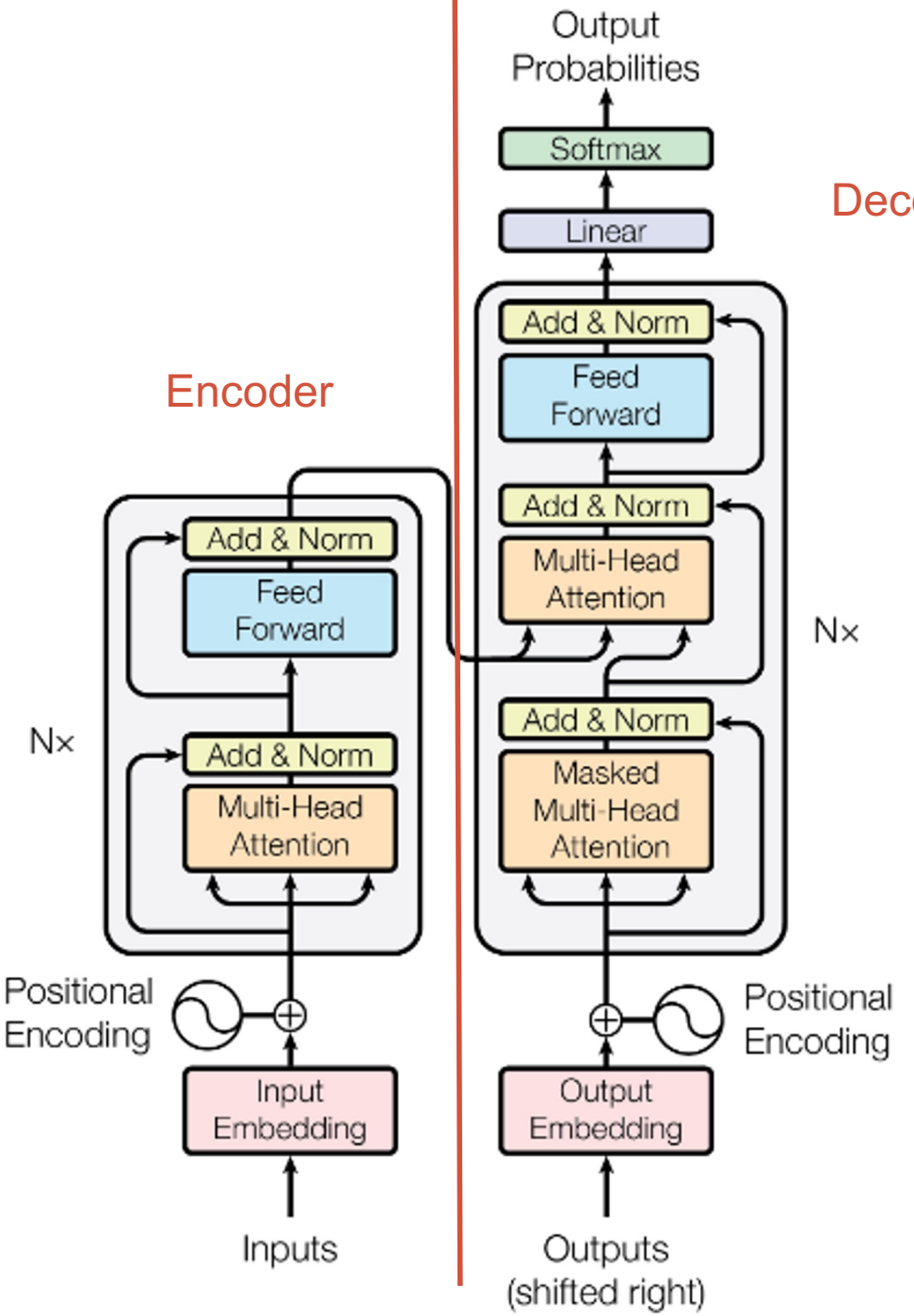
Examples

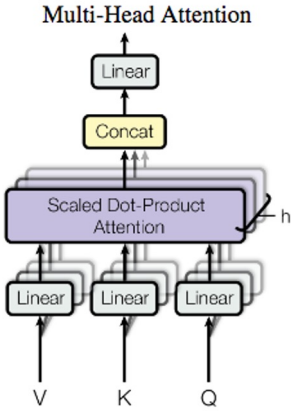
Images

Predict missing patches, predict orientation, etc.



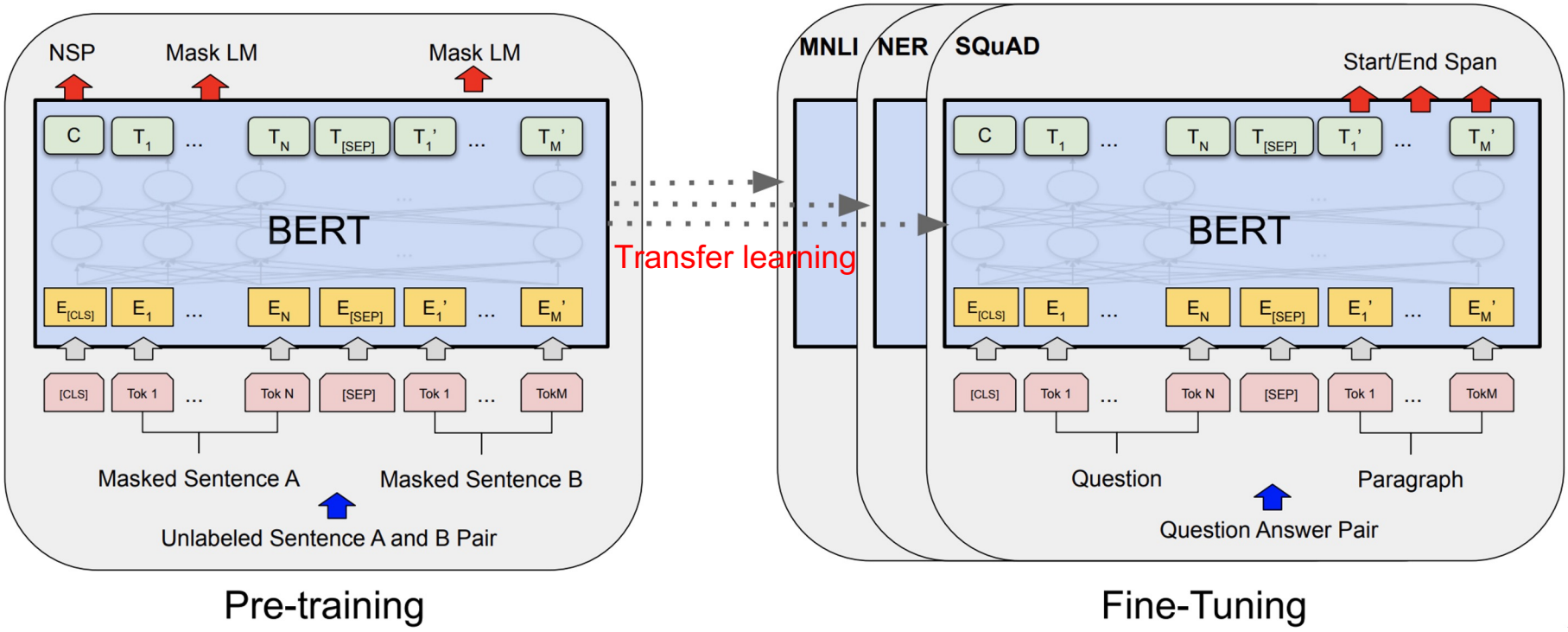
https://arxiv.org/abs/1803.07728 Unsupervised Representation Learning by Predicting Image Rotations


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Output Embedding
Outputs (shifted right)
Encoder
Decoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding


Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

BERT

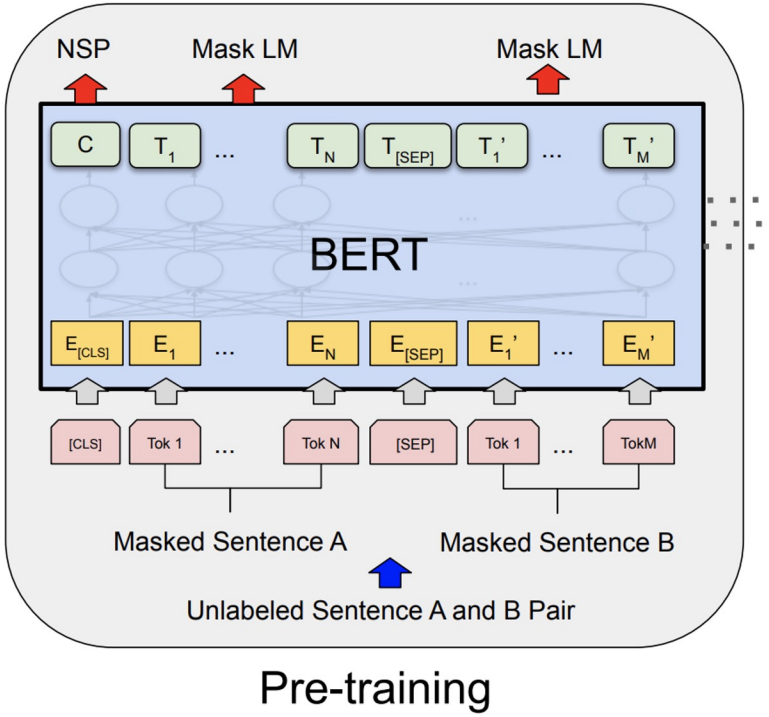
Self-supervised signals
1) Masked words and predict the missing words
2) Next sentence prediction to learn relationship between two sentences



https://arxiv.org/abs/1810.04805

BERT

Self-supervised signals
1) Masked words and predict the missing words
2) Next sentence prediction to learn relationship between two sentences



Pre-training

https://arxiv.org/abs/1810.04805

Mask prediction

Sent A: This is the ___ I used to write with.
Sent B: It belonds to Adam.

Text correction

Sent A: This is the orange I used to write with.
Sent B: The plane's wall burst opens.

Pre-trained transformer types (by training method)

Encoder only (autoencoder)

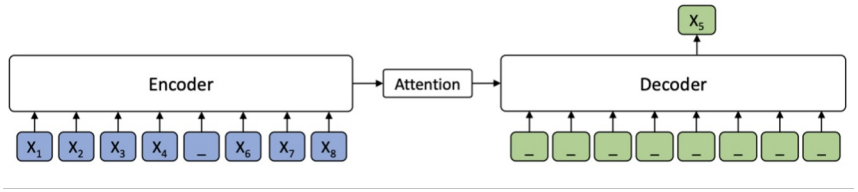
- BERT, ALBERT, RoBERTa
- Seq classification, token classification
- Masked words and predict

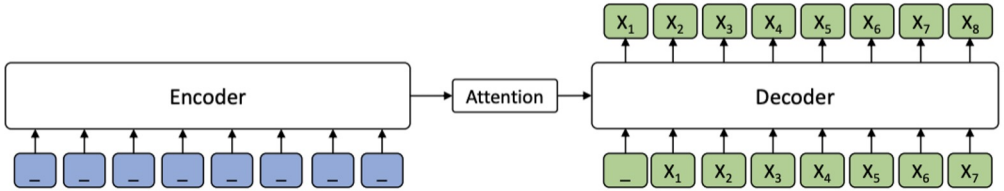
Encoder-decoder (seq2seq)

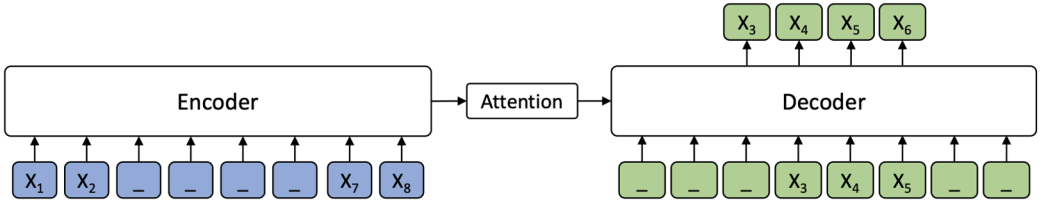
- MASS, BART, T5
- Machine translation, text summary
- Mask phrases

Decoder only (autoregressive)

- GPT, CTRL, XGLM
- Text generation
- Predict next word



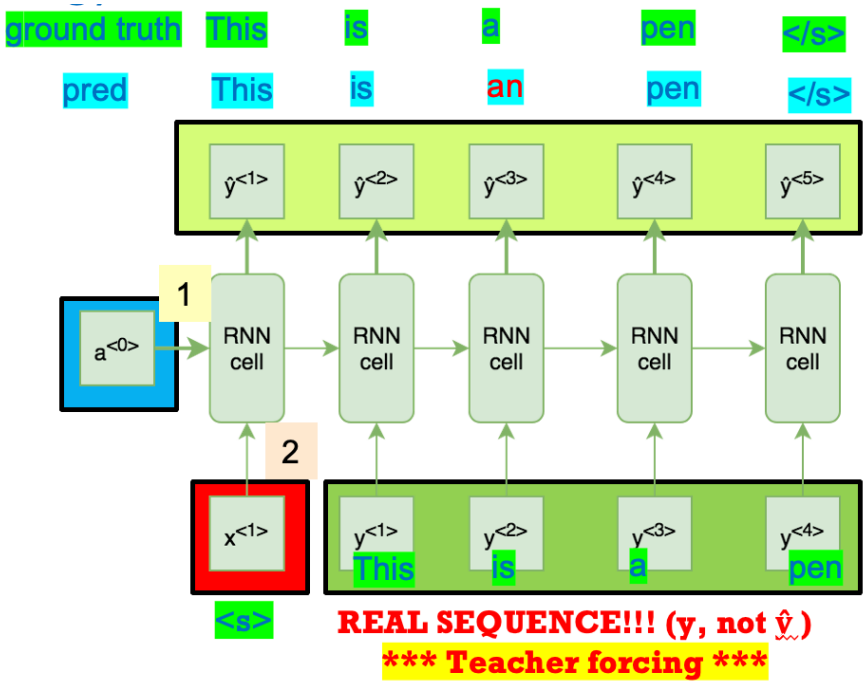




https://huggingface.co/transformers/summary.html https://arxiv.org/pdf/1905.02450.pdf

Autoregressive models

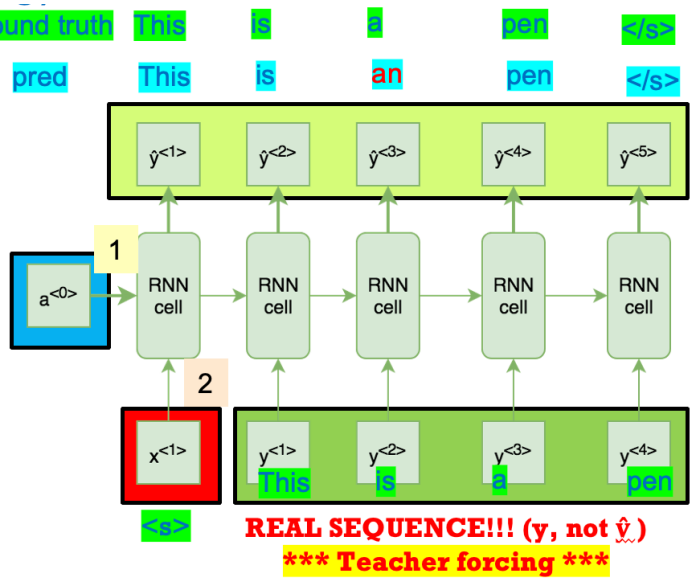
- To generate a sequence of words, we predicts one word at a time.



