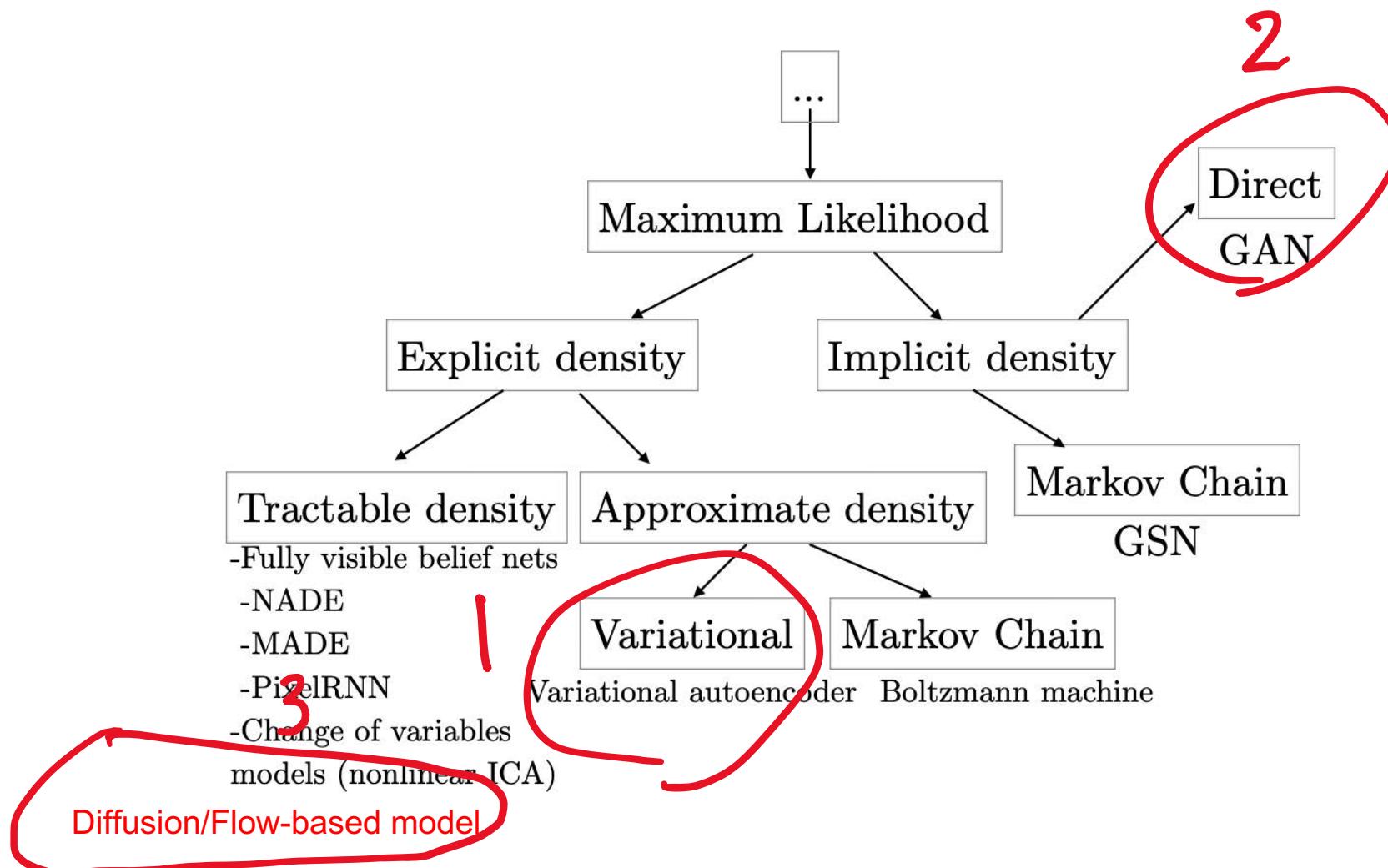


UNSUPERVISED LEARNING

Deep Generative Models

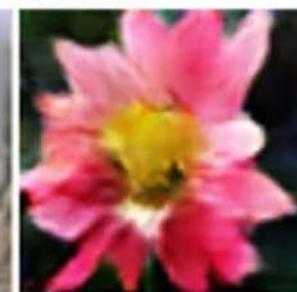
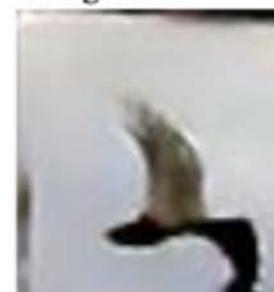
- A deep learning model that can be used to generate X



Other uses

- StackGAN: text to photo

(a) StackGAN
Stage-I
64x64
images



(b) StackGAN
Stage-II
256x256
images

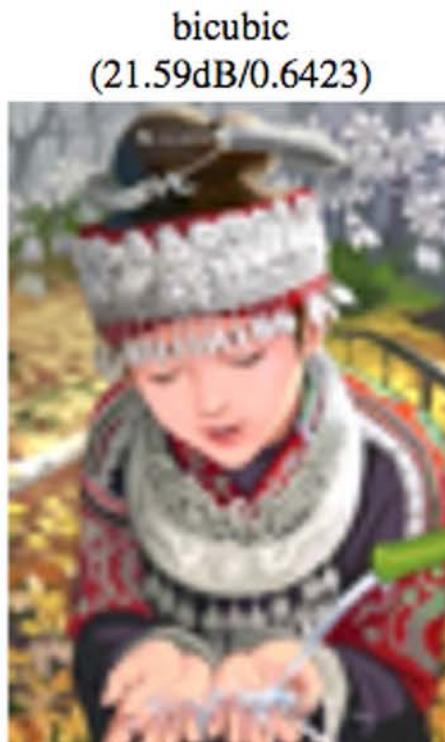


(c) Vanilla GAN
256x256
images



SRGAN

- <https://arxiv.org/pdf/1609.04802.pdf>



Content-aware Filling



Globally and Locally Consistent Image Completion [Iizuka et al.,

Style transfer with cycleGAN



<https://hardikbansal.github.io/CycleGANBlog/>

Zebras  Horses



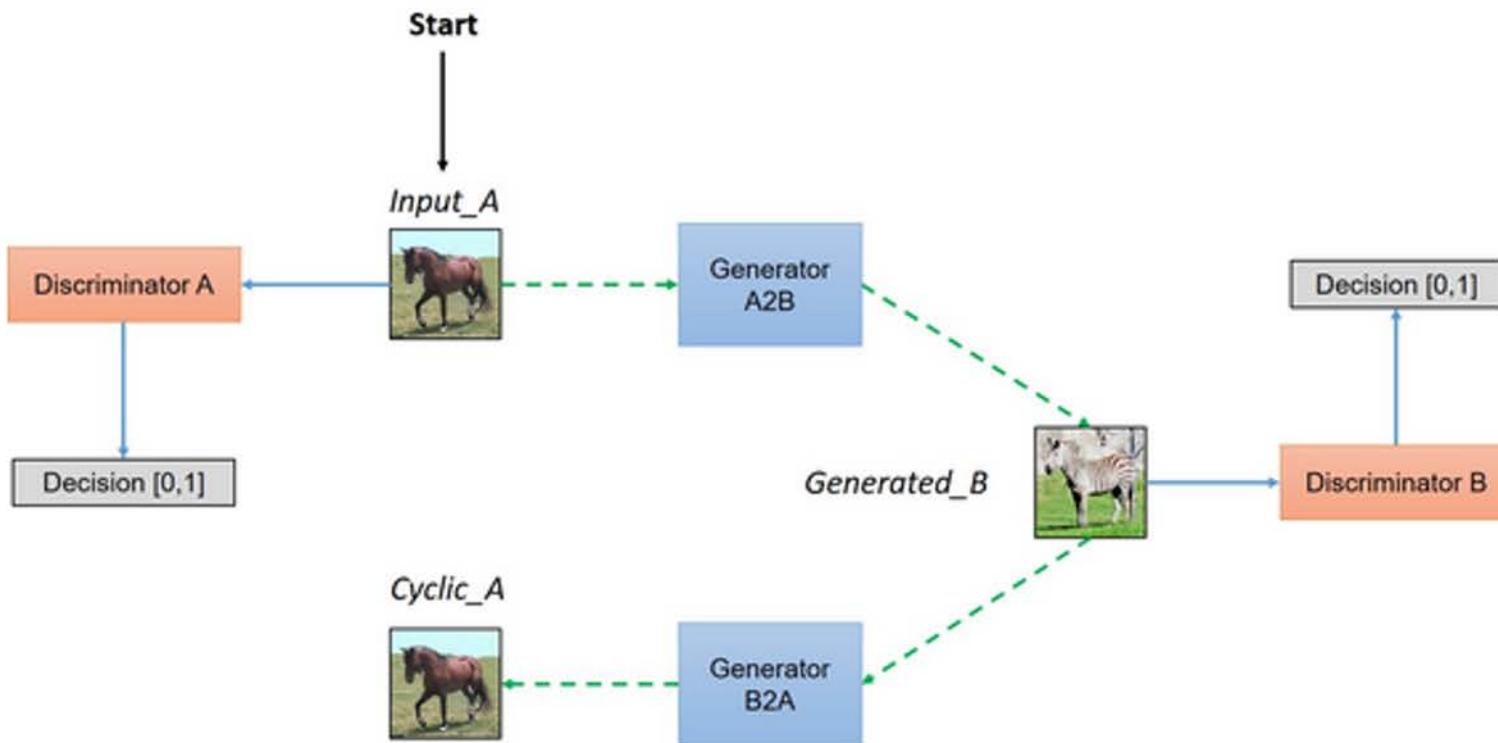
zebra → horse

Cycle consistency



horse → zebra

Note, you don't
need a paired
dataset



Cycle consistency

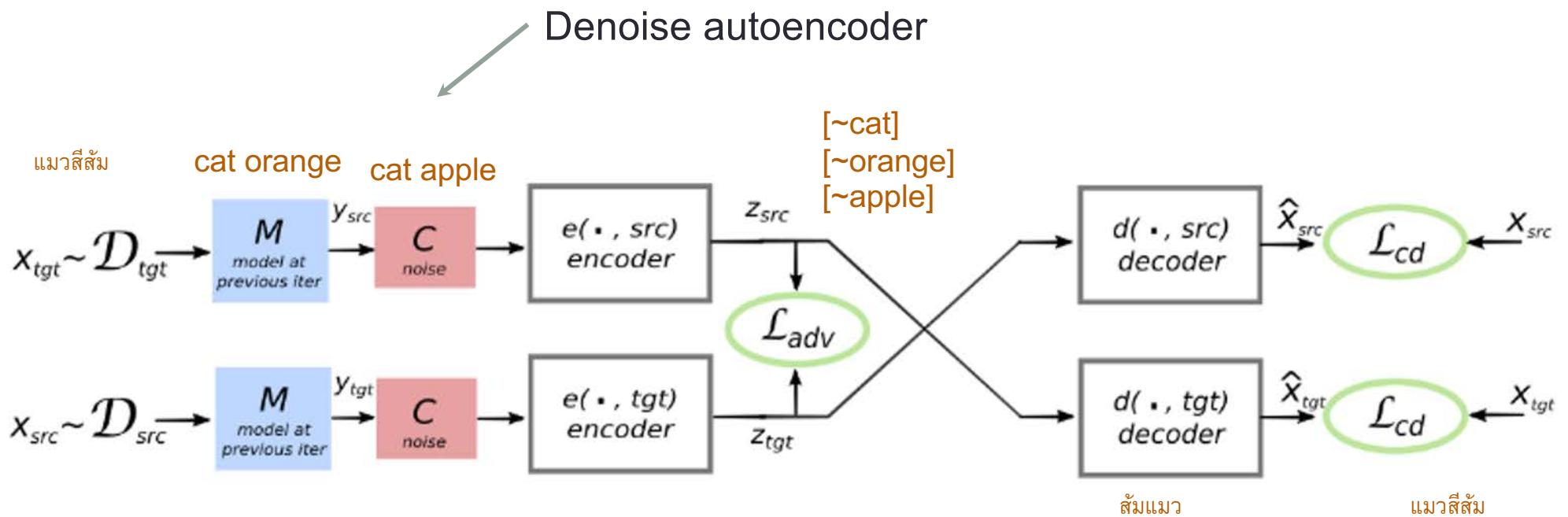
Machine translation with no parallel text

- MT usually requires parallel text

This is a cat

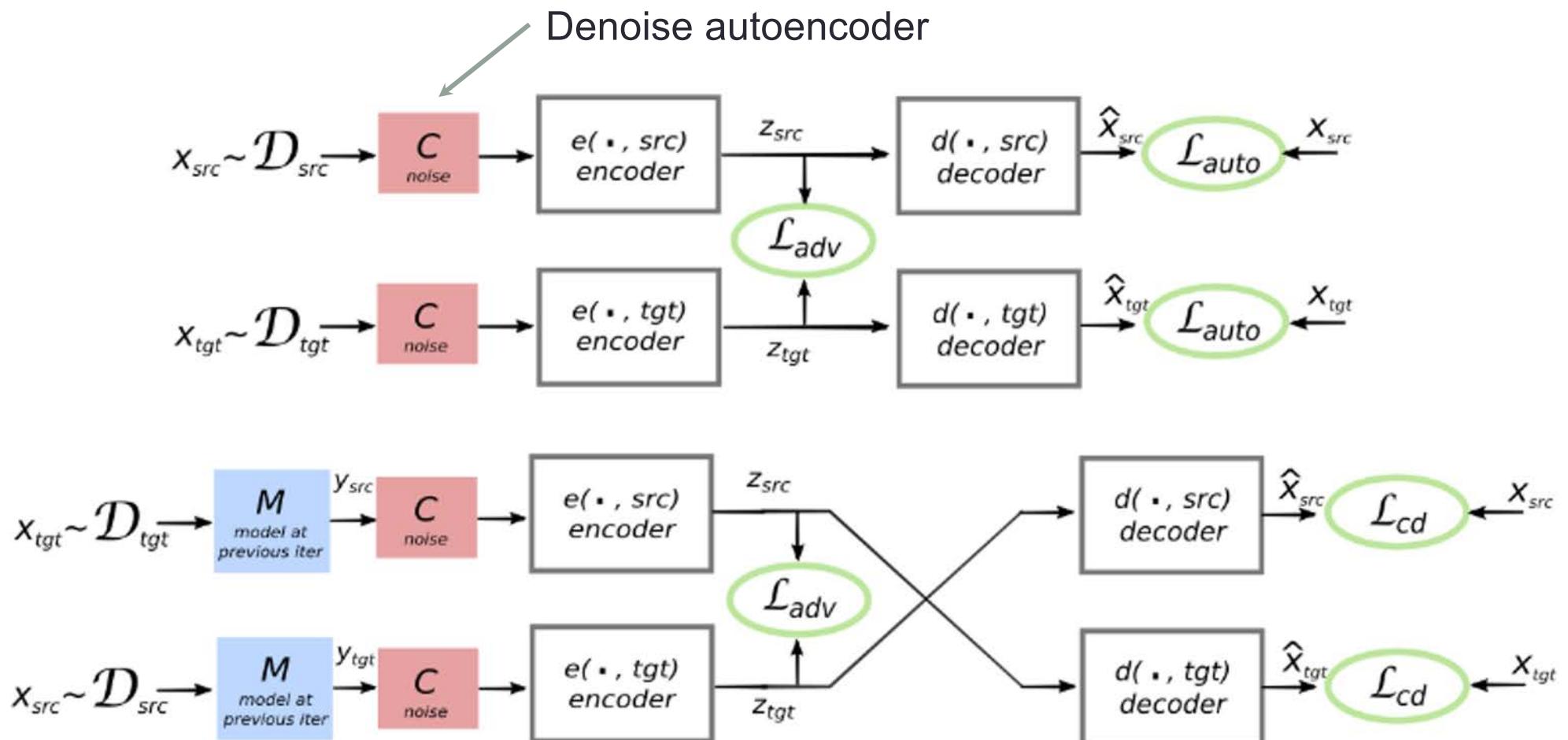
นี่คือแมว

- Most of the time we don't have parallel text. Just text.
- Can we still do MT?
 - Use GANs + Autoencoders



Use GAN Loss (\mathcal{L}_{adv}) to enforce that source and target language pairs share the same distributions.

Then enforce the translation distribution matches.



Results

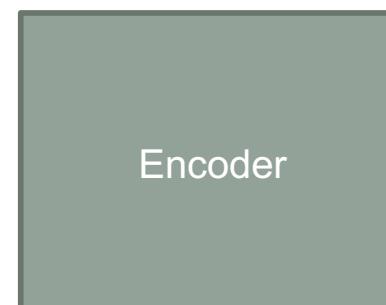
	Multi30k-Task1				WMT			
	en-fr	fr-en	de-en	en-de	en-fr	fr-en	de-en	en-de
Supervised	56.83	50.77	38.38	35.16	27.97	26.13	25.61	21.33
word-by-word	8.54	16.77	15.72	5.39	6.28	10.09	10.77	7.06
word reordering	-	-	-	-	6.68	11.69	10.84	6.70
oracle word reordering	11.62	24.88	18.27	6.79	10.12	20.64	19.42	11.57
Our model: 1st iteration	27.48	28.07	23.69	19.32	12.10	11.79	11.10	8.86
Our model: 2nd iteration	31.72	30.49	24.73	21.16	14.42	13.49	13.25	9.75
Our model: 3rd iteration	32.76	32.07	26.26	22.74	15.05	14.31	13.33	9.64

Adversarial debiasing

- Use the discriminator to force the encoder to drop bias information



Training data



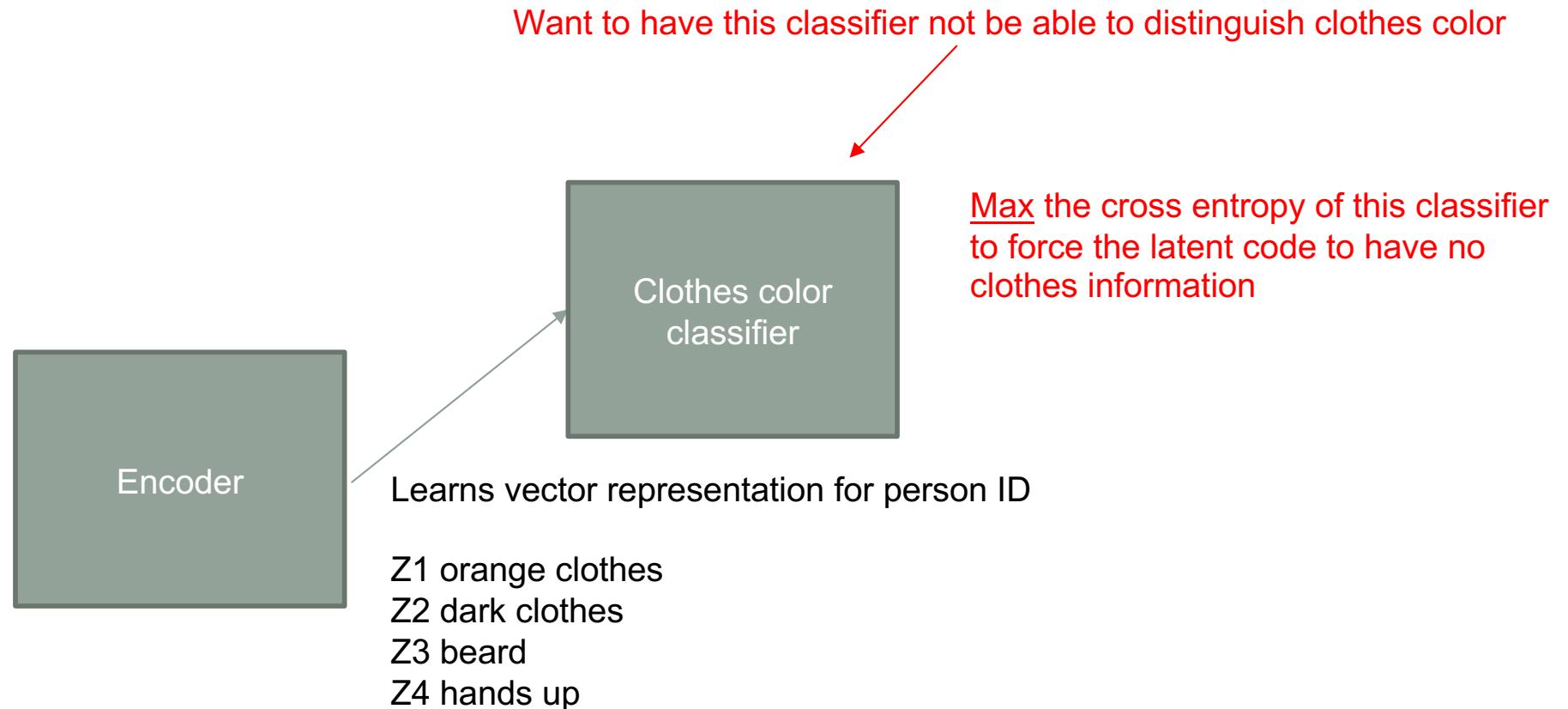
Learns vector representation for person ID

Z1 orange clothes
Z2 dark clothes
Z3 beard
Z4 hands up

How to remove unwanted information in latent code?



