Self-supervised learning Part 1

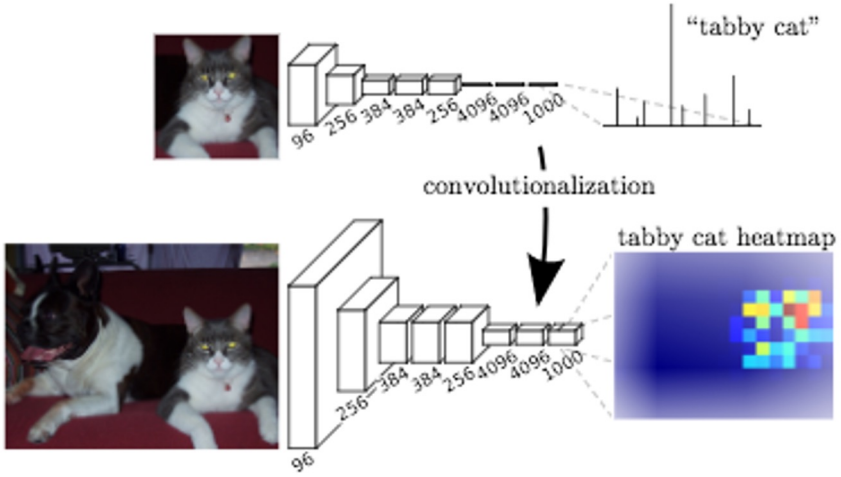
Unsupervised learning
Semi-supervised learning
Self-supervised learning
BERT vs GPT

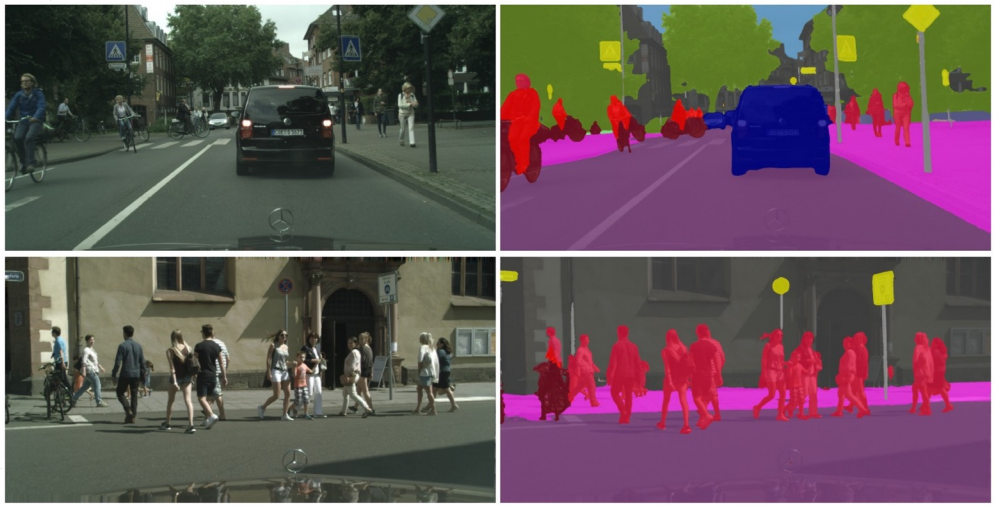
Recap

- Layers
  - Fully connected
  - CNN
  - RNN/GRU
  - Attention
  - Batch/layer/group/instance Norms
  - Dropout
  - Softmax
- Architectures/blocks/connections
  - Transformer
  - Residual connection
  - Depthwise separable convolution (factorized convolution)

Sometimes you want to increase the size of your feature map

Image segmentation

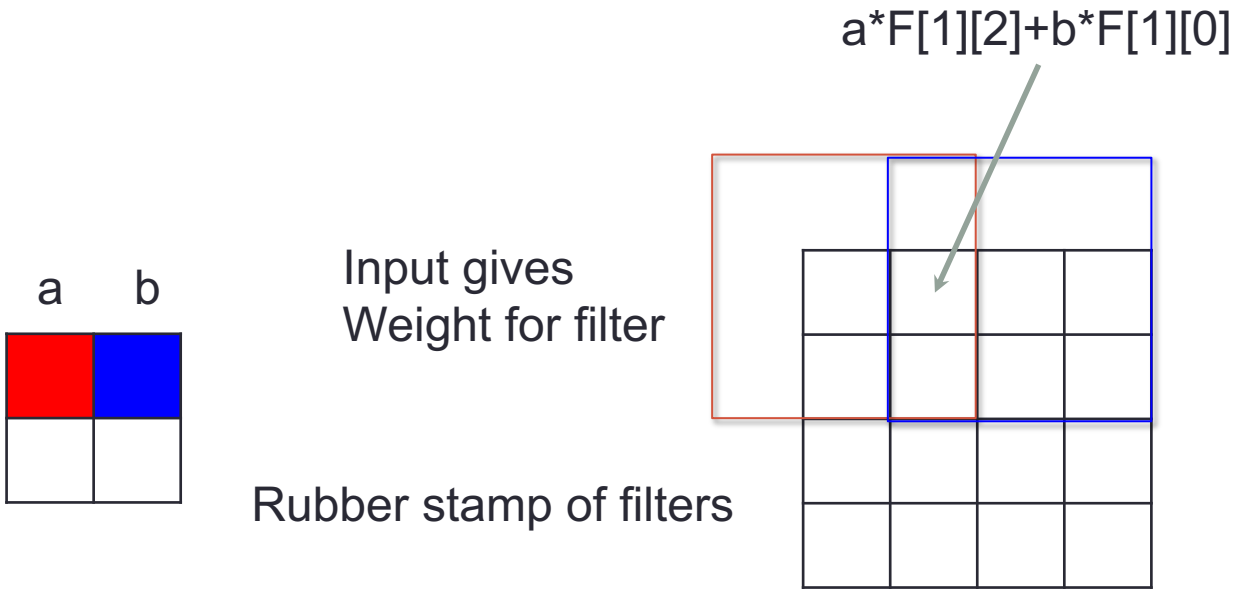




2 main approaches to upsample
De-convolution
resize (unpooling) + convolution

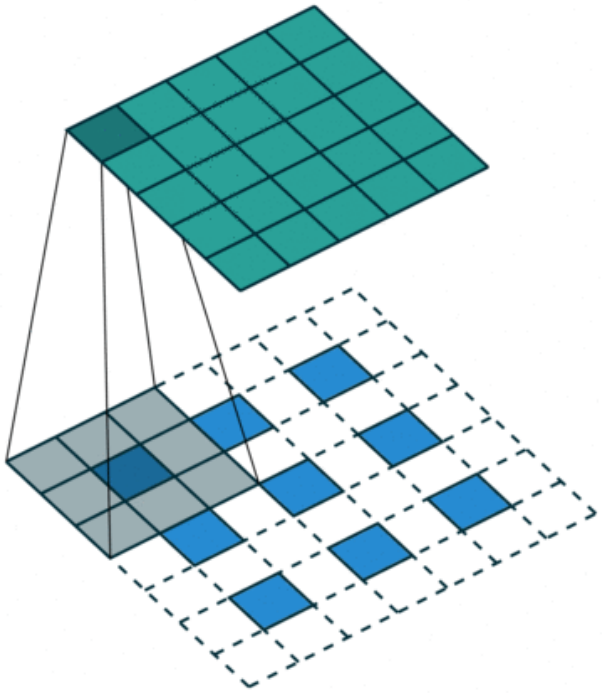
https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf
http://vladlen.info/publications/feature-space-optimization-for-semantic-video-segmentation/

De-convolution (Upsampling)



De-convolution (Upsampling)

- 3x3 de-convolution filter, stride 2, pad 1



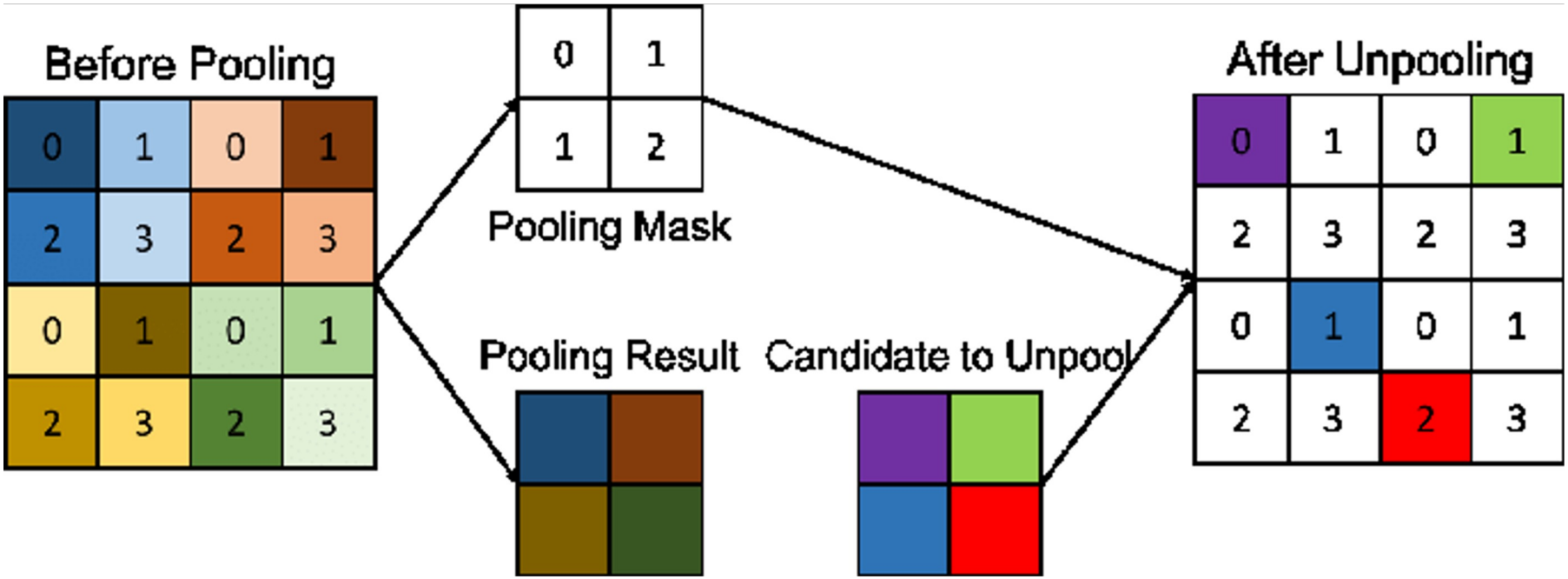
Think if de-convolution as a process that reverse the change in size if you apply a convolution.

Conv: 5x5 input with 3x3 stride 2 pad 1 gives 3x3
Deconv: 3x3 to get 5x5

Note: implementation-wise, de-convolution flip the forward and backward computation of the convolution operation. More info https://arxiv.org/abs/1603.07285

Other names
- Convolution transpose, upconvolution, backward strided convolution

Unpooling + conv



Remember the location of the max value

Put the value to unpool at that location

Fill the rest with zeroes

Follow by regular convolution to smooth the image

Fully Automatic Acute Ischemic Lesion Segmentation in DWI Using Convolutional Neural Networks, 2017.

Upsampling notes

Deconvolution filter size should be a multiple of the stride to avoid checkerboard artifacts

Unpooling can be replaced with regular image resizing techniques (interpolation)

Image using deconv

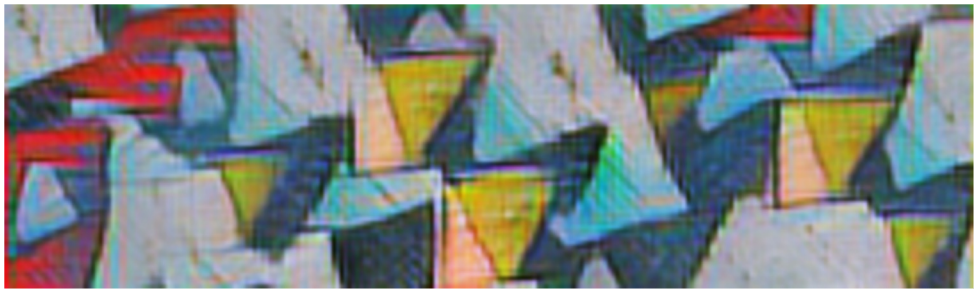


Image using resize upsampling

https://distill.pub/2016/deconv-checkerboard/

Flavors of supervision

Supervised

All labels known

Semi-supervised

Some labels known

Unsupervised

No labels

Representation learning

Transfer learning

Self-supervised learning

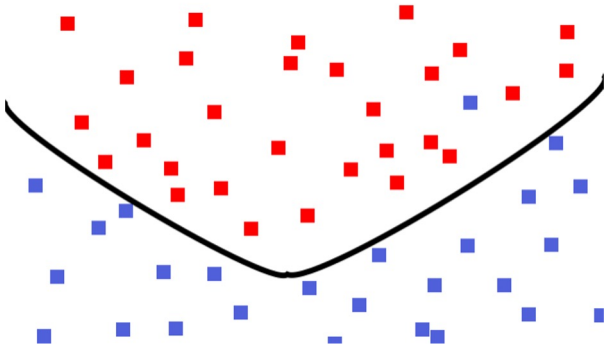
Surrogate task from pseudo labels

Reinforcement learning

Agent gathers data, reward function known

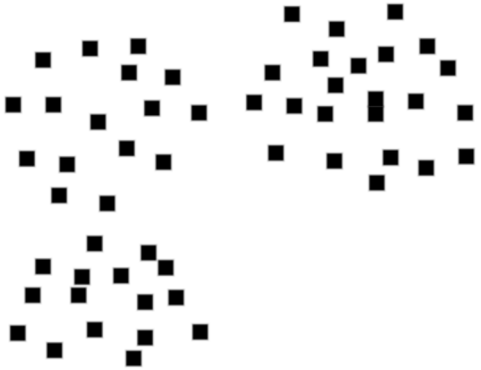
UNSUPERVISED LEARNING

3 Modes of Learning

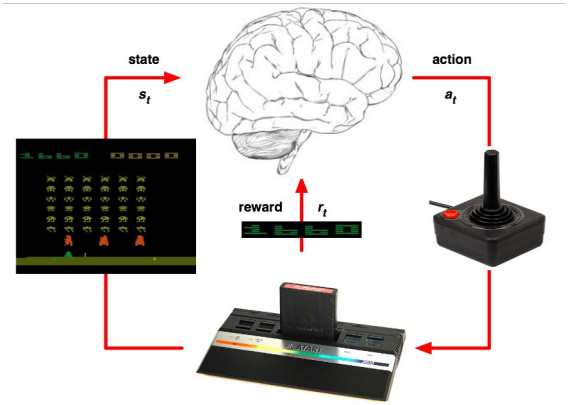


Supervised Learning

Reinforcement Learning



Unsupervised Learning



Unsupervised Learning

- What does it mean to be unsupervised?
- Semi-supervised learning
- Transfer learning
- Self-supervised learning
  - Contrastive learning
  - Consistent loss

“Unsupervised” Learning

- Given an image -> Tell you right away it’s a cat without anyone labeling anything?

  NO

- What does it do exactly?

  Can we somehow utilize unlabeled data to help with some task we care about?

Semi-supervised training

Semi-supervised learning

Can we learn from unlabeled data?

Learn better representations (unsupervised learning)

Can we make a cat classifier out of it?

No, need some labels

Can we use some unlabeled data with labeled data?

Semi-supervised problem

Semi-supervised training using pseudo labels

Idea 1: Bootstrapping with self-training

Use a trained classifier on new data to get fake labels.

Filter data with high scores.

Train/adapt with filtered data.

Caveats:

Scores cannot be trusted (ranking can be trusted)

Learning from same errorful data

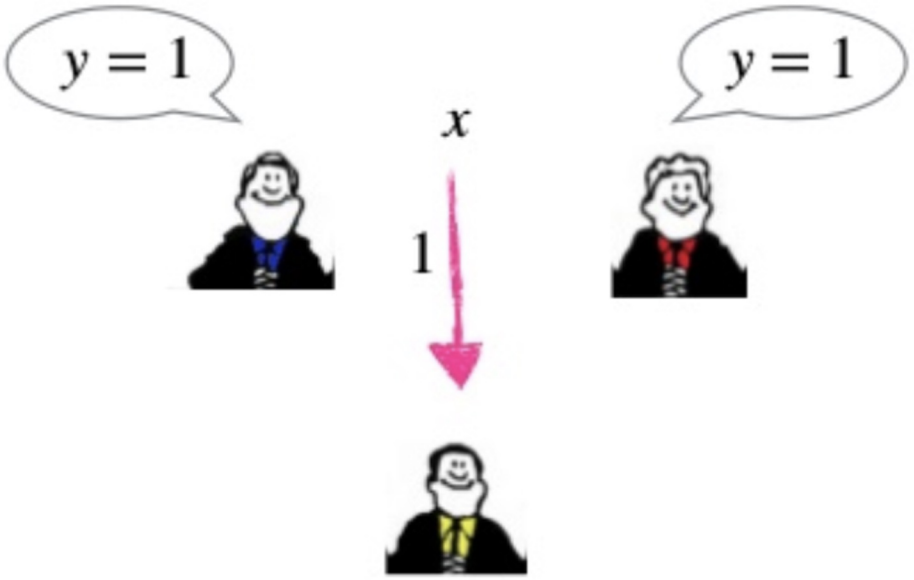
If new data is quite different from the data used to train the classifier, classifier performance is bad

Idea 2: Tri-training

Train 3 models with different subset of source data.