Autoregressive model (training)

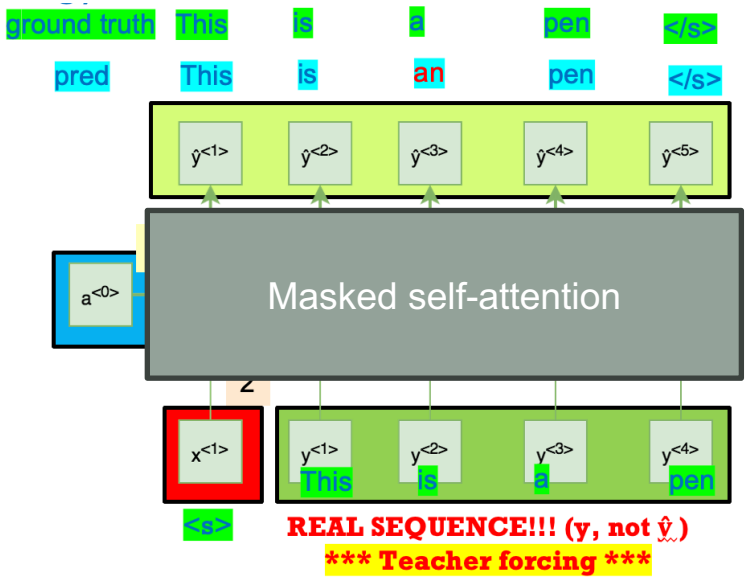
- Training is done by predicting token by token to generate a sequence.
  - Previous output is used as input to ensure consistency of the generation. It knows what it output before.
- Teacher forcing: always force input to be correct

- Learns
  P(next token | history)



Autoregressive model (training)

- For attention-based models, we use a causal mask to ensure the model doesn't cheat with teacher forcing, and make training efficient.



Self Attention map

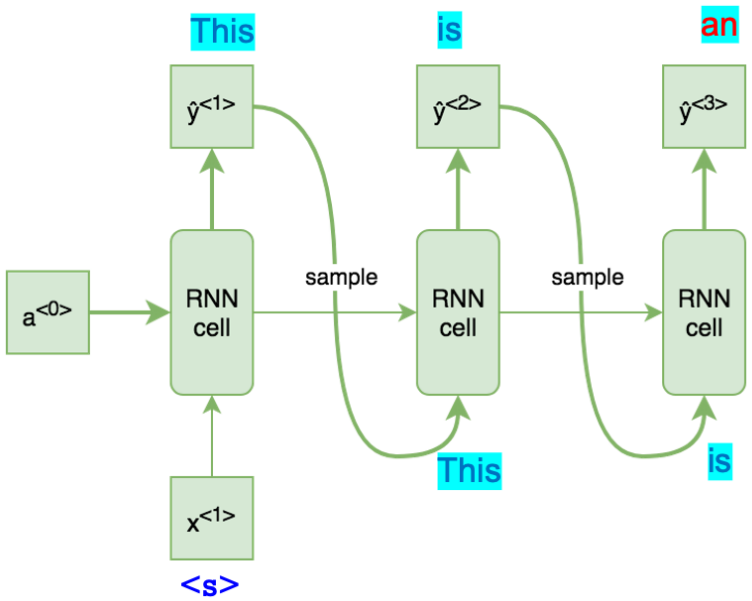
Attends to

Input query

	<s>	This	is	a	pen
<s>		M	M	M	M
This			M	M	M
is				M	M
a					M
pen					

Autoregressive model (inference)

- Inference or decoding is done by predicting token by token (softmax layer)
- Softmax and input use the same tokens (vocabulary)
- This inference operation is serial (cannot parallize)

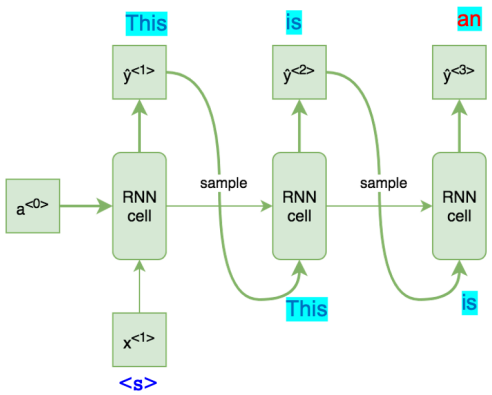


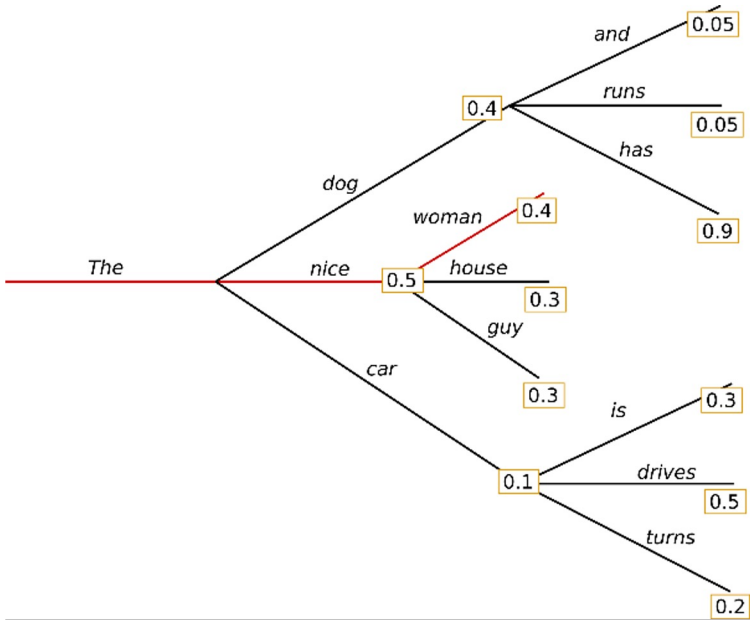
Decoding (inference) for autoregressive model

In sequence generation tasks, the task of selecting what the model outputs as a prediction is called **decoding**.

There are 3 methods for decoding:

1. Greedy decoding
2. Random sampling
3. Beam search



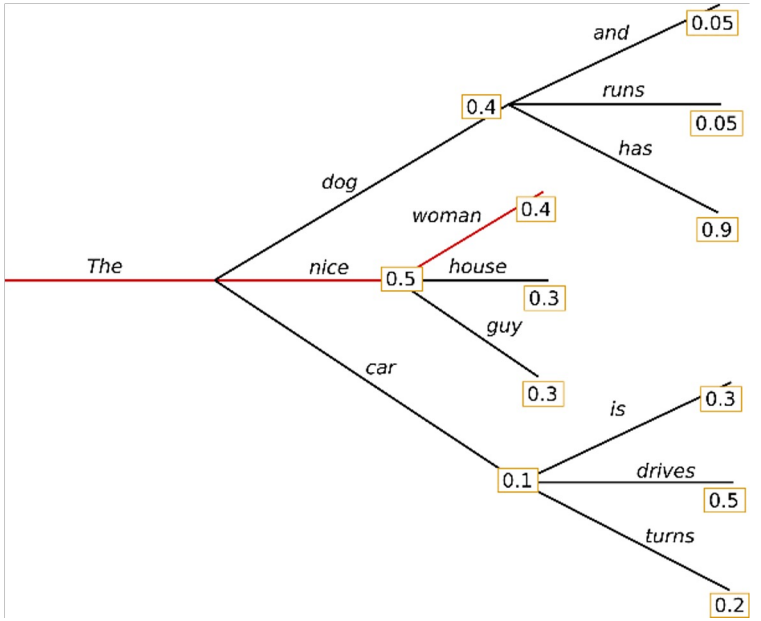


https://huggingface.co/blog/how-to-generate

Greedy decoding

Greedy decoding simply selects the token with **the highest probability** as the next token.

As shown in the picture, after “the”, the continuation with the highest probability is the word “nice” therefore it is selected as the next token. This is done until it reaches the model’s max sequence length or upon encountering an end-of-sentence token.



Greedy decoding

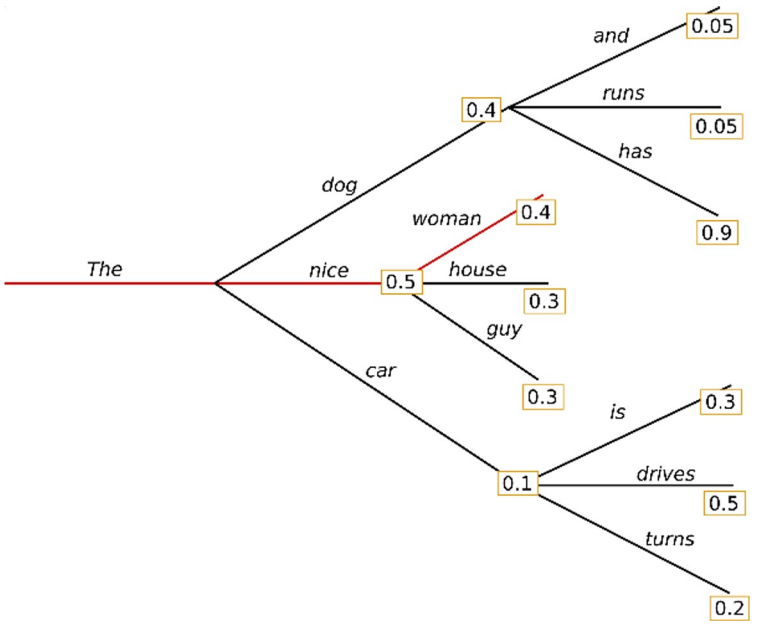
- Greedy decoding is fast and simple, however, the generated text is usually sub-optimal. Sometimes, the model can even repeat itself.

```
He began his premiership by forming a five-man war cabinet which included
Chamerlain as Lord President of the Council, Labour leader Clement Attlee as
Lord Privy Seal (later as Deputy Prime Minister), Halifax as Foreign Secretary
and Labour's Arthur Greenwood as a minister without portfolio. In practice,
+ the cabinet was divided into three parts: the Cabinet of Ministers, the
+ Cabinet of Ministers of the Crown, and the Cabinet of Ministers of the Crown.
+ The Cabinet of Ministers was the most important part of the government. The
+ Cabinet of Ministers was the most important part of the government. The
+ Cabinet of Ministers was the most important part of the government. The
+ Cabinet of Ministers was the most important part of the government. The
+ Cabinet of Ministers was the most important part of the government. The
+ Cabinet of Ministers was the most important part of the government. The
+ Cabinet of Ministers was the most important part of the government. The
+ Cabinet of Ministers...
```

Random sampling

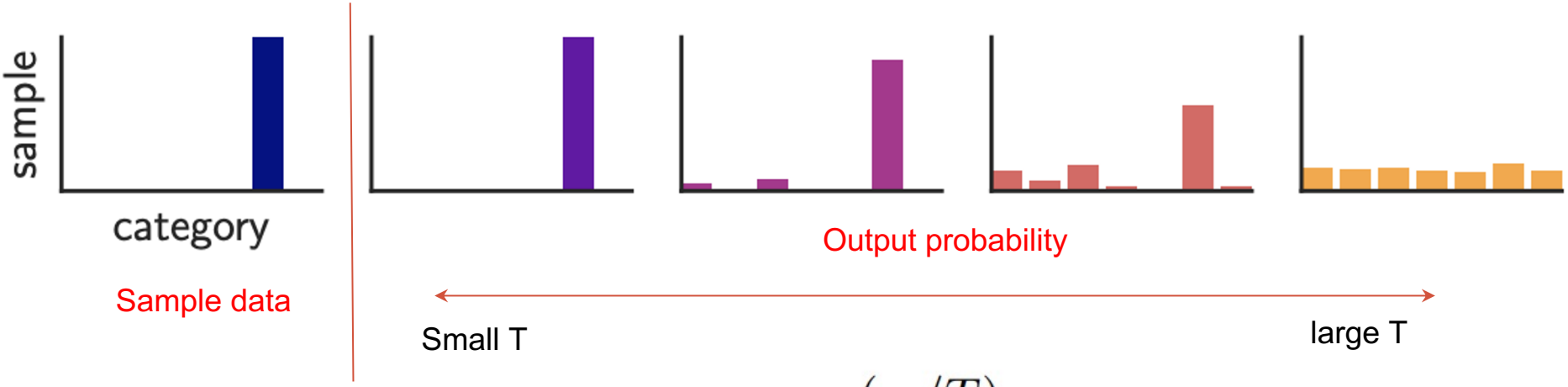
Random sampling chooses the next token **based on the probabilities**.