Adversarial debiasing



Reading <https://arxiv.org/pdf/1801.07593.pdf>

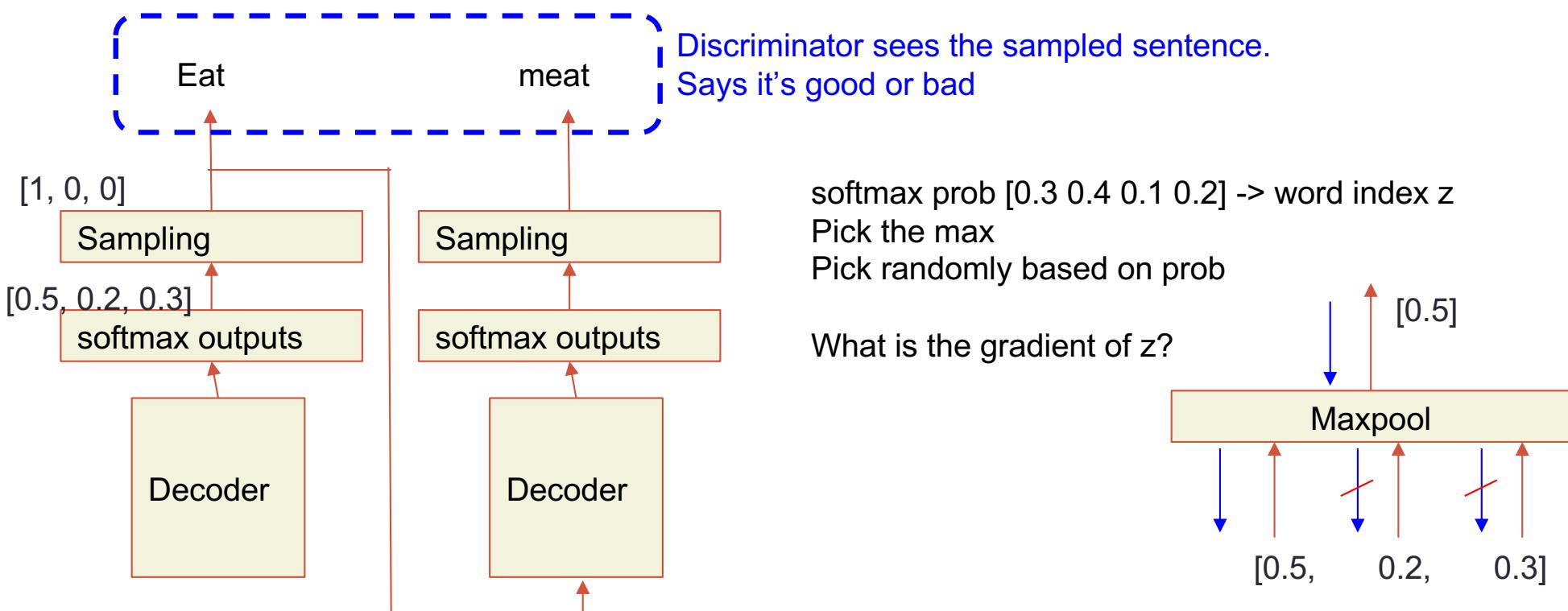
Example application usages

Bioinformatics <https://www.biorxiv.org/content/10.1101/2021.04.14.439903v1.full.pdf>

Face recognition <https://arxiv.org/pdf/1911.08080.pdf>

GAN for text generation

Autoregressive decoding includes a sampling process
Cannot gradient descent

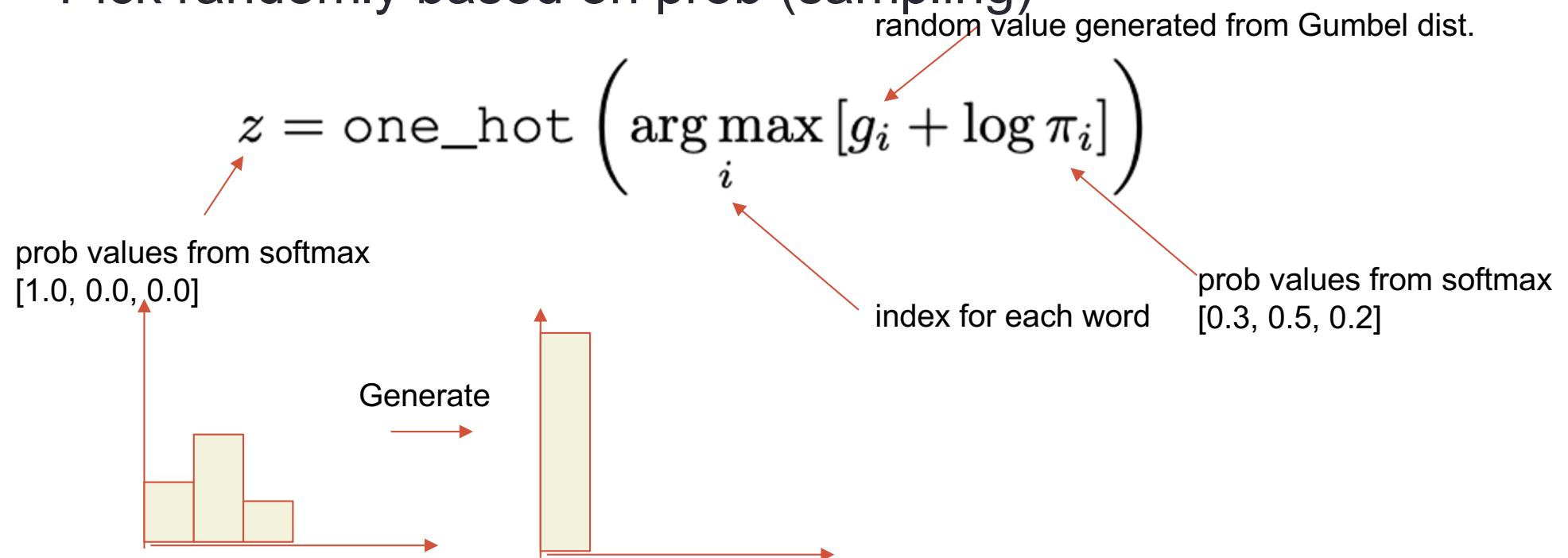


Two popular methods: REINFORCE, Gumbel-Softmax approximation (<https://arxiv.org/abs/1611.01144>)

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z

Pick randomly based on prob (sampling)



Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z

Pick randomly based on prob

$$z = \text{one_hot} \left(\arg \max_i [g_i + \log \pi_i] \right)$$

random value generated from Gumbel dist.

prob values from softmax

index for each word

estimate using Gumbel trick

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^k \exp((\log(\pi_j) + g_j)/\tau)} \quad \text{for } i = 1, \dots, k.$$

Gumbel Softmax

From softmax prob [0.3 0.4 0.1 0.2] -> word index z

Pick randomly based on prob

$$z = \text{one_hot} \left(\arg \max_i [g_i + \log \pi_i] \right)$$

random value generated from Gumbel dist.

prob values from softmax

index for each word

Not a one hot

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^k \exp((\log(\pi_j) + g_j)/\tau)}$$

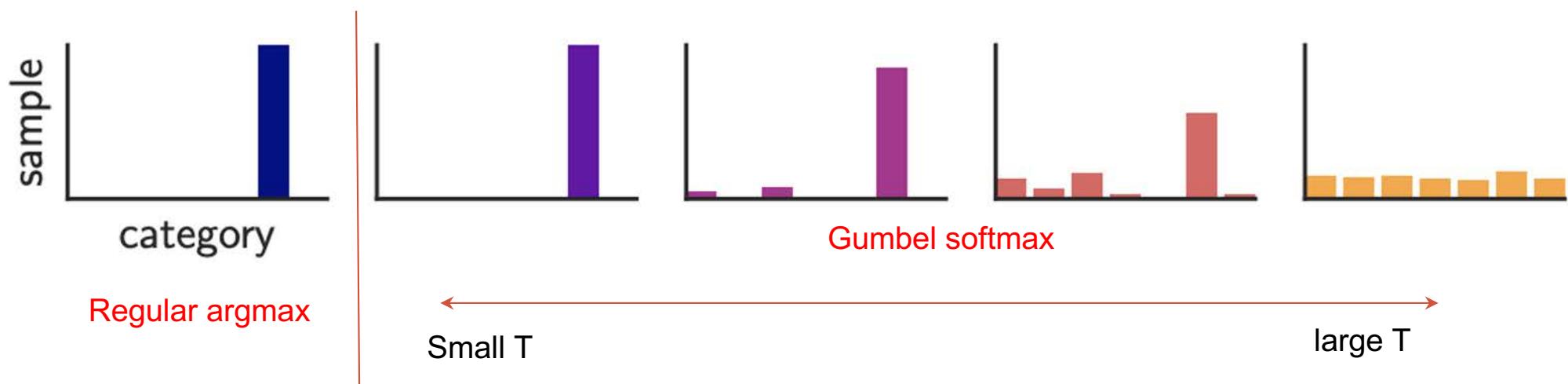
Temperature parameter

for $i = 1, \dots, k$.

This rescales the distribution

Gumbel Softmax

Temperature can be scaled
y can be back propagated through



Not a one hot

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^k \exp((\log(\pi_j) + g_j)/\tau)}$$

Temperature parameter

for $i = 1, \dots, k.$

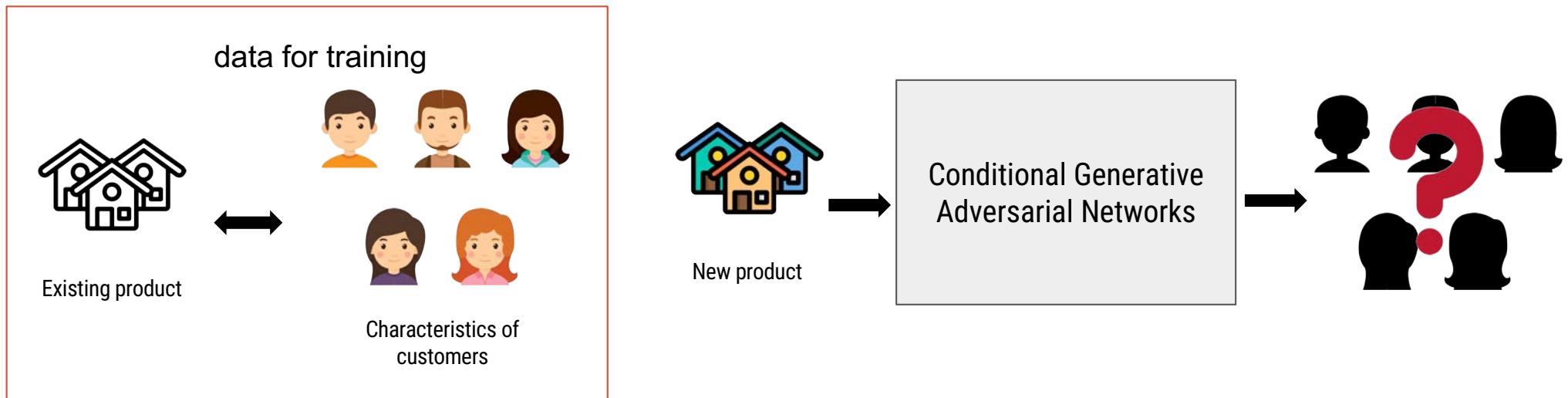
This rescales the distribution

ML for product development

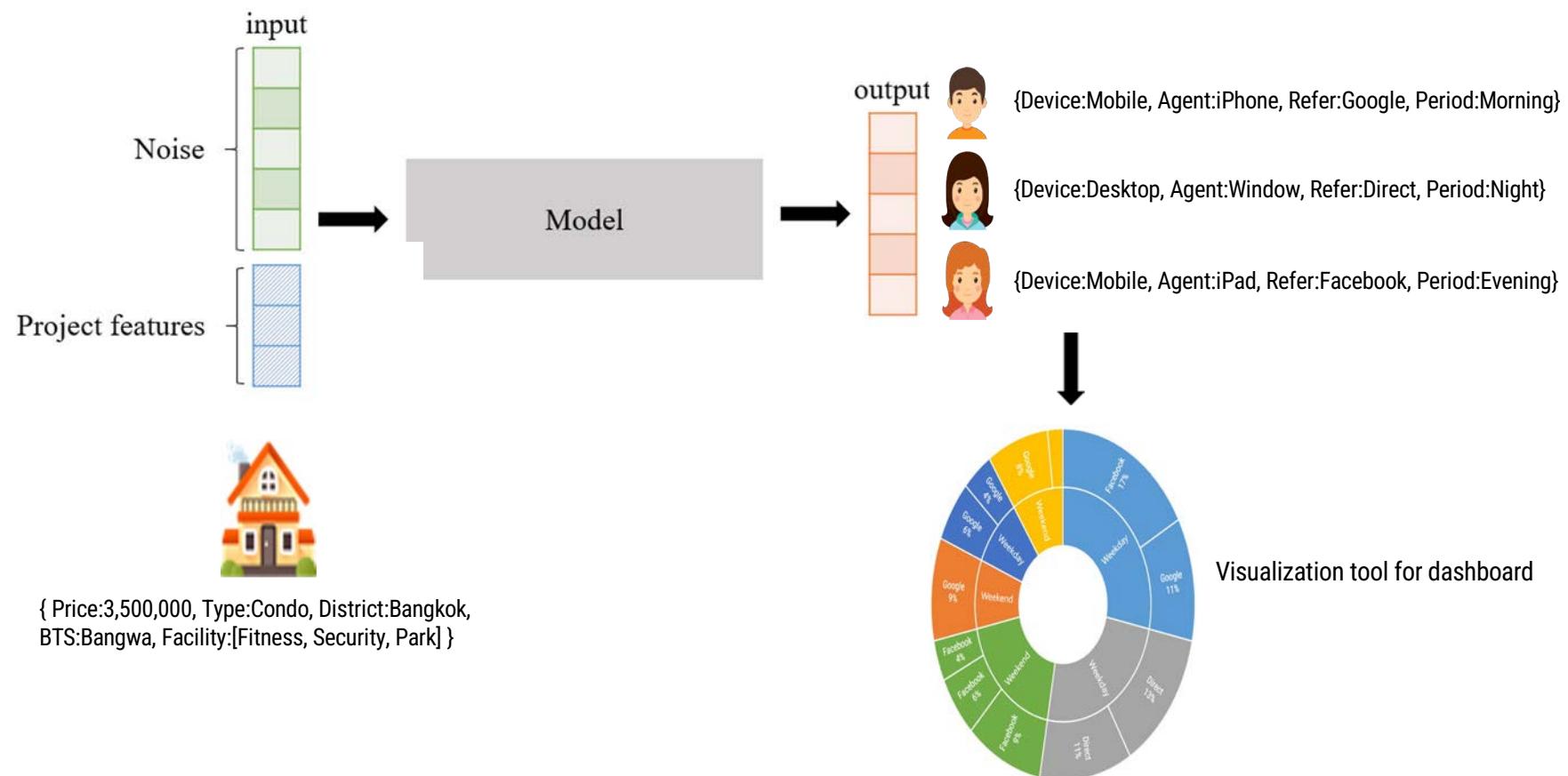
For real estates, no two products are the same. Development based on gut feeling.

Make some informative guess about a new product

- popularity
- the type of potential buyers
- whether to add or remove some features
- the best marketing channel

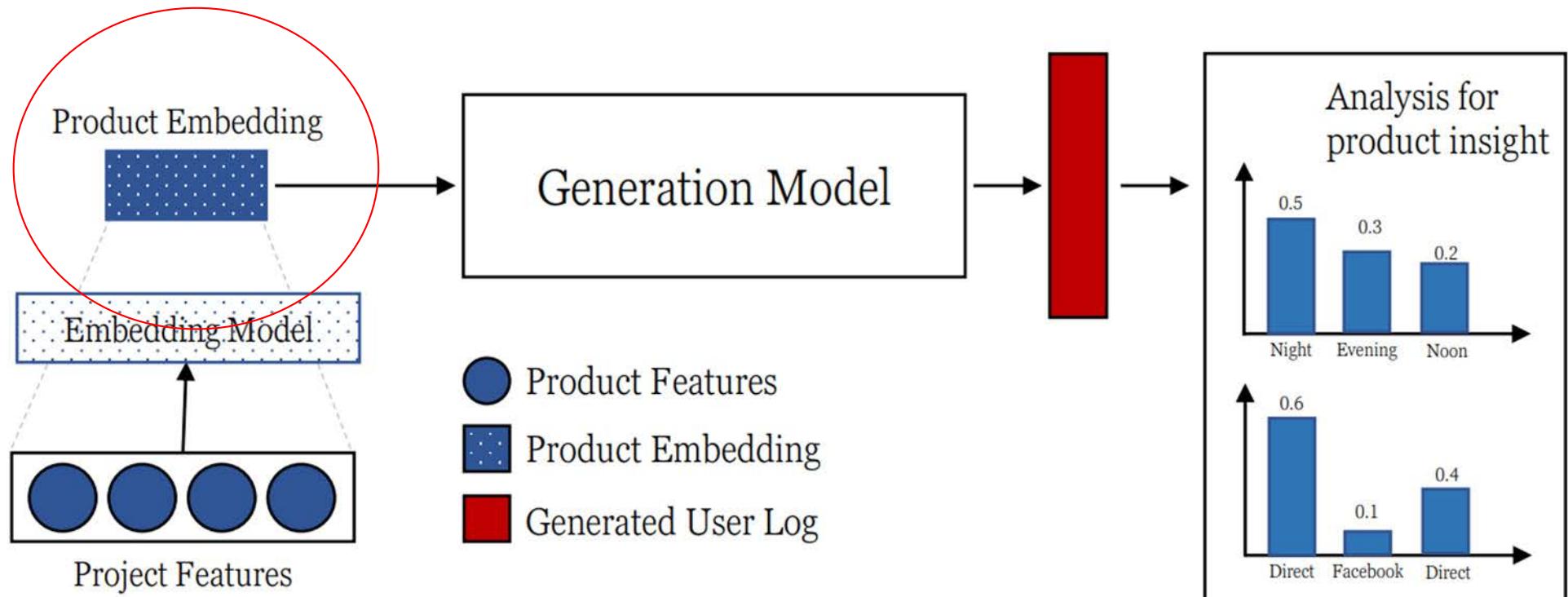


System overview



Generating Realistic Users Using Generative Adversarial Network With Recommendation-Based Embedding, IEEE Access, 2020

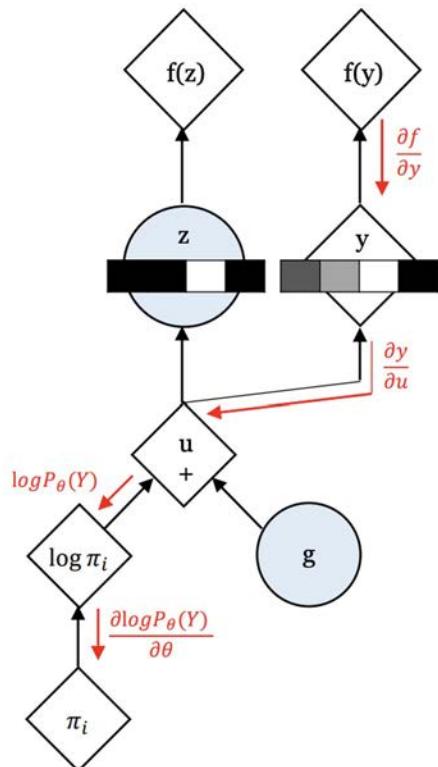
Embedding learned from our recommender system



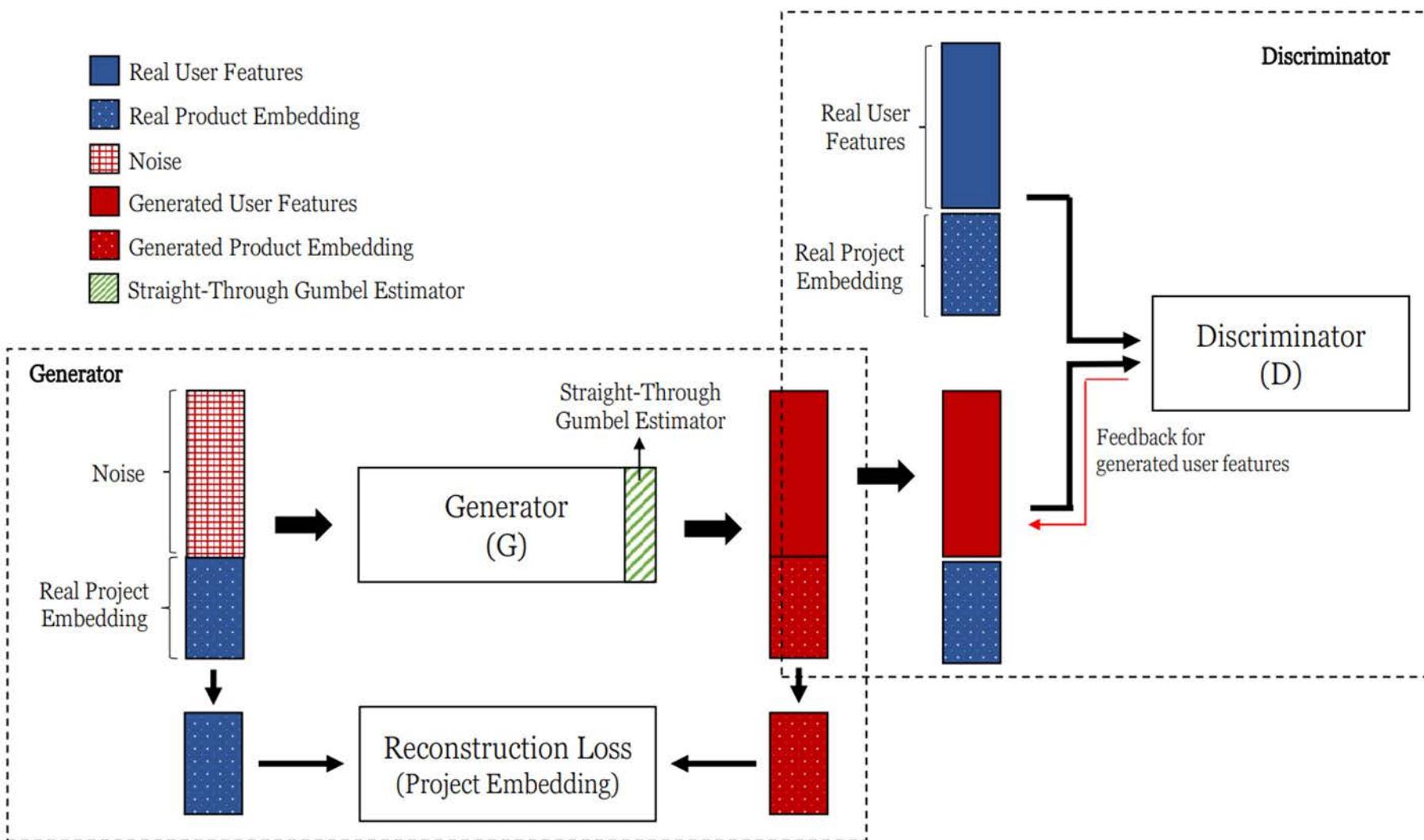
Straight-through Gumbel estimator

Forward to
the discrim.