If two models agree on a label on target data, add this data to train the third.

Diversify each model



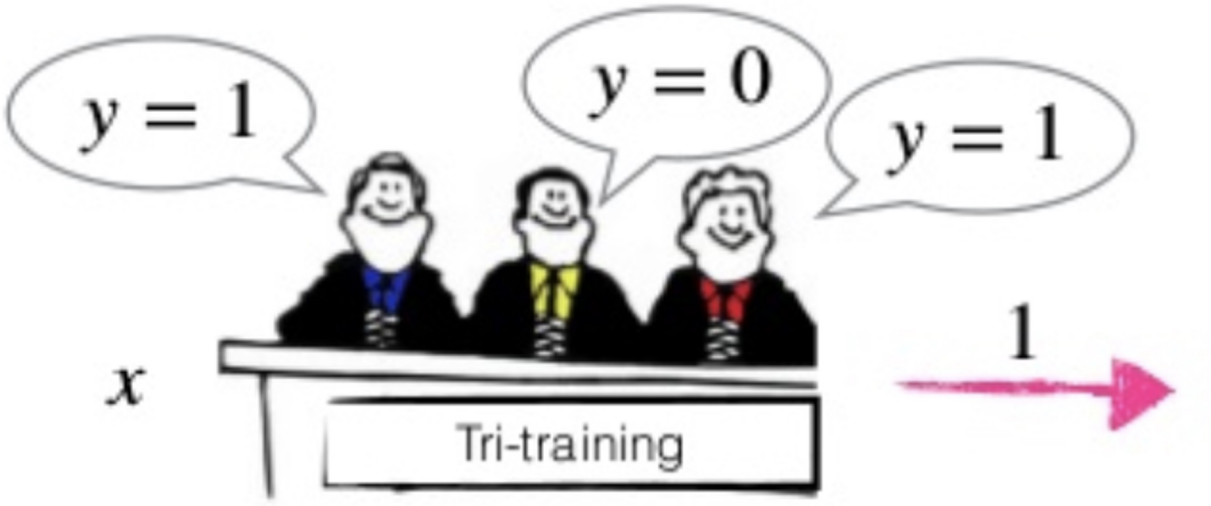
https://www.slideshare.net/SebastianRuder/strong-baselines-for-neural-semisupervised-learning-under-domain-shift

Idea 2: Tri-training with disagreement

If all models agree, it might be an easy data point.

Not so useful

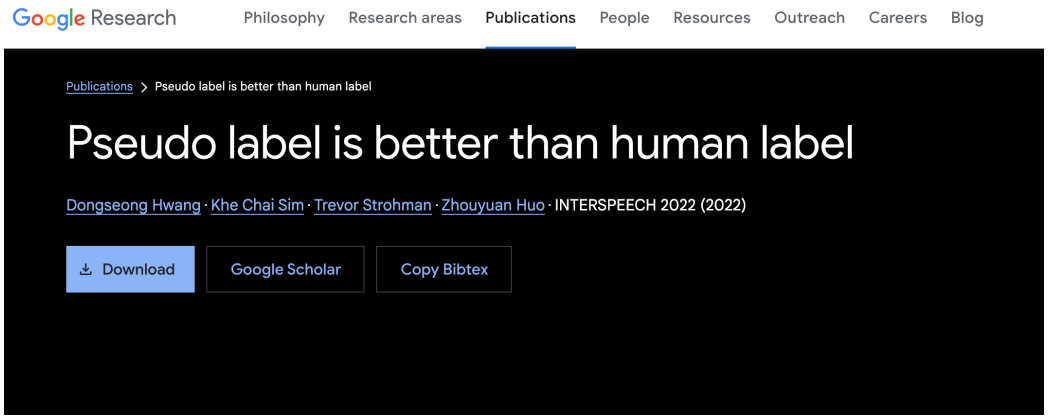
Only use data when two agree and the third disagree



https://www.slideshare.net/SebastianRuder/strong-baselines-for-neural-semisupervised-learning-under-domain-shift

Notes on psuedo labels

- Modern techniques try to use soft labels (rather than hard labels – think k-mean vs GMM)
- Quite useful in practice
  - Can be even better than true label sometimes. Human label can be noisy. However, need a strong starting model.



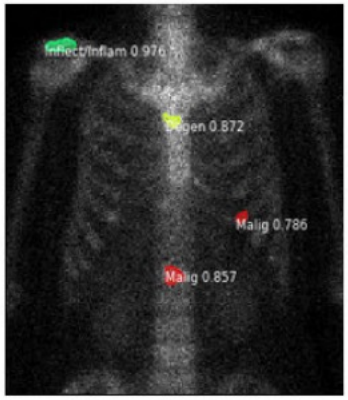
https://arxiv.org/abs/2203.12668

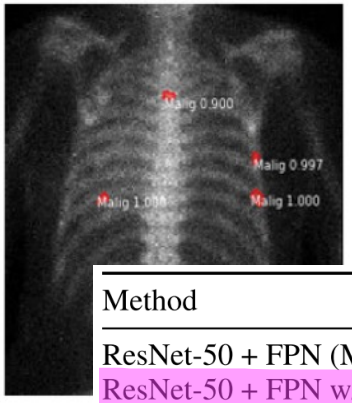
Abstract

Human labeling is expensive. Labeling is the most painful step for ML production. It's widely believed that data is the new gold and big tech companies have an unfair advantage. Is it true that unlimited data unlimits model performance? In this study, we show 1k hrs human labeled data is enough for the best ASR model. The model trained with 1k hrs human labels and 26k hrs pseudo labels has better WERs than the model with 27k hrs human labels. Pseudo label training improves WERs of the production model by a significant margin; 5.9 to 5.1 on voice search. It means pseudo label quality is better than human label. To have quality pseudo labels, we utilized recent self/semi-supervised learning for a large ASR model.

Notes on psuedo labels

- Modern techniques try to use soft labels (rather than hard labels – think k-mean vs GMM)
- Quite useful in practice
  - Can be even better than true label sometimes. Human label can be noisy. However, need a strong starting model.





Data type	Amount of data	Training data	Validation data	Testing data
Labeled data	1,088	741	231	116
Unlabeled data	18,560	14,786	3,774	0
Total data	19,648	15,527	4,005	116

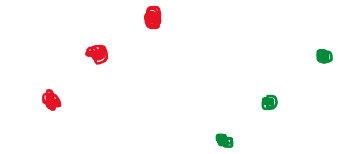
Method	Mean precision	Mean sensitivity	Mean F1-score
ResNet-50 + FPN (Mask R-CNN)	0.827	0.811	0.816
ResNet-50 + FPN w/ global features	0.829	0.826	0.824
ResNet-50 + LFPN w/o global features	0.838	0.839	0.835
ResNet-50 + LFPN w/ global features (MaligNet)	**0.852**	**0.856**	**0.848**
ResNet-50 + FPN + self-training	0.849	0.843	0.840
ResNet-50 + LFPN w/ global features + self-training	**0.867**	**0.844**	**0.851**

https://ieeexplore.ieee.org/document/8979392

Unsupervised learning

First Unsupervised Learning Idea

- Clustering!
- Assign “Id” to each instance



Version 1

Version 2

- And then what?
  - What makes a clustering solution useful?

2nd Unsupervised Learning Idea

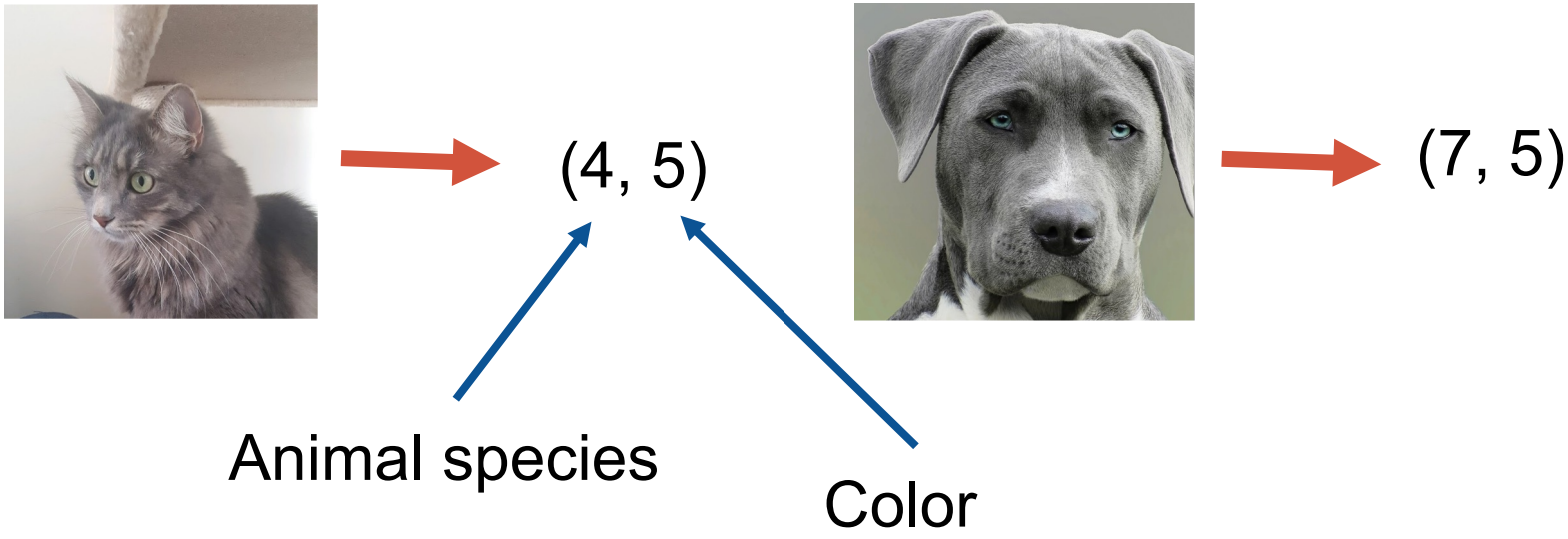
- Suppose we have a magic function f that maps an image to two numbers

f

(4.1, 5.0)

2nd Unsupervised Learning Idea

- Ideal magic function f



Getting this f would be so nice!

Given f, we can recognize animals with a linear classifier!

2nd Unsupervised Learning Idea

An alternative magic function f

(4, 5)

(7, 5)

Value of top-left pixel

Standard deviation of pixel colors

This f isn't so useful

Task: We try to learn the most "useful" f() from unlabeled data.

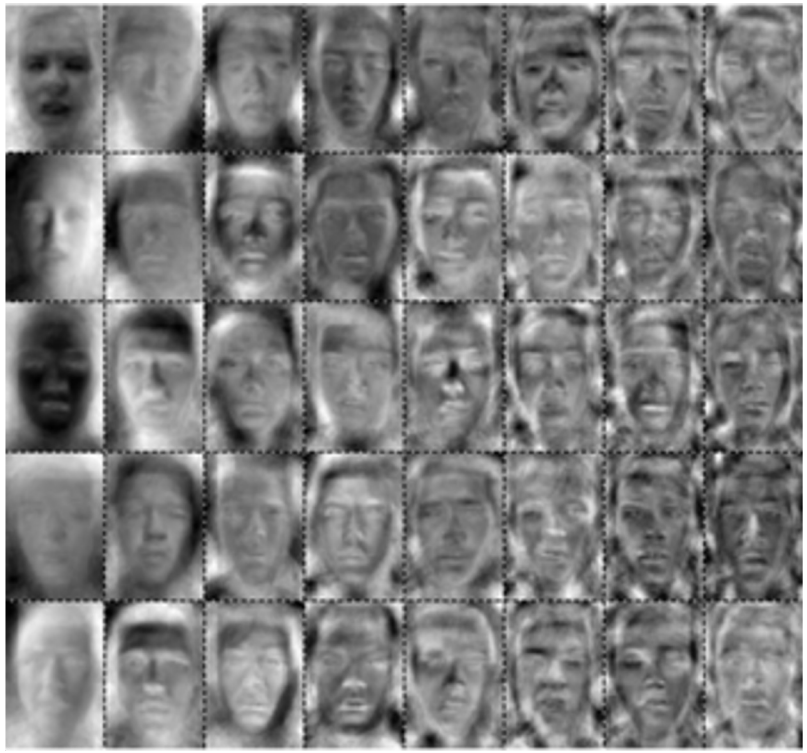
How to define usefulness? [subjective: Natural, human tasks]

PCA

- PCA is such a function
  - But PCA is limited due to being just a linear function

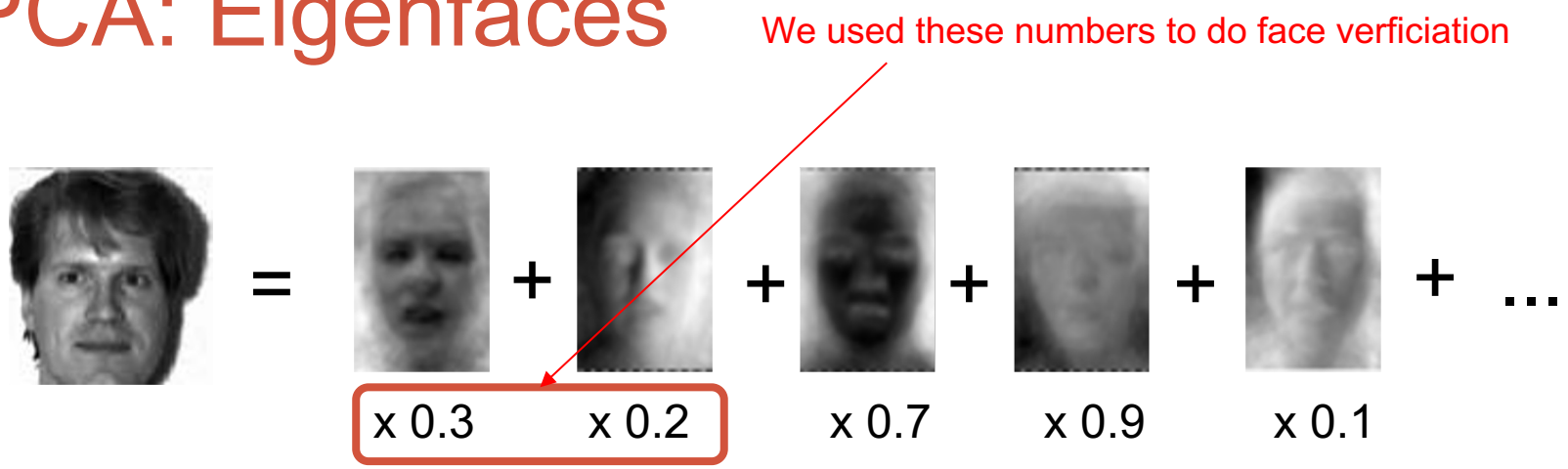
Encoding and reconstructing an image

- We know we can reconstruct a face using PCA to map high dimension data to low dimension data



PCA basis (Eigenfaces)

PCA: Eigenfaces



Problems:

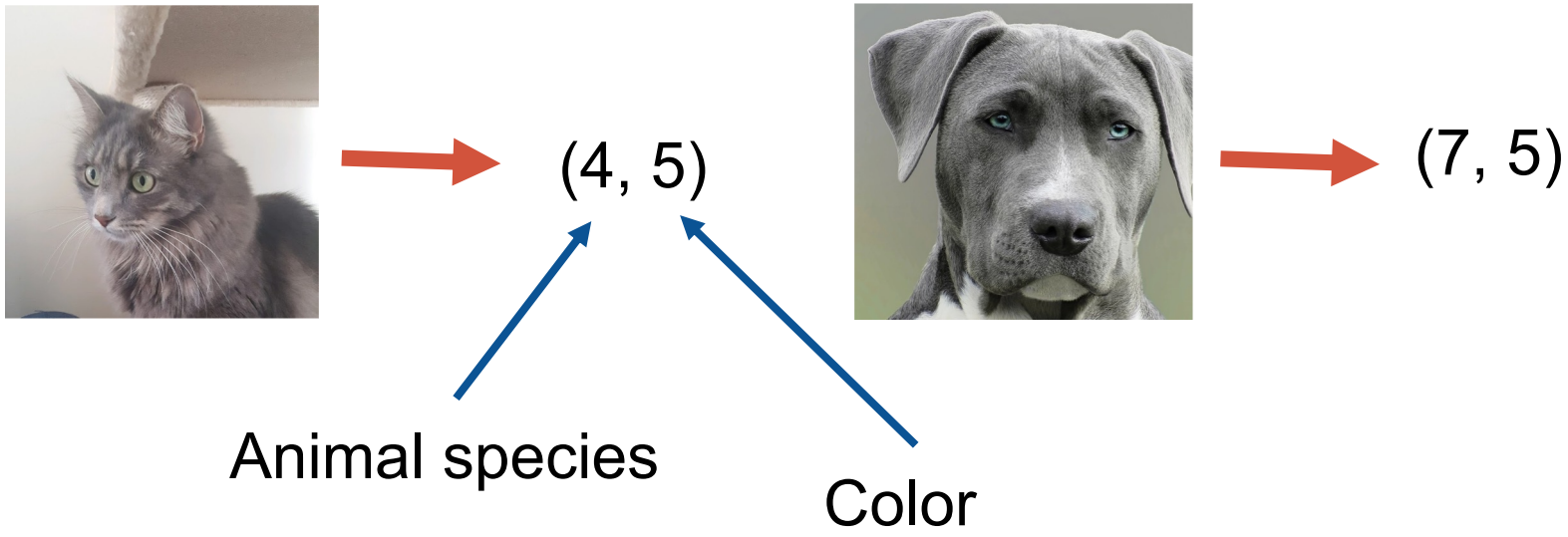
1. PCA is a linear operation (as well as LDA or fisherface). Many interesting attributes are highly non-linear, e.g., identity, emotion, and cannot be factored linearly.
2. Faces have to be aligned and centered for this to work
3. Illumination dominates!