Using random sampling, the probability of selecting the token “nice”, “dog”, and “car” as the continuation is 50%, 40%, and 10%, respectively. The decoding also proceeds until reaching max sequence length or encountering the end-of-sentence token.



Temperature

- In a softmax function, we can add a temperature term to encourage lower probability items
  - In a maximum likelihood setup, rare items are often under-estimated
  - Can also used to boost the gradient size during training

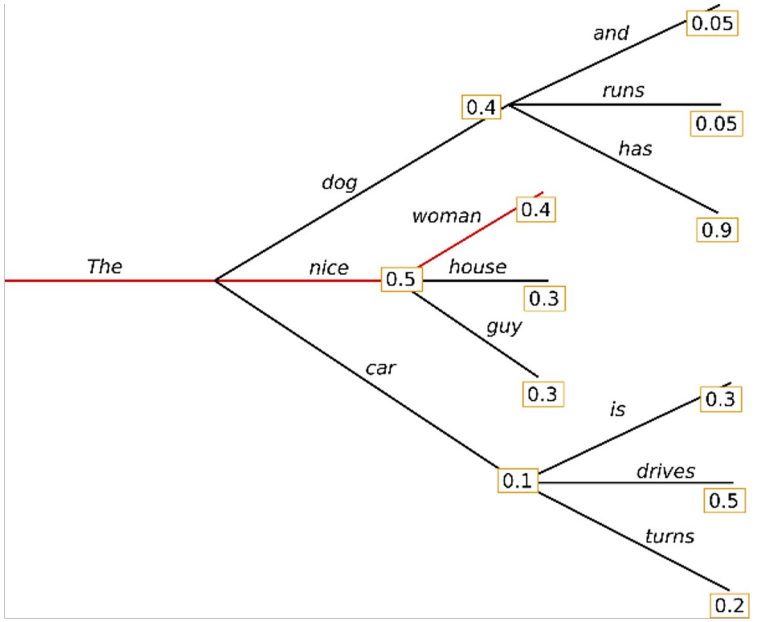


$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Random sampling

By the laws of probability, you are bound to eventually generating something gibberish by selecting multiple low probability tokens in a row.

To prevent this problem, top-k and top-p (nucleus sampling) are often used to improve the generation quality.



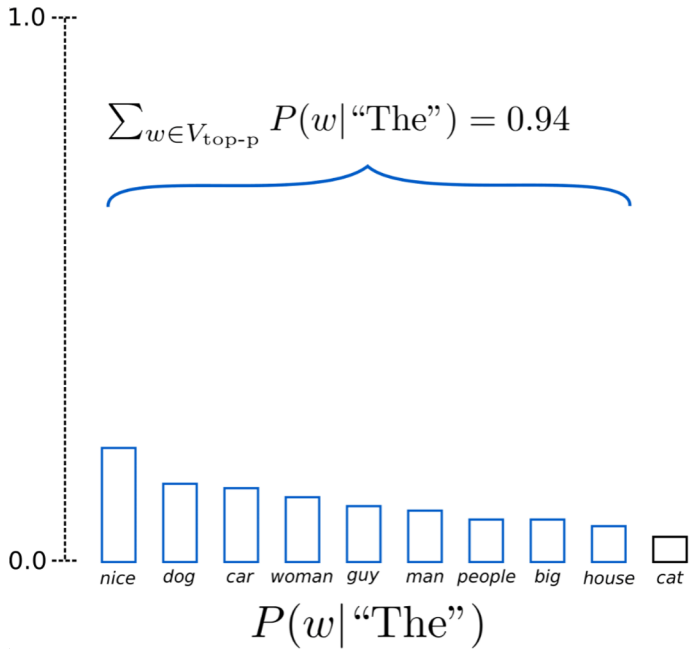
Random sampling

Top-k sampling simply limits the token selection to just top k (usually 20-40) words with the highest probabilities.

Top-p sampling or nucleus sampling dynamically limits the number of words by setting a probability threshold. Top-p sampling chooses from the smallest possible set of tokens whose cumulative probability exceeds the probability threshold.

For example, if we set p as 0.92, top-p sampling will select the *minimum* number of tokens whose cumulative probability is more than 92%

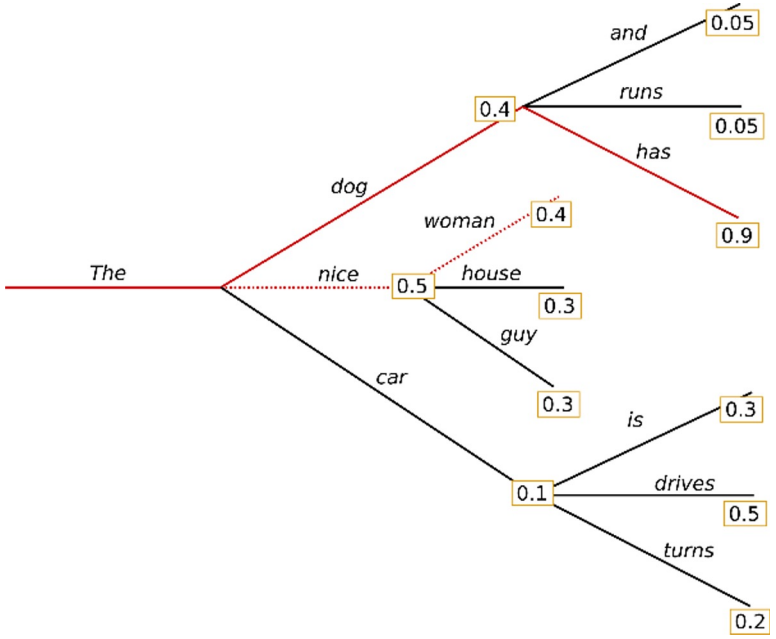
Note that both top-k and top-p can also be used together.



Beamsearch

The two techniques mentioned before selects the next token only based on its probability. On the other hand, Beam search allows us to explore further into each continuation until completion before choosing.

Beam search is relatively computationally expensive since it basically generates multiple sequences but it always find an output sequence that is more probable than greedy decoding.



Beamsearch

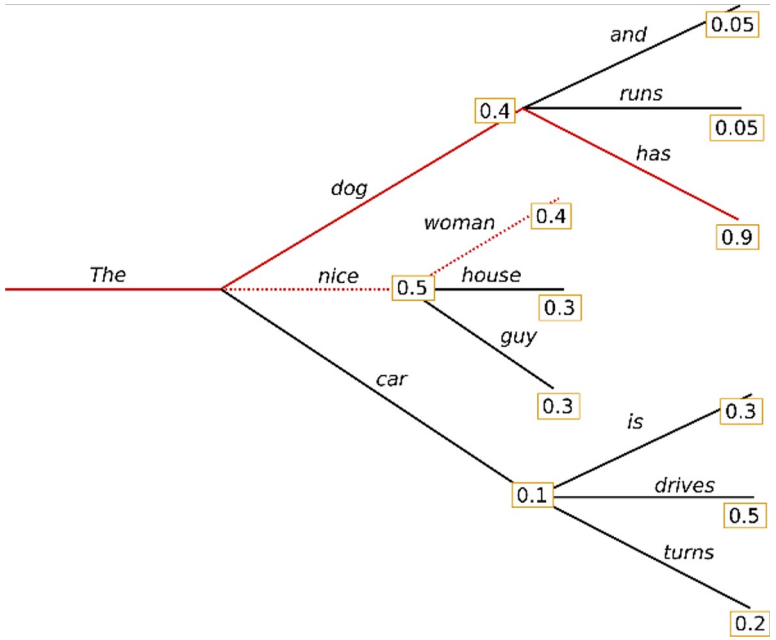
From the example, we consider 2 “beams”, i.e., we only keep the top 2 most probable sequence while we go through the generation process.

At time step 1, the beam search algorithm keeps tab on 2 most probable continuations: (“The”, “nice”) and (“The”, “dog”).

At time step 2, it continues to find the next word for each beam:

(“The”, “dog”, “has”) = 0.4*0.9 = **0.36**
(“The”, “nice”, “woman”) = 0.5*0.4 = 0.2

Suppose this is the end of the generation, the output of the algorithm will be “The dog has”.



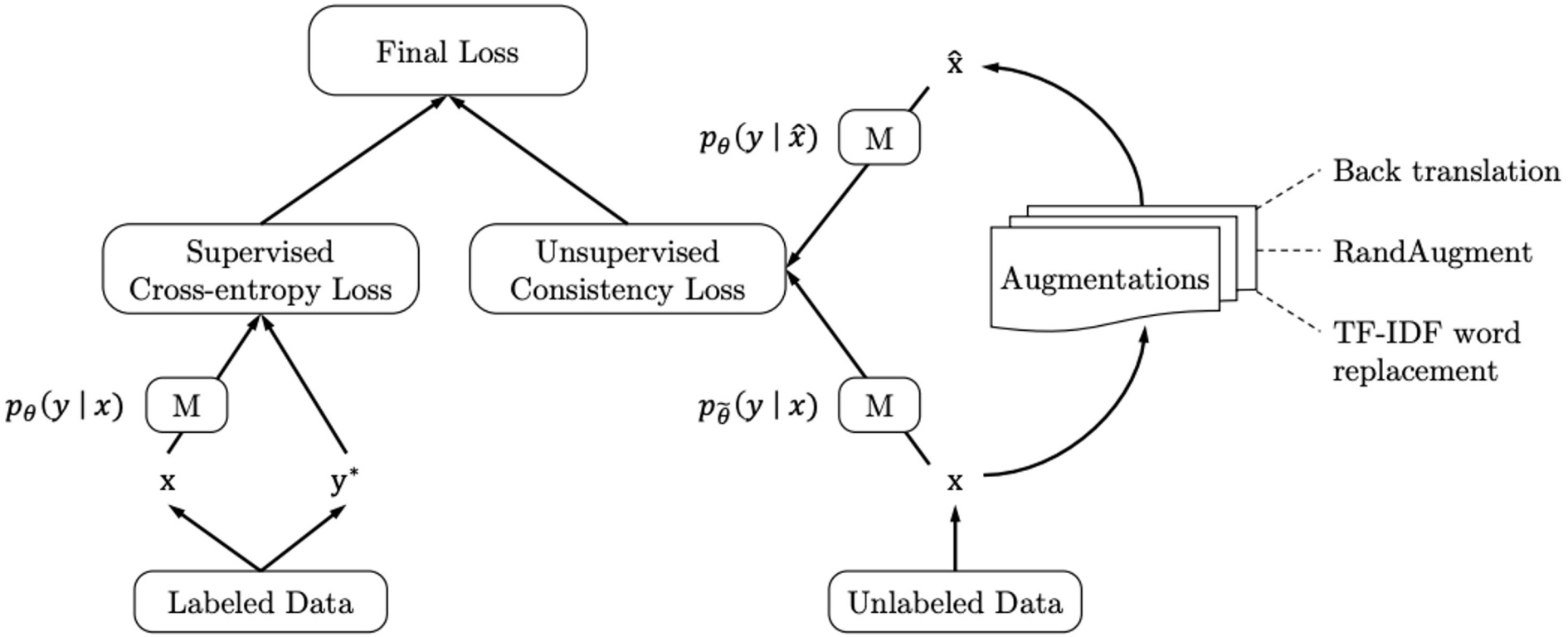
Contrastive learning

Disclaimer

- The subfield evolves fast. Multiple communities working on similar concepts
  - Different names for the same thing
  - Things get rebranded but actually equivalent
  - The same term might mean slightly techniques for different community

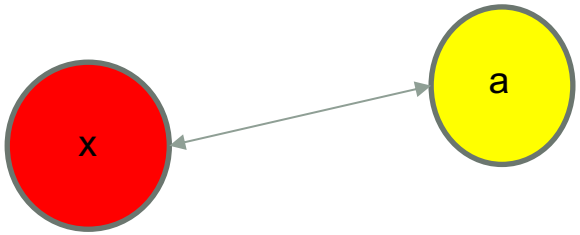
Consistency training

- Consistency loss can be considered as a self-supervised loss

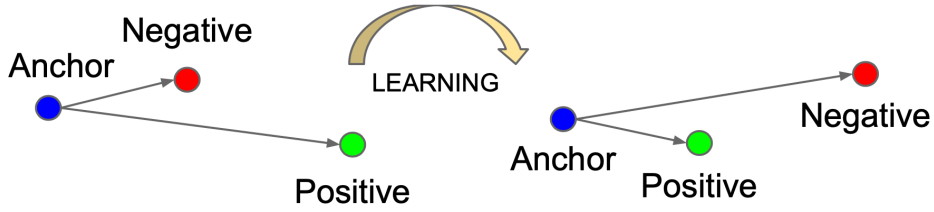


Contrastive training

- Consistency training focus on pulling similar things together while ignoring noise
- Contrastive training focus on pushing different things away
- Contrastive loss are key in face verficiation task
- These two are often times used together, and the clear differentiation between the two are vague