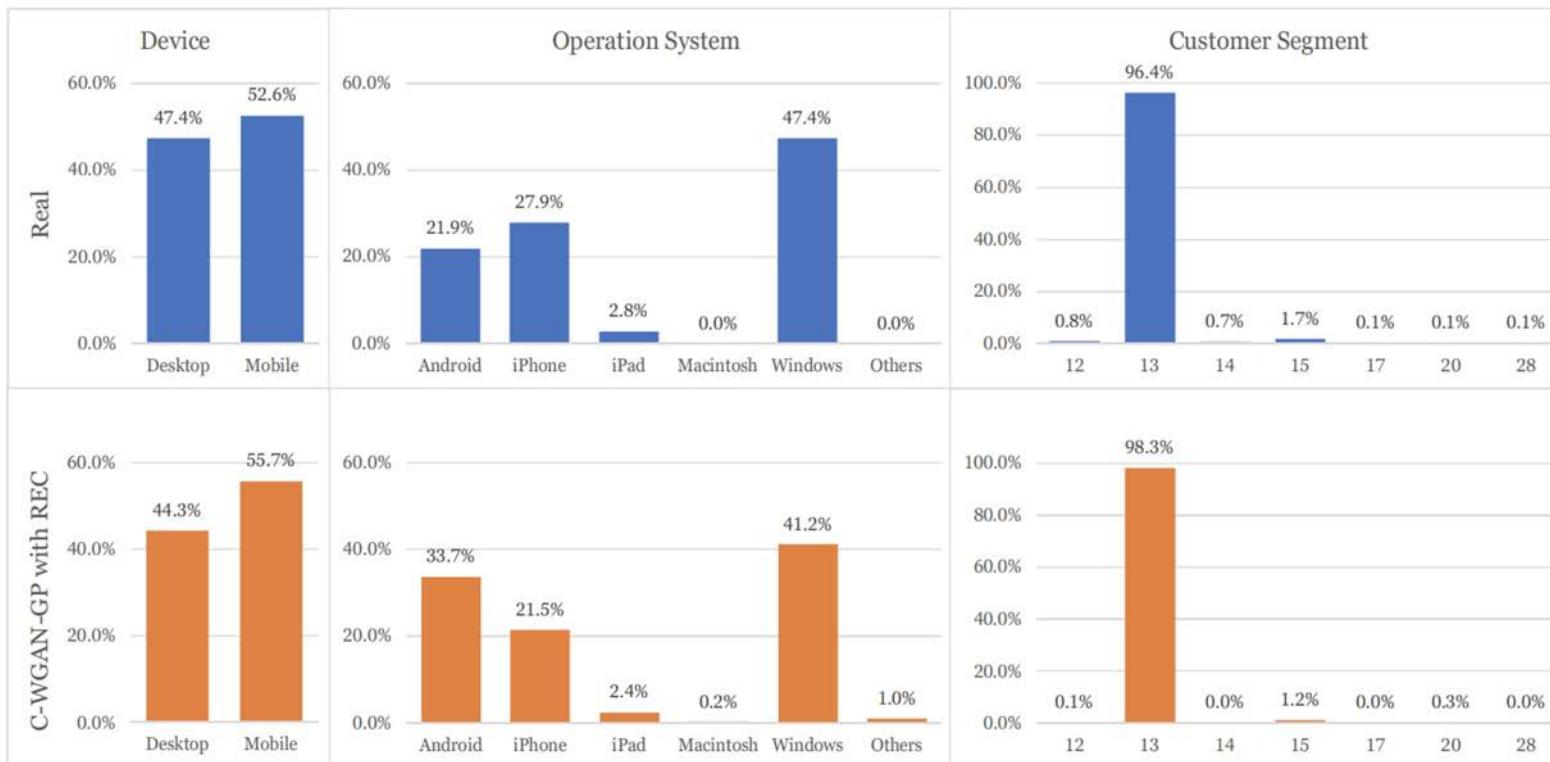
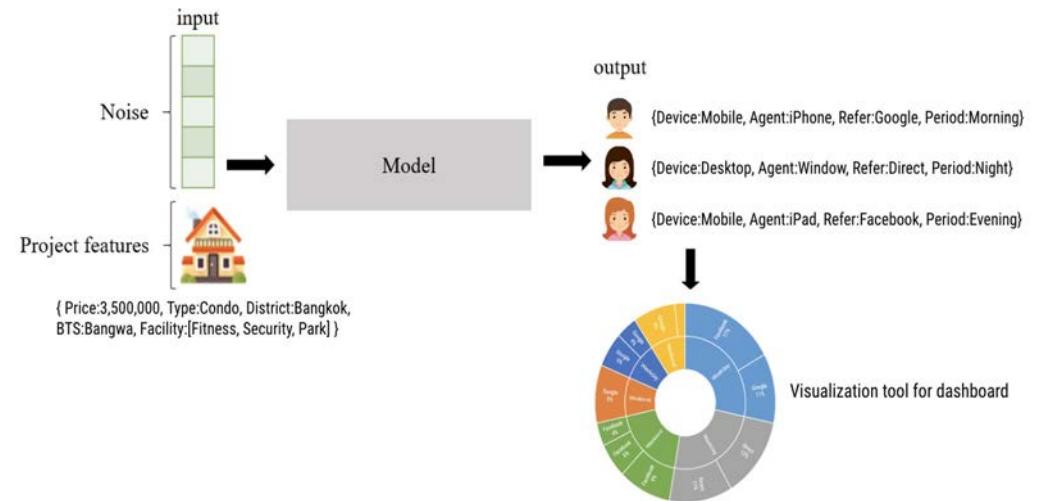
Backward from
the discrim.



The generator generates both the argmax and the Gumbel version. The discriminator uses the argmax version as input. However, the gradient is passed through the Gumbel version.



Predicting users



Why GAN?

vs supervised learning

- supervised learning yields one correct answer (not learning the distribution)
- cannot be used to generate examples

vs other distribution learning methods

- non-parametric
- better than other methods for multi-modal distributions
- generate things that differ from the training data but still “realistic”

GAN vs VAE

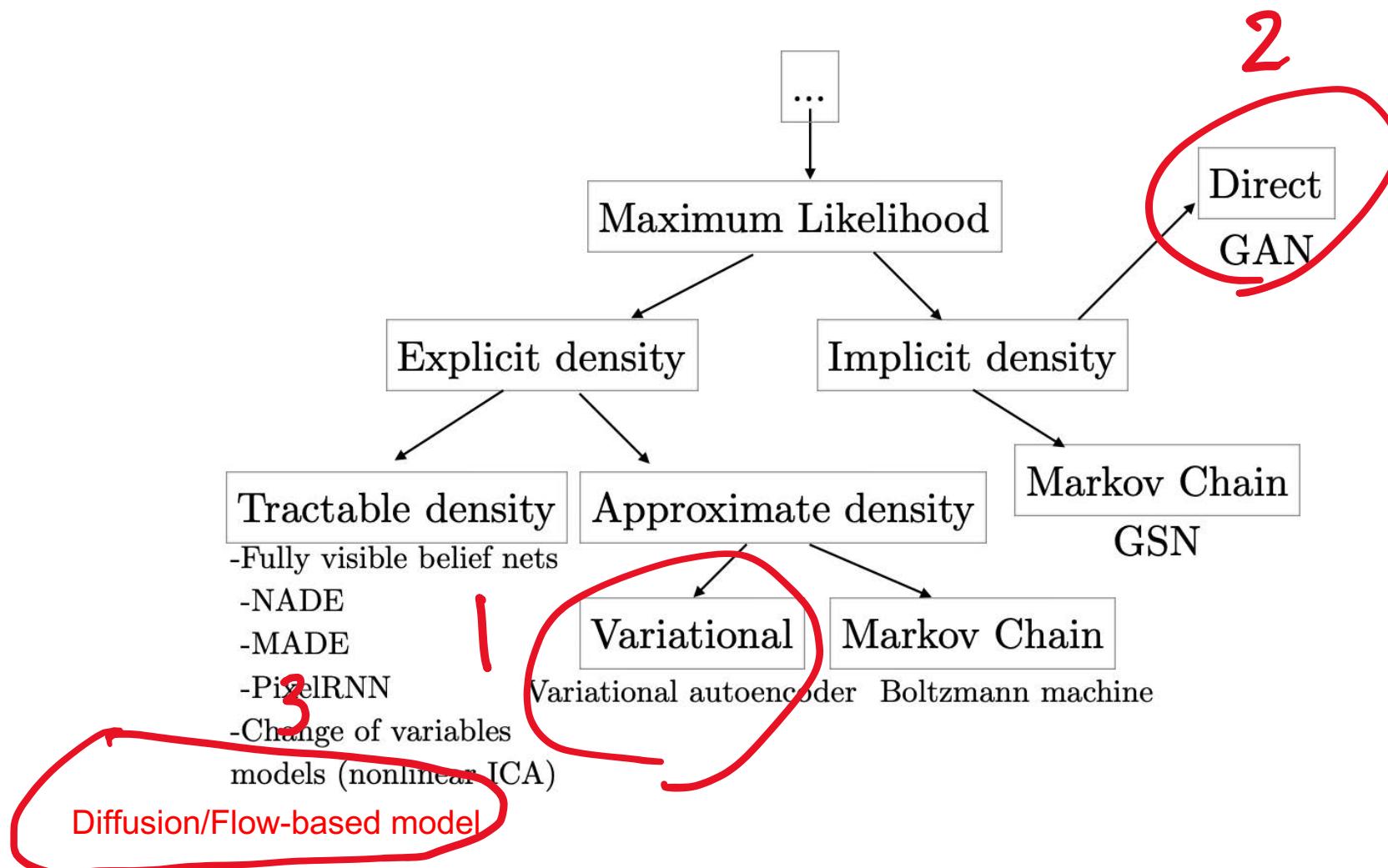
- Typically GAN provides better quality
- Typically VAE provides more controllability
 - Representation more disentangled
 - Can easily interpolate two inputs
- VAE can encode nice embeddings/features for other task
- Research exist to fill both gaps

Deep generative models

- Autoencoders
 - Variational autoencoders
- GAN
- Diffusion/Flow

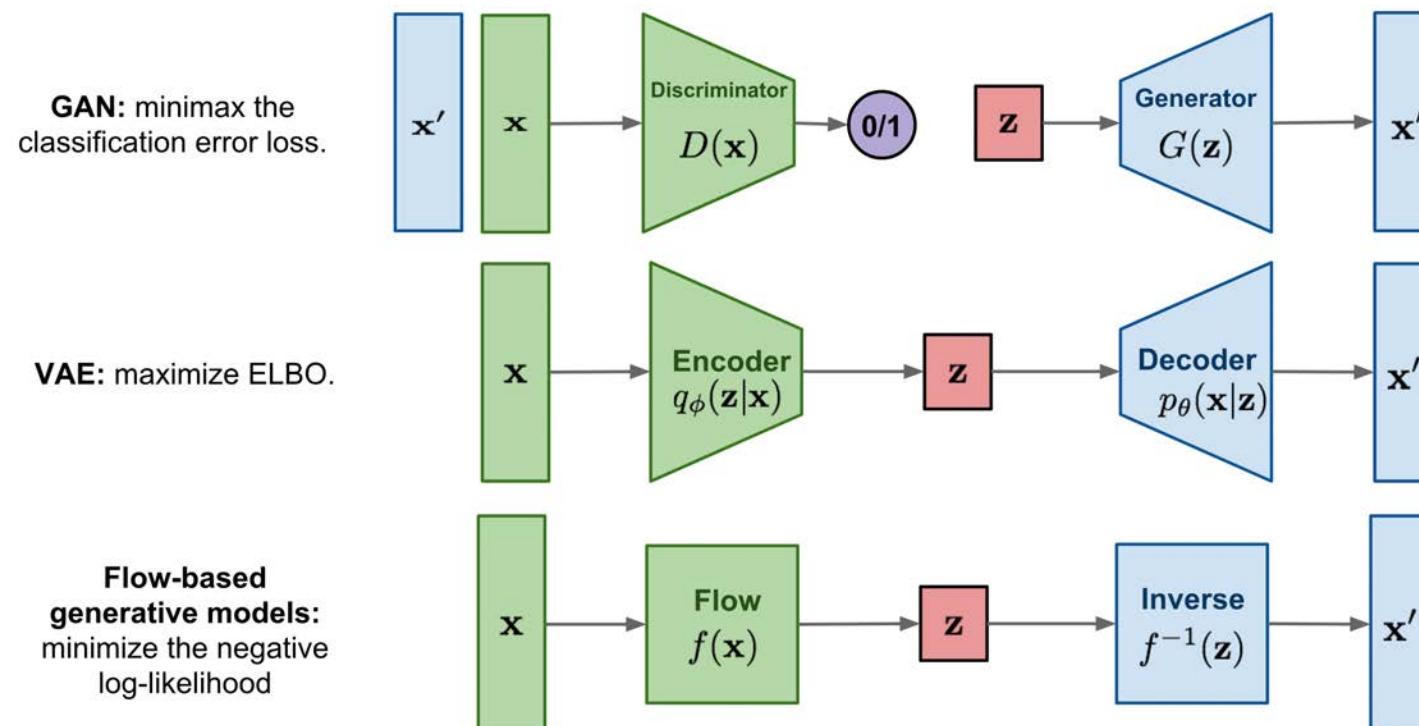
Deep Generative Models

- A deep learning model that can be used to generate X

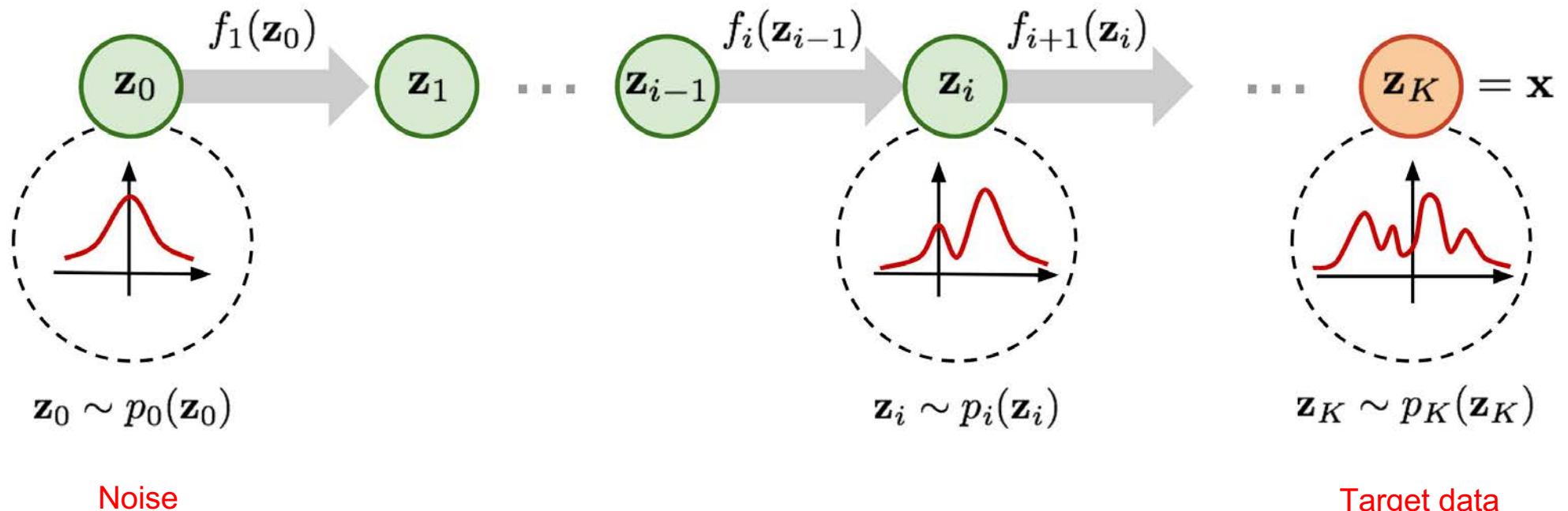


Flow-based model

- Tweak distribution until you get the target distribution
- Deterministic reversible/invertable tweaks

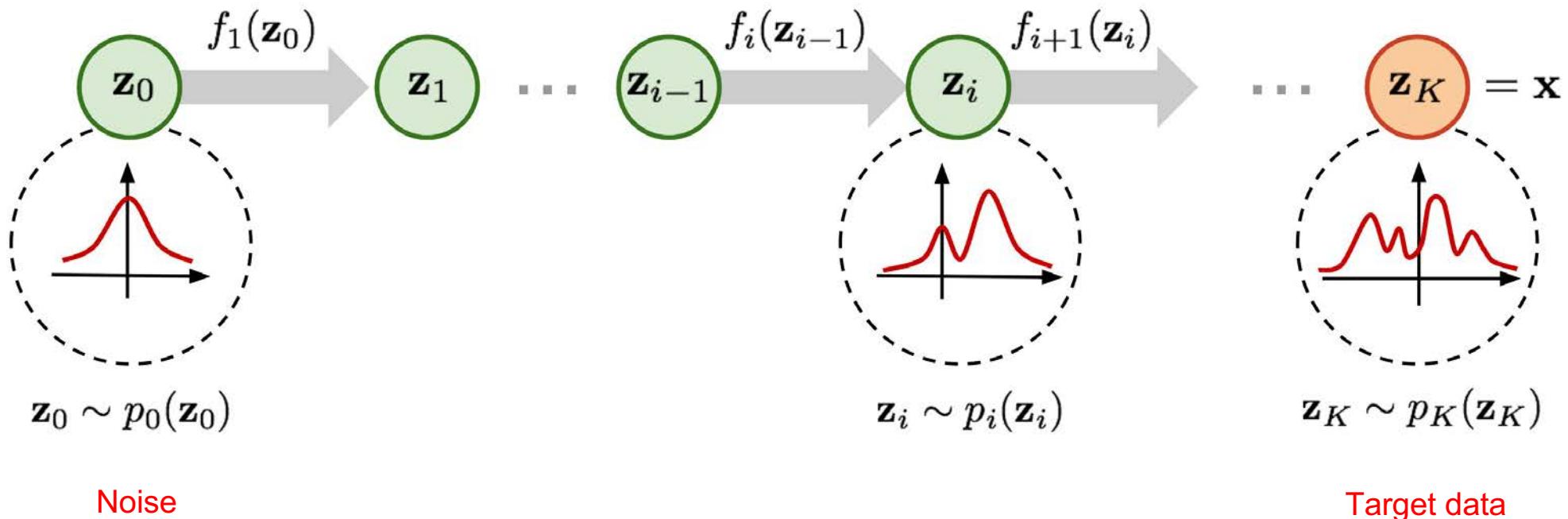


Flow-based model



$$\begin{aligned}
\mathbf{x} = \mathbf{z}_K &= f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0) \\
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \dots \\
&= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^K \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|
\end{aligned}$$

Flow-based model



Find f that maximizes the likelihood of the data

Constrain f so that it is easily invertable and det easy to compute

f is a neural network (needs networks that's easily invertable by design)

Example of invertible layer

- RealNVP

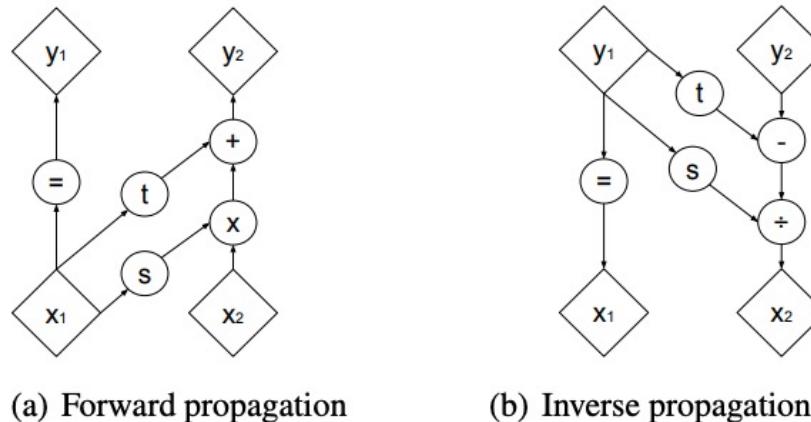
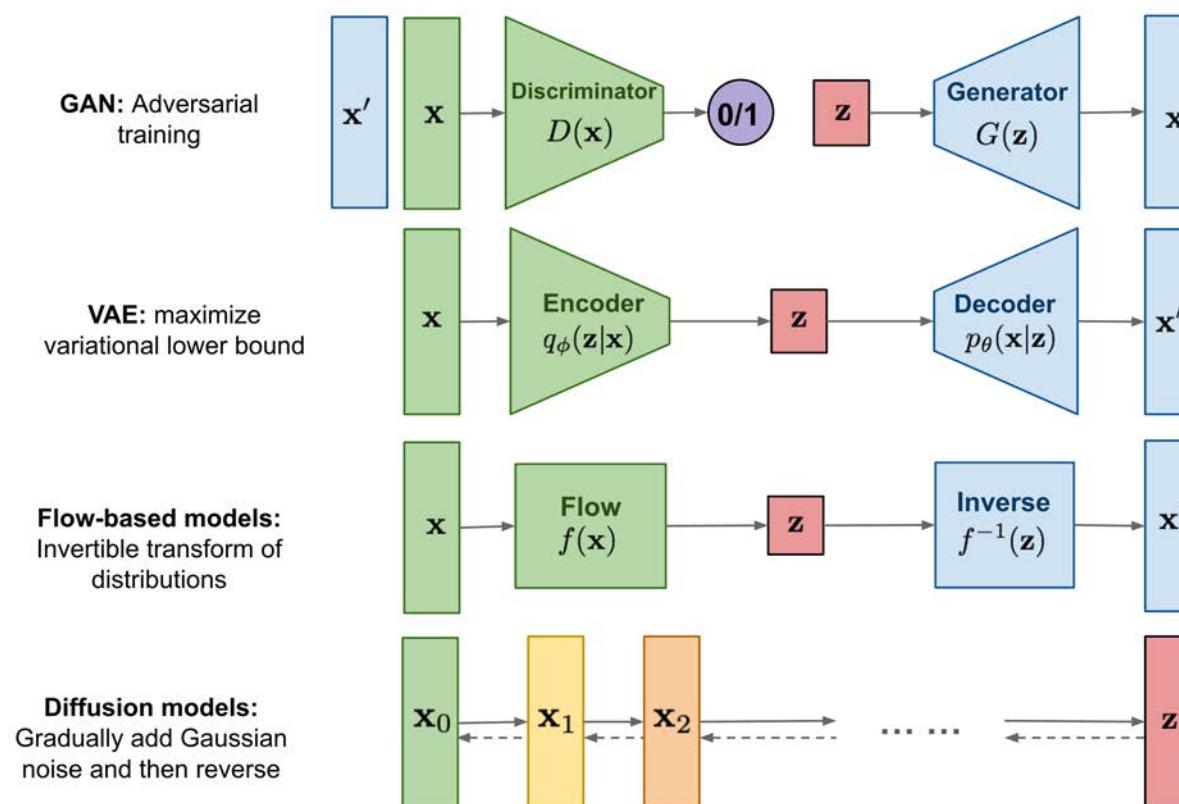


Figure 2: Computational graphs for forward and inverse propagation. A coupling layer applies a simple invertible transformation consisting of scaling followed by addition of a constant offset to one part x_2 of the input vector conditioned on the remaining part of the input vector x_1 . Because of its simple nature, this transformation is both easily invertible and possesses a tractable determinant. However, the conditional nature of this transformation, captured by the functions s and t , significantly increase the flexibility of this otherwise weak function. The forward and inverse propagation operations have identical computational cost.