f( ) through deep learning

- A lot of the techniques uses clever tricks to learn the model f( )
  - This will be the topic of this and the next lecture
- Can learn more sophisticated structure in the data

Steps

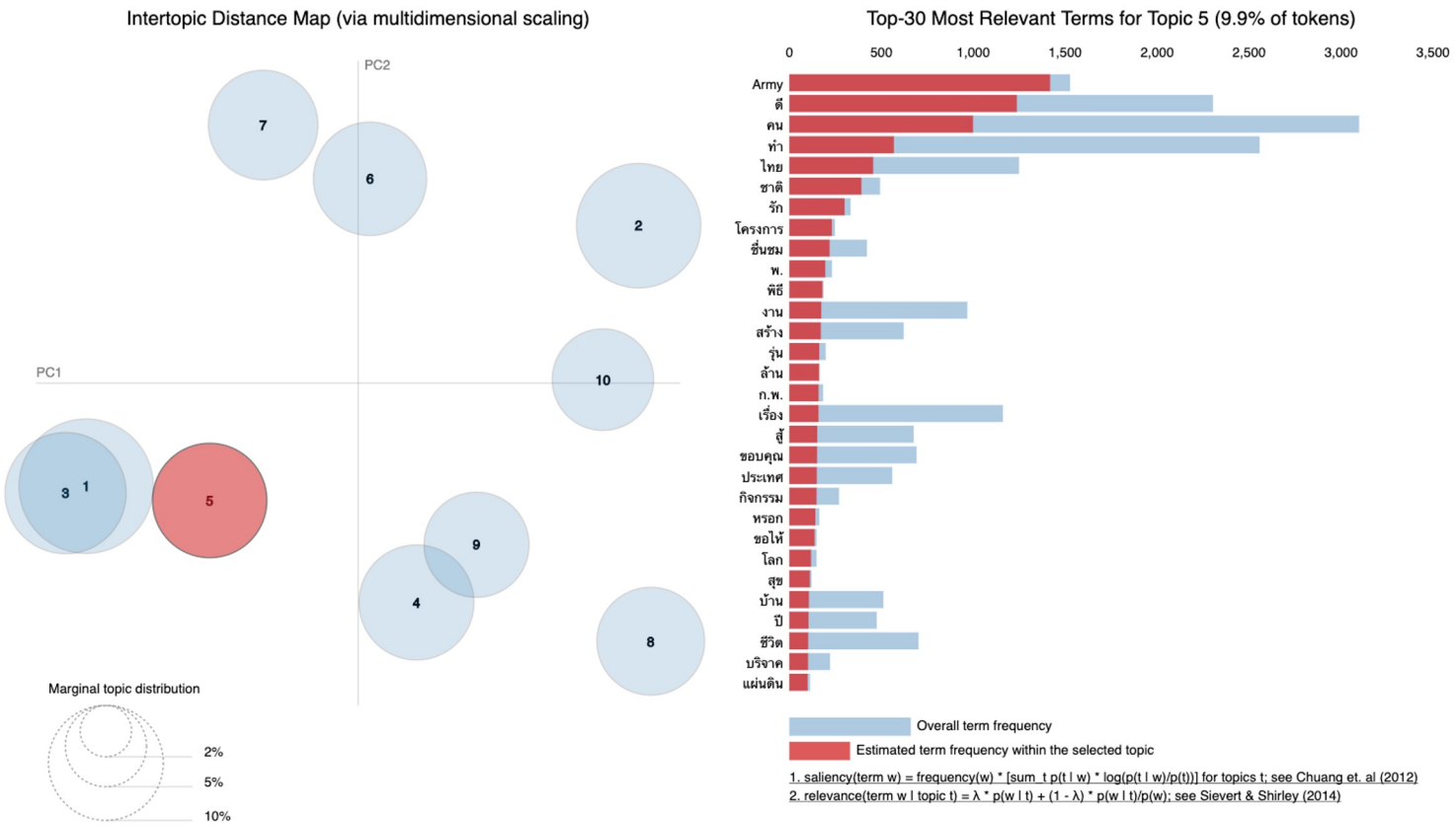
- Learning f (somehow)



- Use f via transfer learning, few/zero-shot

Other notable (classical) unsupervised learning methods

- LDA (Latent Dirichlet Allocation)
  - Learns hidden topics in documents



https://www.facebook.com/teattapol/posts/3337686199651109?_rdc=1&_rdr

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

Representation learning

Finding the magical function f( )

We know that DNN can model highly complex mapping.
How about modeling f with a neural net?

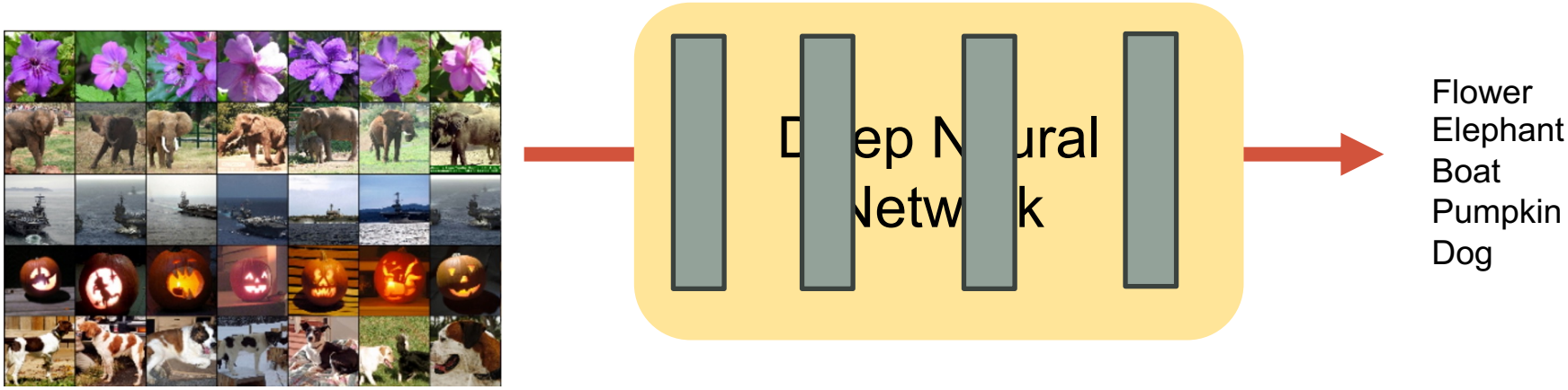
Deep Neural Network

(4.1, 5.0)

But how to train it? We don't know the target, groundtruth, two numbers.

Finding the magical function f( )

One way is to use a supervised model and extract hidden values from the network



Drawbacks: need labelled data.
Can we just use unlabelled data to learn the neural network?

Autoencoder

Auto means self!

Input

X

Encoder

(?, ?)

Decoder

Output

X'

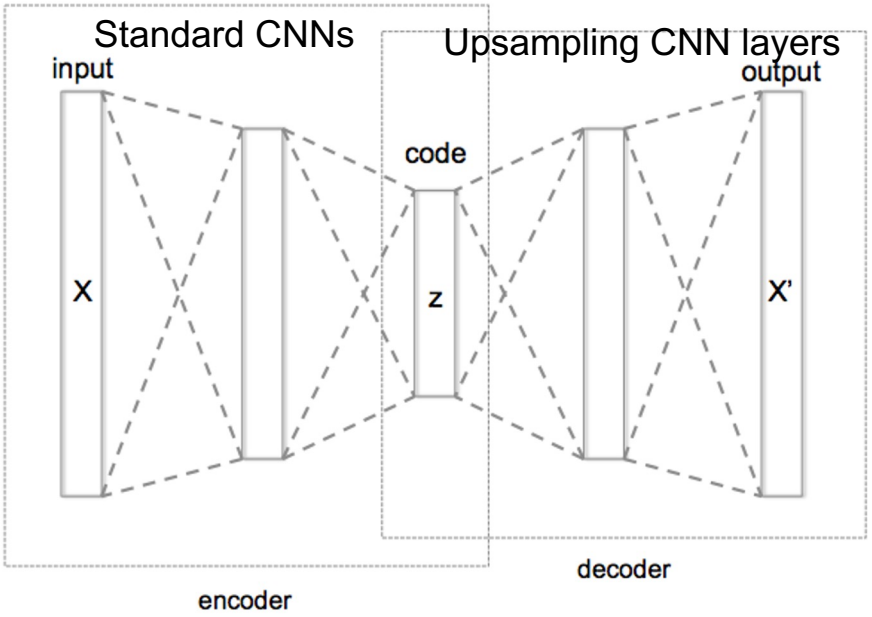
Now we can train the network with L2 Loss: $\|X - X'\|^2$

Autoencoder

Auto means self!

Input

X

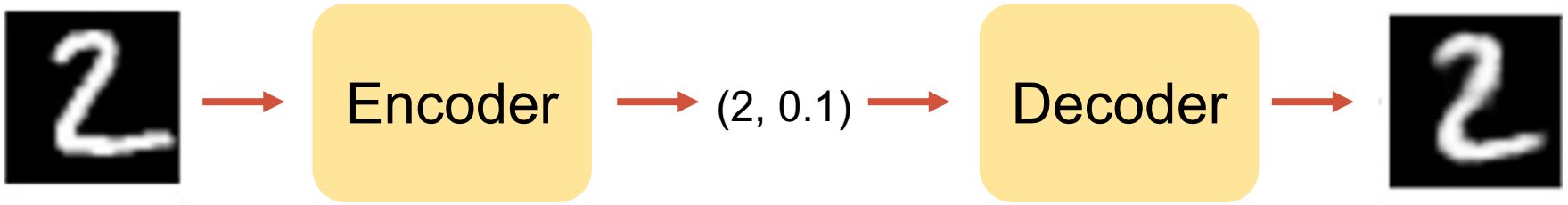


Output

X'

Now we can train the network with L2 Loss: $\|X - X'\|^2$

Autoencoding Game



Both Encoder and Decoder see the entire dataset.

Encoder: What two numbers should I send you so that you can re-draw my input as accurate as possible?

Encoder: Maybe the first number would directly correspond to what digit it is. The second, perhaps, the thickness of the stroke?

Turns out DNN does surprisingly well at encoding important attributes.

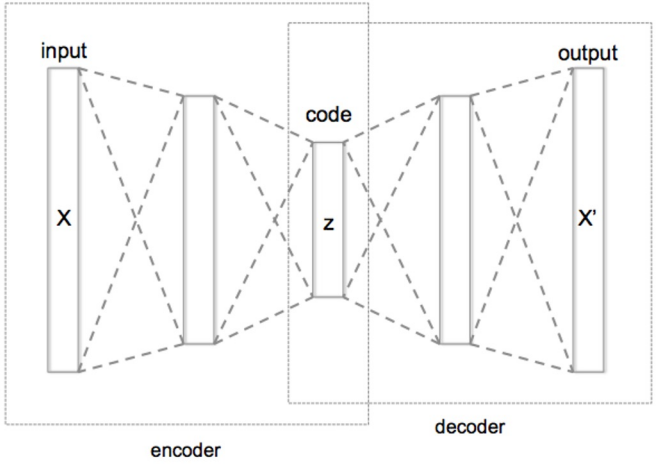
Autoencoder

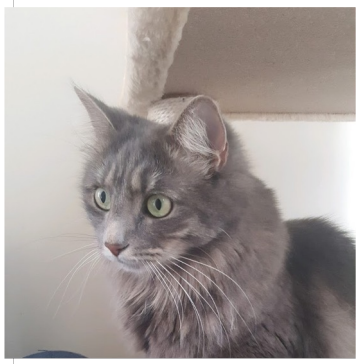
Can we pick an arbitrary dimension for the code?

- When z has the same dimension as the input and output, NN cheats and outputs an identity mapping -> useless.

Input

X



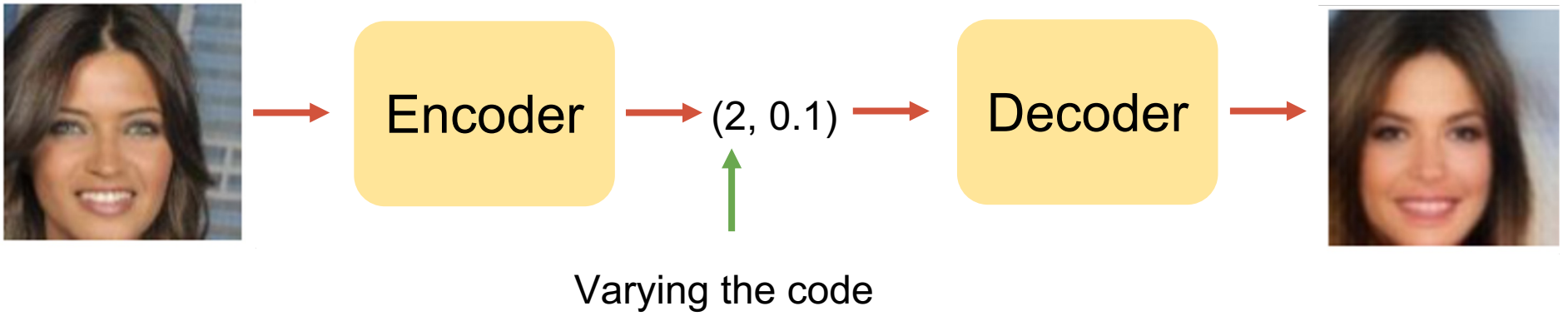


Output

X'

- When z dim << input dim, NN has to work hard!

Inspecting attributes learned from AE



Deep Feature Consistent Variational Autoencoder [Hou et al. 2016]

Attributes from autoencoder

Deep Feature Consistent Variational Autoencoder [Hou et al. 2016]

Utilizing autoencoder for supervised tasks

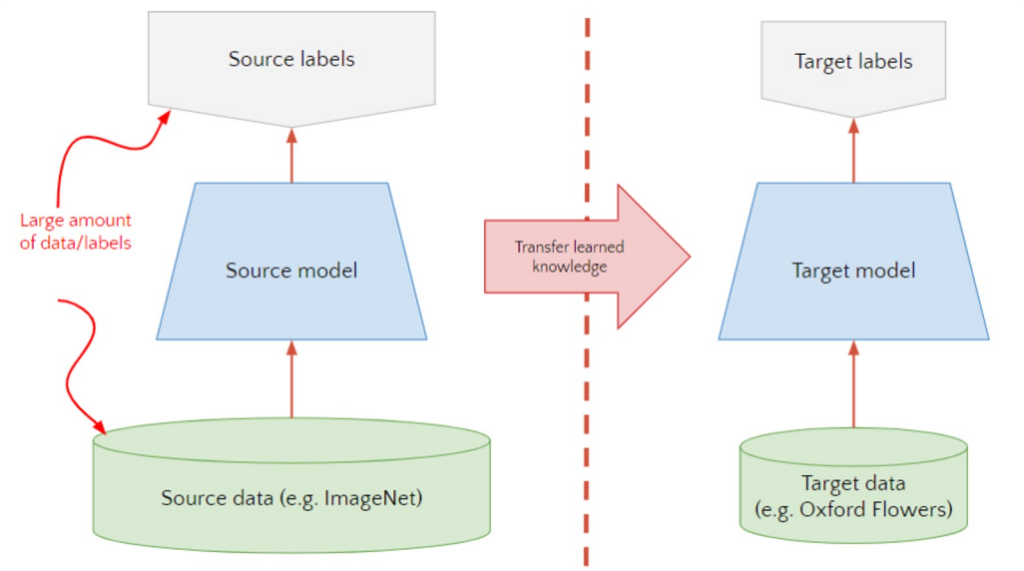
Many ways to help with a supervised task such as recognition.

1. Append the input with the code from the encoder
2. Stick a classifier on top of the encoder
   Can even be a linear classifier – “linear probe”
3. Used for pretraining a network

Transfer learning

Transfer learning (basics)

- We know networks captures good representations
- Can we use it for other tasks?
- Use trained networks to initialize a new network for a different task.
- Re-train the network using SGD on new data.