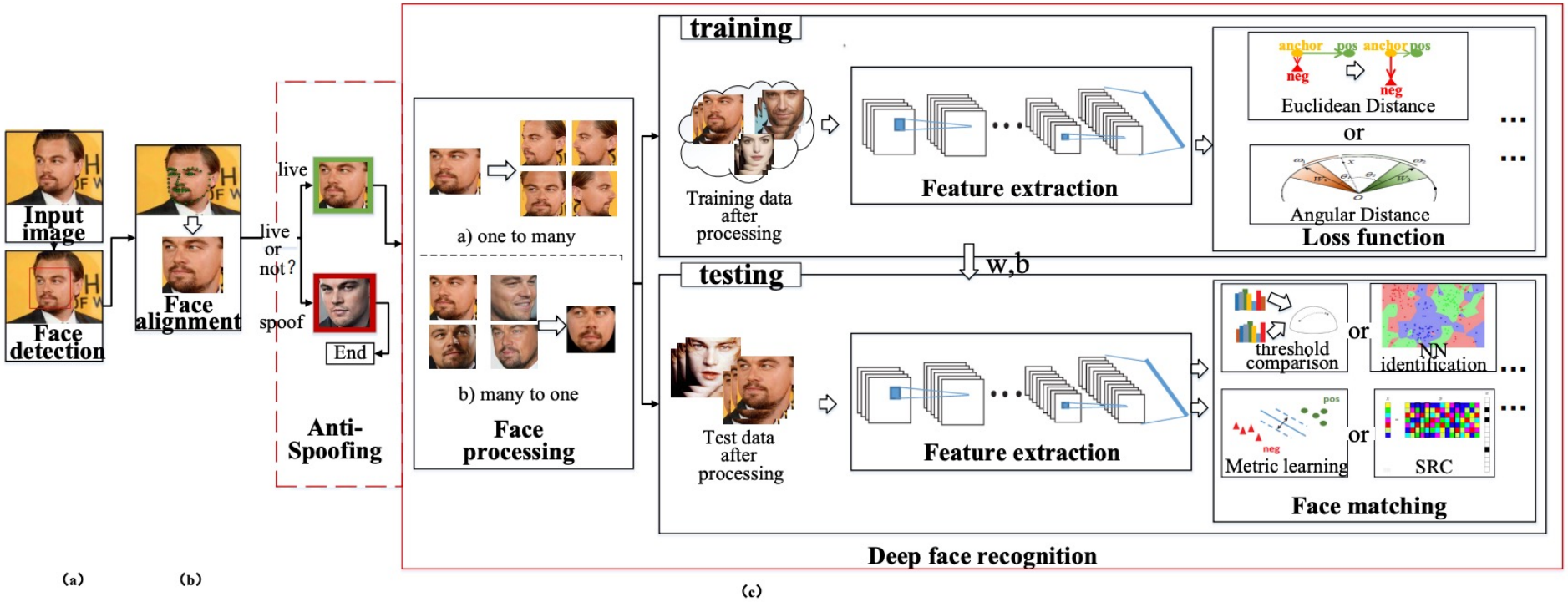


https://arxiv.org/abs/1503.03832
Deepface 2014 (Siamese network)



https://arxiv.org/abs/1503.03832
FaceNet 2015

Deep face verification



https://arxiv.org/pdf/1804.06655.pdf

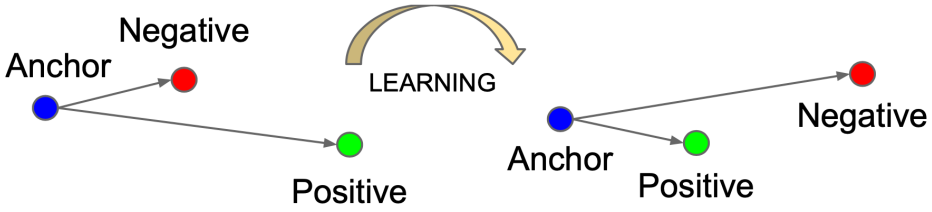
The next set of slides are all supervised frameworks

Triplet loss

- One of the earliest deep learning contrastive loss
- Positive must be closer to anchor than some margin
- Uses Euclidean distance (features are normalized to unit norm)

$$\sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

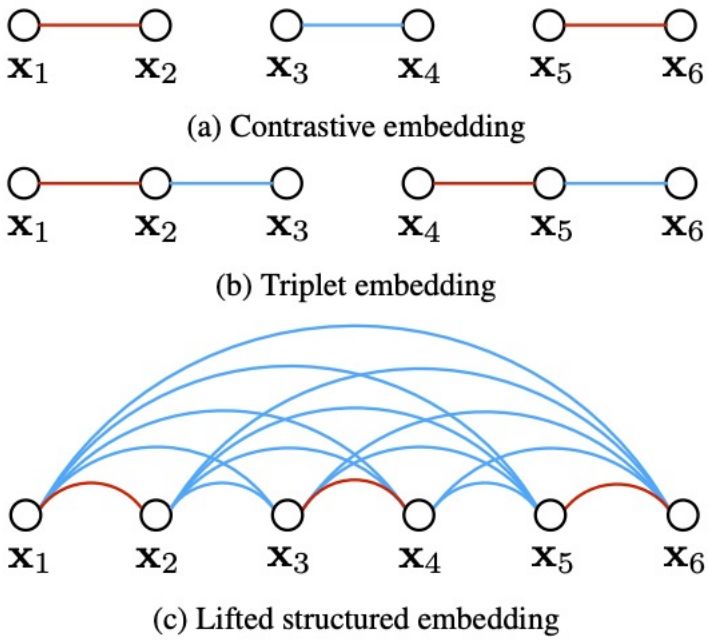
Take positive only max(0,x)
Makes gradient not smooth



https://arxiv.org/abs/1503.03832

Dealing with minibatches

- Since we train in minibatches, most modern losses pair positive and negative samples within a minibatch for more efficient computation (in-batch negative)
  - Compute all pairwise distance within the minibatch



https://arxiv.org/pdf/1511.06452.pdf

NCE (Noise constrastive estimation) loss (2010)

- Maximize training data probability while reducing noise probability
- Learn in a constrastive way to reduce overhead for normalization (energy-based models)
  - Max LogP(data) – Log P(noise or negative samples)
  - Ex: used to train word embeddings such as W2V, too many classes in the softmax output

http://proceedings.mlr.press/v9/gutmann10a.html

InfoNCE

- Similar to NCE but just for categorical cross entropy (instead of binary cross entropy) https://arxiv.org/pdf/1807.03748.pdf
- Given a context vector c, the positive x should be selected rather than the negative x
- Effectively maximize mutual information between c and positive x

Note that this is a weighted cosine similiarity instead of Euclidean distance. But in general f can be any distance/similarity measure.

$$L_{InfoNCE} = -E[log\frac{f(x,c)}{\sum_{x'} f(x',c)}]$$

$$f(x,c) = exp(\mathbf{z}^T W c)$$

z is encoded x

- f( ) can be any function that describes similarity
- Can be extended to have multiple positive examples in a batch (soft nearest neighbor loss) https://arxiv.org/abs/1902.01889

InfoNCE (2018) – N-pair loss (2016)

- A similar paper to InfoNCE, N-pair loss use the same concept
- Use a softmax-like function to keep positive together and push negative away
- A common form is something like this

$$InfoNCE\ Loss_a = \frac{1}{B \cdot P} \sum_{i=1}^{B} \sum_{p \in P(i)} \frac{e^{cos(f_i, f_p)}}{e^{cos(f_i, f_p)} + \sum_{n \in N(i)} e^{cos(f_a, f_n)}}$$

Might also use temperature here

Notation

- B = batch size
- P = the number of positive images per anchor
- i = index of the anchor feature vector
- $P(i)$ = the sampled positive feature vectors
- $N(i)$ = the sampled negative feature vectors
- $cos(f_i, f_p)$ = cosine similarity between f_i and f_p
- f_i = anchor feature vector
- f_p = positive feature vector
- f_n = negative feature vector

https://papers.nips.cc/paper/2016/hash/6b180037abbebea991d8b1232f8a8ca9-Abstract.html

Soft nearest neighbor loss (2019)

- Multiple positive and negative
- Adds temperature (either hyperparameter, or learned)

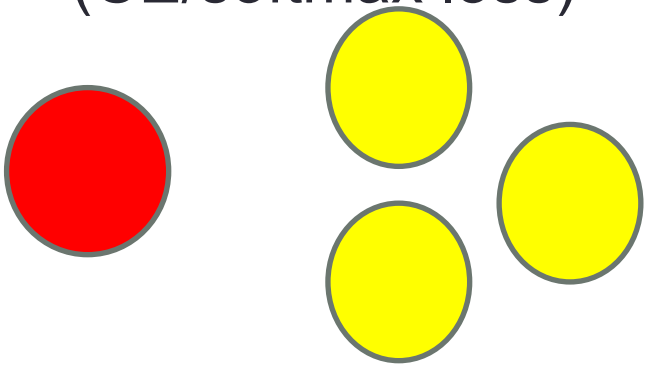
Definition. The *soft nearest neighbor loss* at temperature T, for a batch of b samples (x, y), is:

$$l_{sn}(x, y, T) = -\frac{1}{b} \sum_{i \in 1..b} \log \left(\frac{\sum_{\substack{j \in 1..b \\ j \neq i \\ y_i = y_j}} e^{-\frac{||x_i - x_j||^2}{T}}}{\sum_{\substack{k \in 1..b \\ k \neq i}} e^{-\frac{||x_i - x_k||^2}{T}}} \right) \quad (1)$$

https://arxiv.org/pdf/1902.01889.pdf

Key details to make contrastive learning work

- Large batch (There are works that show batch size of 10,000+ helps)
- Temperature is very important to tune
- Hard/semi-hard negative mining (less important with large batch
- Augmentation on the anchor and postive (consistency training)
- Other improvement includes - adding classification loss (CE/softmax loss)



Softmax/angular-based loss

- Another popular construction of the loss is based on angular distance
- Consider a regular softmax/CE loss (only the correct class term, the wrong class is zerod out)

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

- Can be written as, where W is the weight associated with the class

$$L_i = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i})}}{\sum_j e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right)$$

Margin in the softmax (L-softmax)

- To classify as class 1

$$\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2)$$

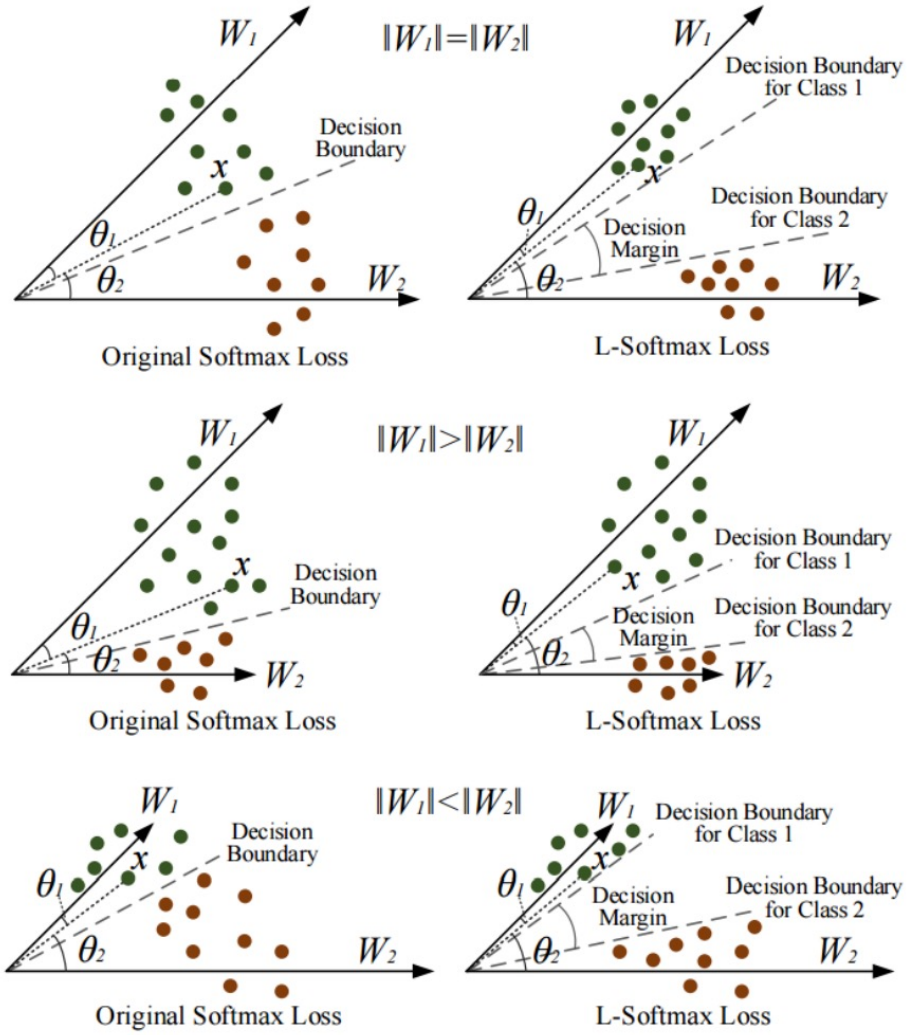
must be true

- We introduce an angular margin m

$$\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) \geq \|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(m\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2).$$

$$(0 \leq \theta_1 \leq \frac{\pi}{m})$$

Angular margin effect (2016)



https://wyliu.com/papers/LiuICML16_slides.pdf

Center loss (2016)

- Makes the data bunch up towards the centroid
- Unlike k-mean cluster, centroid is learned via gradient descent (for speed)
- Iterative update between centroid, feature, and classification

Algorithm 1. The discriminative feature learning algorithm

Input: Training data $\{\boldsymbol{x}_i\}$. Initialized parameters θ_C in convolution layers. Parameters W and $\{\boldsymbol{c}_j | j = 1, 2, ..., n\}$ in loss layers, respectively. Hyperparameter λ, α and learning rate μ^t. The number of iteration $t \leftarrow 0$.

Output: The parameters θ_C.