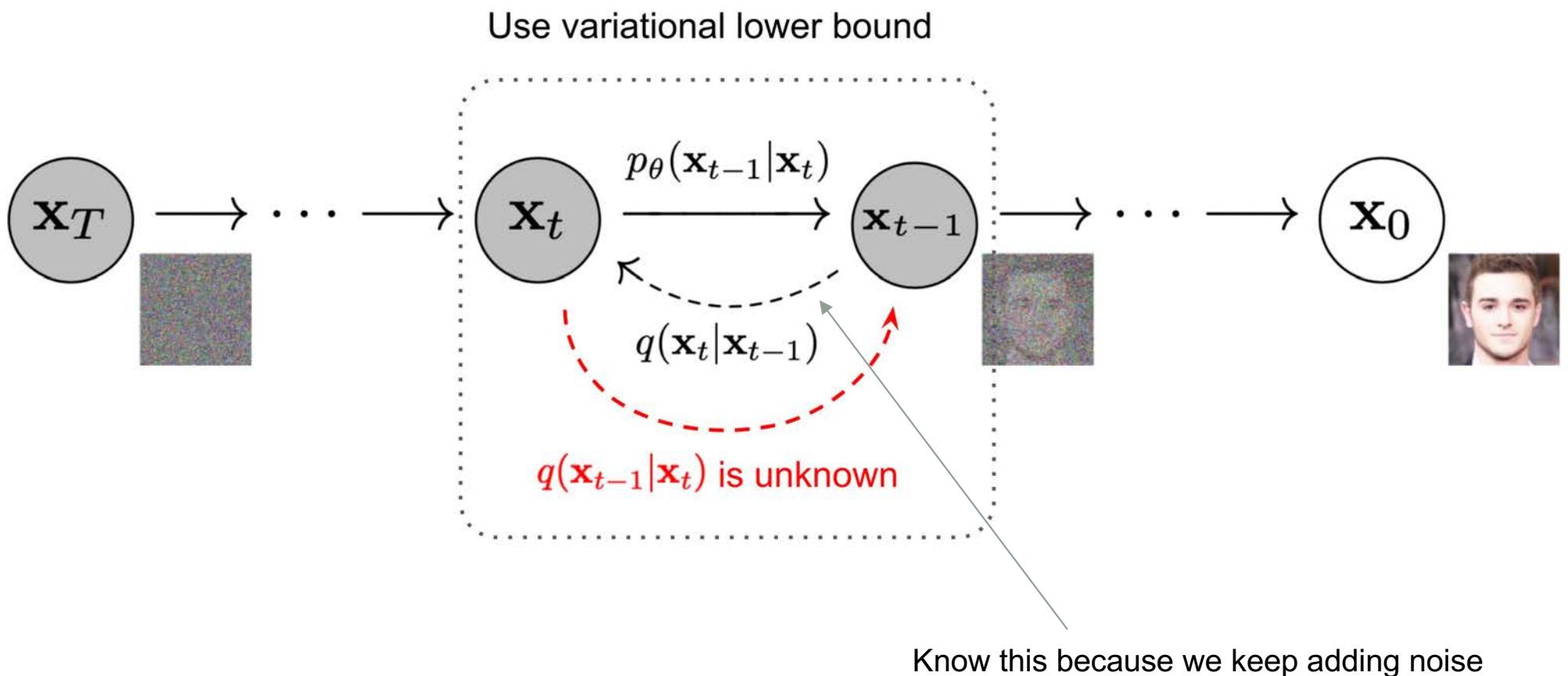
Diffusion model

- Similar to flow
 - No longer constrained to revertible operations
 - Add noise to image until you get just noise



Diffusion model

Diffusion model tries to learn how to remove the noise in each step



Notes

- Diffusion model is now state-of-the-art in many generation tasks
- Very slow (need hundreds iterative steps of adding noise)
- The latent of diffusion and flow models are of the same size of the original data (does not compress).
- Diffusion + flow = Diffusion normalization flow

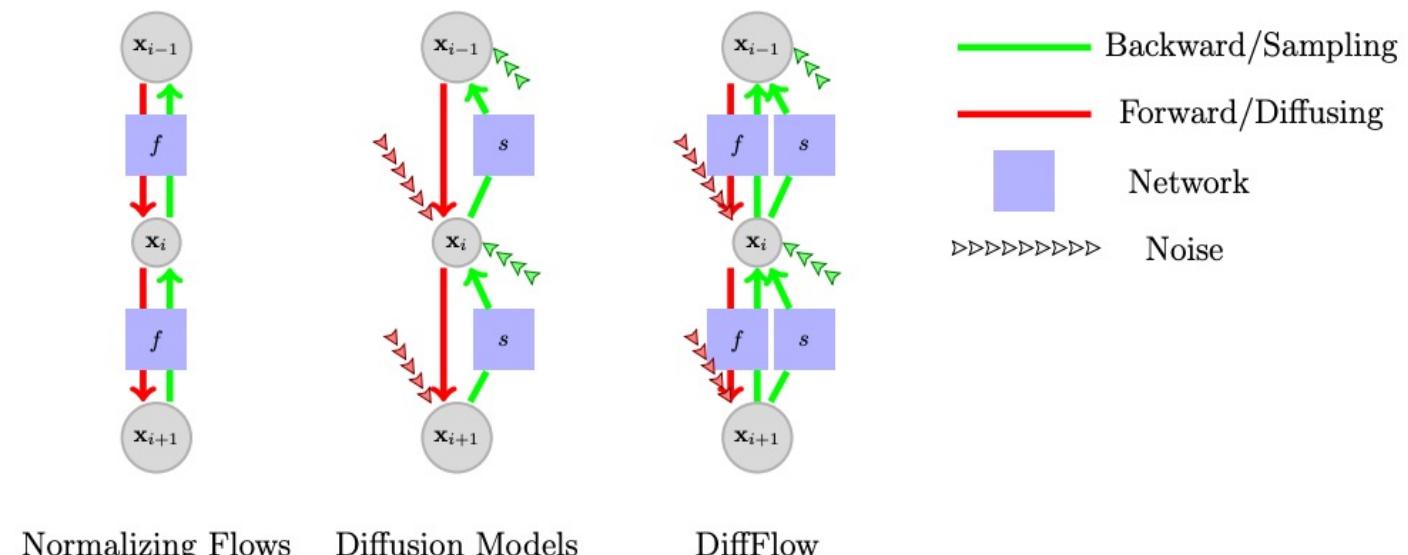
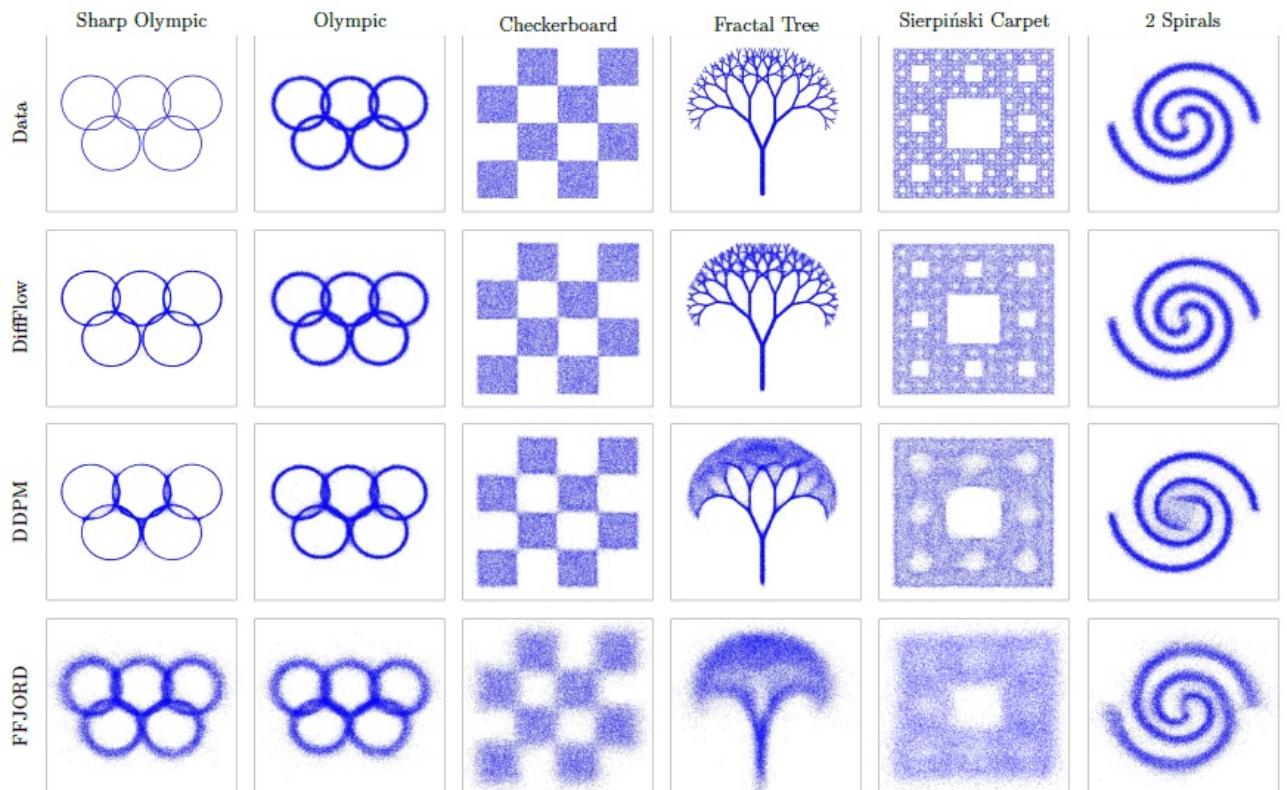
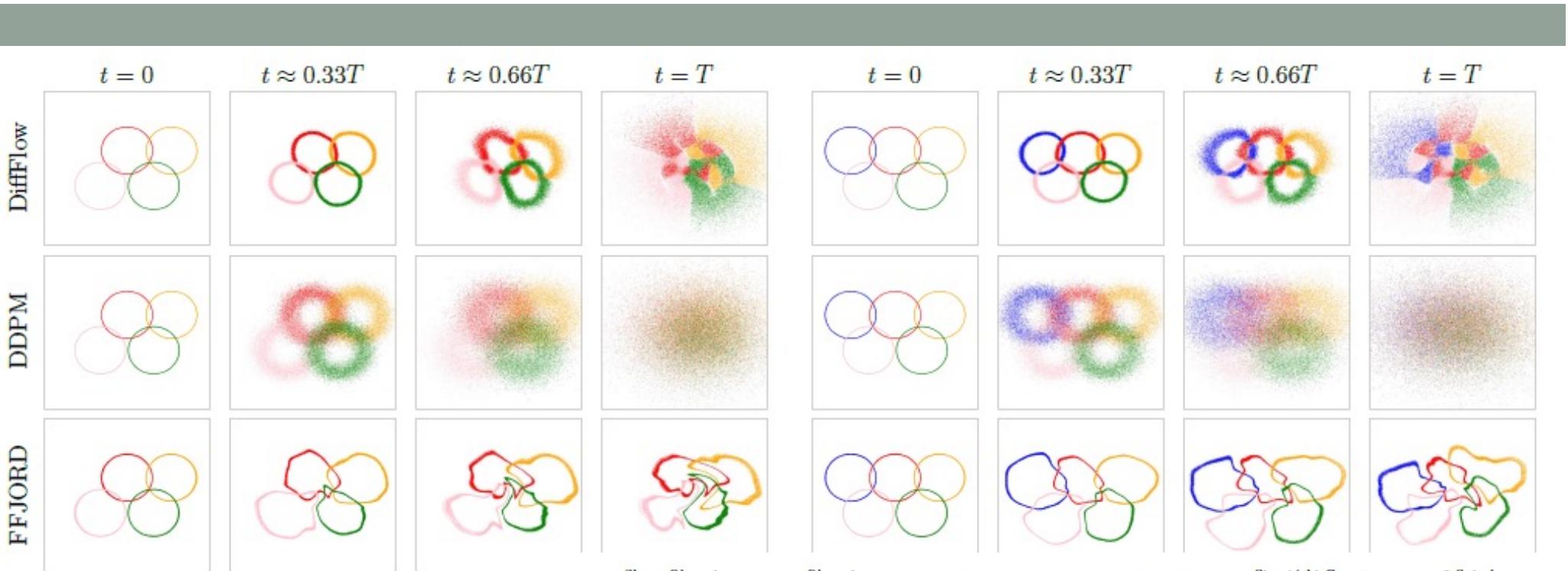


Figure 1: The schematic diagram for normalizing flows, diffusion models, and DiffFlow. In normalizing flow, both the forward and the backward processes are deterministic. They are the inverse of each other and thus collapse into a single process. The diffusion model has a fixed forward process and trainable backward process, both are stochastic. In DiffFlow, both the forward and the backward processes are trainable and stochastic.

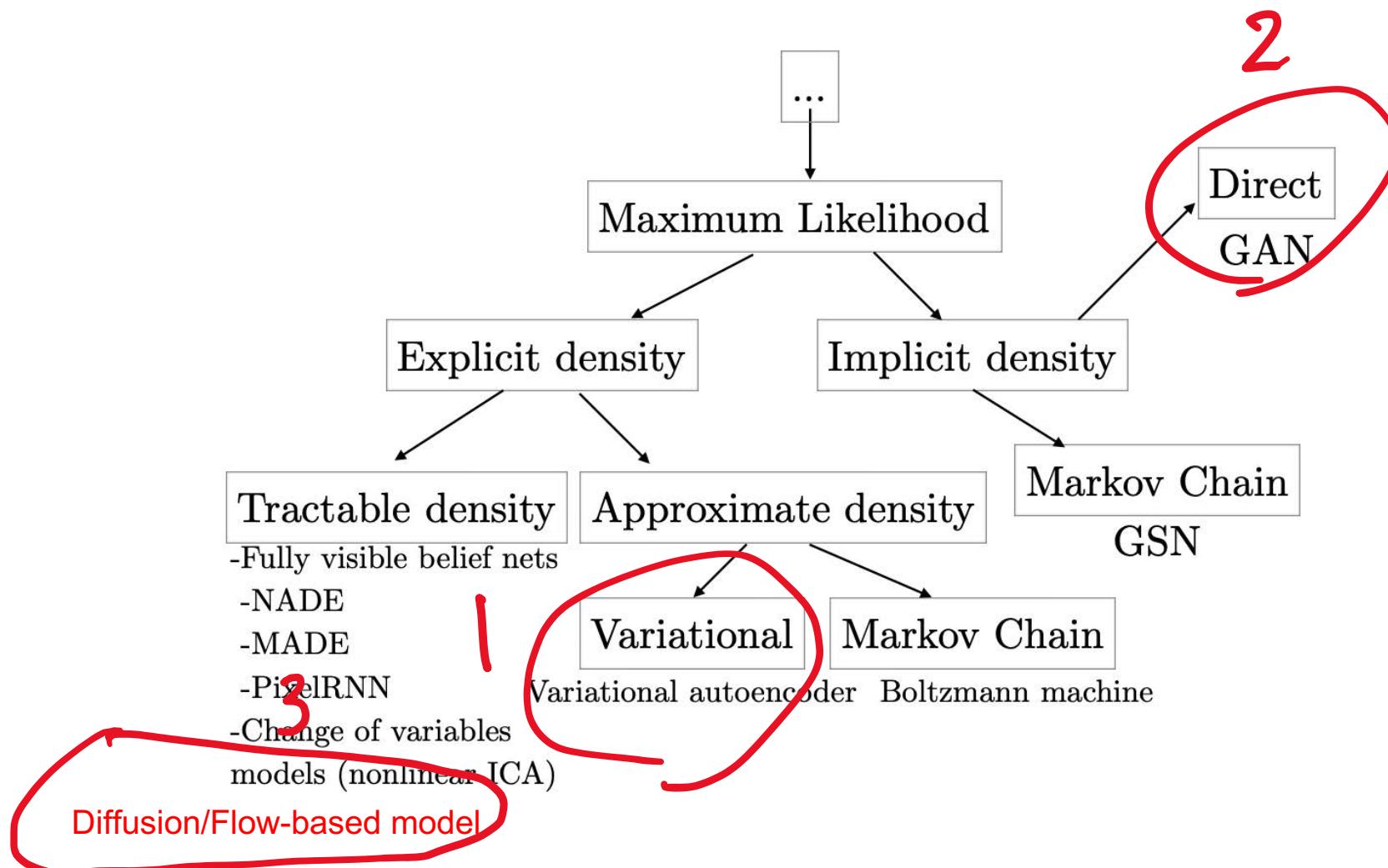


Recommend reading

https://ai-scholar.tech/en/articles/diffusion-model/diffusion_normalizing_flow

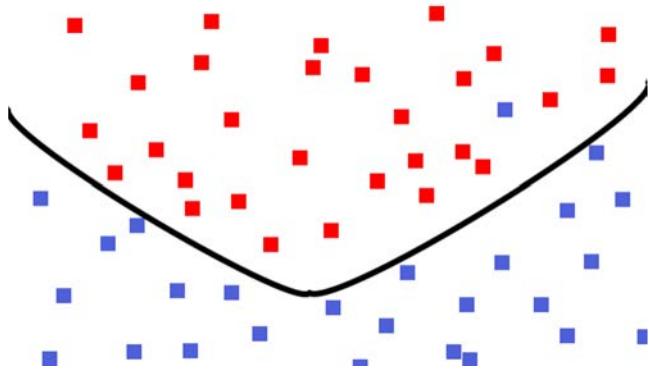
Deep Generative Models

- A deep learning model that can be used to generate X



UNSUPERVISED LEARNING

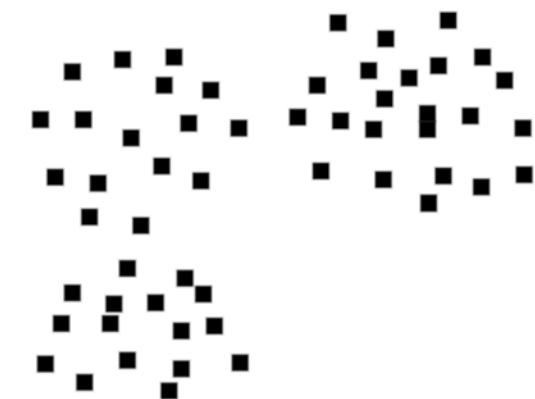
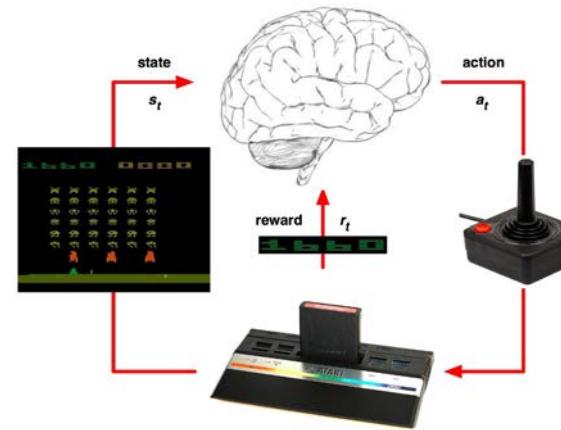
3 Modes of Learning



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?