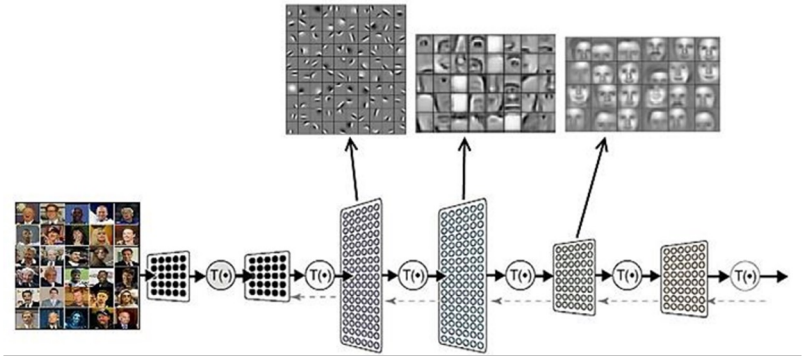


For CV tasks, we call the pre-trained network backbones

Transfer learning idea

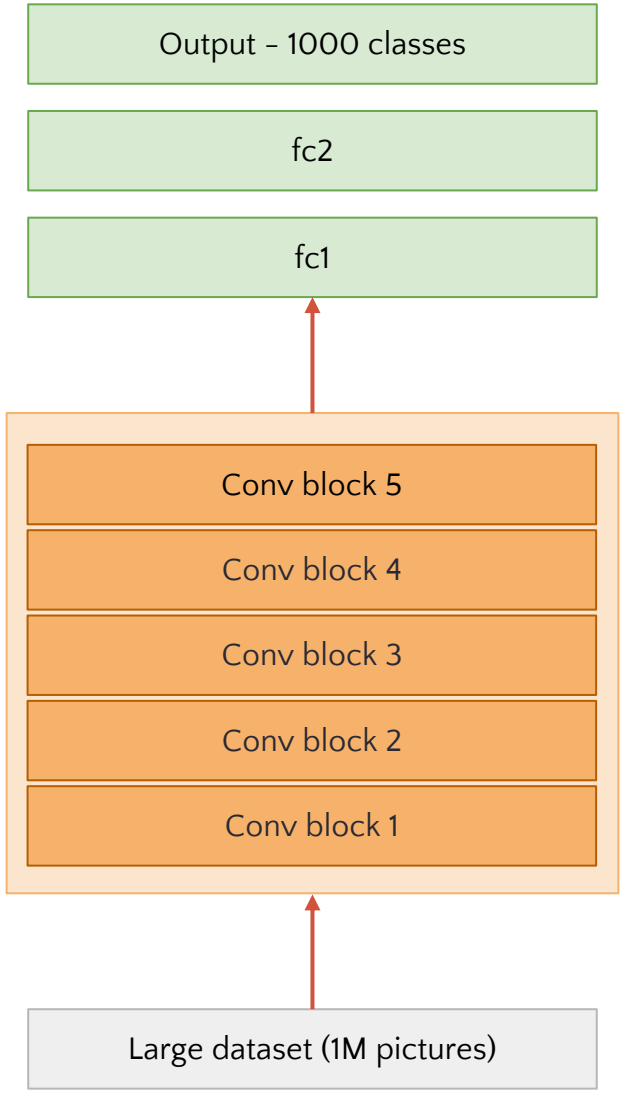
Instead of training a deep network from scratch for your task, you can

- Take a network trained on **a different domain** for a **different source task**
- **Adapt (fine-tune)** it for your domain and your **target task**

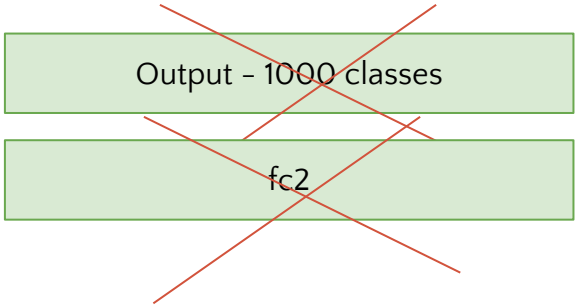


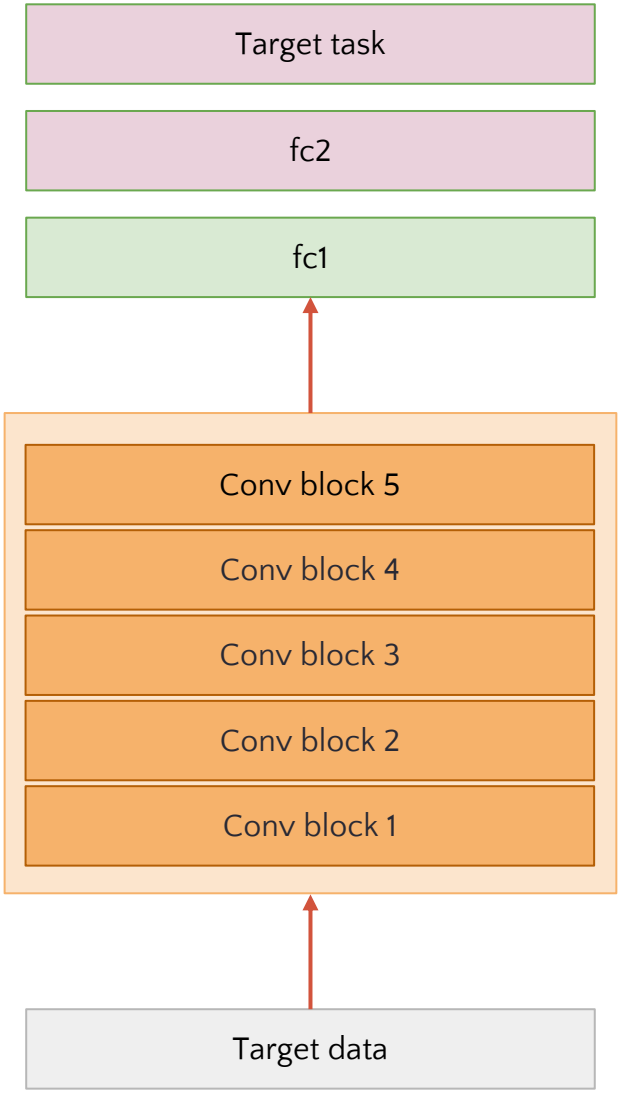
Transfer learning

1. A model is trained on large dataset
   Ex ImageNet



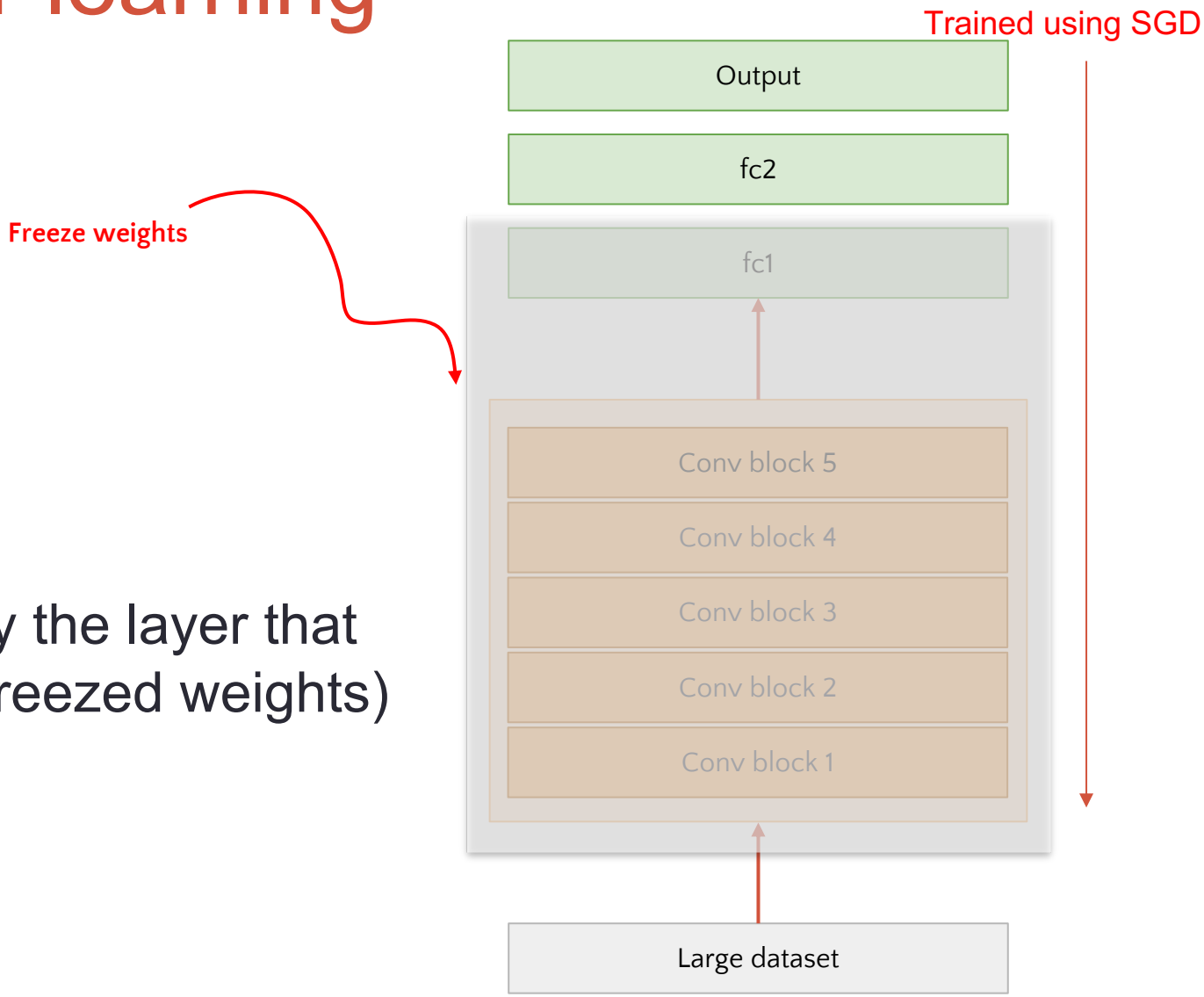
Transfer learning





2. Replace the top layers with target task

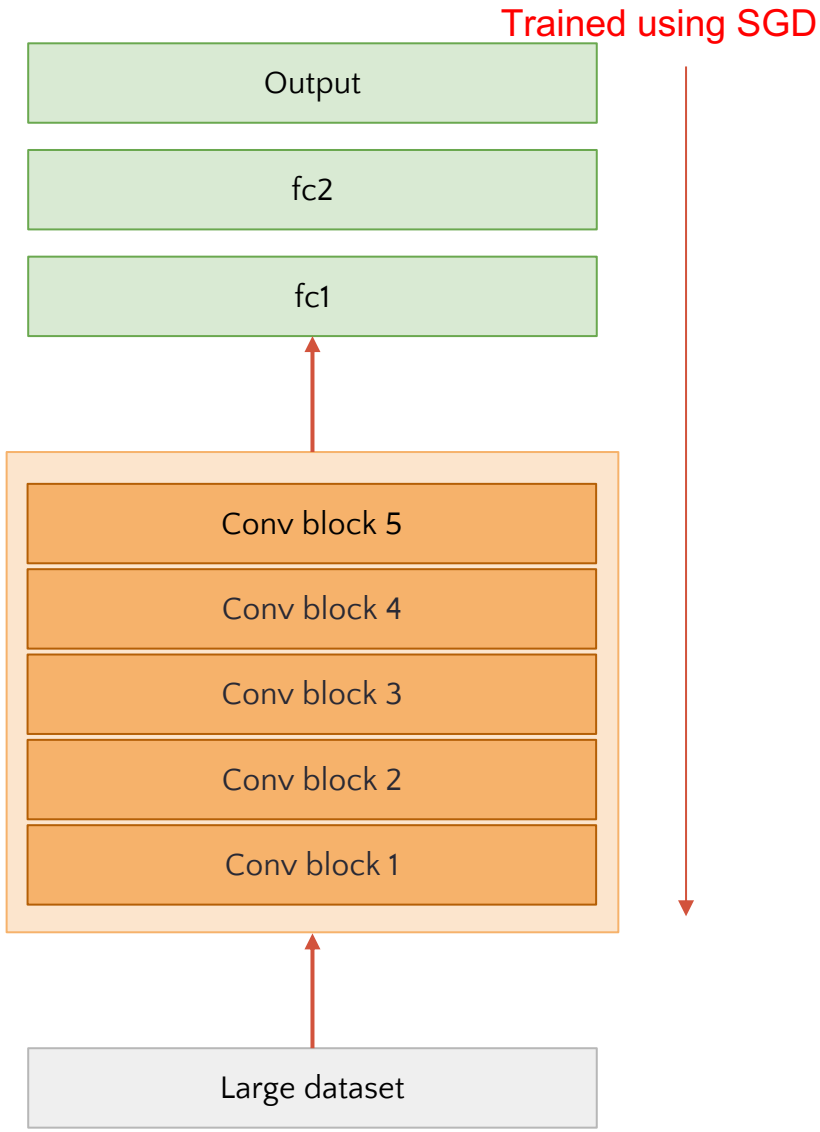
Transfer learning



3A. Train only the layer that is replaced (freezed weights)

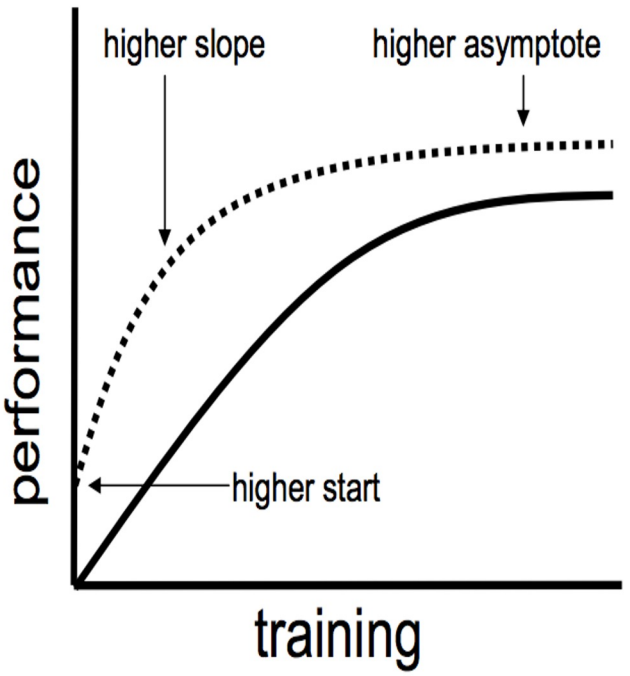
Transfer learning

3B. Train all layers
(unfreezed weights)



Benefits to transfer learning

1. Higher start
2. Higher slope (converge faster)
3. Higher asymptote (if small data)

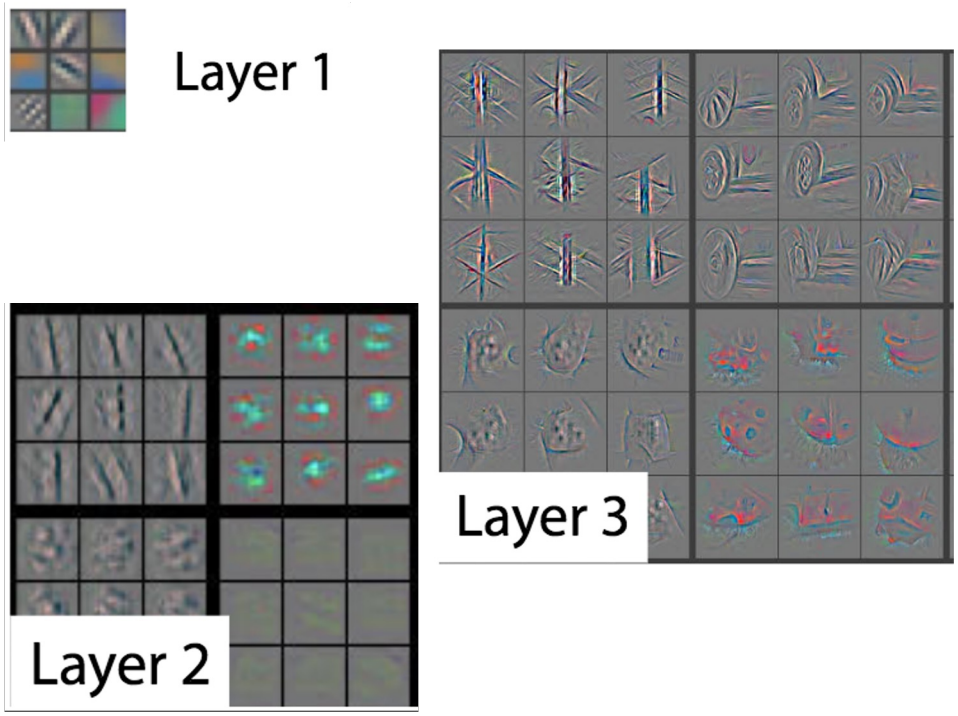


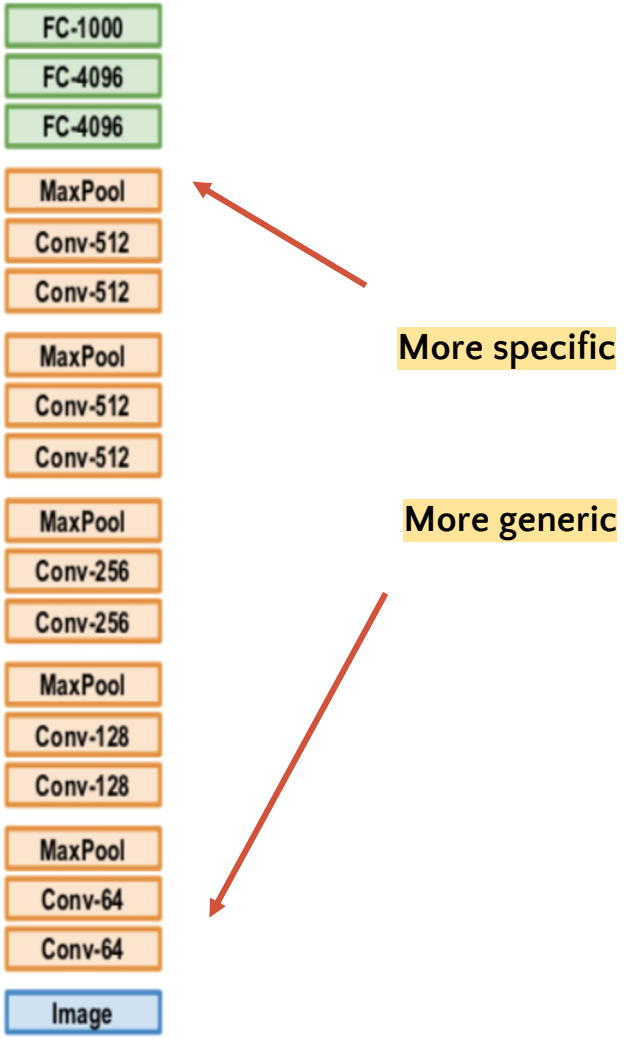
Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009.