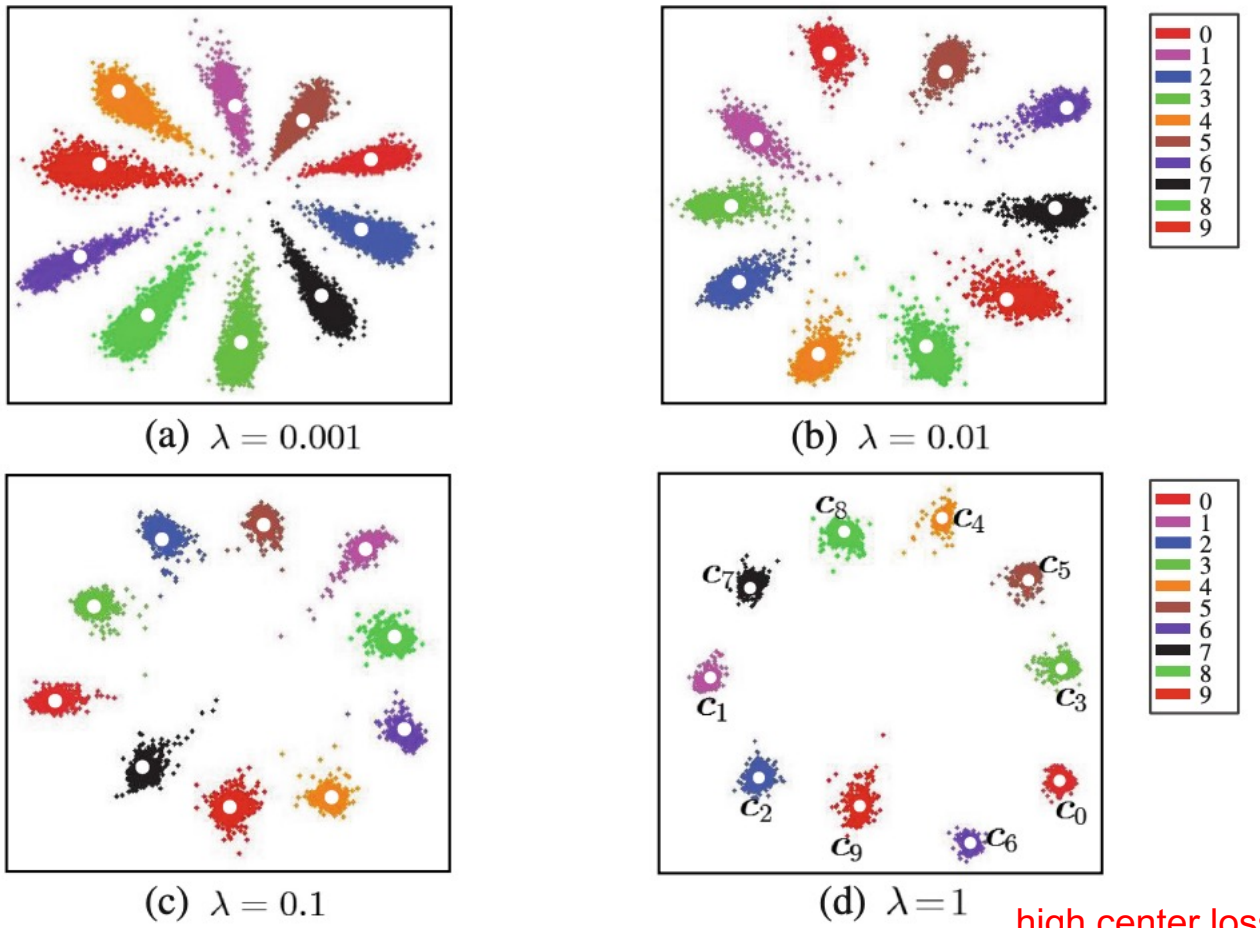
1: **while** not converge **do**
2: $t \leftarrow t + 1$.
3: Compute the joint loss by $\mathcal{L}^t = \mathcal{L}_S^t + \mathcal{L}_C^t$.
4: Compute the backpropagation error $\frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t}$ for each i by $\frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t} = \frac{\partial \mathcal{L}_S^t}{\partial \boldsymbol{x}_i^t} + \lambda \cdot \frac{\partial \mathcal{L}_C^t}{\partial \boldsymbol{x}_i^t}$.
5: Update the parameters W by $W^{t+1} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}^t}{\partial W^t} = W^t - \mu^t \cdot \frac{\partial \mathcal{L}_S^t}{\partial W^t}$.
6: Update the parameters $\boldsymbol{c}_j$ for each j by $\boldsymbol{c}_j^{t+1} = \boldsymbol{c}_j^t - \alpha \cdot \Delta \boldsymbol{c}_j^t$.
7: Update the parameters θ_C by $\theta_C^{t+1} = \theta_C^t - \mu^t \sum_i^m \frac{\partial \mathcal{L}^t}{\partial \boldsymbol{x}_i^t} \cdot \frac{\partial \boldsymbol{x}_i^t}{\partial \theta_C^t}$.
8: **end while**

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_C$$

$$= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

https://ydwen.github.io/papers/WenECCV16.pdf

low center loss



high center loss

Fig. 3. The distribution of deeply learned features under the joint supervision of softmax loss and center loss. The points with different colors denote features from different classes. Different λ lead to different deep feature distributions ($\alpha = 0.5$). The white dots ($\boldsymbol{c}_0$, $\boldsymbol{c}_1$,...,$\boldsymbol{c}_9$) denote 10 class centers of deep features. **Best viewed in color.** (Color figure online)

Arc face (2018)

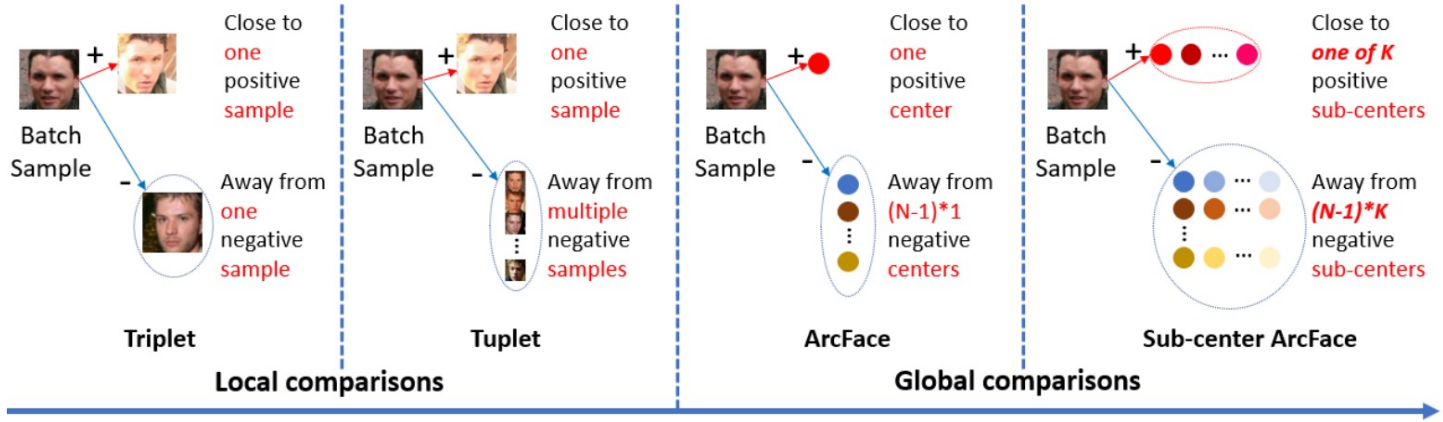


Fig. 1. Comparisons of Triplet [3], Tuplet [12], ArcFace and sub-center ArcFace. Triplet and Tuplet conduct local sample-to-sample comparisons with Euclidean margins within the mini-batch. By contrast, ArcFace and sub-center ArcFace conduct global sample-to-class and sample-to-subclass comparisons with angular margins.

$$L_4 = -\log \frac{e^{s(\cos(m_1\theta_{y_i}+m_2)-m_3)}}{e^{s(\cos(m_1\theta_{y_i}+m_2)-m_3)} + \sum_{j=1,j\neq y_i}^{N} e^{s\cos\theta_j}}. \quad (4)$$

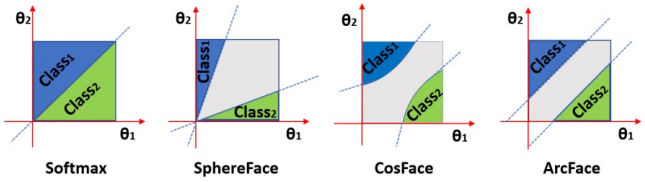
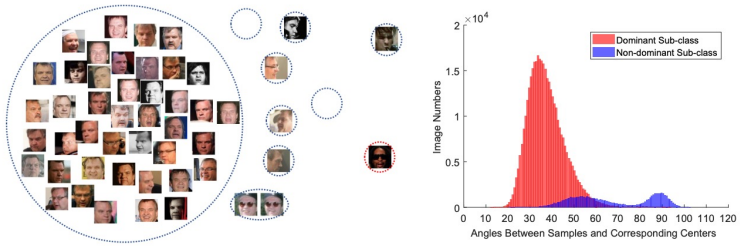


Fig. 5. Decision margins of different loss functions under binary classification case. The dashed line represents the decision boundary, and the grey areas are the decision margins.



(a) Example of Sub-classes (b) Clean Data Isolation

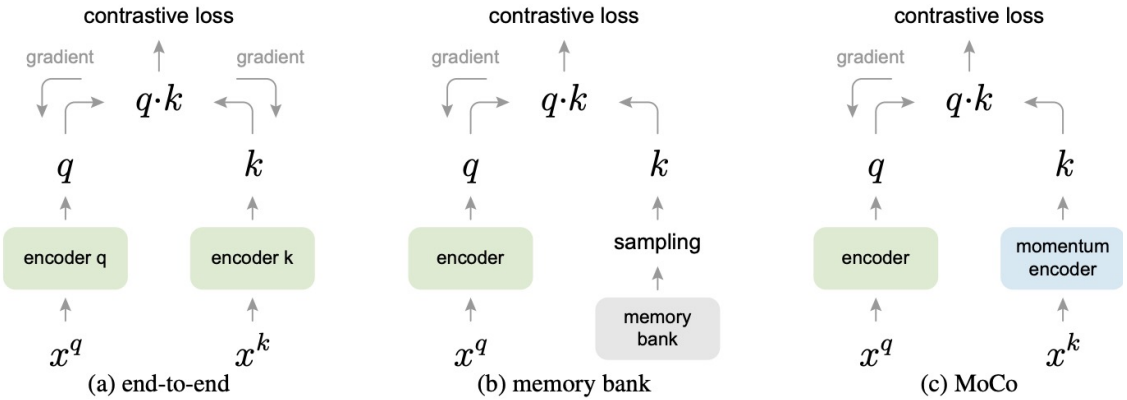
Fig. 6. (a) The sub-classes of one identity from the CASIA dataset [56] after using the sub-center ArcFace loss ($K = 10$). Noisy samples and hard samples (e.g. profile and occluded faces) are automatically separated from the majority of clean samples. (b) Angle distribution of samples from the dominant and non-dominant sub-classes. Clean data are automatically isolated by the sub-center ArcFace.

https://arxiv.org/abs/1801.07698

Dealing with large batch size

- So if we have a large batch, let's say 65536. We have to do 65k forward passes for 1 batch update!
- Is there a better way?
  - Use cache eg buffer/queue for the instances

Momentum Contrast for Unsupervised Visual Representation Learning (MOCO) 2019



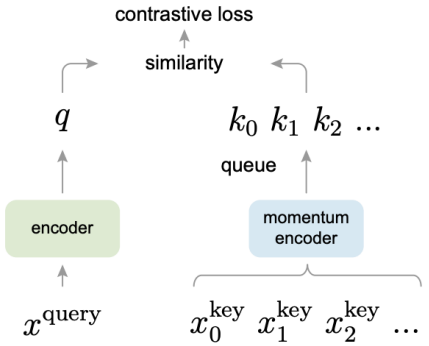
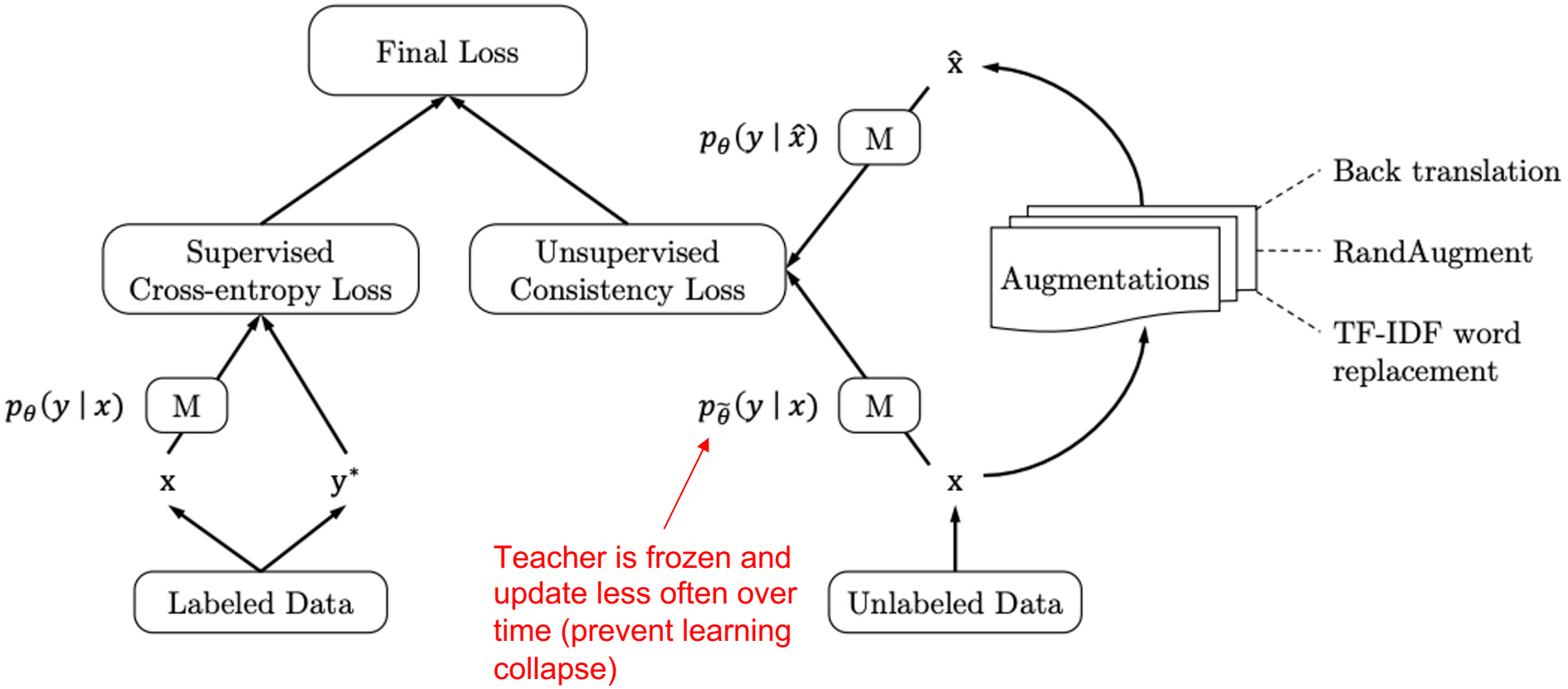


Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, ...\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from the mini-batch size. The keys are encoded by a slowly progressing encoder, driven by a momentum update with the query encoder. This method enables a large and consistent dictionary for learning visual representations.

https://arxiv.org/abs/1911.05722

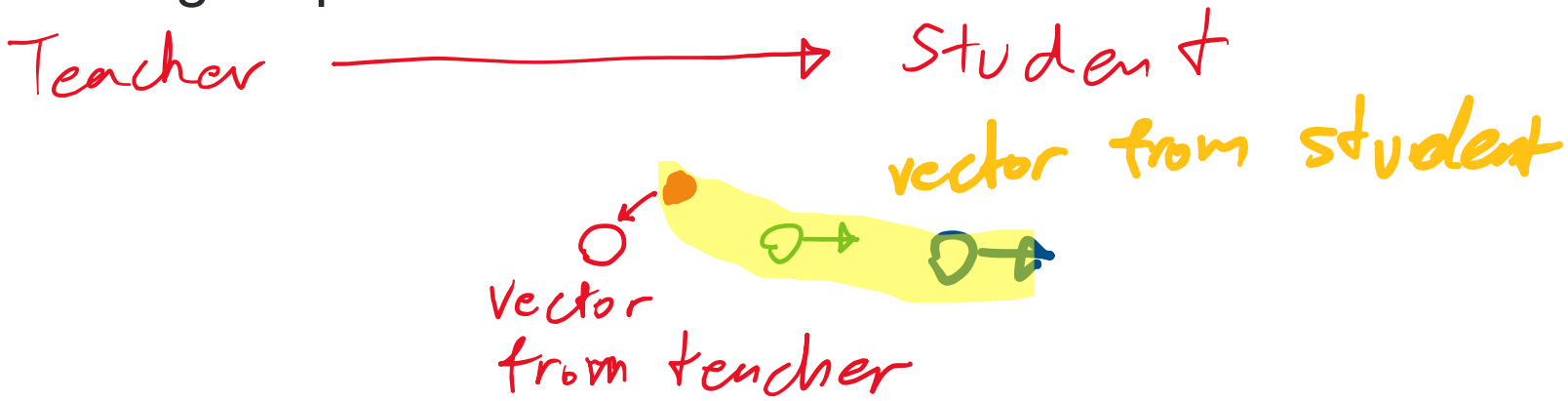
Back to unsupervised/self-supervised

The key difference between supervised and unsupervised contrastive is how we get the positive example
Unsupervised contrastive – use an augmented version of itself
Supervised contrastive – use another data point from the same class

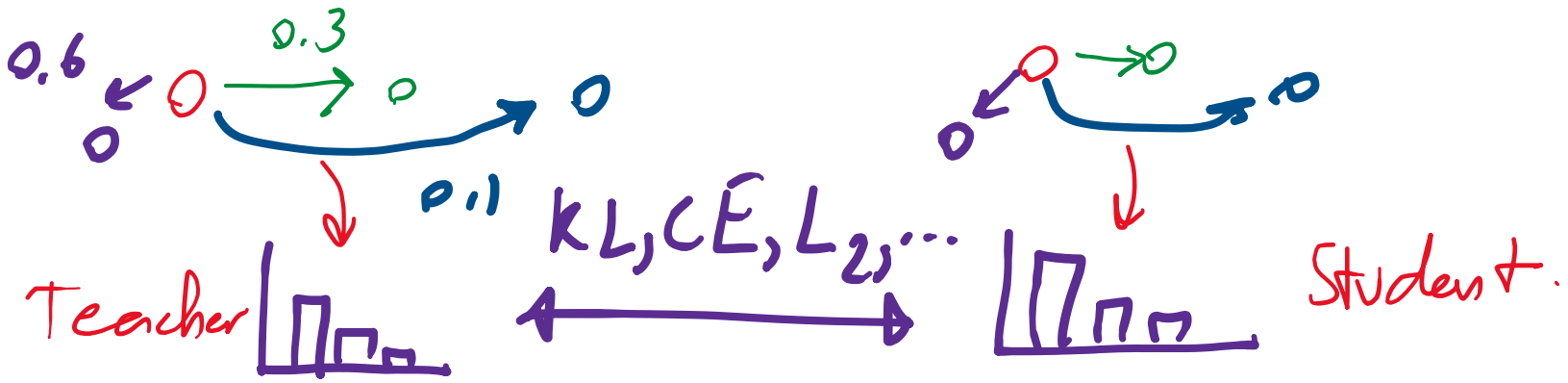


Contrastive on a distribution rather than points

- Learning on points:



- Learning on distributions



Unsupervised contrastive details (redux)

- How to construct the teacher? (average/latest) – stabilize learning
- How to post process labels? (hard/soft, all/some, sharpen/no) – minimizing entropy
- How to augment? (weak/strong) – coping with OOD
- Loss – L2, CE, KL, Jensen-Shannon

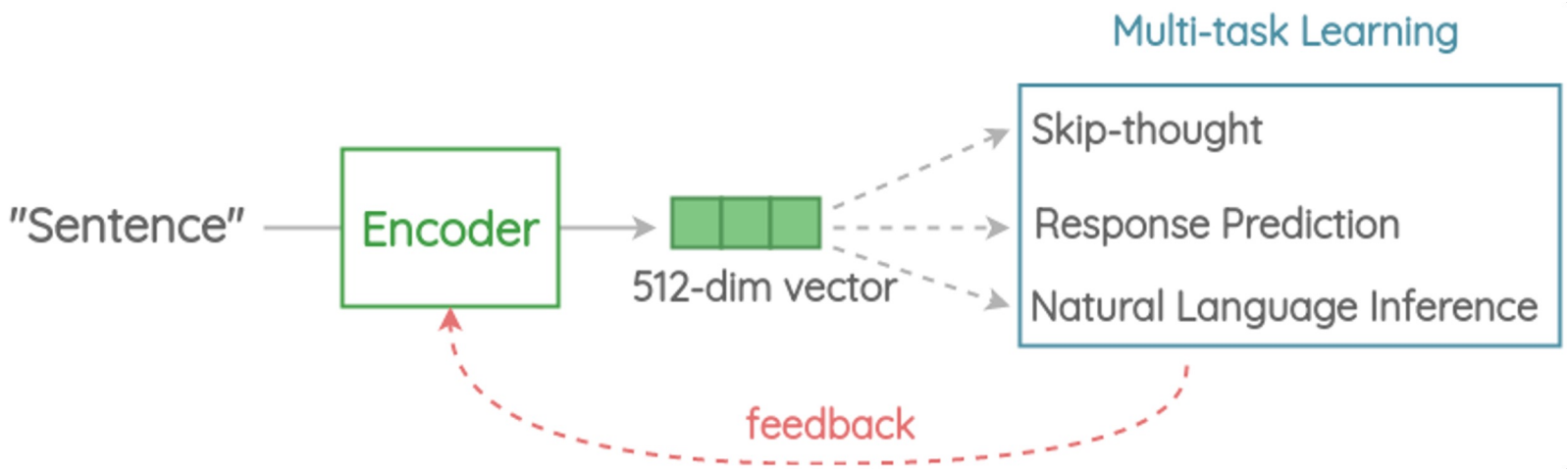
Teacher augment Student augment

Algorithm	Artificial label augmentation	Prediction augmentation	Artificial label post-processing	Notes
TS / Π-Model	Weak	Weak	None	
Temporal Ensembling	Weak	Weak	None	Uses model from earlier in training
Mean Teacher	Weak	Weak	None	Uses an EMA of parameters
Virtual Adversarial Training	None	Adversarial	None	
UDA	Weak	Strong	Sharpening	Ignores low-confidence artificial labels
MixMatch	Weak	Weak	Sharpening	Averages multiple artificial labels
ReMixMatch	Weak	Strong	Sharpening	Sums losses for multiple predictions
FixMatch	Weak	Strong	Pseudo-labeling	

https://arxiv.org/pdf/2001.07685.pdf

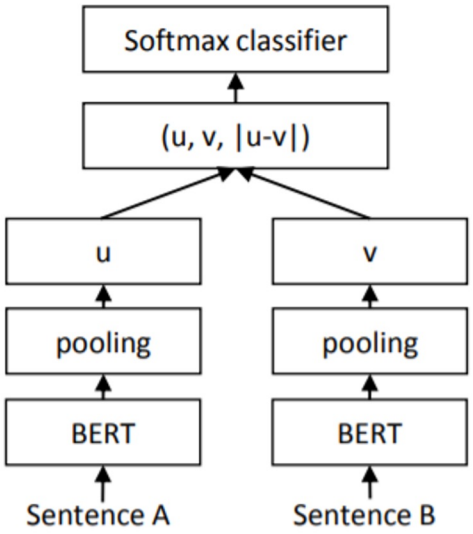
Case study: NLP representation learning (sentence embeddings)

Early days: supervised model
USE https://arxiv.org/abs/1803.11175 2018

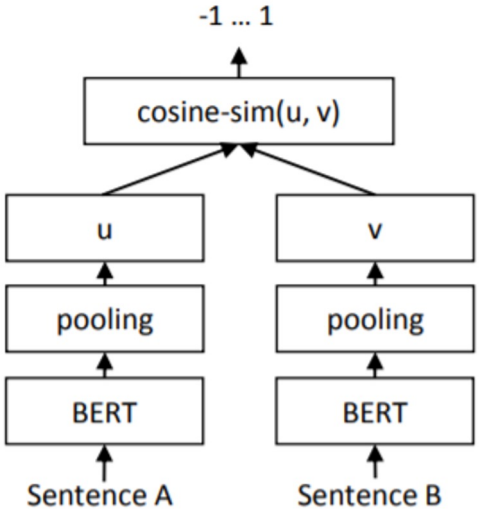


SBERT

Language Understanding



Semantic Understanding



Model	Spearman
Not trained for STS	
Avg. GloVe embeddings	58.02
Avg. BERT embeddings	46.35
InferSent - GloVe	68.03
Universal Sentence Encoder	74.92
SBERT-NLI-base	77.03
SBERT-NLI-large	79.23
Trained on STS benchmark dataset	
BERT-STSb-base	84.30 ± 0.76
SBERT-STSb-base	84.67 ± 0.19
SRoBERTa-STSb-base	**84.92** ± 0.34
BERT-STSb-large	**85.64** ± 0.81
SBERT-STSb-large	84.45 ± 0.43
SRoBERTa-STSb-large	85.02 ± 0.76
Trained on NLI data + STS benchmark data	
BERT-NLI-STSb-base	**88.33** ± 0.19
SBERT-NLI-STSb-base	85.35 ± 0.17
SRoBERTa-NLI-STSb-base	84.79 ± 0.38
BERT-NLI-STSb-large	**88.77** ± 0.46
SBERT-NLI-STSb-large	86.10 ± 0.13
SRoBERTa-NLI-STSb-large	86.15 ± 0.35

https://arxiv.org/abs/1908.10084 2019

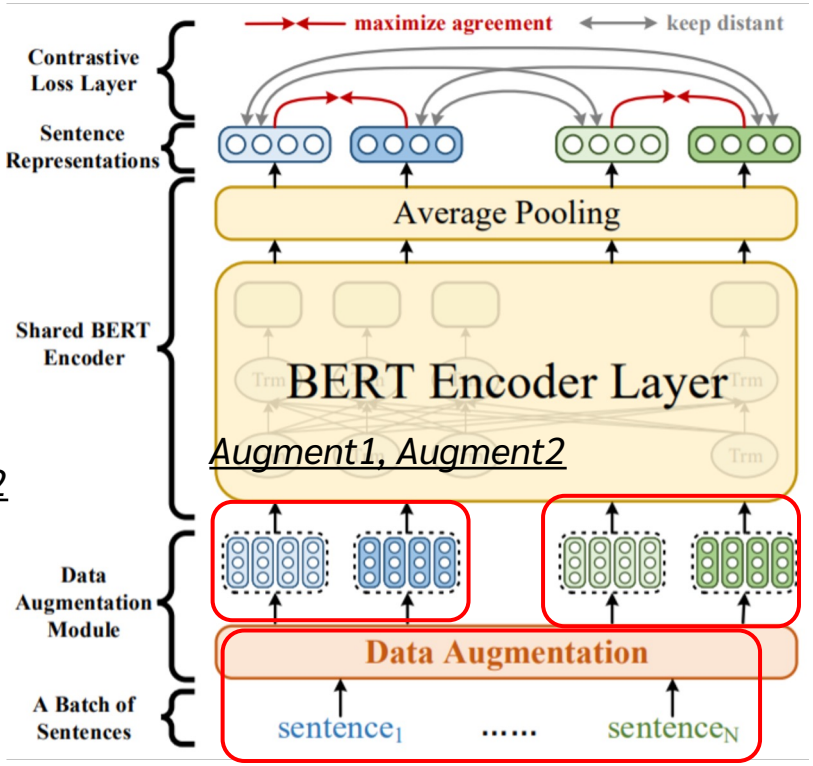
Sentence level contrastive learning

- We can learn better sentence representation with some additional supervised (or unsupervised) sentence level contrastive learning