First Unsupervised Learning Idea

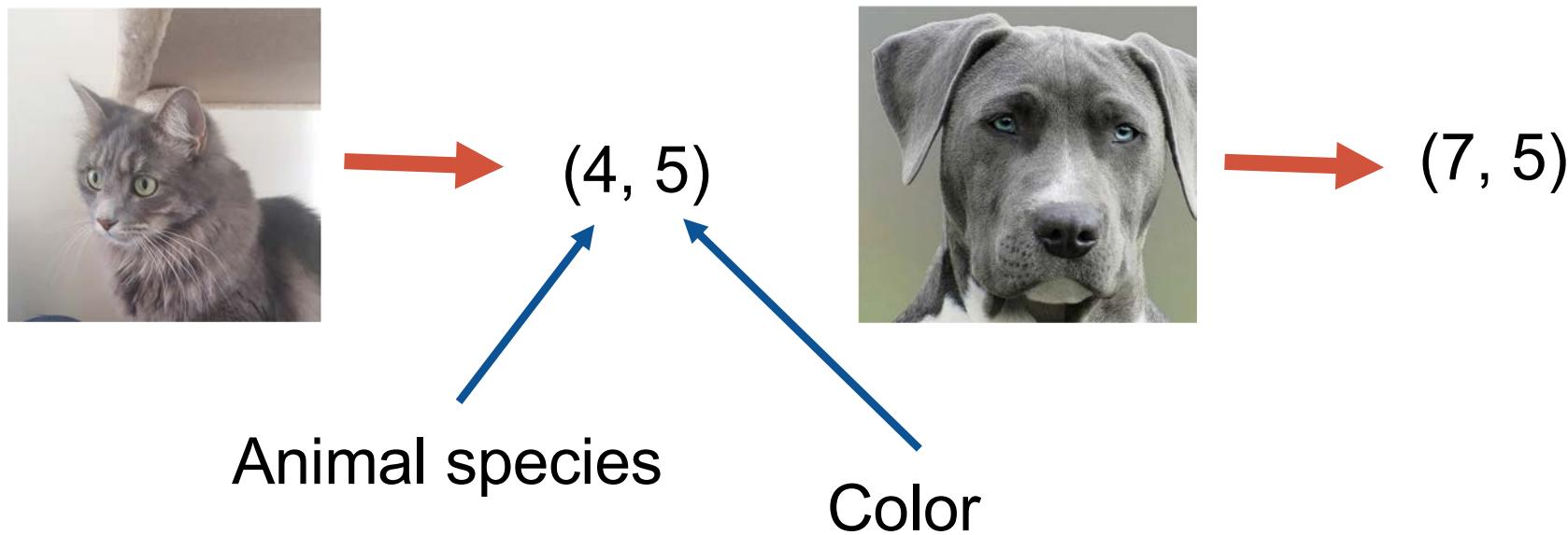
- Suppose we have a magic function f that maps an image to two numbers



$$\xrightarrow{f} (4.1, 5.0)$$

First Unsupervised Learning Idea

- Ideal magic function f

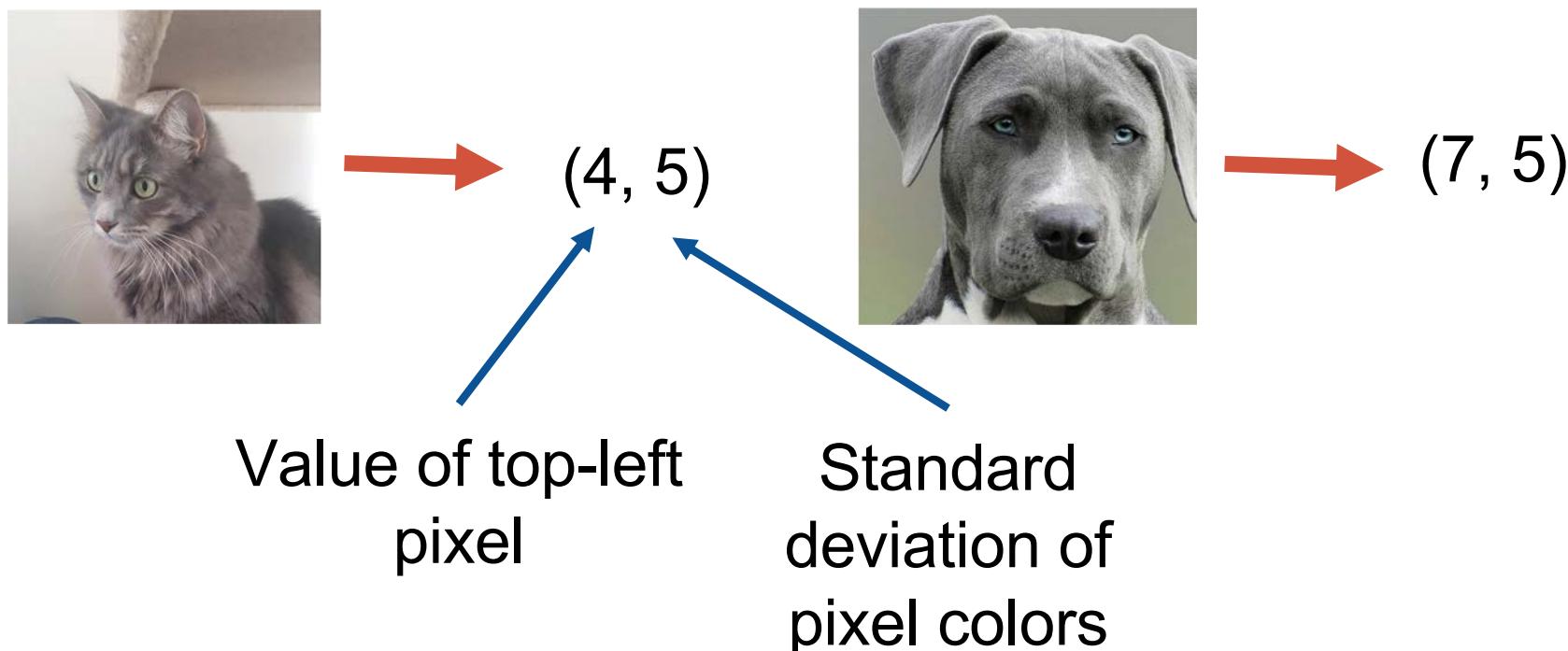


Getting this f would be so nice!

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

First Unsupervised Learning Idea

An alternative magic function f



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]

$f(\cdot)$ through deep learning

- Last time we talk about how VAE can learn meaningful codes
- A lot of the techniques in this lecture employ clever tricks to learn the model

Flavors of supervision

Supervised

 All labels known

Unsupervised

 No labels

Semi-supervised

 Some labels known

Representation learning

 Transfer learning

Self-supervised learning

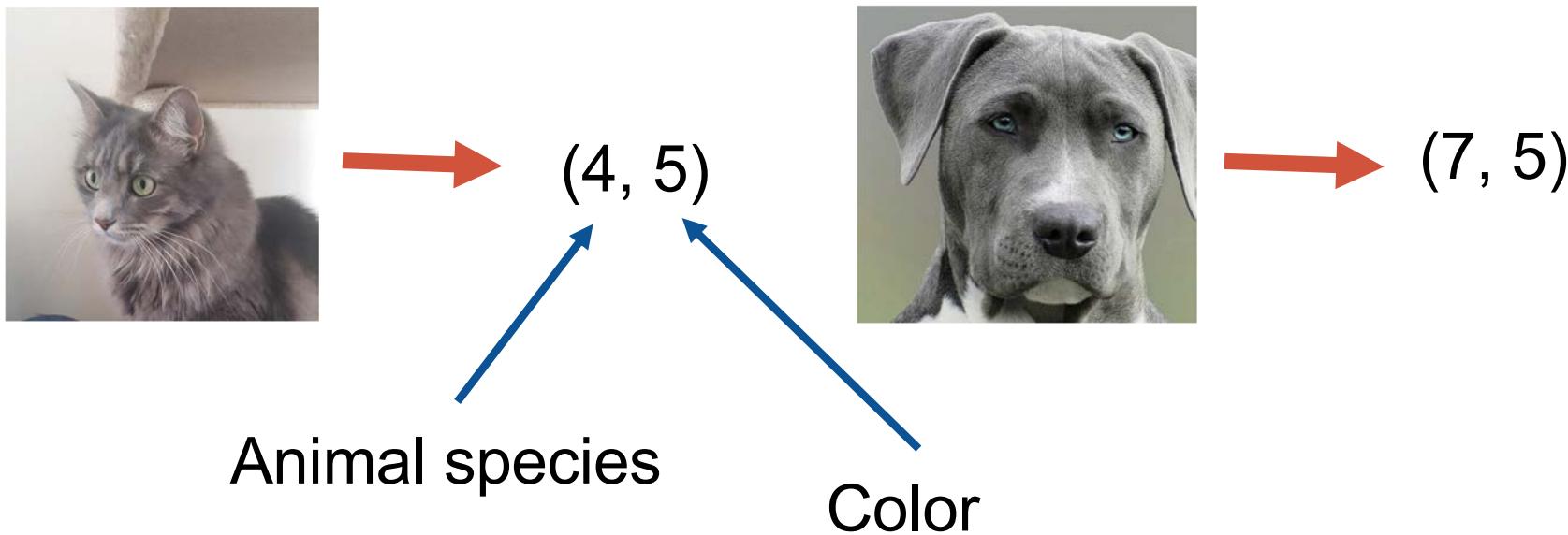
 Surrogate task from pseudo labels

Reinforcement learning

 Agent gathers data, reward function known

Steps

- Learning f (somehow)



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

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

Idea 0: learning better representations

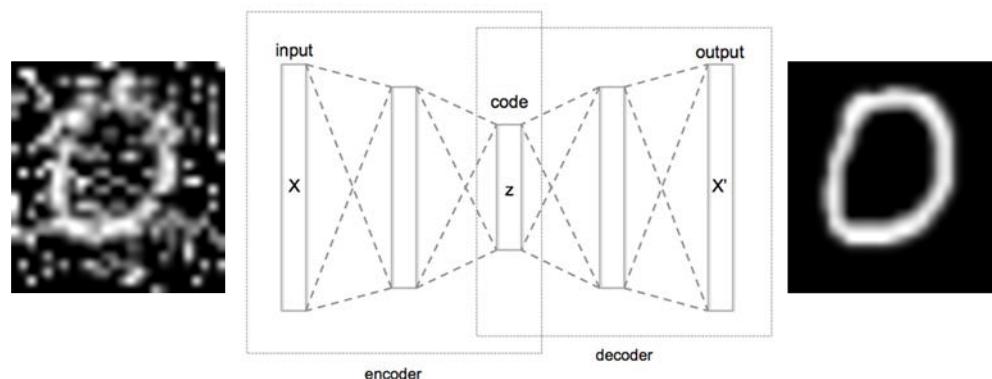
Use denoising autoencoders to learn representations on unlabelled data

- Use latent codes as additional features

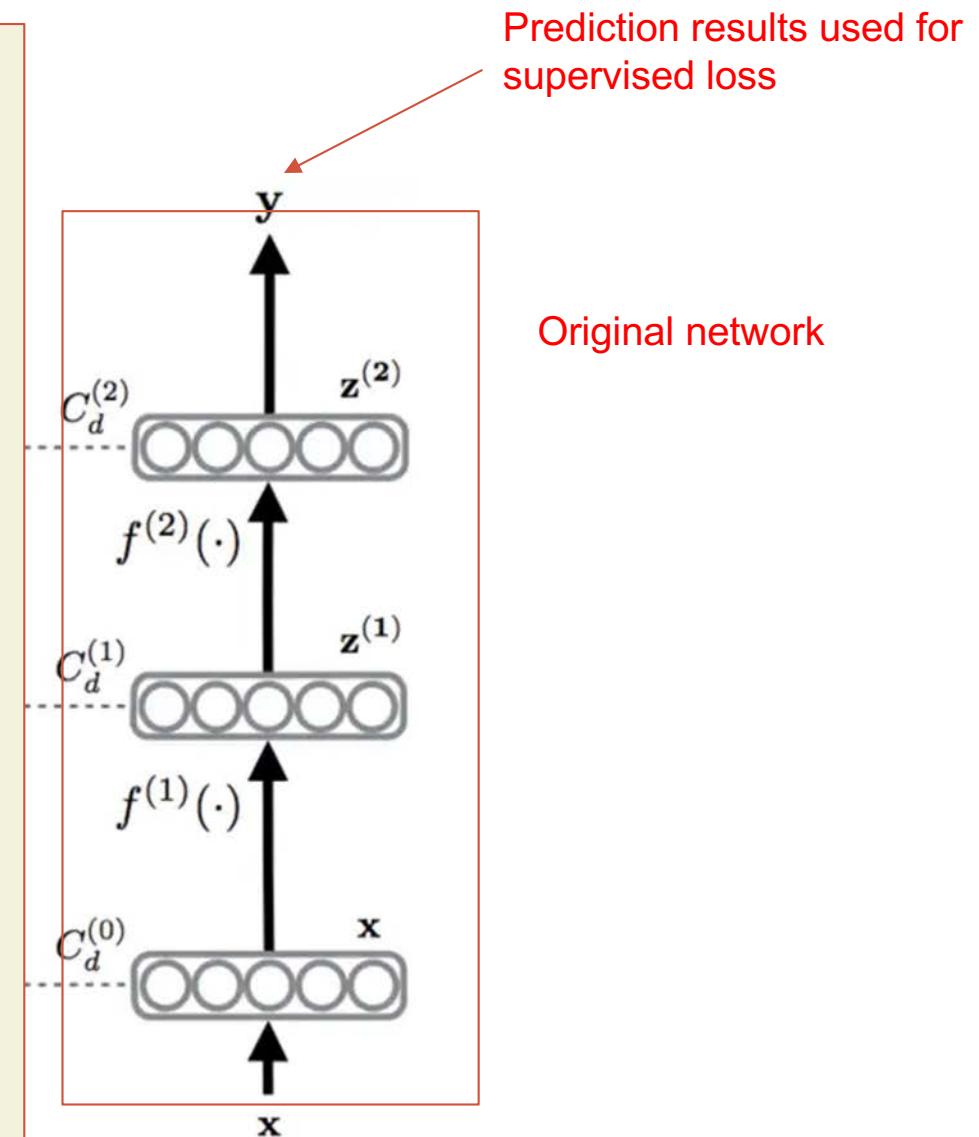
- Or bake in denoising autoencoders into the model
(LADDER networks)

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 $x \rightarrow x''$
 - Use x'' as the input, and minimize the same loss
 - Loss = $\|x-x'\|^2$

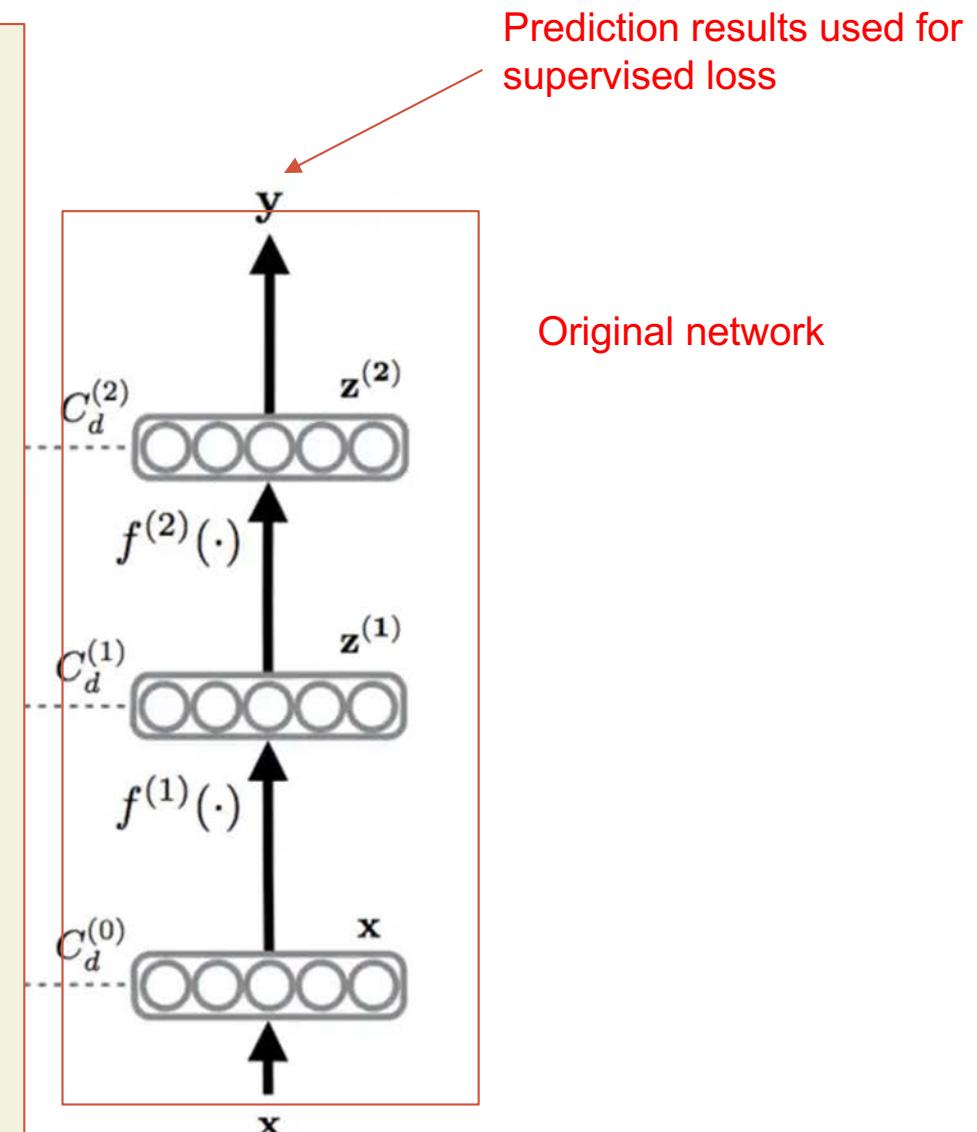
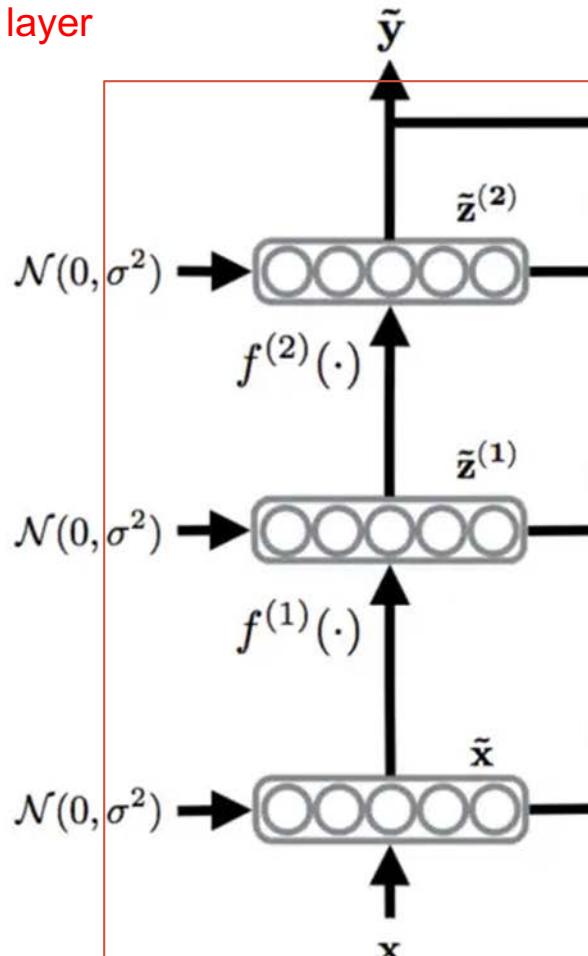


LADDER networks

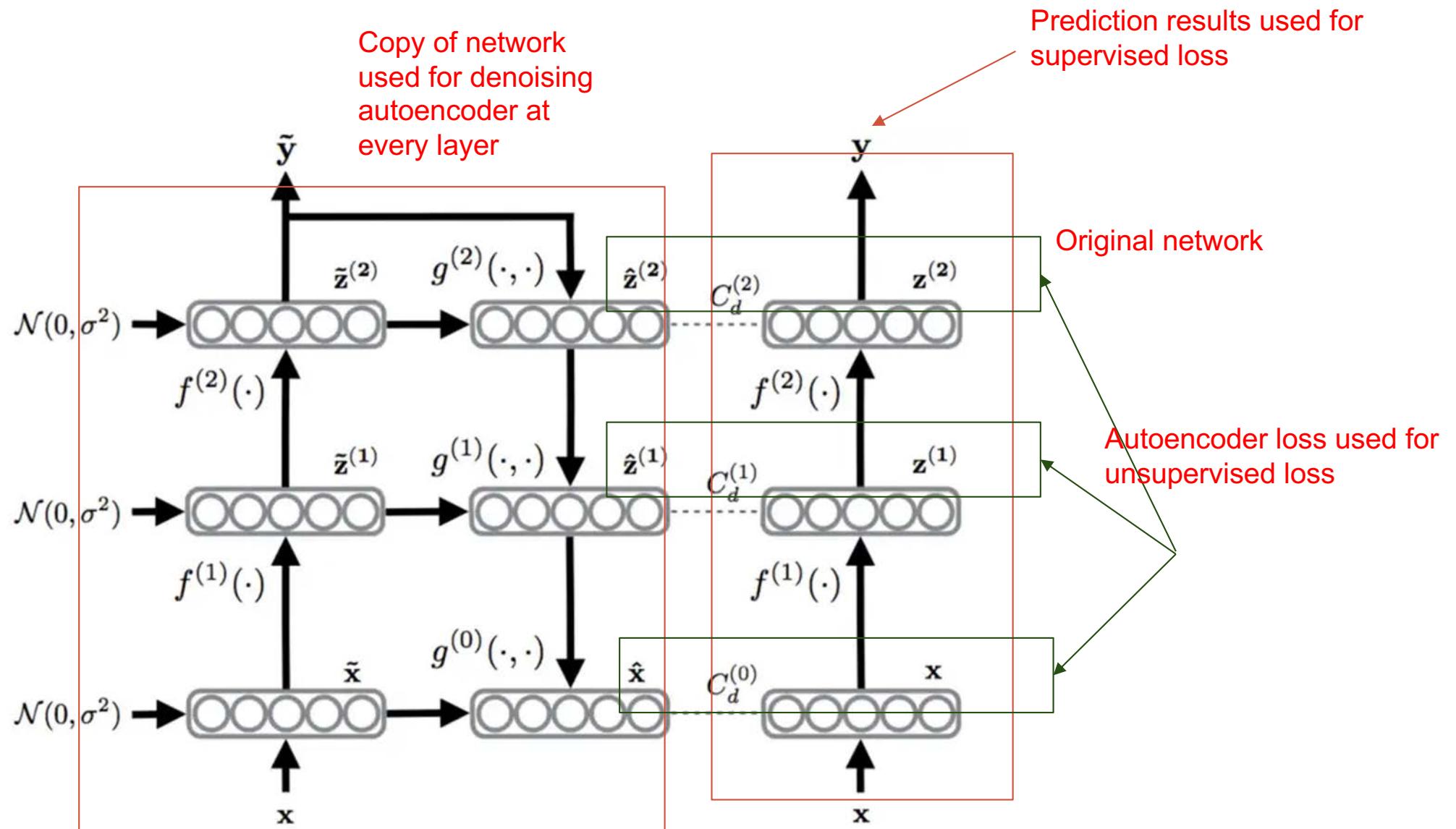


LADDER networks

Copy of network
but add noise to every
intermediate output of each
layer



LADDER networks



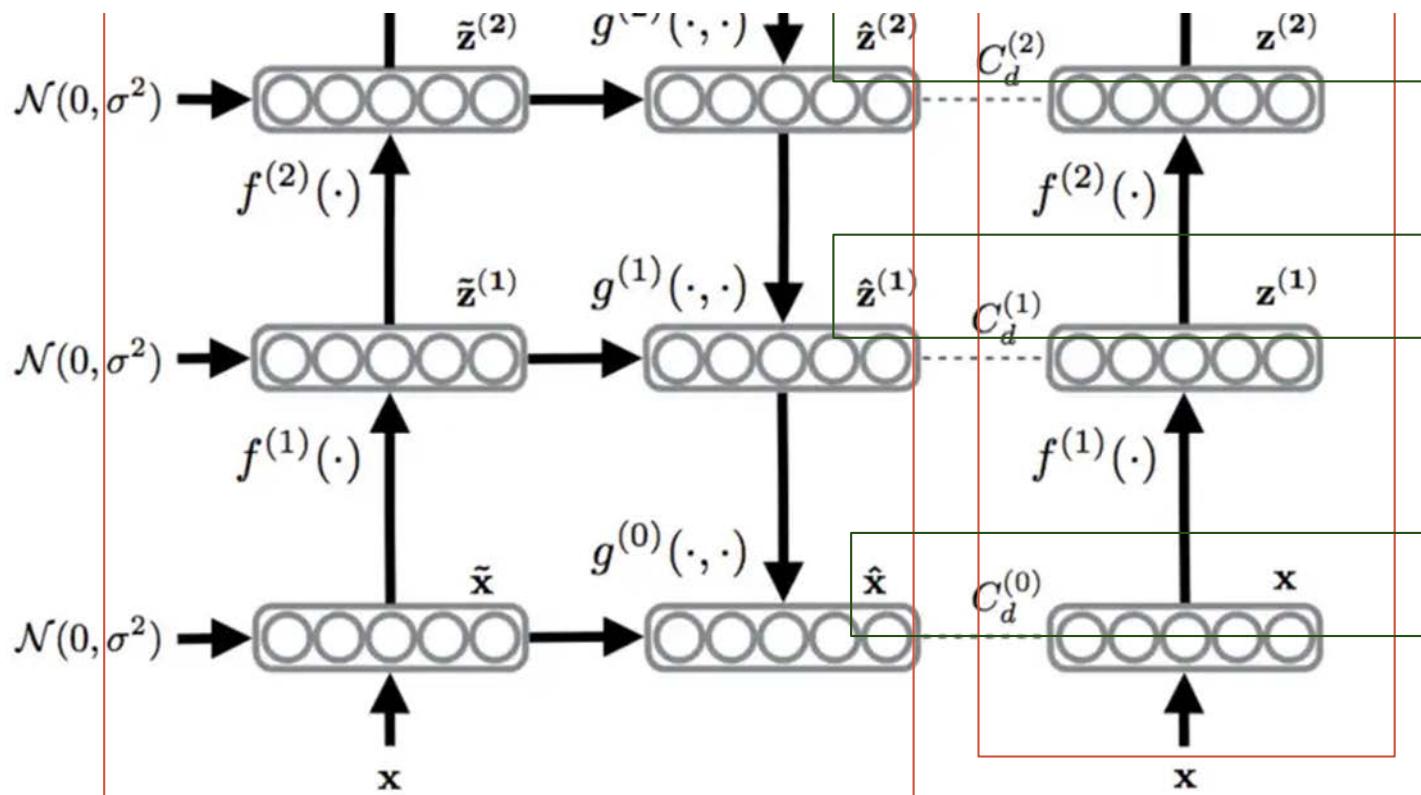
LADDER networks

Overall loss is the sum of both losses.