Layers

Lower layers: more general

Higher layers: more specific





Domains

Similar domain can transfer easier

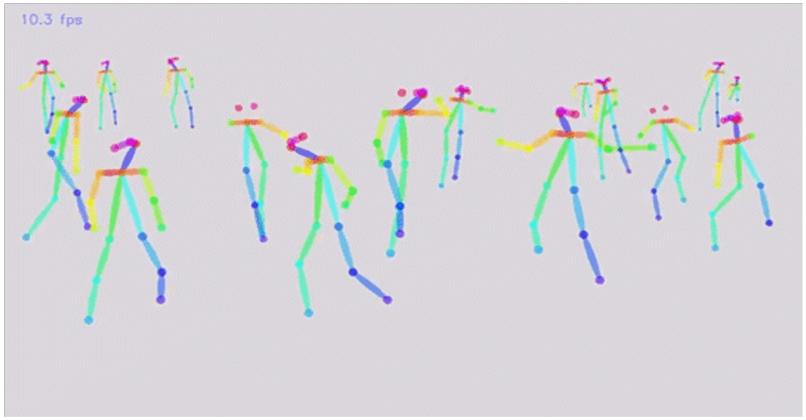
Fine-tuning (adaptation) is crucial

Aircraft images
Different from ImageNet

Natural images
Similar to ImageNet

Kornblith et al., Do Better ImageNet Models Transfer Better?, arxiv 2018 https://arxiv.org/abs/1805.08974

Example: pose estimation



https://arxiv.org/abs/1611.08050

VGG19-finetuned

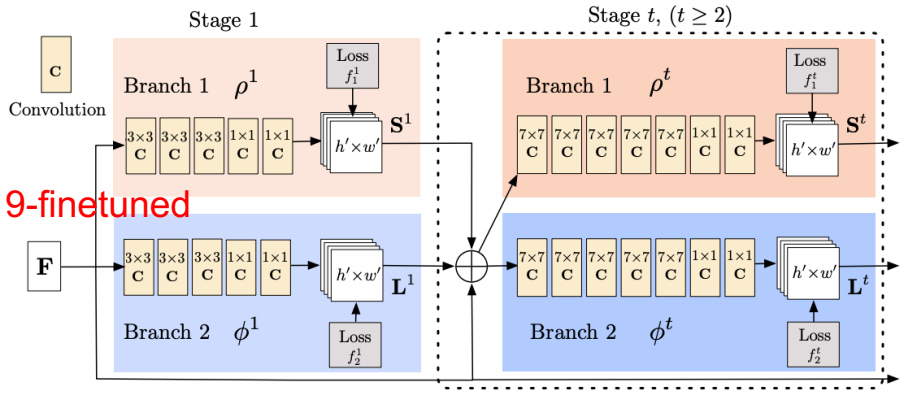
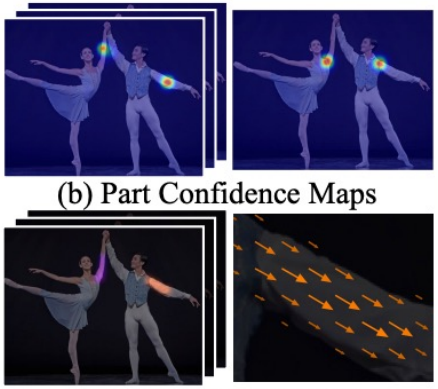


Figure 3. Architecture of the two-branch multi-stage CNN. Each stage in the first branch predicts confidence maps $\mathbf{S}^t$, and each stage in the second branch predicts PAFs $\mathbf{L}^t$. After each stage, the predictions from the two branches, along with the image features, are concatenated for next stage.

(a) Input Image



(b) Part Confidence Maps

(c) Part Affinity Fields

(d) Bipartite Matching

(e) Parsing Results

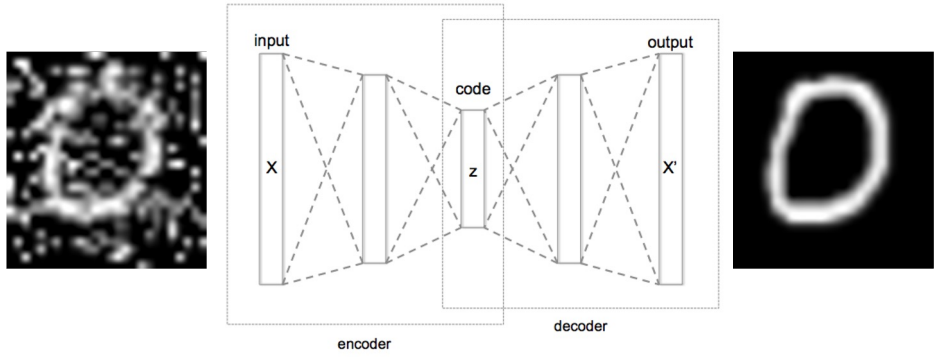
Figure 2. Overall pipeline. Our method takes the entire image as the input for a two-branch CNN to jointly predict confidence maps for body part detection, shown in (b), and part affinity fields for parts association, shown in (c). The parsing step performs a set of bipartite matchings to associate body parts candidates (d). We finally assemble them into full body poses for all people in the image (e).

Representation learning (cont)

Denoising autoencoder

- Motivation: Making AE more robust -- really learn what's important

- An autoencoder that denoise corrupted inputs
  - Blur image, sound corrupted by noise
- How?
  - Corrupt your input x -> x''
  - Use x'' as the input, and minimize the same loss
  - Loss = $||x-x'||^2$



Example

Original Corrupted Generated

https://www.doc.ic.ac.uk/~js4416/163/website/autoencoders/denoising.html

Noise? What noise?

- Turns out the core idea of denoising autoencoder is simple yet very powerful.
  - Key limitation: what kind of noise should we add?

- Important criterions for noise
  - Doesn’t change what we care about
  - Should be something that force the model to learn to focus on the important things

Consistency training/Teacher-student training

One main model (teacher) that does prediction.

Can be on unlabelled data, if trained on labeled data first

Other models (students) see limited view but try to answer like the teacher

Student1: They really _____.

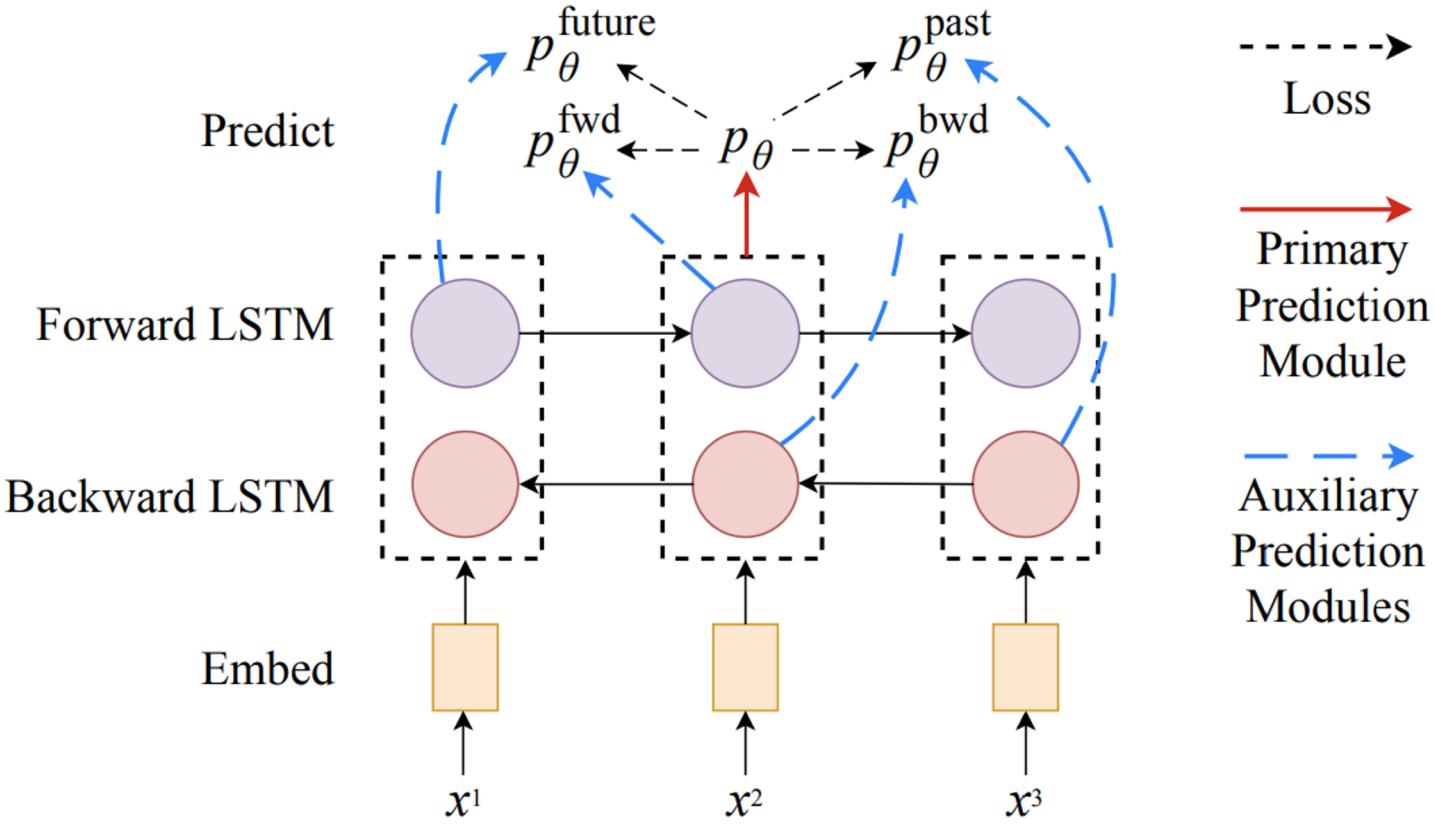
Student2: They really **like**_____.

Teacher: They really **like** the idea.

Student3: _____ **like** the idea.

Student4: _____ the idea.

Future: They really _____.
Forward: They really **like**_____.
Primary: They really **like** the idea.
Backward: _____ **like** the idea.
Past: _____ the idea.

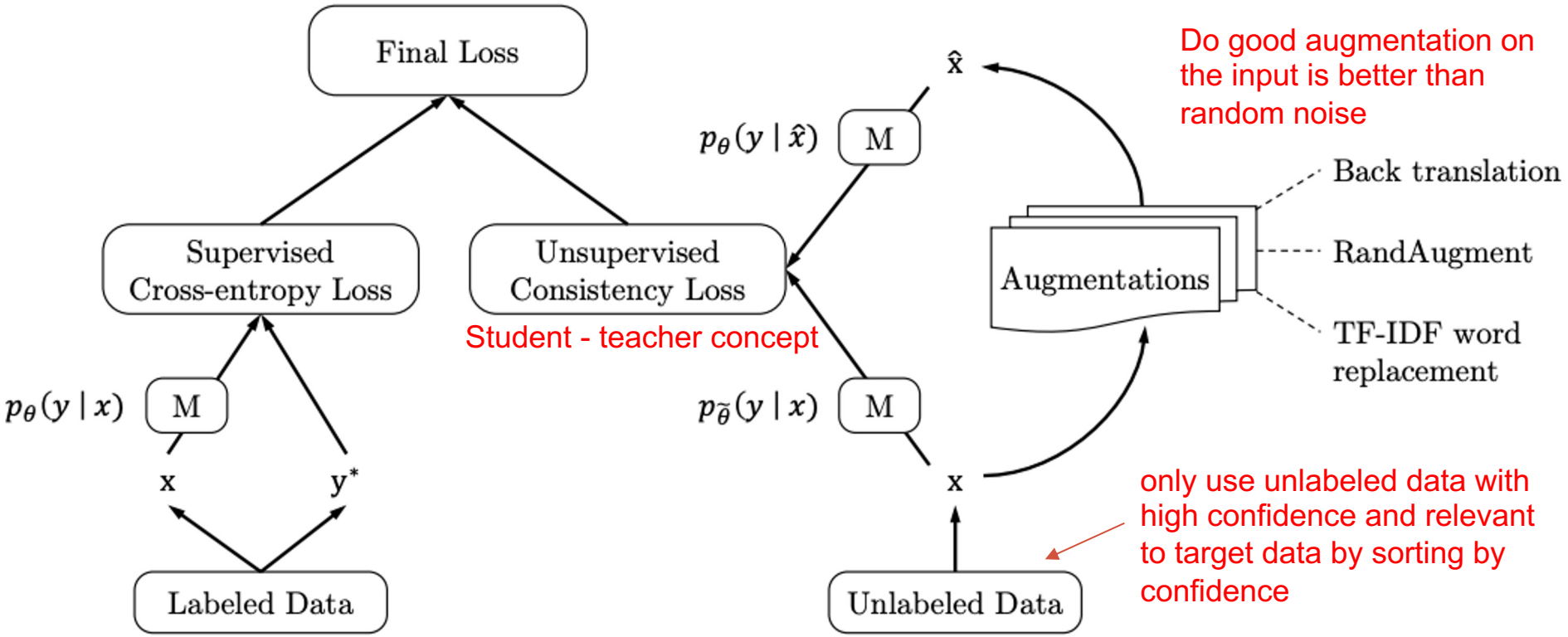


https://aclweb.org/anthology/D18-1217

Method	CCG Acc.	Chunk F1	NER F1	FGN F1	POS Acc.	Dep. Parse UAS	Dep. Parse LAS	Translate BLEU
Shortcut LSTM (Wu et al., 2017)	95.1				97.53			
ID-CNN-CRF (Strubell et al., 2017)			90.7	86.8				
JMT† (Hashimoto et al., 2017)		95.8			97.55	94.7	92.9	
TagLM* (Peters et al., 2017)		96.4	91.9					
ELMo* (Peters et al., 2018)			92.2					
Biaffine (Dozat and Manning, 2017)						95.7	94.1	
Stack Pointer (Ma et al., 2018)						95.9	94.2	
Stanford (Luong and Manning, 2015)								23.3
Google (Luong et al., 2017)								26.1
Supervised	94.9	95.1	91.2	87.5	97.60	95.1	93.3	28.9
Virtual Adversarial Training*	95.1	95.1	91.8	87.9	97.64	95.4	93.7	–
Word Dropout*	95.2	95.8	92.1	88.1	97.66	95.6	93.8	29.3
ELMo (our implementation)*	95.8	96.5	92.2	88.5	97.72	96.2	94.4	29.3
ELMo + Multi-task*†	95.9	96.8	92.3	88.4	**97.79**	96.4	94.8	–
CVT*	95.7	96.6	92.3	88.7	97.70	95.9	94.1	**29.6**
CVT + Multi-task*†	96.0	96.9	92.4	88.4	97.76	96.4	94.8	–
CVT + Multi-task + Large*†	**96.1**	**97.0**	**92.6**	**88.8**	97.74	**96.6**	**95.0**	–

Table 1: Results on the test sets. We report the mean score over 5 runs. Standard deviations in score are around 0.1 for NER, FGN, and translation, 0.02 for POS, and 0.05 for the other tasks. See the supplementary materials for results with them included. The +Large model has four times as many hidden units as the others, making it similar in size to the models when ELMo is included. * denotes semi-supervised and † denotes multi-task.