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\mathrm{sim}(r_i, r_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(r_i, r_k)/\tau)}$$

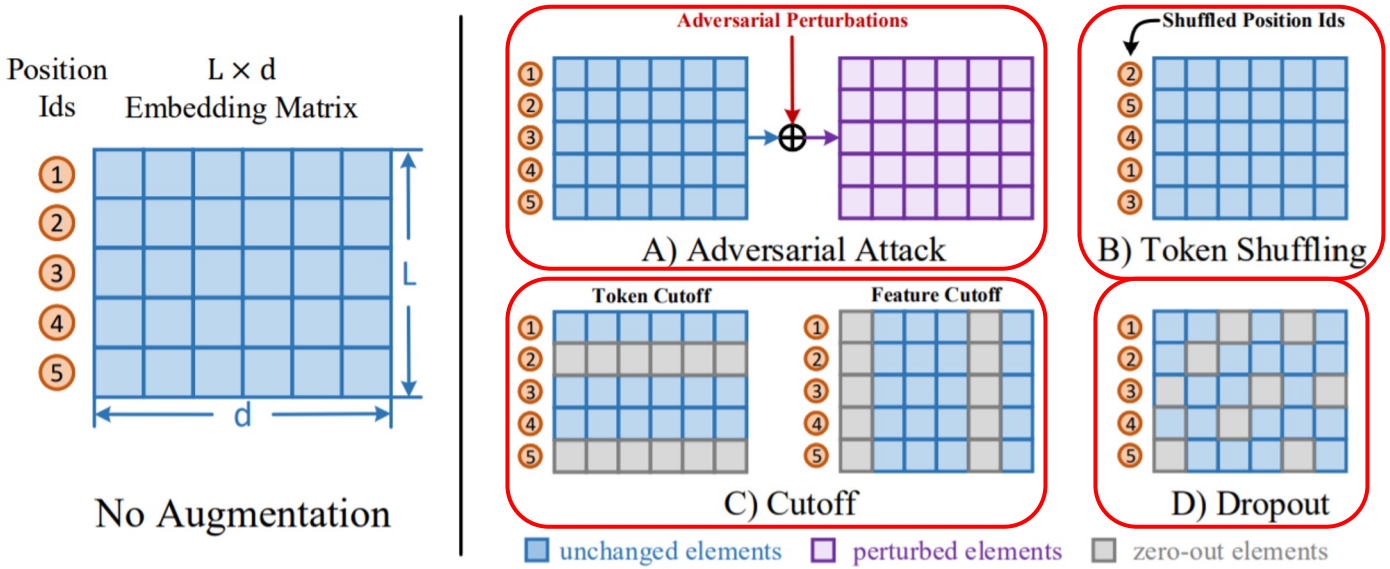
Augment1, Augment2

Not augment2

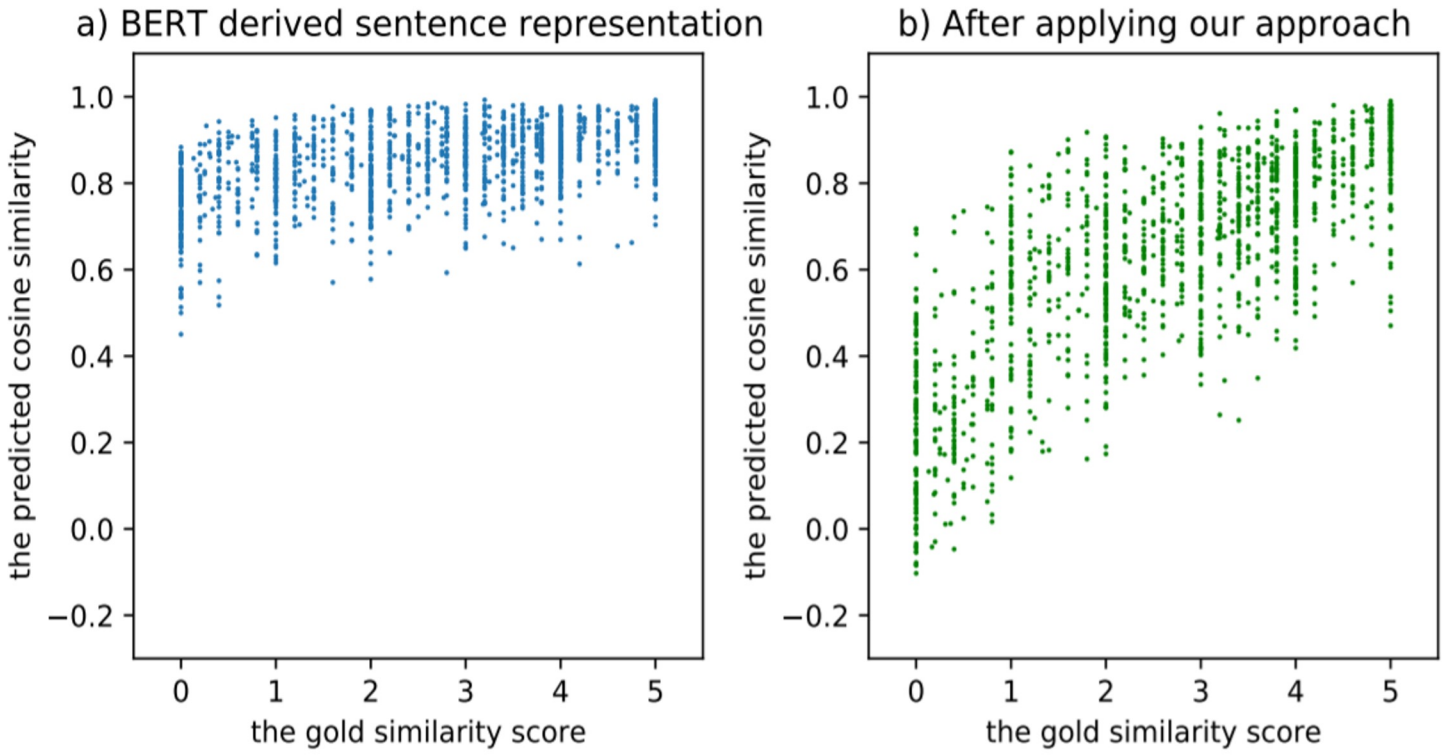


ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer (2021)

ConSERT augmentations

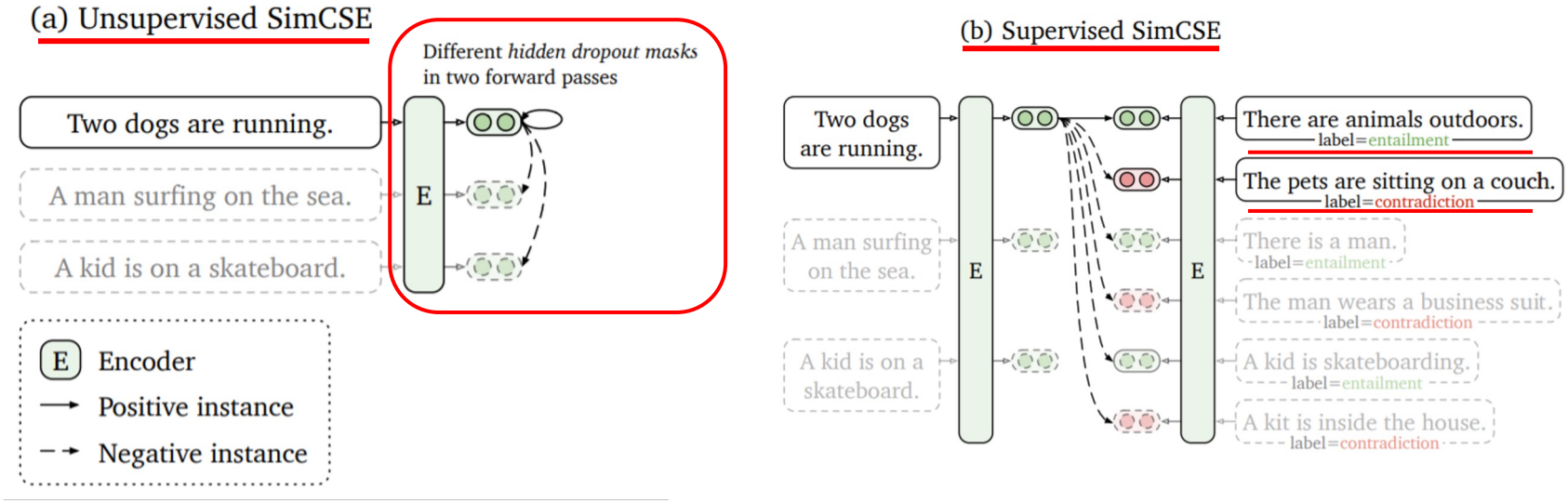


ConSERT alignment



SimCSE

- Use simple dropout in the model to create different versions of the same sentence



SimCSE

Data augmentation			STS-B
None (unsup. SimCSE)			**82.5**
Crop	*10%*	*20%*	*30%*
	77.8	71.4	63.6
Word deletion	*10%*	*20%*	*30%*
	75.9	72.2	68.2
Delete one word			75.9
w/o dropout			74.2
Synonym replacement			77.4
MLM 15%			62.2

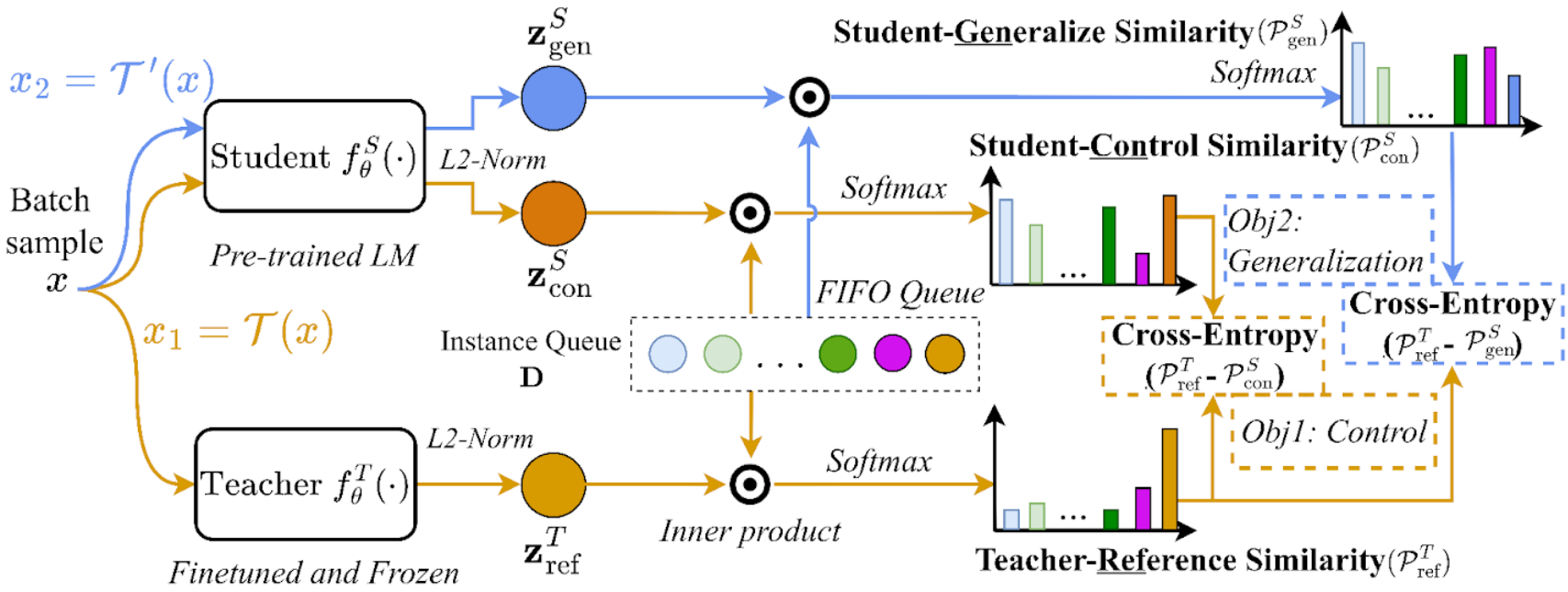
Other augmentations technique

Training objective	f_θ	$(f_{\theta_1}, f_{\theta_2})$
Next sentence	67.1	68.9
Next 3 sentences	67.4	68.8
Delete one word	75.9	73.1
Unsupervised SimCSE	**82.5**	80.7

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(r_i, r_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(r_i, r_k)/\tau)}$$