Training via SGD.

Have labeled and unlabeled data in the mini-batch.

If the data is unlabelled, only calculate the unsupervised loss.



Idea 1: Bootstrapping with self-training

Same task, different domain or same task, unlabelled data

- Use a trained classifier on new domain.

- Filter data with high scores.

- Train/adapt with filtered data.

Caveats:

- Scores cannot be trusted (Ranking can be trusted)

- Learning from same errorful data

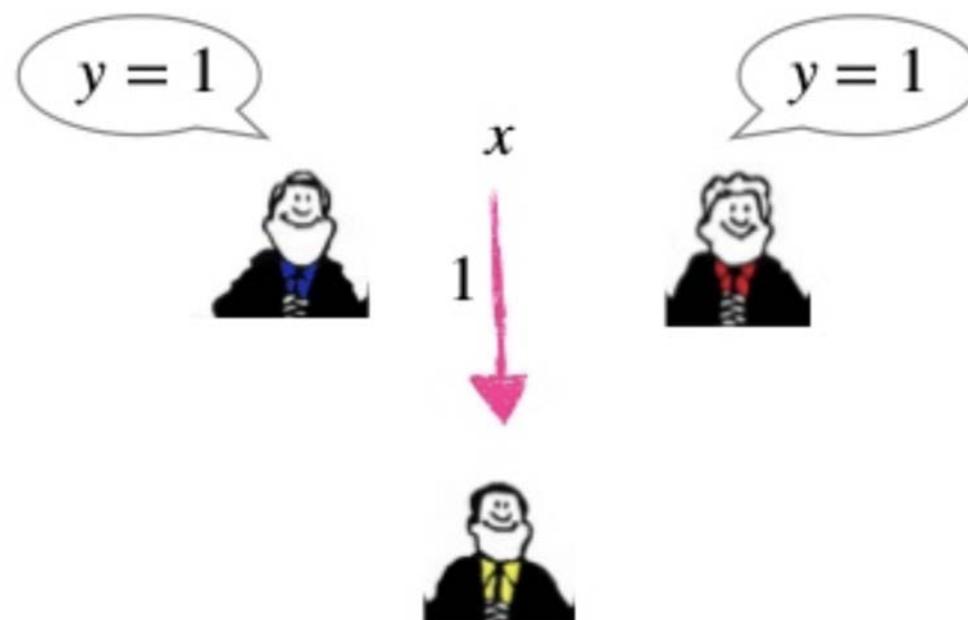
- If different domain, classifier performance is bad

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

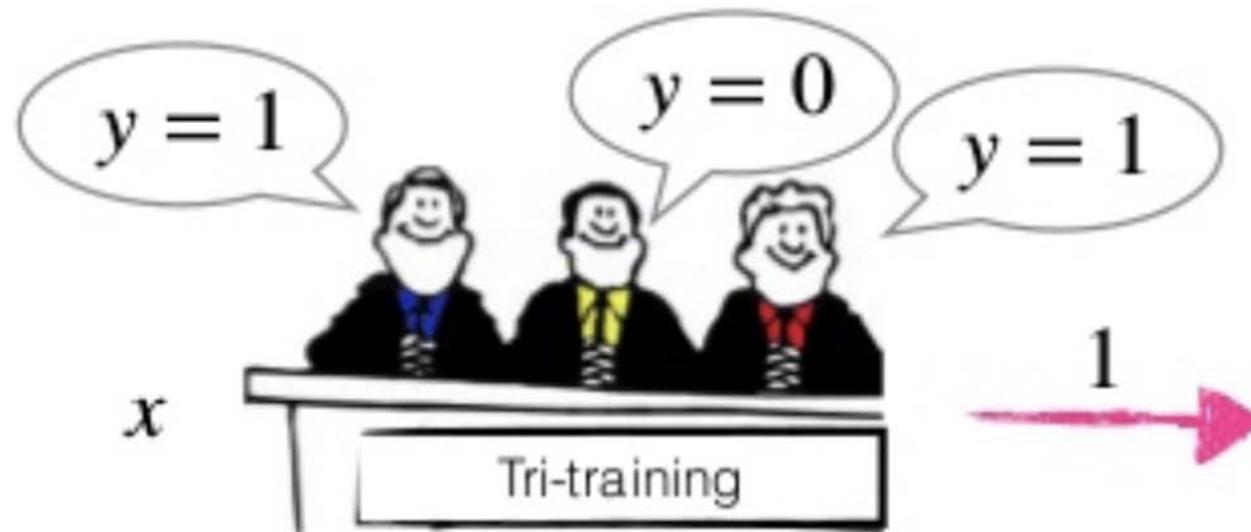


Idea 2: Tri-training with disagreement

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

Not so useful

Only use data when two agree and the third disagree



Idea 3: consistency training

Tri-training requires maintaining three models

Training each of them means 3x space and runtime

How do we create three different models?

Different data - **tri-training**

Different features (different viewpoint) -
multi/cross-view

Share some parts of the model to reduce computation and
space and faster convergence

Idea 3: consistency 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 _____.

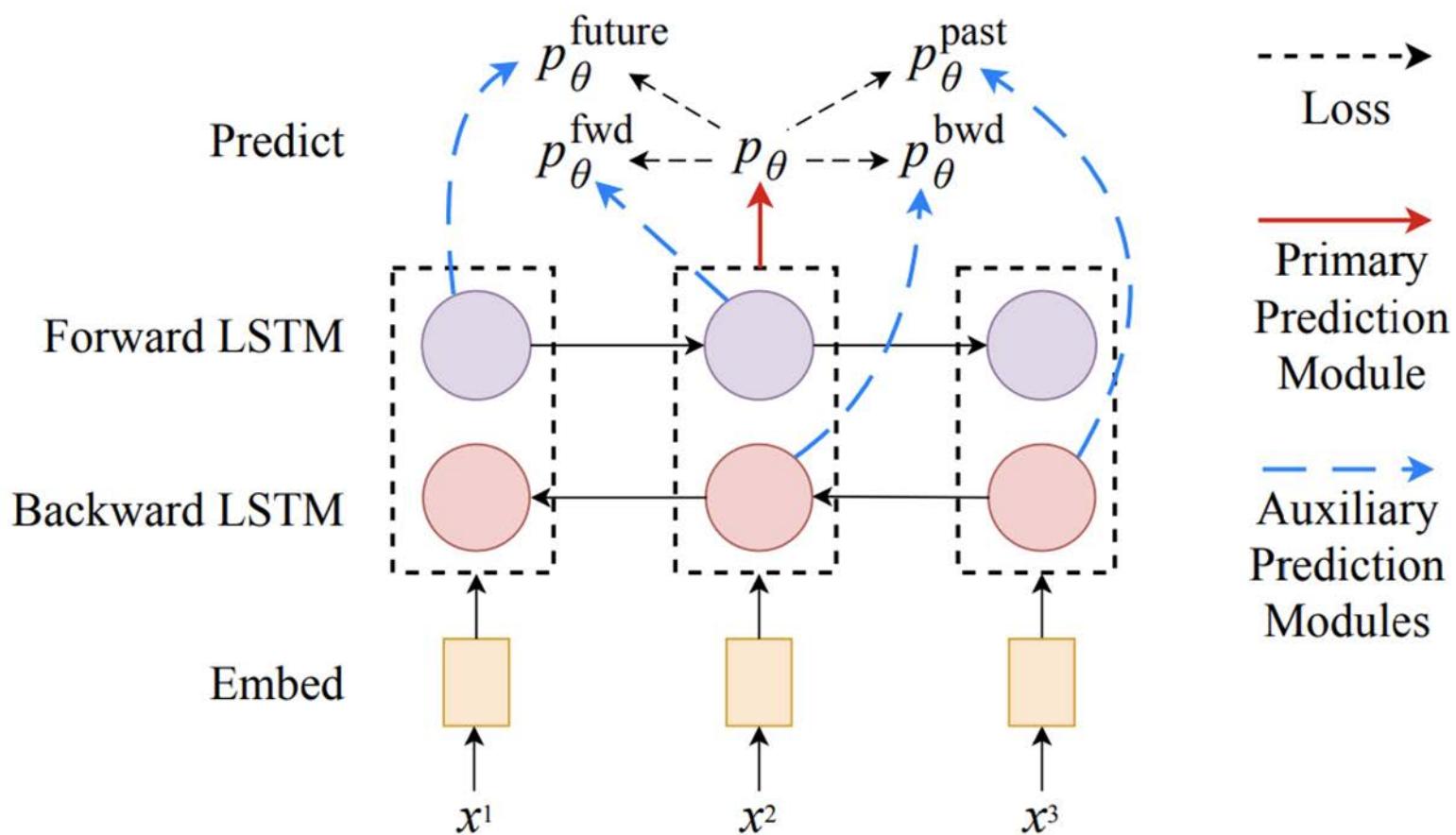
Student2: They really **like** _____.

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

Student3: _____ **like** the idea.

Student4: _____ the idea.

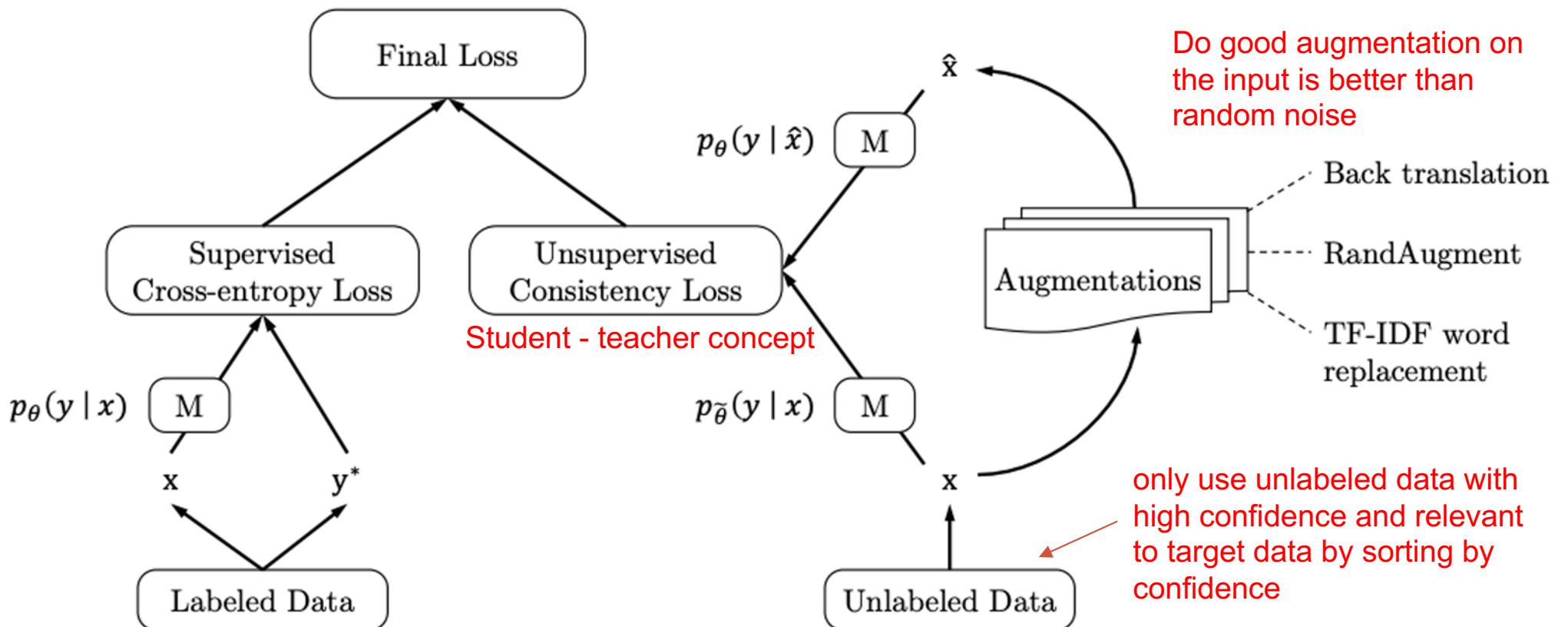
Future: They really _____.
Forward: They really **like** _____.
Primary: They really **like** the idea.
Backward: _____ **like** the idea.
Past: _____ the idea.



Method	CCG Acc.	Chunk F1	NER F1	FGN F1	POS Acc.	Dep. Parse UAS	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



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

Overview

(picture from a different paper)

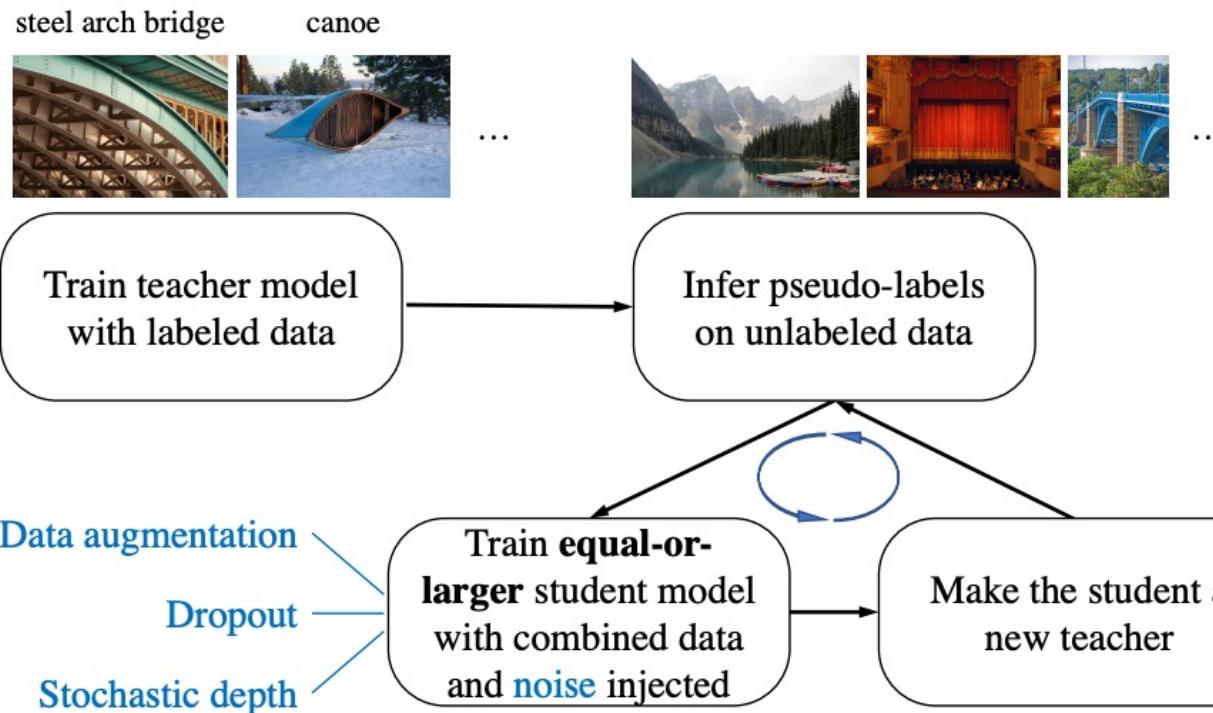
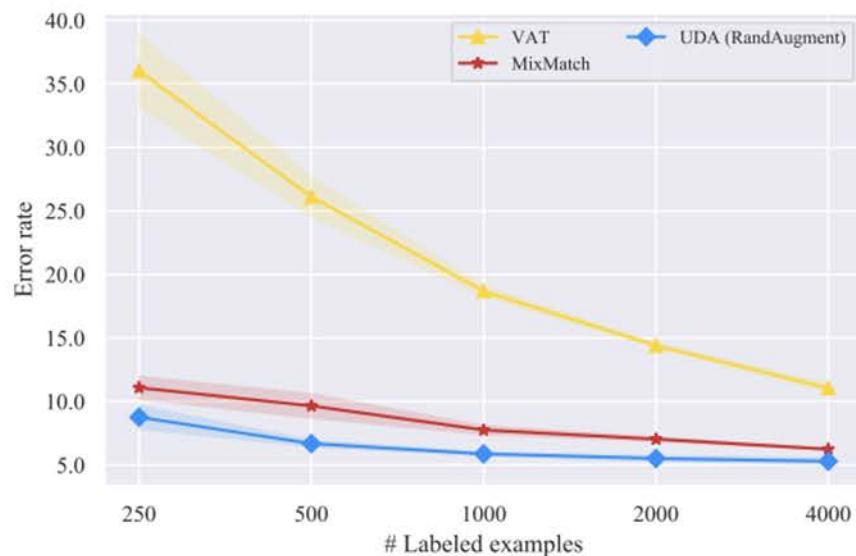
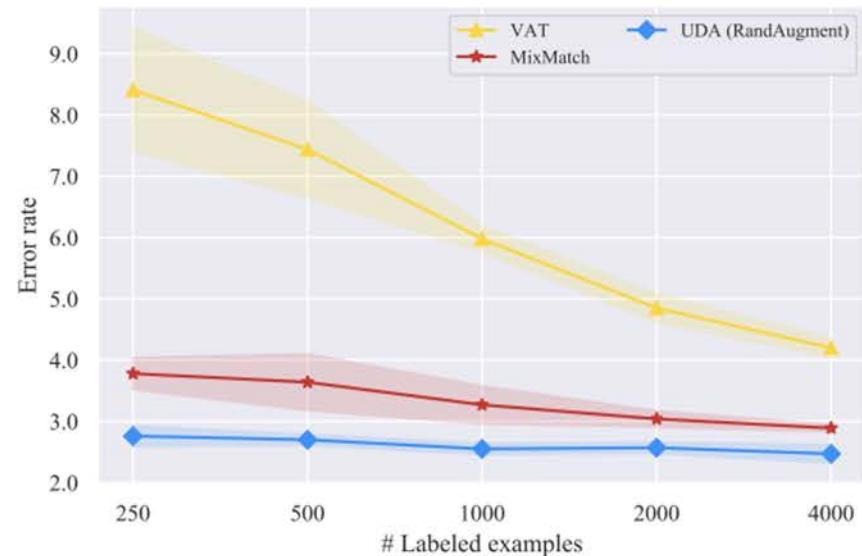


Figure 1: Illustration of the Noisy Student Training. (All shown images are from ImageNet.)

Results



(a) CIFAR-10



(b) SVHN

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 / PI-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	

Semi-supervised learning notes

Both tries to predict with constraints (with noise or with missing information)

Consistency training can be used in unsupervised manner, but still need some label to do useful classification

Adding noise to text is harder to do than in image.