Consistency training



$$\min_\theta \; \mathcal{J}(\theta) = \mathbb{E}_{x,y^* \in L}\left[-\log p_\theta(y^* \mid x)\right] + \lambda \mathbb{E}_{x \in U} \mathbb{E}_{\hat{x} \sim q(\hat{x} \mid x)}\left[\mathcal{D}_{\mathrm{KL}}\left(p_{\tilde{\theta}}(y \mid x) \,\middle\|\, p_\theta(y \mid \hat{x})\right)\right]$$

https://arxiv.org/abs/1904.12848 Unsupervised Data Augmentation for Consistency Training

Overview of teacher-student training

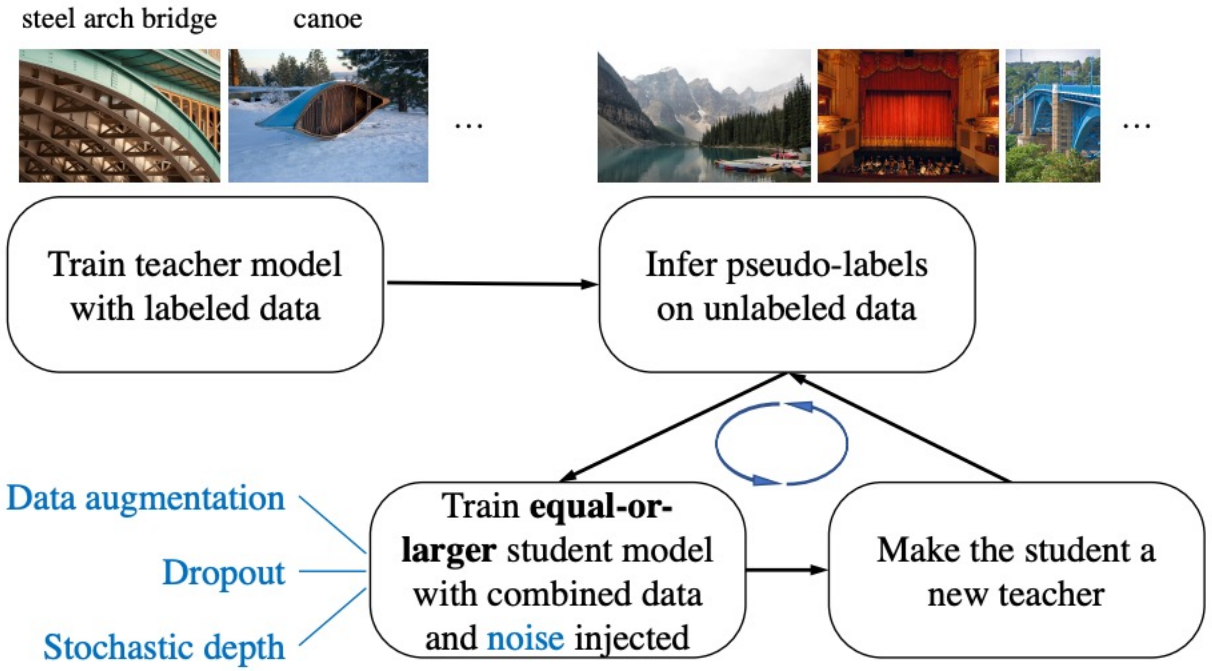
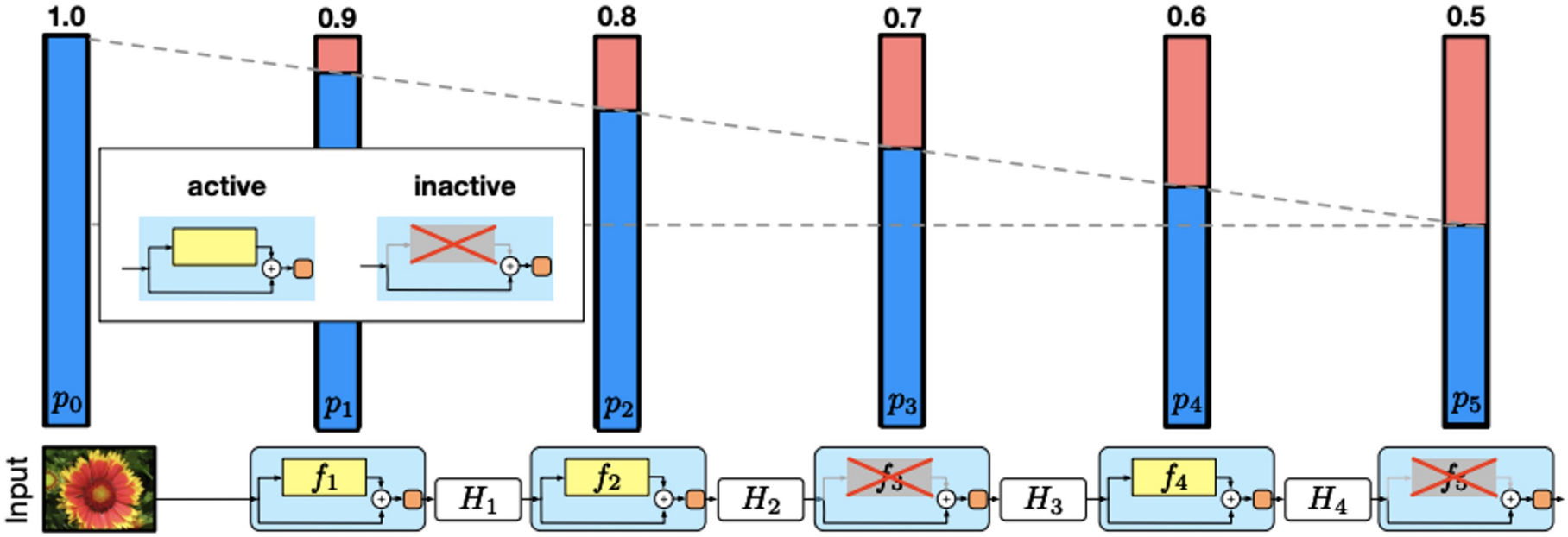


Figure 1: Illustration of the Noisy Student Training. (All shown images are from ImageNet.)

https://arxiv.org/pdf/1911.04252.pdf

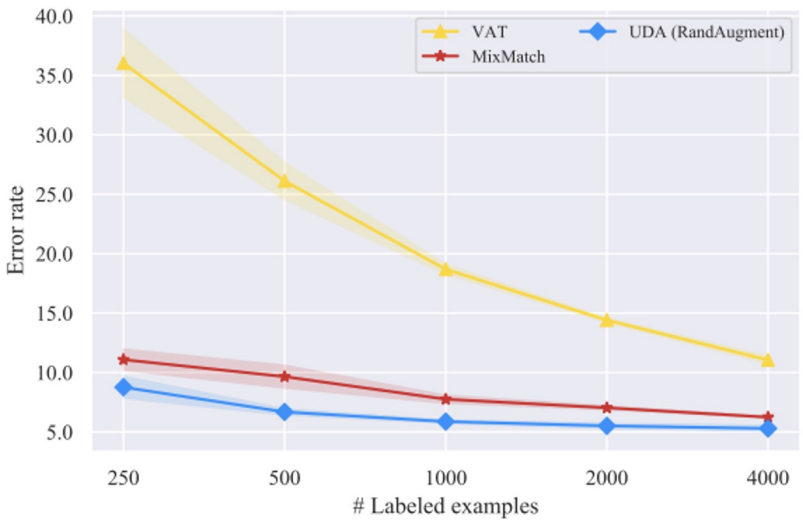
Augmenting (noising) more

Besides augmenting data, we can also noise the network via dropout and stochastic depth

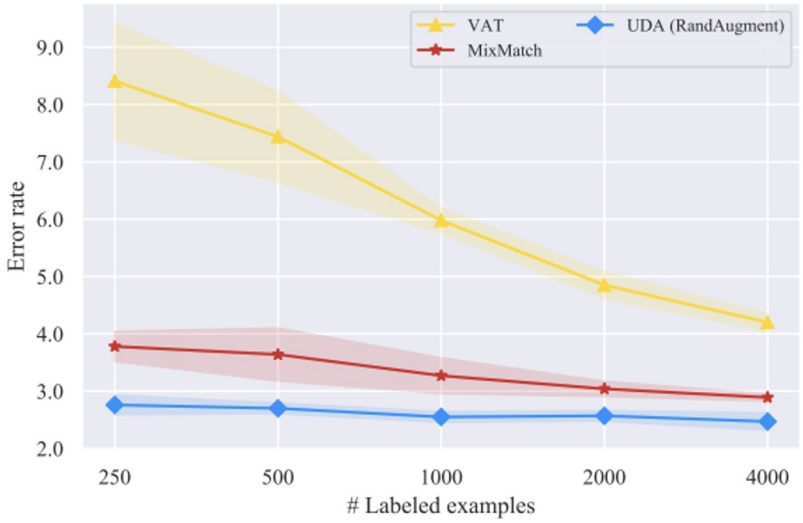


https://arxiv.org/abs/1603.09382

Results



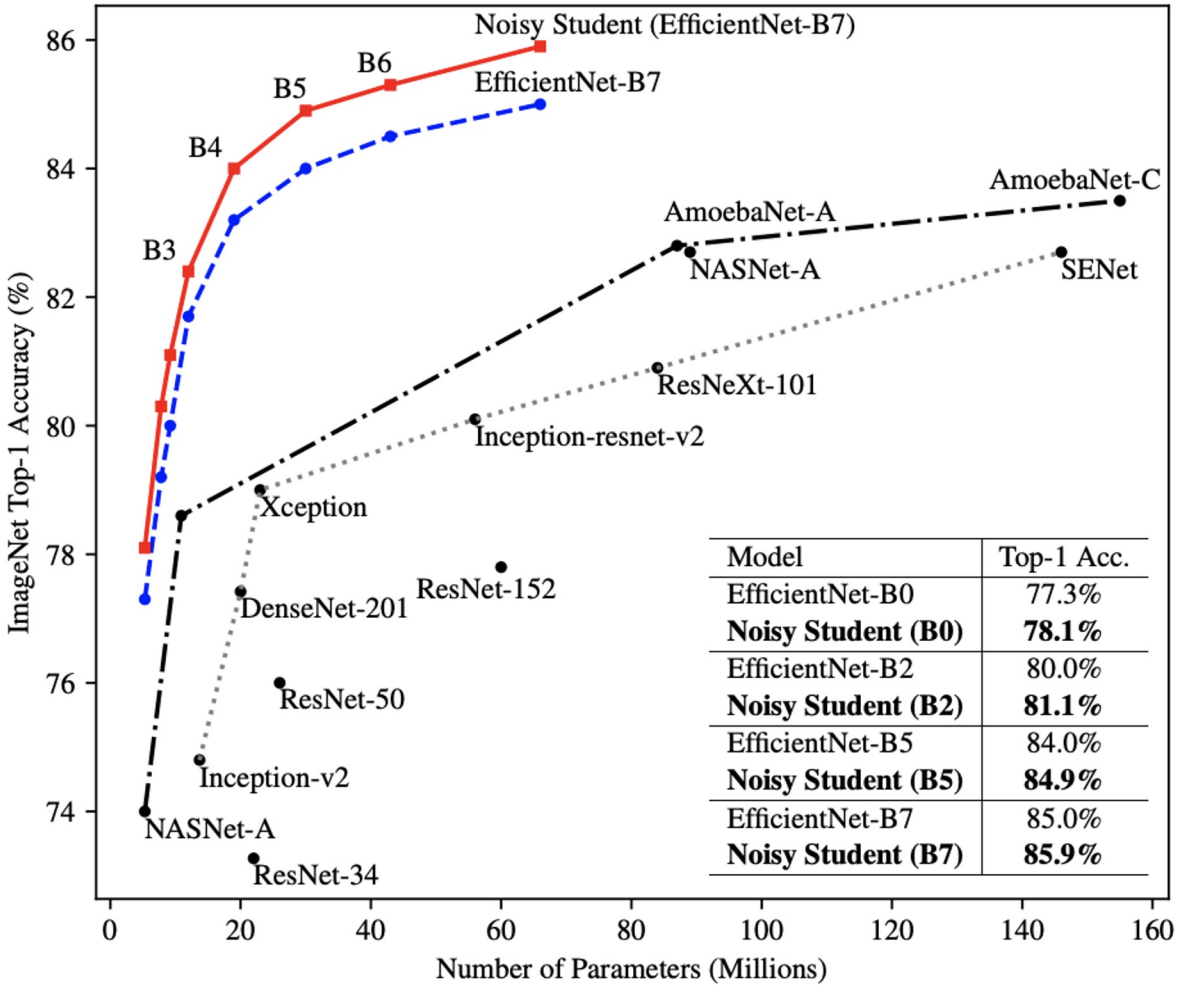
(a) CIFAR-10



(b) SVHN

Results

	ImageNet top-1 acc.	ImageNet-A top-1 acc.	ImageNet-C mCE	ImageNet-P mFR
Prev. SOTA	86.4%	16.6%	45.7	27.8
Ours	**87.4%**	**74.2%**	**31.2**	**16.1**



Model	Top-1 Acc.
EfficientNet-B0	77.3%
Noisy Student (B0)	**78.1%**
EfficientNet-B2	80.0%
Noisy Student (B2)	**81.1%**
EfficientNet-B5	84.0%
Noisy Student (B5)	**84.9%**
EfficientNet-B7	85.0%
Noisy Student (B7)	**85.9%**

Imagenet + 300M unlabelled images

https://arxiv.org/abs/1911.04252 Self-training with Noisy Student improves ImageNet classification

Consistency training details

- 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

Drawbacks of previous approaches

Noising by random noise might not be useful

- Label might change
- Might need to filter by model confidence

Size of supervised set is much smaller than unsupervised set. Data can overfit to supervised set too quickly.

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.

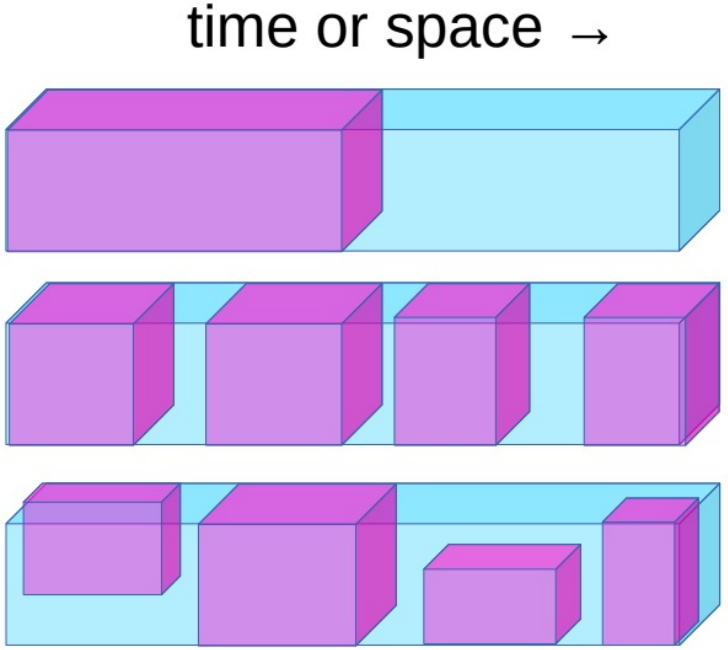
So far, similar approaches haven't worked quite as well for images or videos because of the difficulty of representing distributions over high-dimensional continuous spaces.