Rather than contrastive, predict next sentence, 1 of 3 next sentences

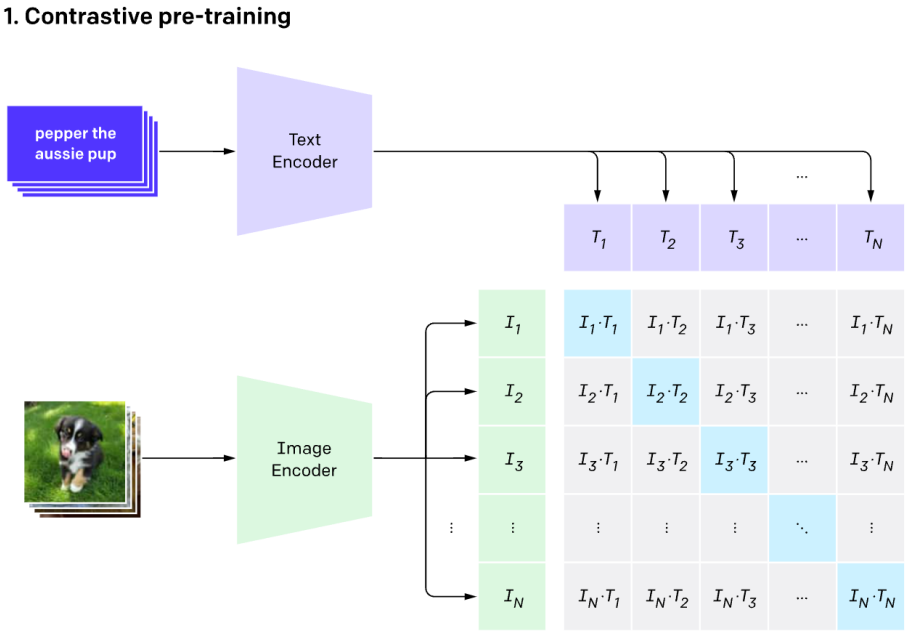
ConGEN (2022)

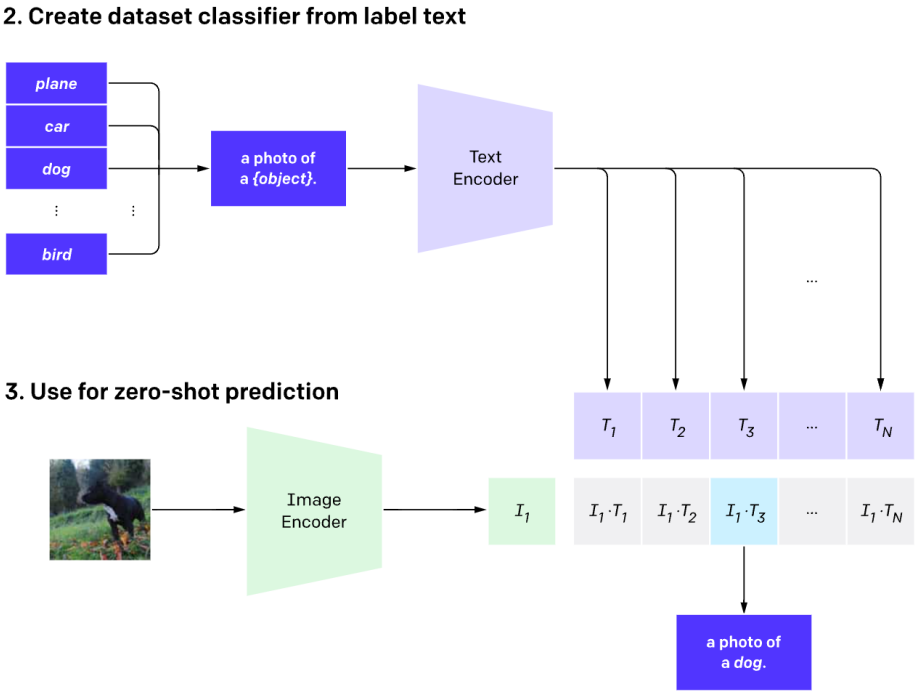


https://aclanthology.org/2022.findings-emnlp.483/
And a more upgrade version SCT https://arxiv.org/abs/2311.03228 (2023)

Vision

CLIP





https://openai.com/research/clip

CLIP

- Can be used for text->image or image->text retreival (dense retreival)
- Can be used for caption generation (Using ZeroCap https://arxiv.org/abs/2111.14447 a kind of embedding inversion)

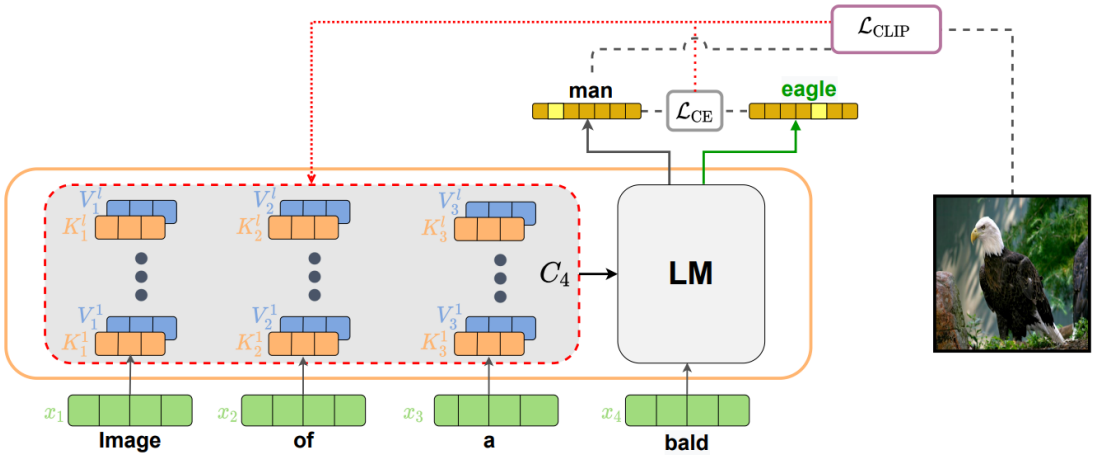


Figure 2. An overview of our approach. We guide the model towards the phrase 'eagle' instead of 'man'. We do this by adjusting the context (C_4), using the gradients of CLIP loss ($\mathcal{L}_{\text{CLIP}}$) illustrated with a red arrow. To maintain language attributes, we optimize the minimum distance to the original distribution of the LM ($\mathcal{L}_{\text{CE}}$).

Embedding inversion

- Simple technique to go from embedding to feature space
  - Boils down to: let's gradient descent from the target to the input feature

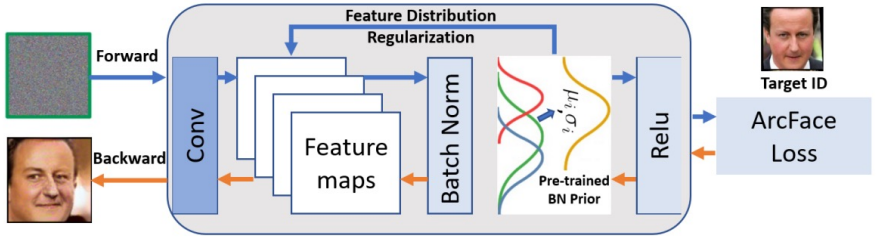


Fig. 8. ArcFace is not only a discriminative model but also a generative model. Given a pre-trained ArcFace model, a random input tensor can be gradually updated into a pre-defined identity by using the gradient of the ArcFace loss as well as the face statistic priors stored in the Batch Normalization layers.

Algorithm 1 Face Image Generation from the ArcFace Model

Input: model $\mathcal{M}$ with L BN layers, class label y_i
Output: a batch of generated face images: I^r
Generate random data I^r from Gaussian $(\mu = 0, \sigma = 1)$
Get μ_i, σ_i from BN layers of $\mathcal{M}$, $i \in 0, \ldots, L$
for j $= 1, 2, \ldots, T$ **do**
  Forward propagate $\mathcal{M}(I^r)$ and calculate ArcFace loss
  Get $\tilde{\mu}_i$ and $\tilde{\sigma}_i$ from intermediate activations, $i \in 0, \ldots, L$
  Compute statistic loss min $\sum_{i=0}^{L} \|\tilde{\mu}_i^r - \mu_i\|_2^2 + \|\tilde{\sigma}_i^r - \sigma_i\|_2^2$,
  Backward propagate and update I^r
end for

https://arxiv.org/abs/1801.07698

Masked auto encoder

- Inspired by BERT. Turns patches into tokens
- Probably the best self-supervised vision training method right now.

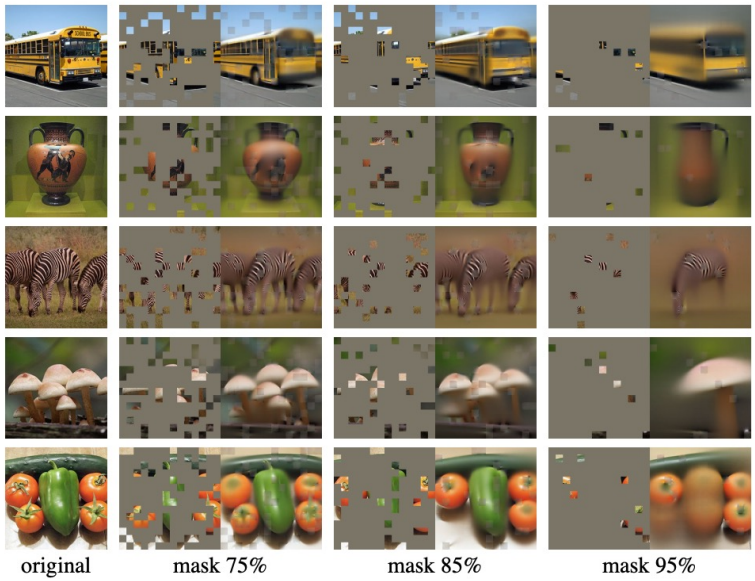


Figure 4. Reconstructions of ImageNet *validation* images using an MAE pre-trained with a masking ratio of 75% but applied on inputs with higher masking ratios. The predictions differ plausibly from the original images, showing that the method can generalize.

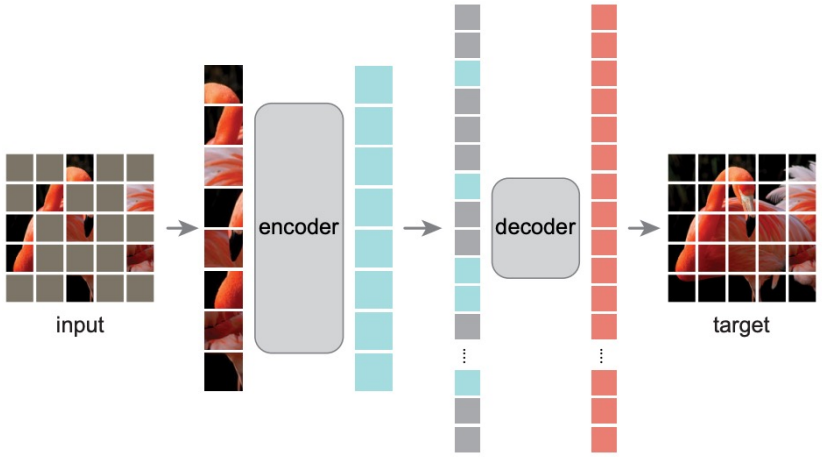


Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images (full sets of patches) for recognition tasks.

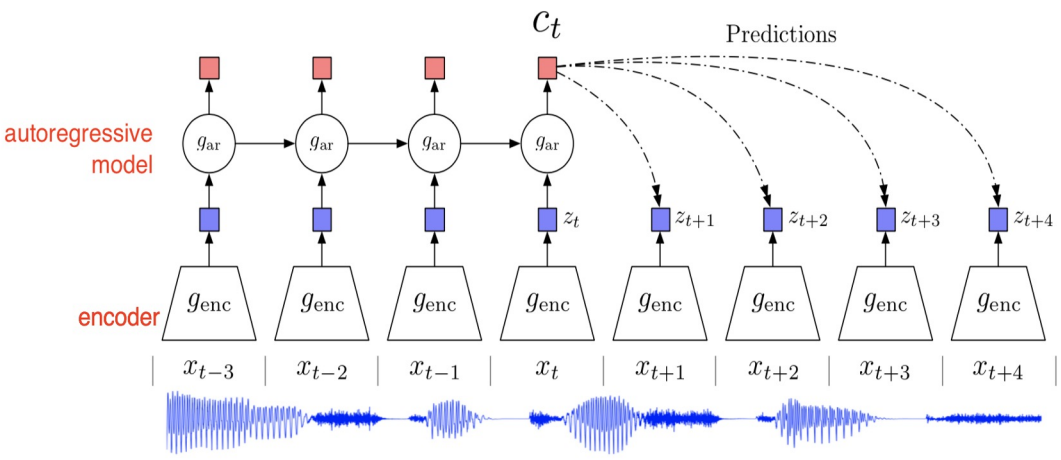
https://arxiv.org/abs/2111.06377

Other Self-supervised Examples

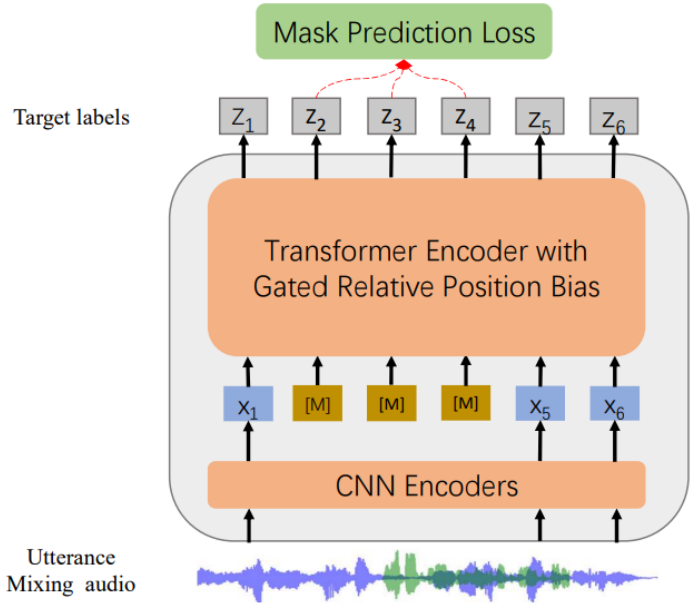
Sound

Cluster data into discrete classes then predict masked portions

More examples here https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html



https://arxiv.org/abs/1807.03748 Representation Learning with Contrastive Predictive Coding



https://arxiv.org/abs/2110.13900
WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing

Summary

Self-supervised

- Contrastive learning tricks!
- How to noise the data

Caution: self-supervised is powerful only if LARGE data.

- Do not try this at home.