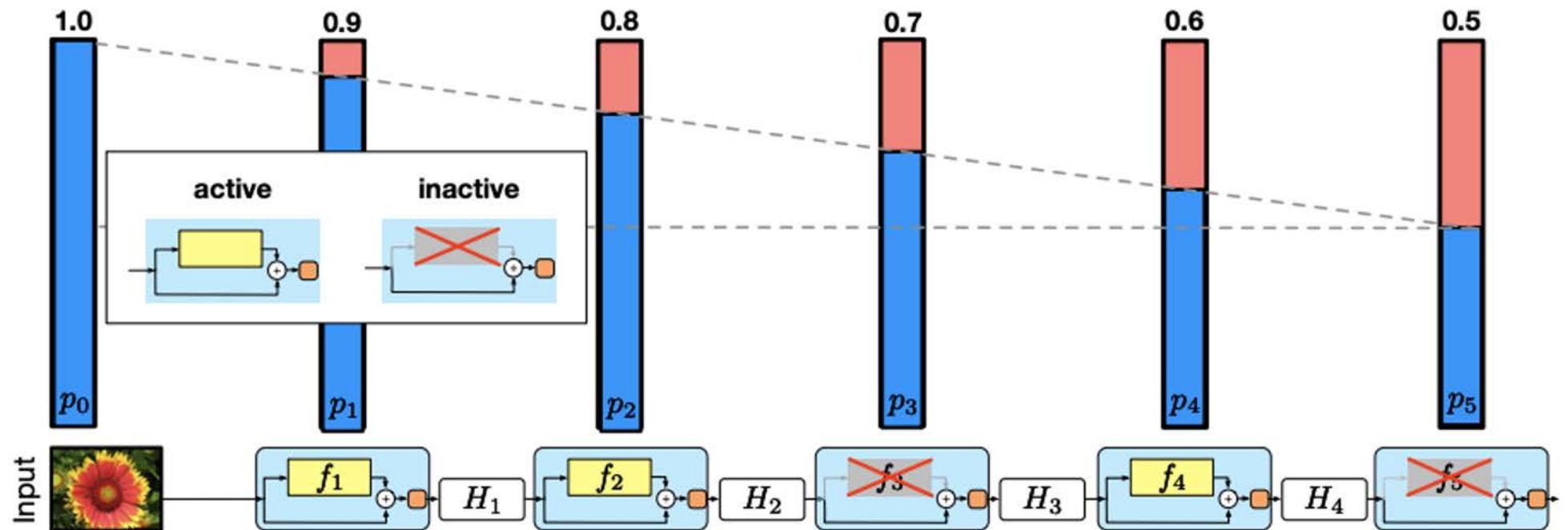
Drawbacks of previous approaches

- 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.

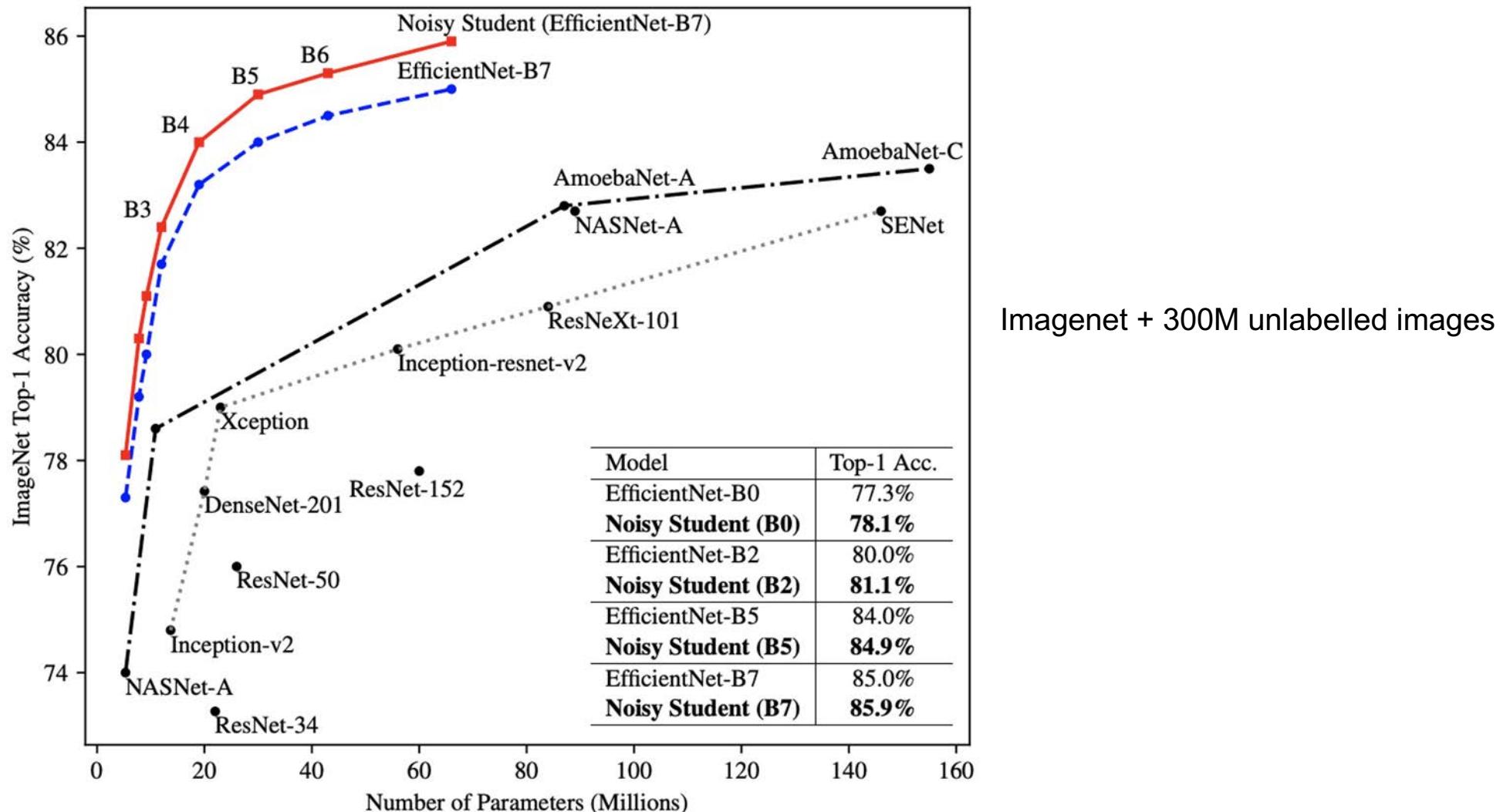
Augmenting (noising) more

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



Results

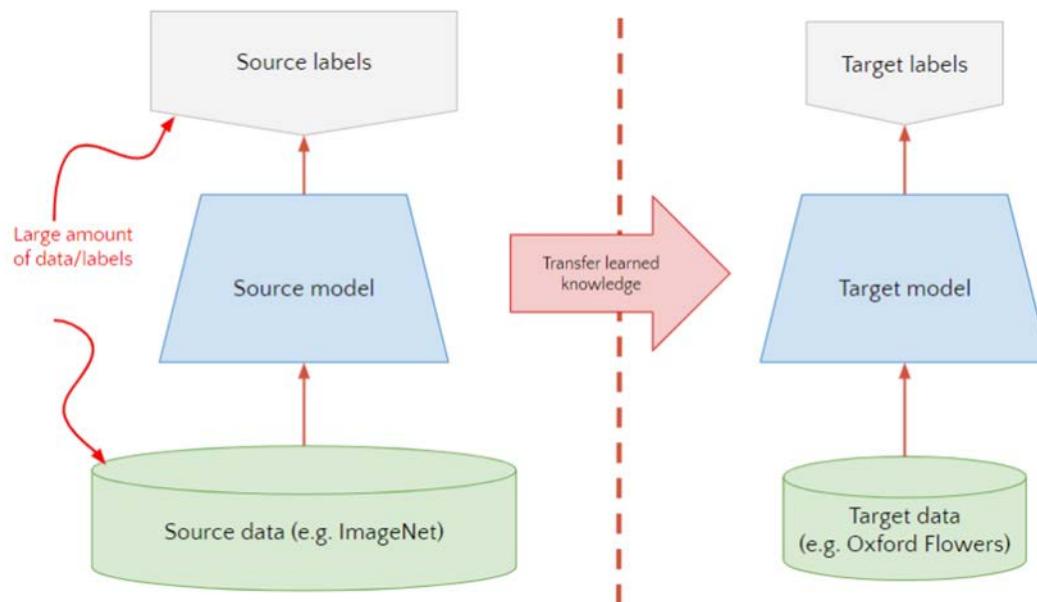
	ImageNet top-1 acc.	ImageNet-A top-1 acc.	ImageNet-C mCE	ImageNet-P mFR
Prev. SOTA	86.4%	16.6%	45.7	27.8
Ours	87.4%	74.2%	31.2	16.1



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

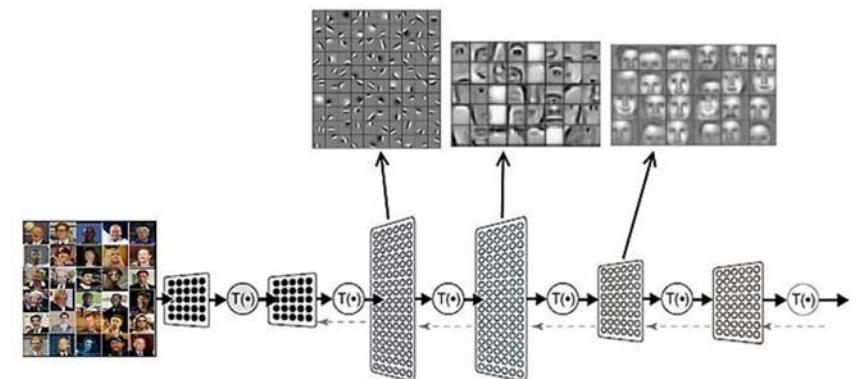
Instead of training a deep network from scratch for your task, you can

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

This lecture will talk about how to do this.

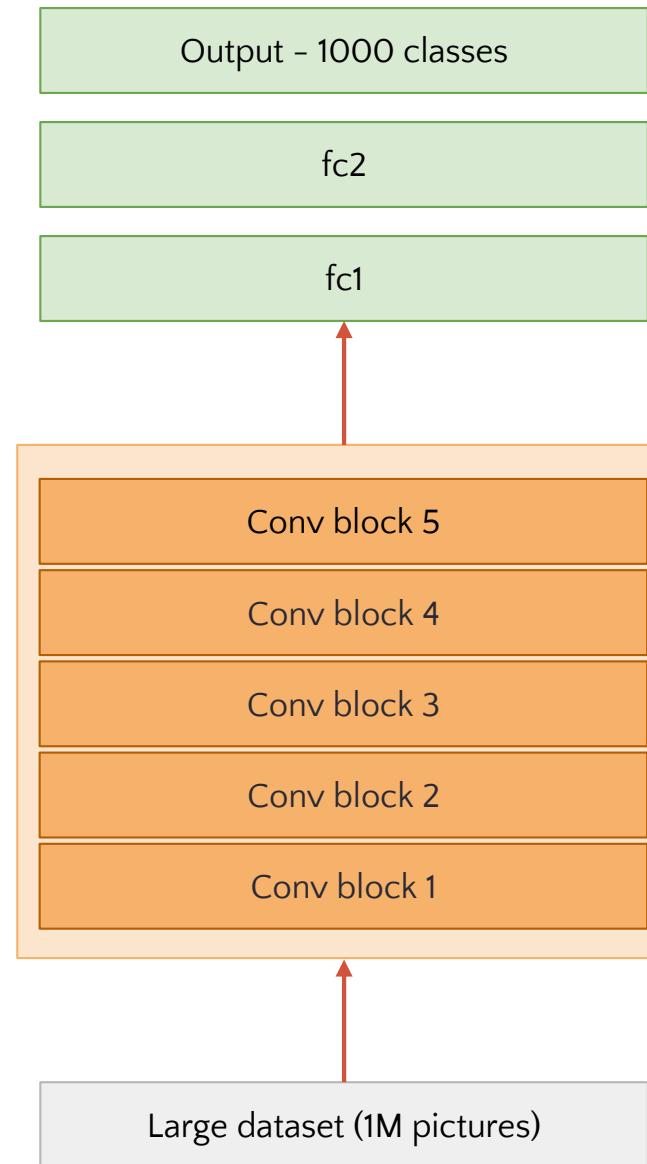
Variations:

- Different domain, same task
- Different domain, different task

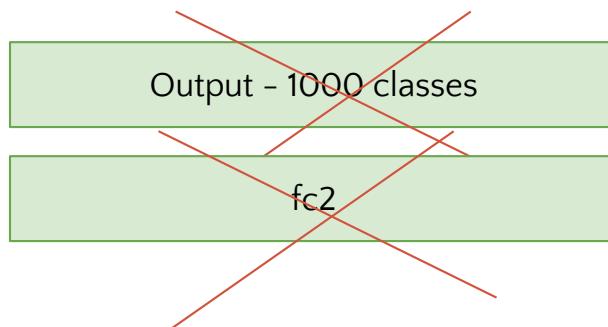


Transfer learning

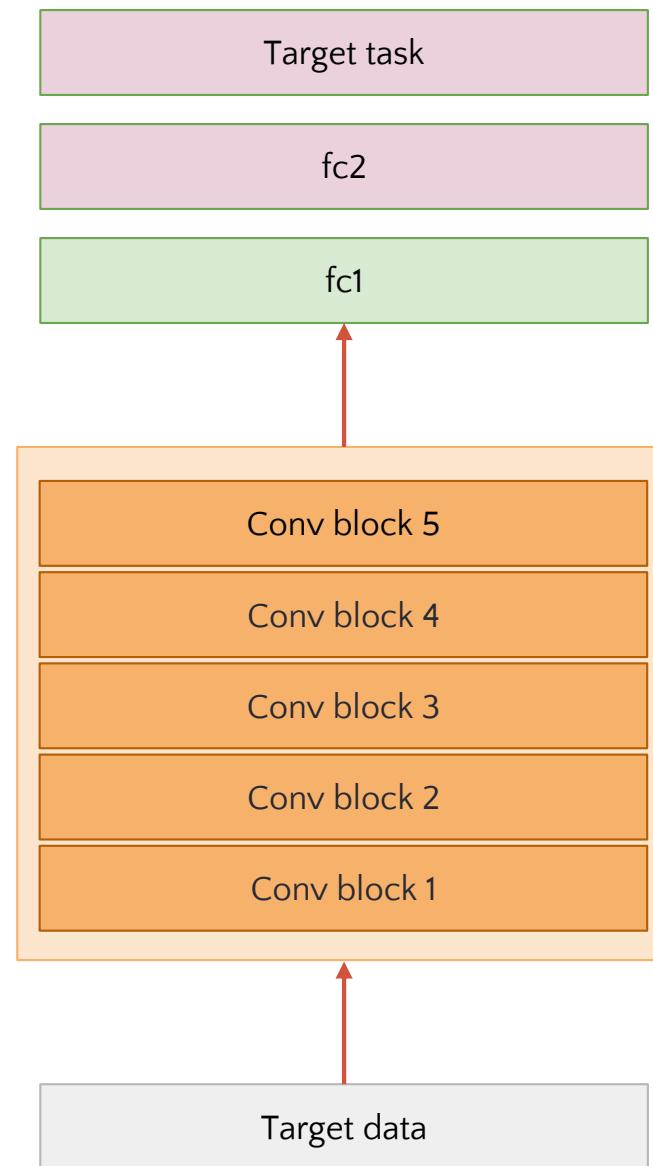
1. A model is trained on large dataset
Ex ImageNet



Transfer learning



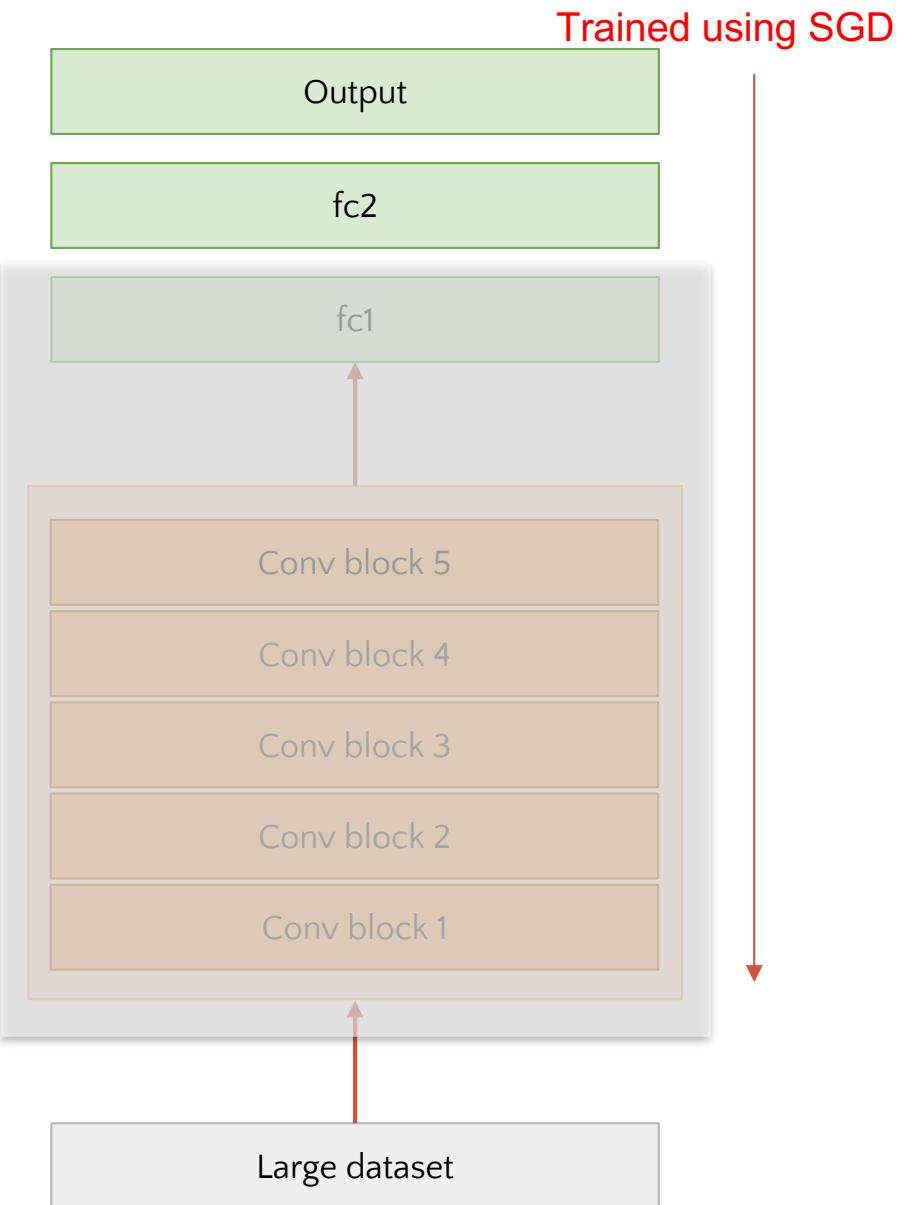
2. Replace the top layers with target task



Transfer learning

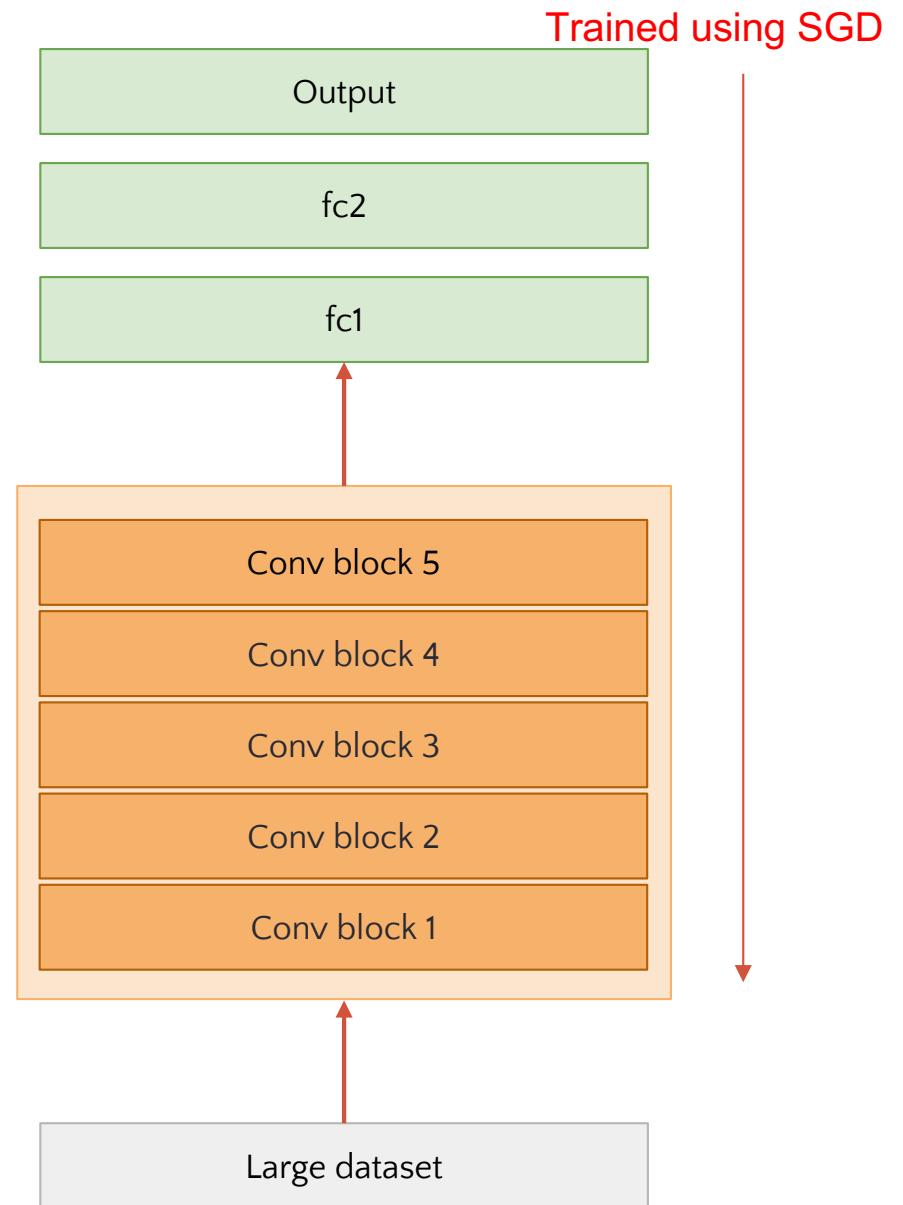
Freeze weights

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



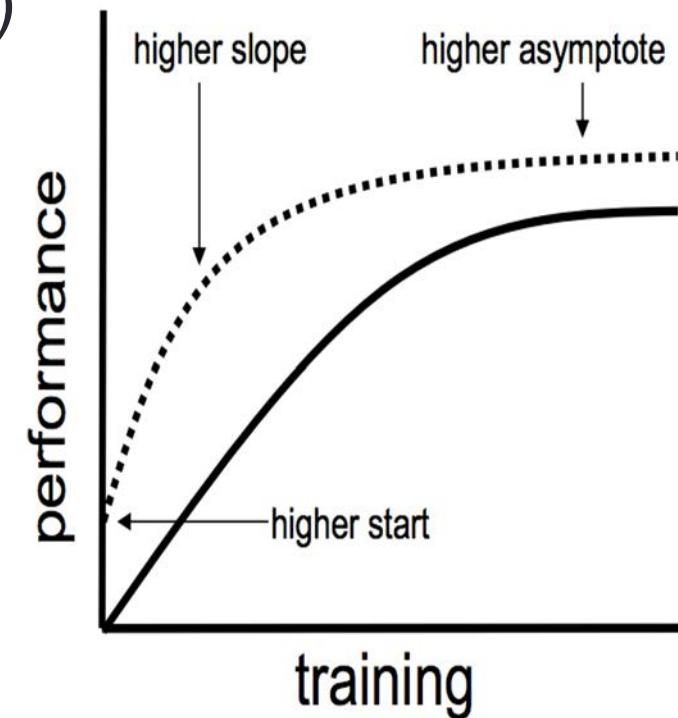
Transfer learning

3B. Train all layers
(unfreezed weights)