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.**



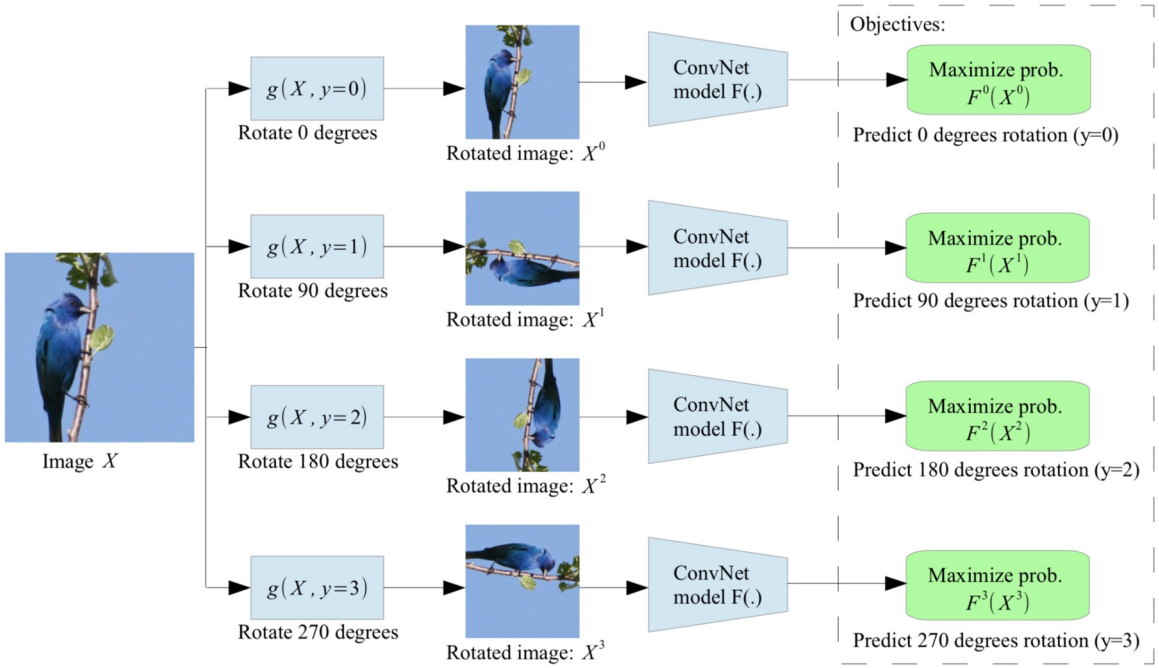
- **Pretend there is a part of the input you don't know and predict that.**
- **Reconstruction = SSL when any part could be known or unknown**

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

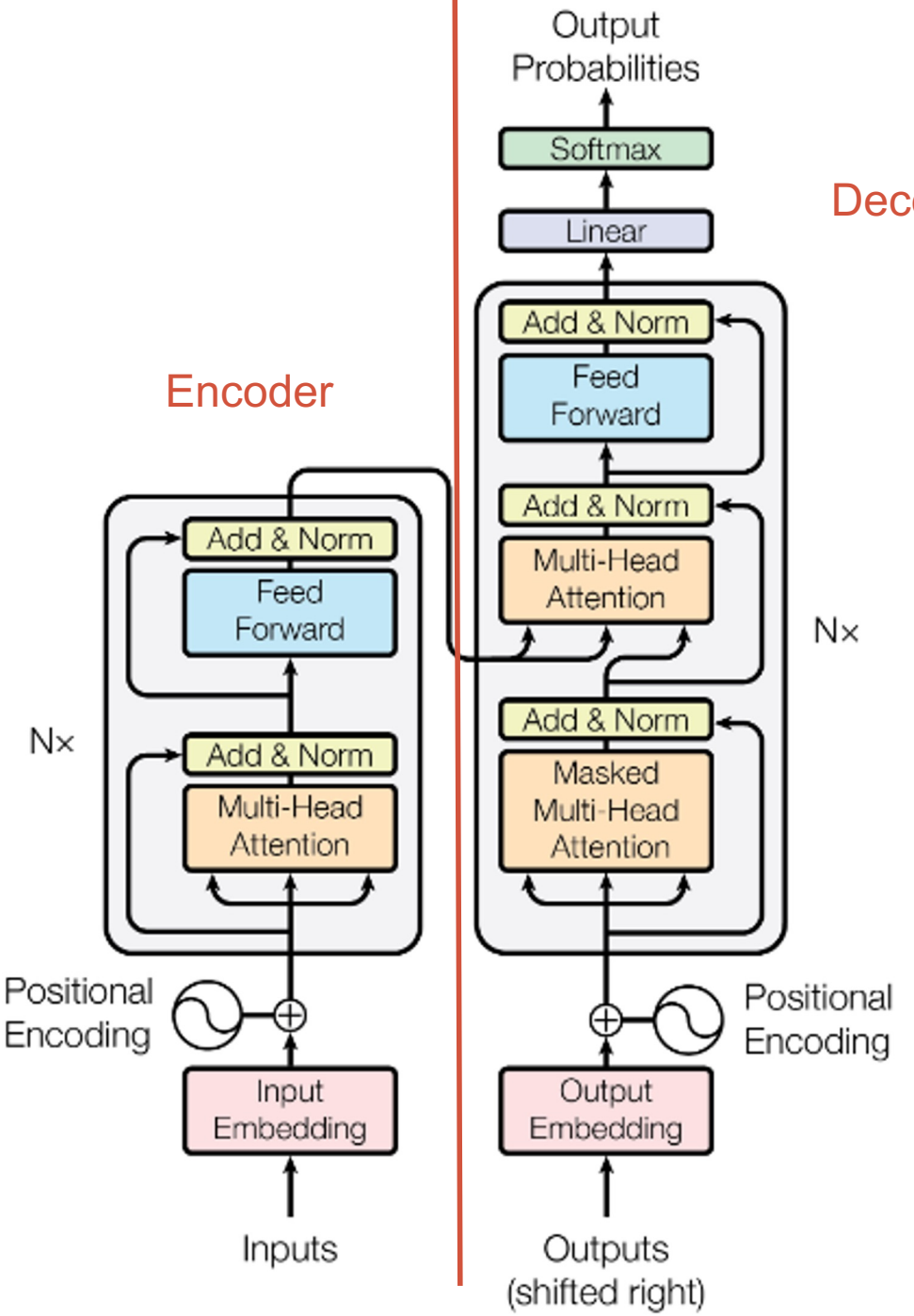
Examples

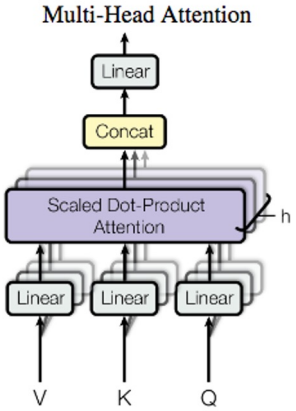
Images

Predict missing patches, predict orientation, etc.



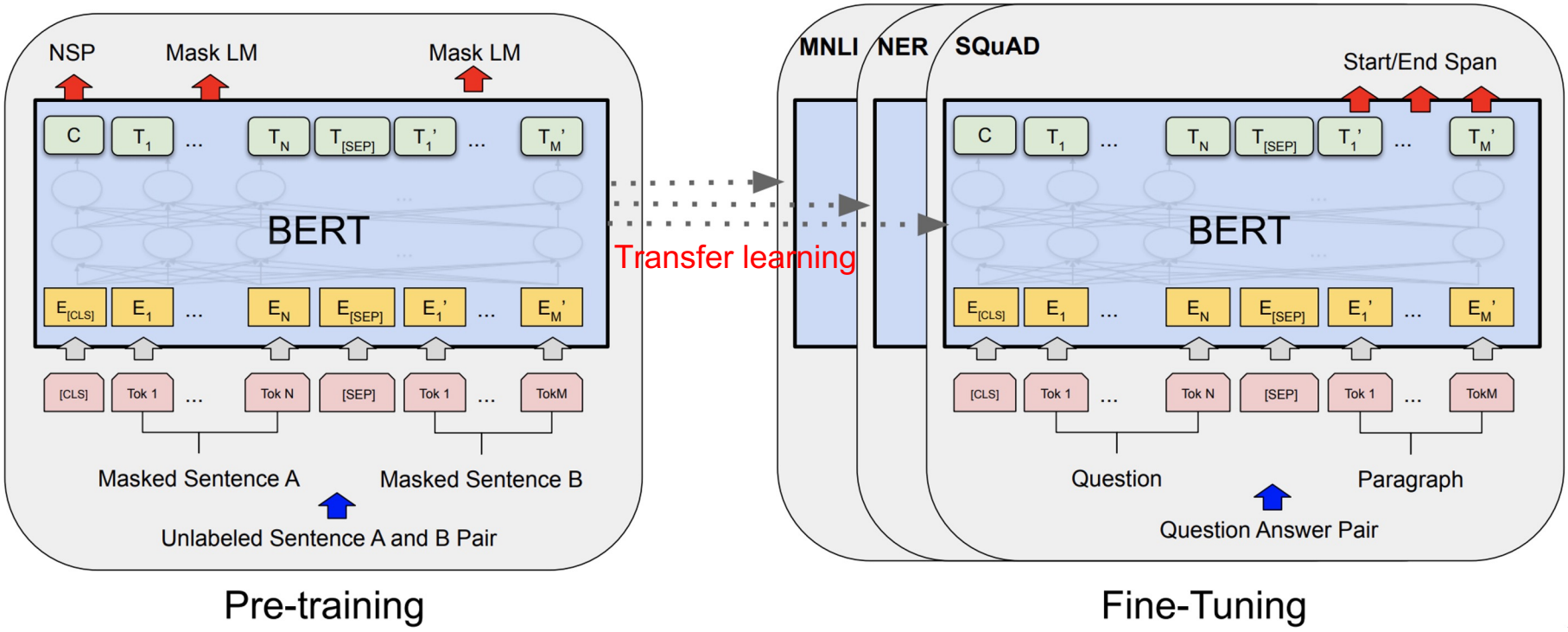
https://arxiv.org/abs/1803.07728 Unsupervised Representation Learning by Predicting Image Rotations


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

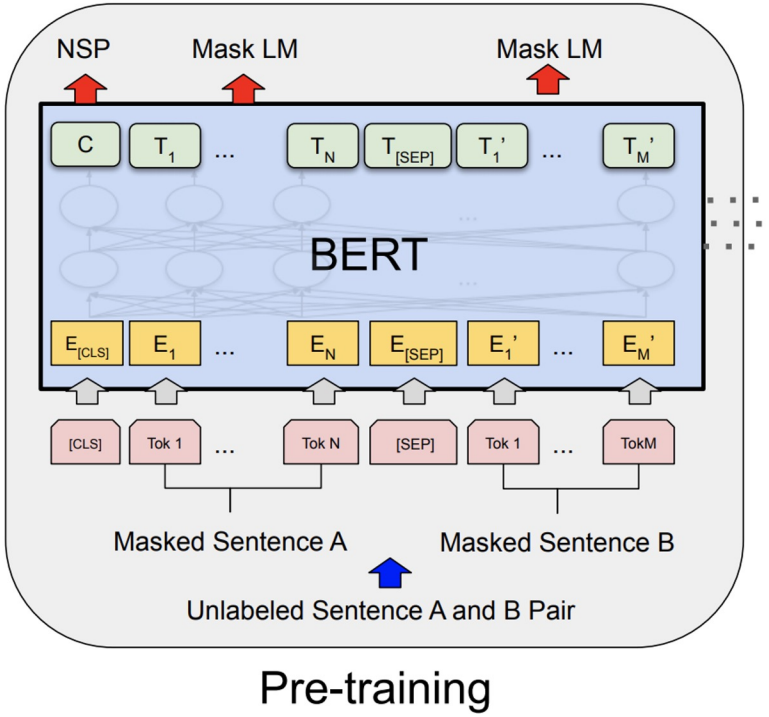
Self-supervised signals
1) Masked words and predict the missing words
2) Next sentence prediction to learn relationship between two sentences



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)

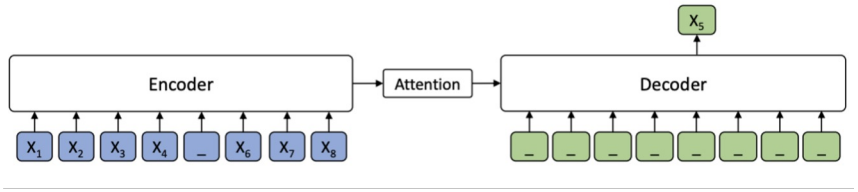
- BERT, ALBERT, RoBERTa
- Seq classification, token classification
- Masked words and predict

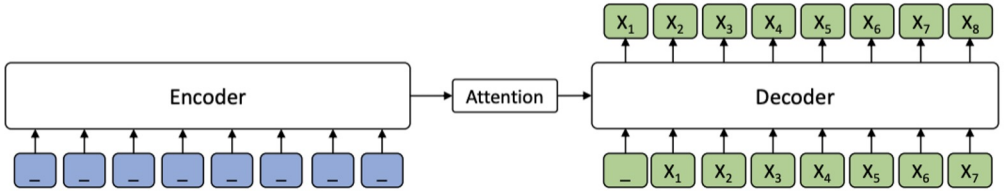
Encoder-decoder (seq2seq)

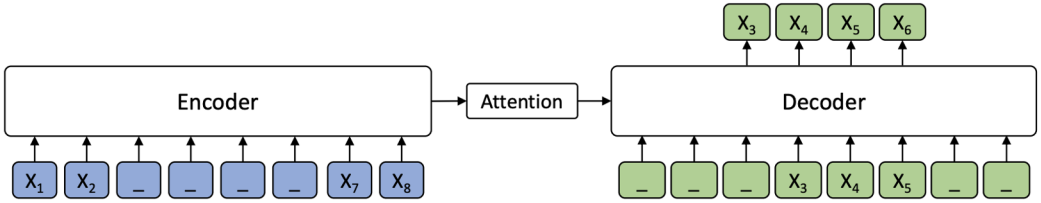
- MASS, BART, T5
- Machine translation, text summary
- Mask phrases

Decoder only (autoregressive)

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



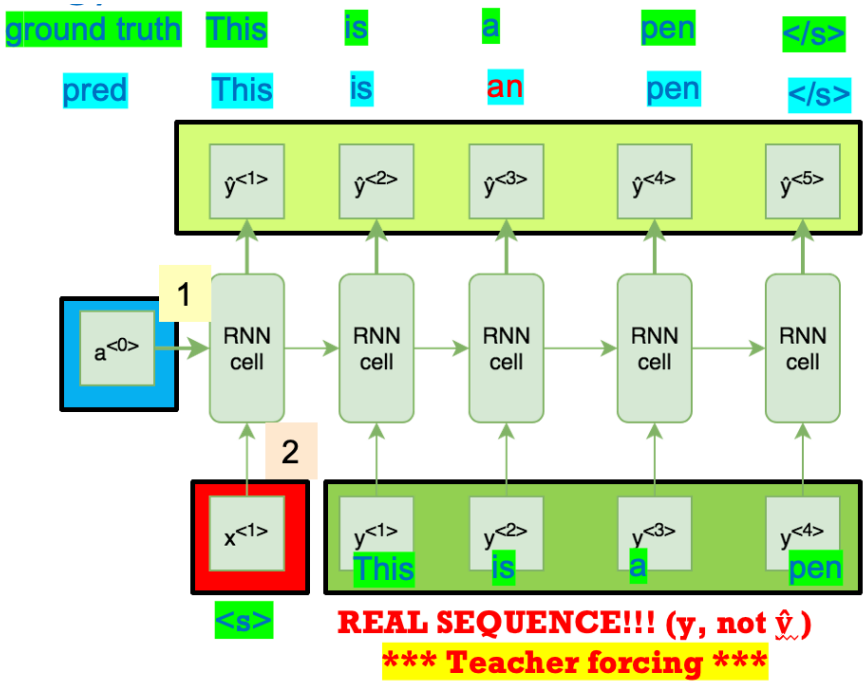




https://huggingface.co/transformers/summary.html https://arxiv.org/pdf/1905.02450.pdf

Autoregressive models

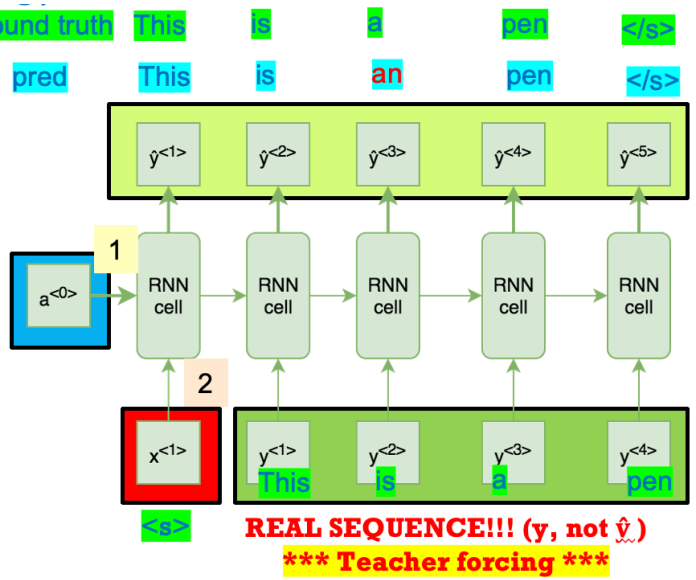
- To generate a sequence of words, we predicts one word at a time.



Autoregressive model (training)

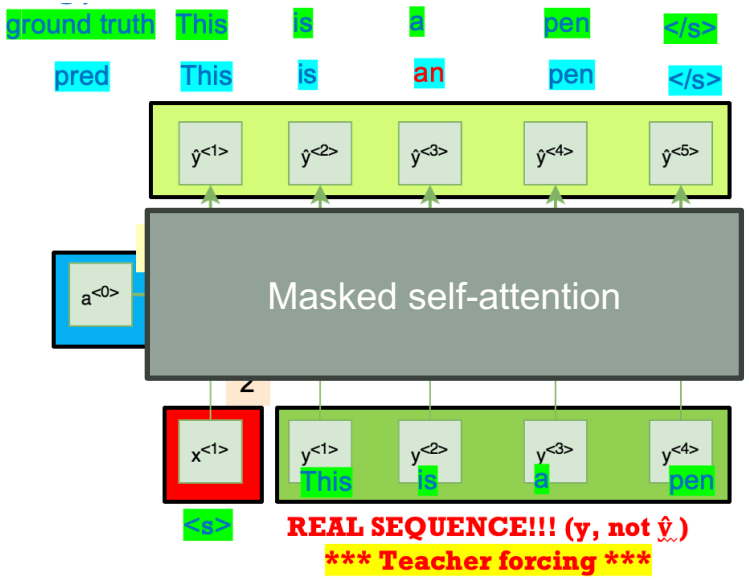
- Training is done by predicting token by token to generate a sequence.
  - Previous output is used as input to ensure consistency of the generation. It knows what it output before.
- Teacher forcing: always force input to be correct

- 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

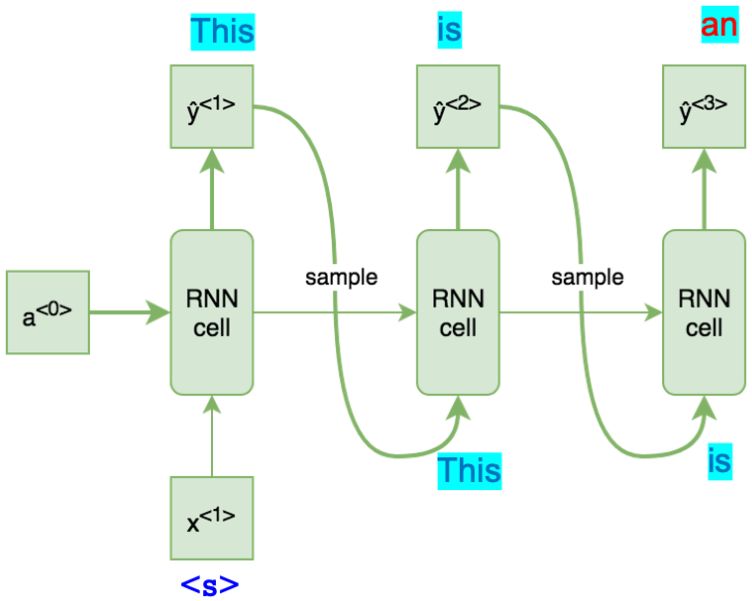
Attends to

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)

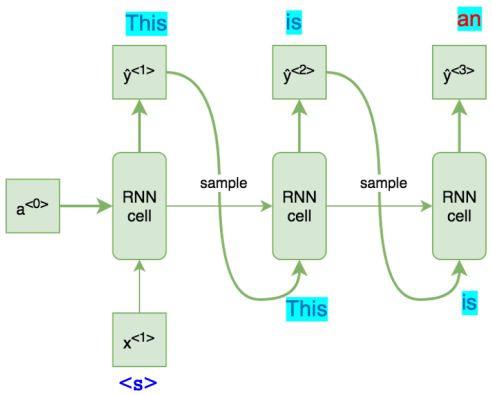


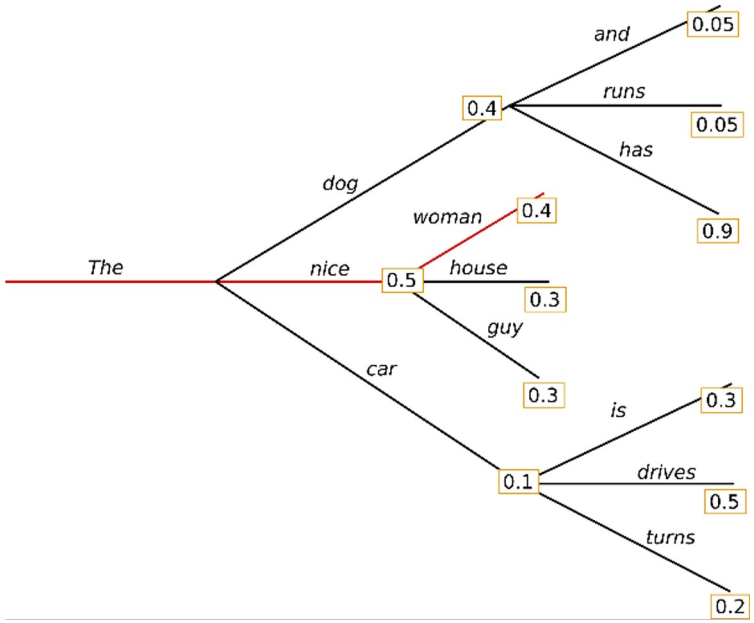
Decoding (inference) for autoregressive model

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



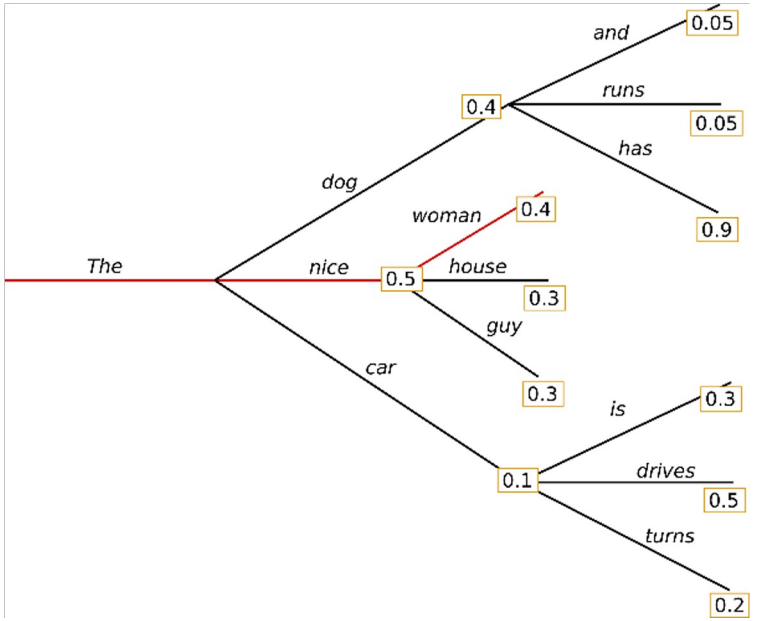


https://huggingface.co/blog/how-to-generate

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

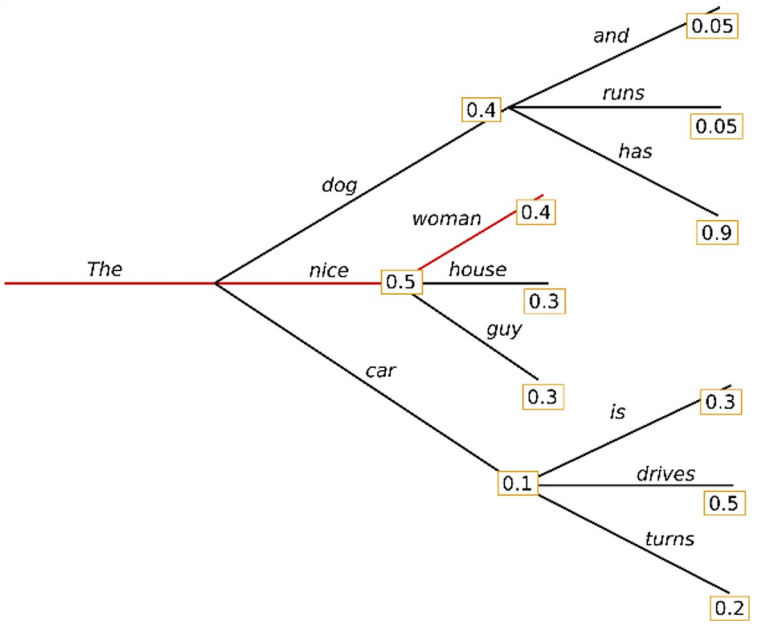
- Greedy decoding is fast and simple, however, the generated text is usually sub-optimal. Sometimes, the model can even repeat itself.

```
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**.

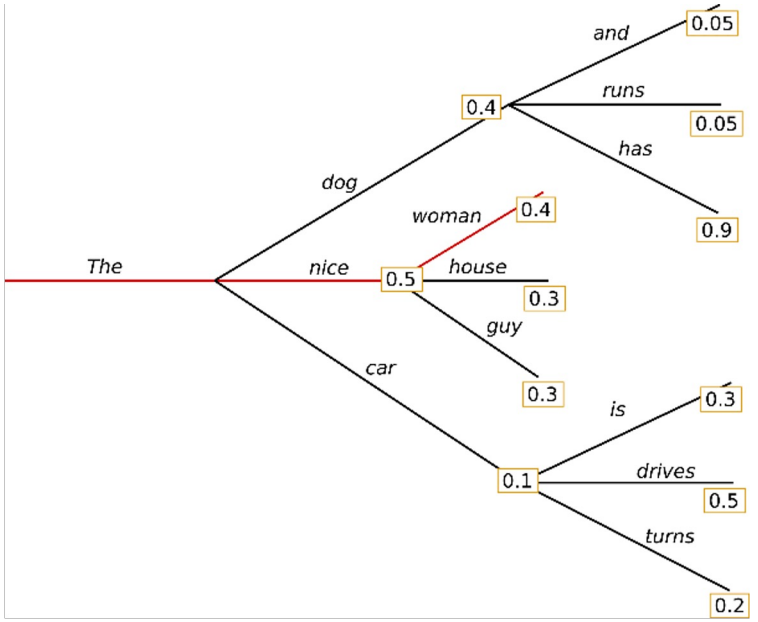
Using random sampling, the probability of selecting the token “nice”, “dog”, and “car” as the continuation is 50%, 40%, and 10%, respectively. The decoding also proceeds until reaching max sequence length or encountering the end-of-sentence token.



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.



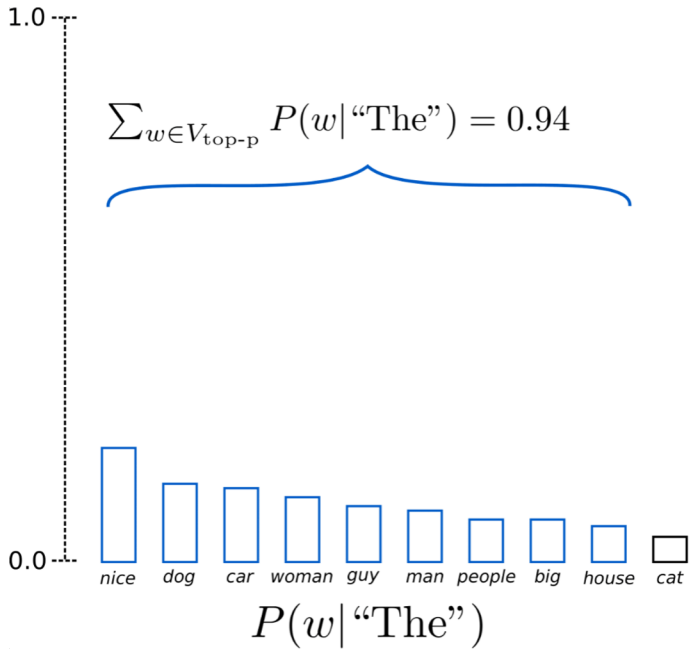
Random sampling

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%

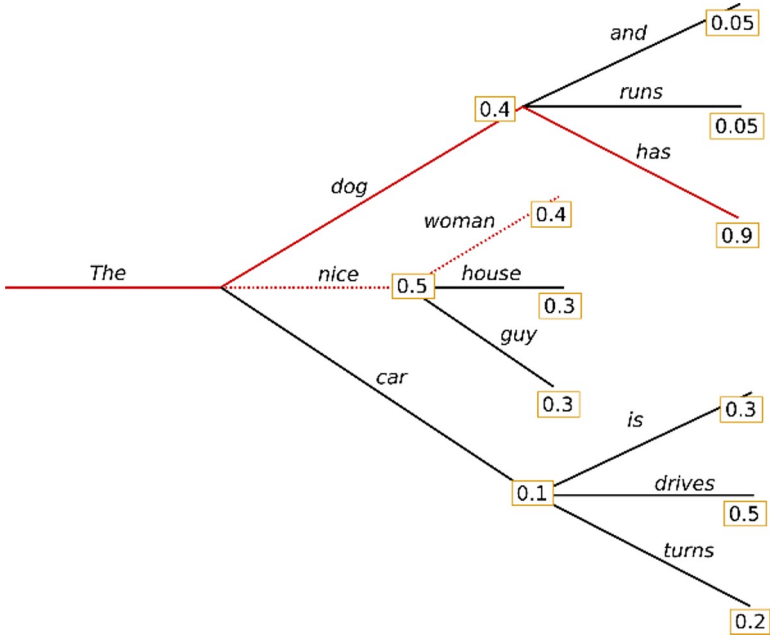
Note that both top-k and top-p can also be used together.



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

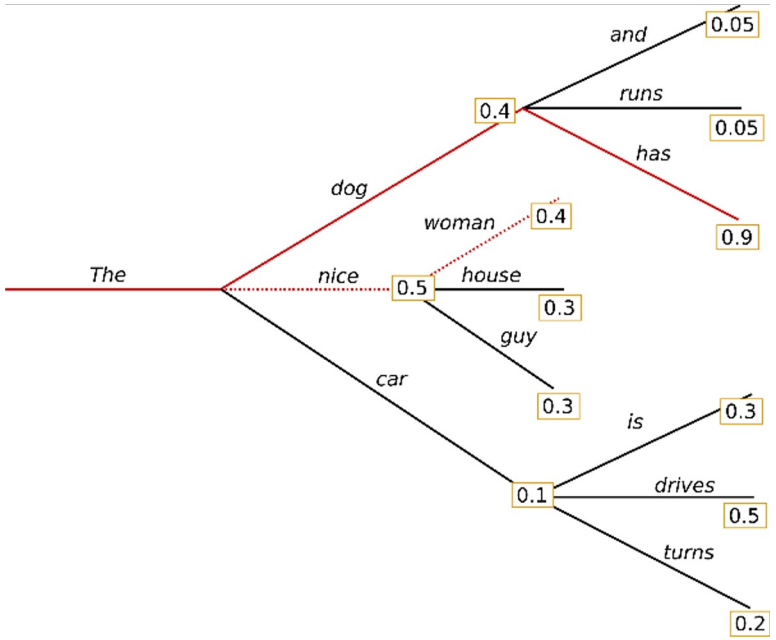
From the example, we consider 2 “beams”, i.e., we only keep the top 2 most probable sequence while we go through the generation process.

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”.

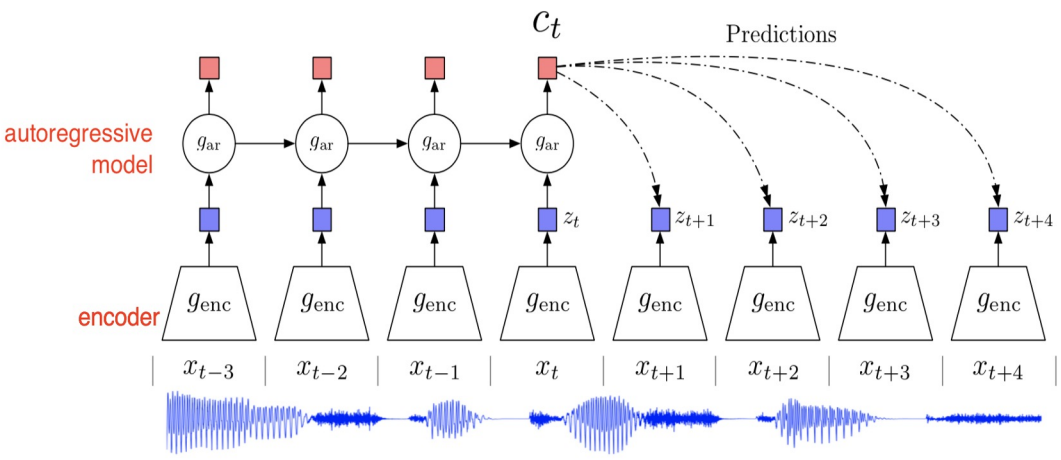


Other Self-supervised Examples

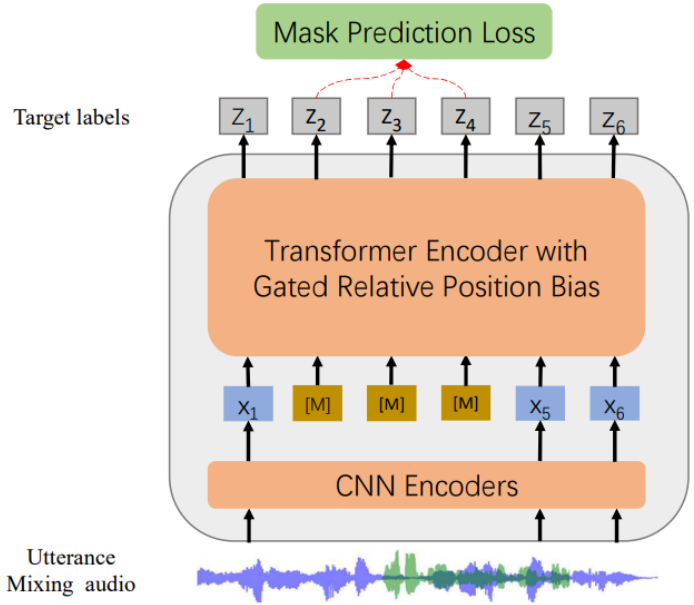
Sound

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

Semi-supervised

- Creating pseudo labels

Representation learning

- Learn representations to use in transfer learning
- Learn via supervised or self-supervise

Self-supervised learning

- Surrogate task from pseudo labels

Autoregressive and Decoding

Next time more details on self-supervised/contrastive/consistency learning