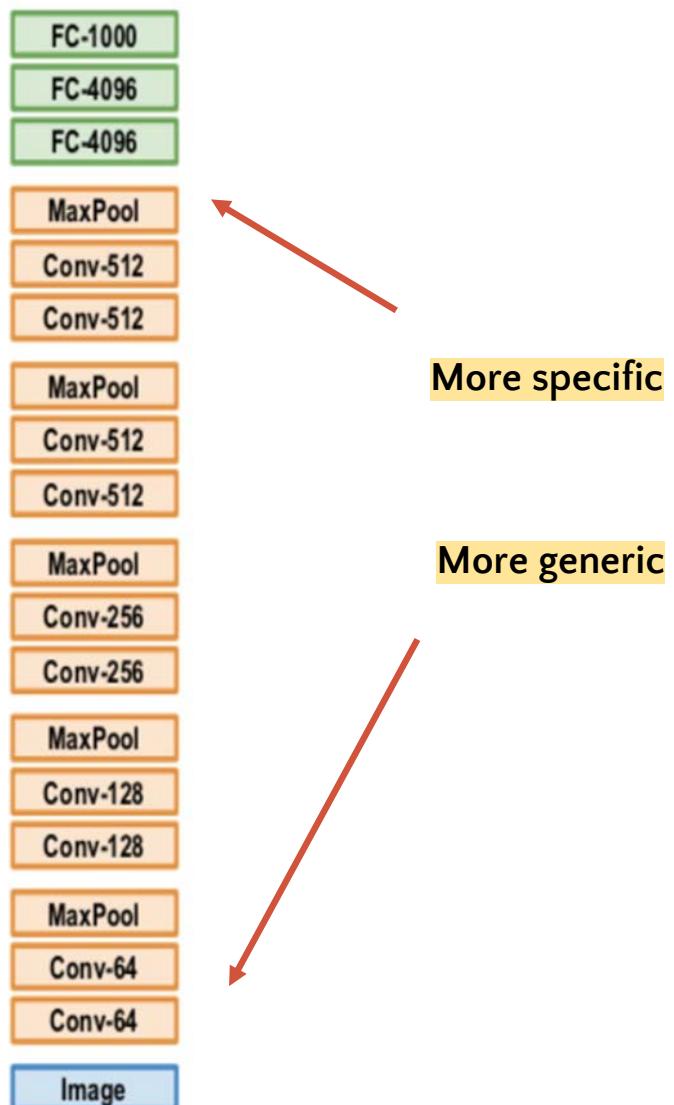
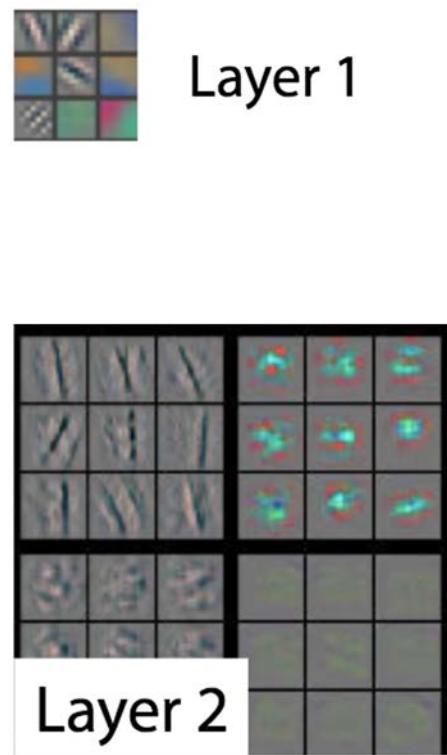
Benefits to transfer learning

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



Layers

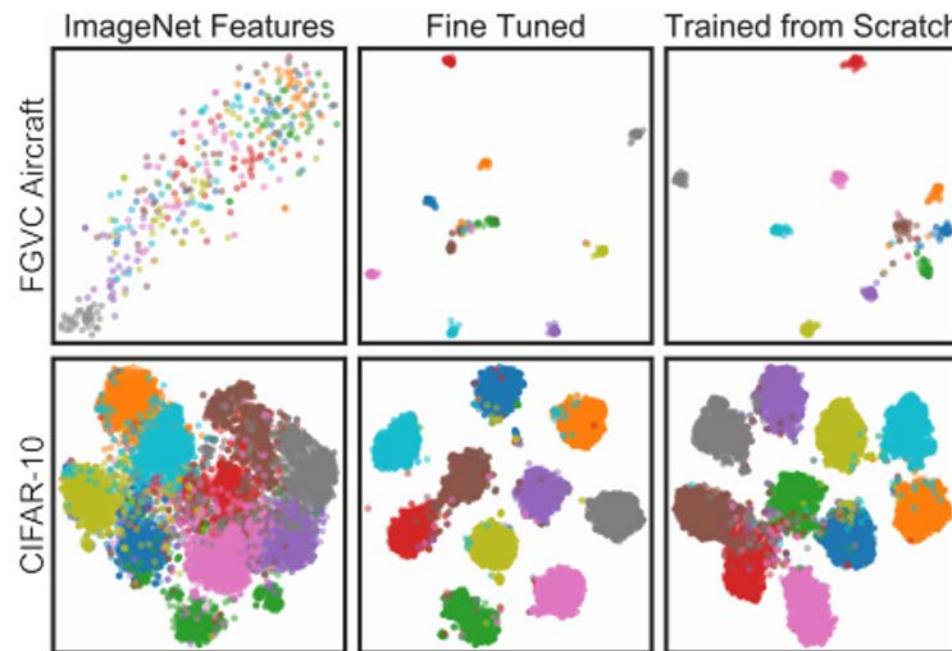
Lower layers: more general
Higher layers: more specific



Domains

Similar domain can transfer easier
Fine-tuning (adaptation) is crucial

Aircraft images
Different from ImageNet



Natural images
Similar to ImageNet

Example: pose estimation



<https://arxiv.org/abs/1611.08050>

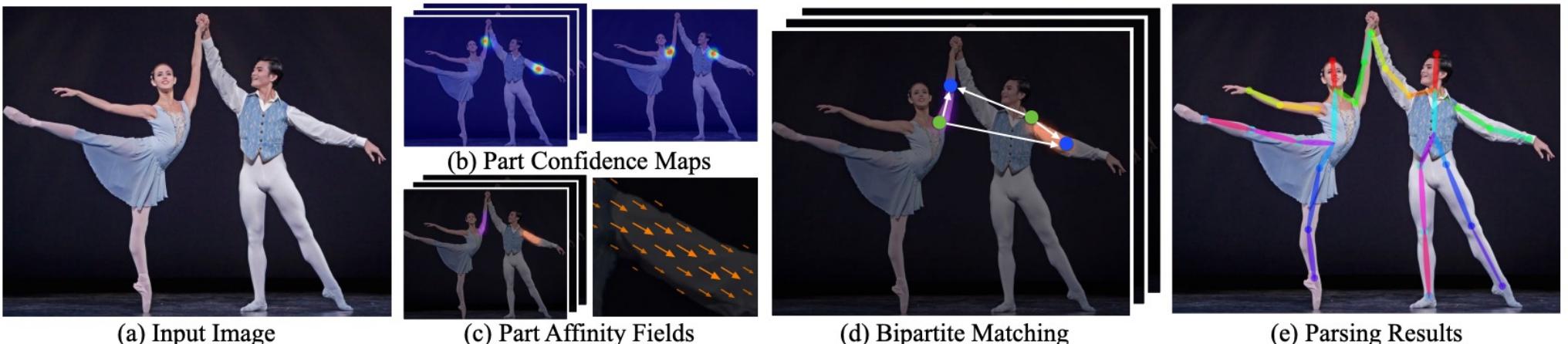


Figure 2. Overall pipeline. Our method takes the entire image as the input for a two-branch CNN to jointly predict confidence maps for body part detection, shown in (b), and part affinity fields for parts association, shown in (c). The parsing step performs a set of bipartite matchings to associate body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

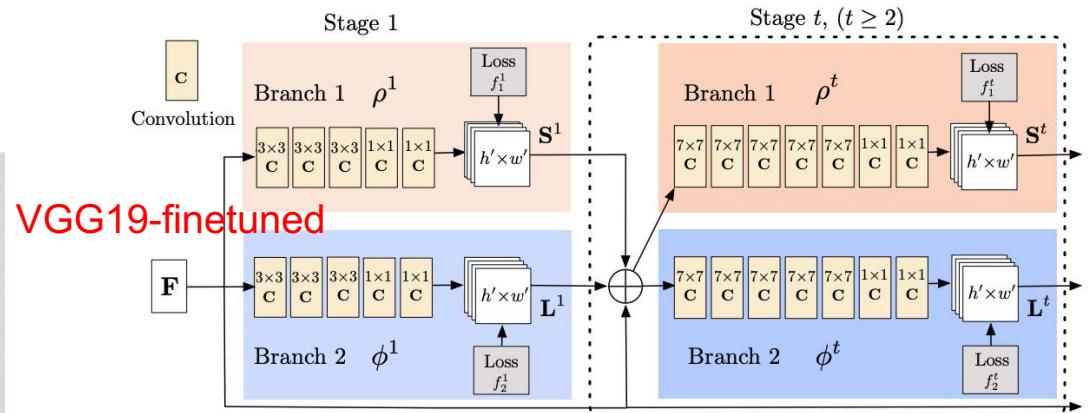


Figure 3. Architecture of the two-branch multi-stage CNN. Each stage in the first branch predicts confidence maps S^t , and each stage in the second branch predicts PAFs L^t . After each stage, the predictions from the two branches, along with the image features, are concatenated for next stage.

Unsupervised Learning

- What does it mean to be unsupervised?
- Semi-supervised learning
- Transfer learning
- Self-supervised learning
 - Consistency training
 - Contrastive learning

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 its 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 is 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.

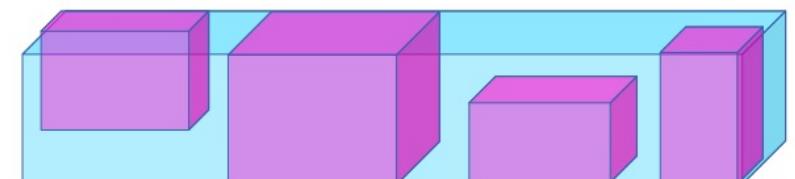
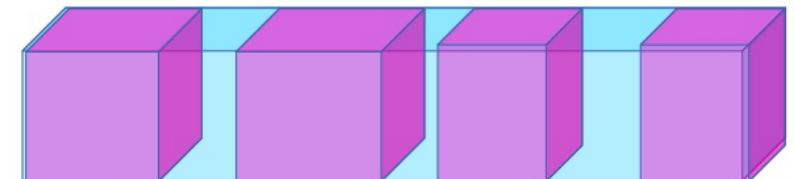
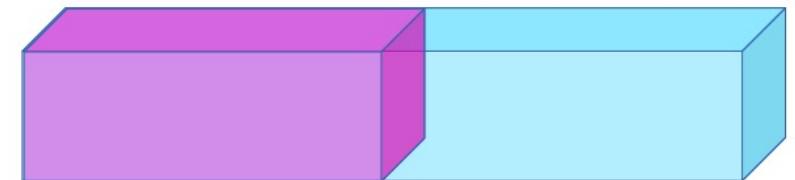
So far, similar approaches haven't worked quite as well for images or videos because of the difficulty of representing distributions over high-dimensional continuous spaces.

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

- ▶ 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

time or space →



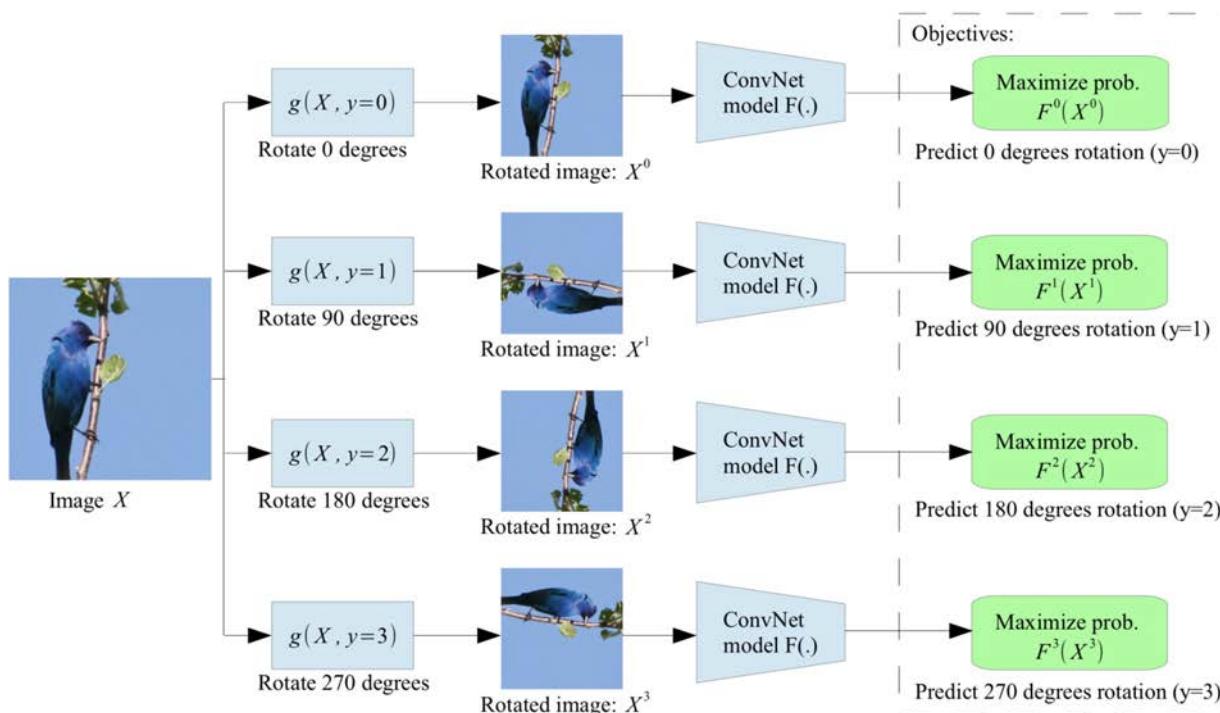
Examples

Text

Predict masked text - BERT

Images

Predict missing patches, predict orientation, etc.

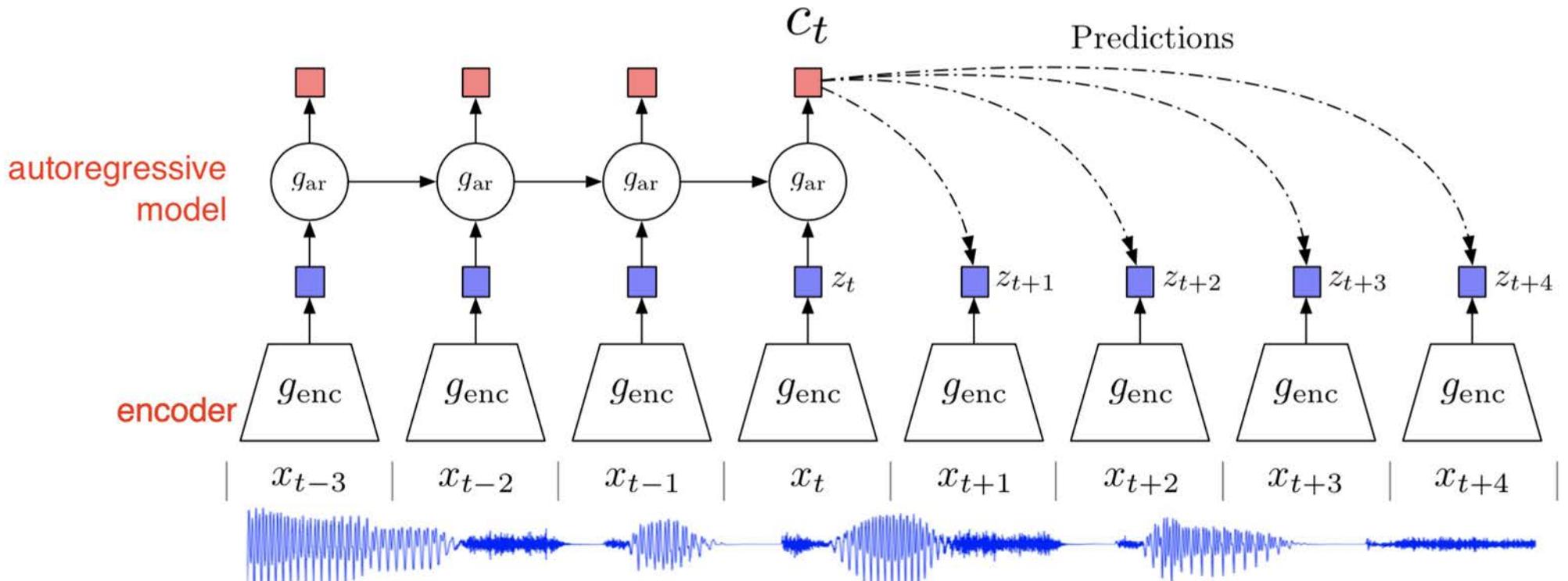


Examples

Sound

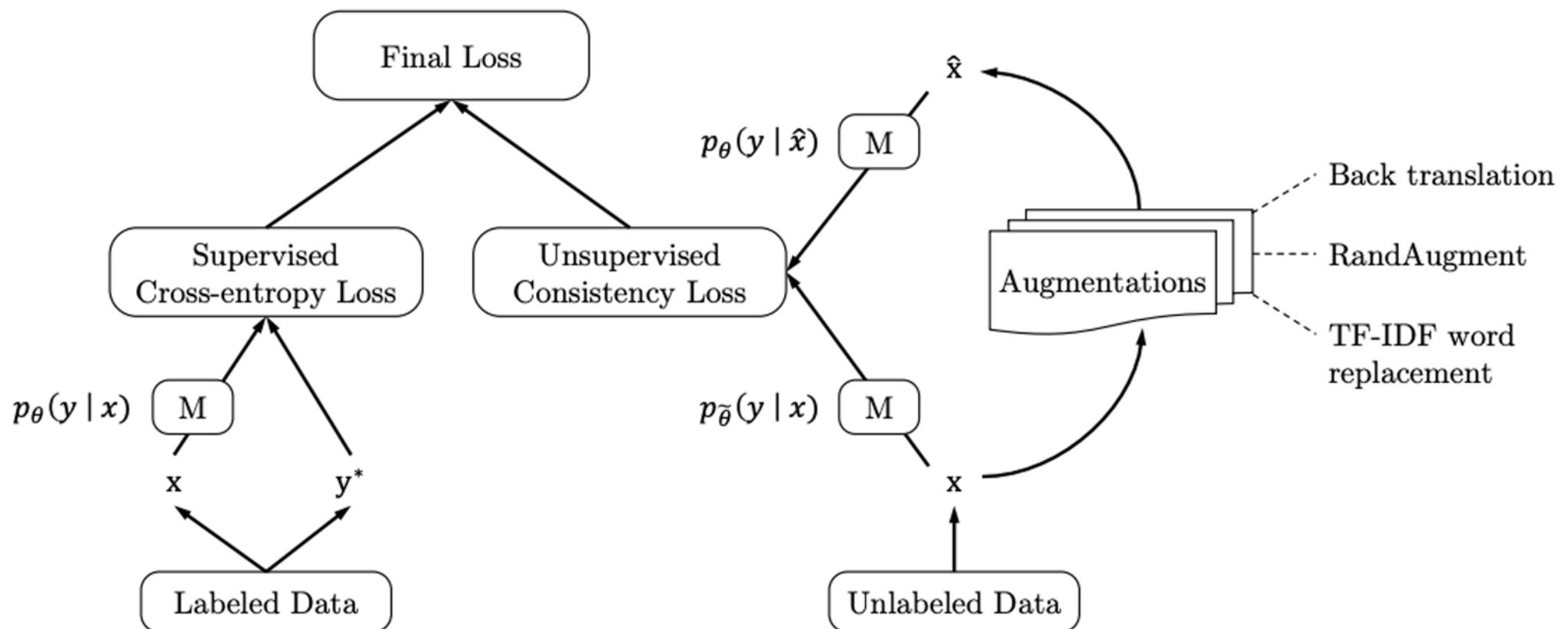
Cluster data into discrete classes then predict masked portions

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



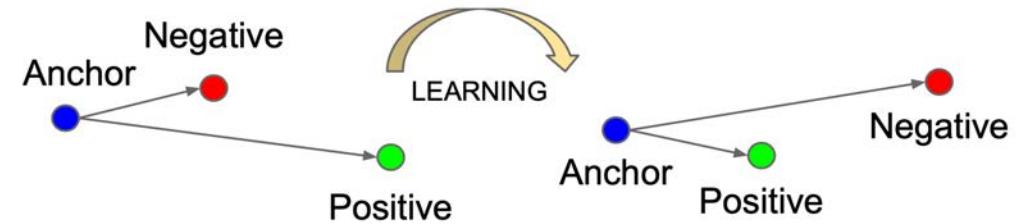
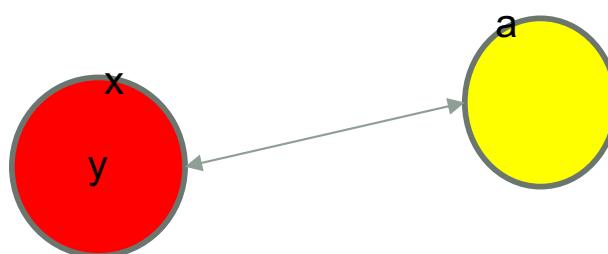
Consistency training

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



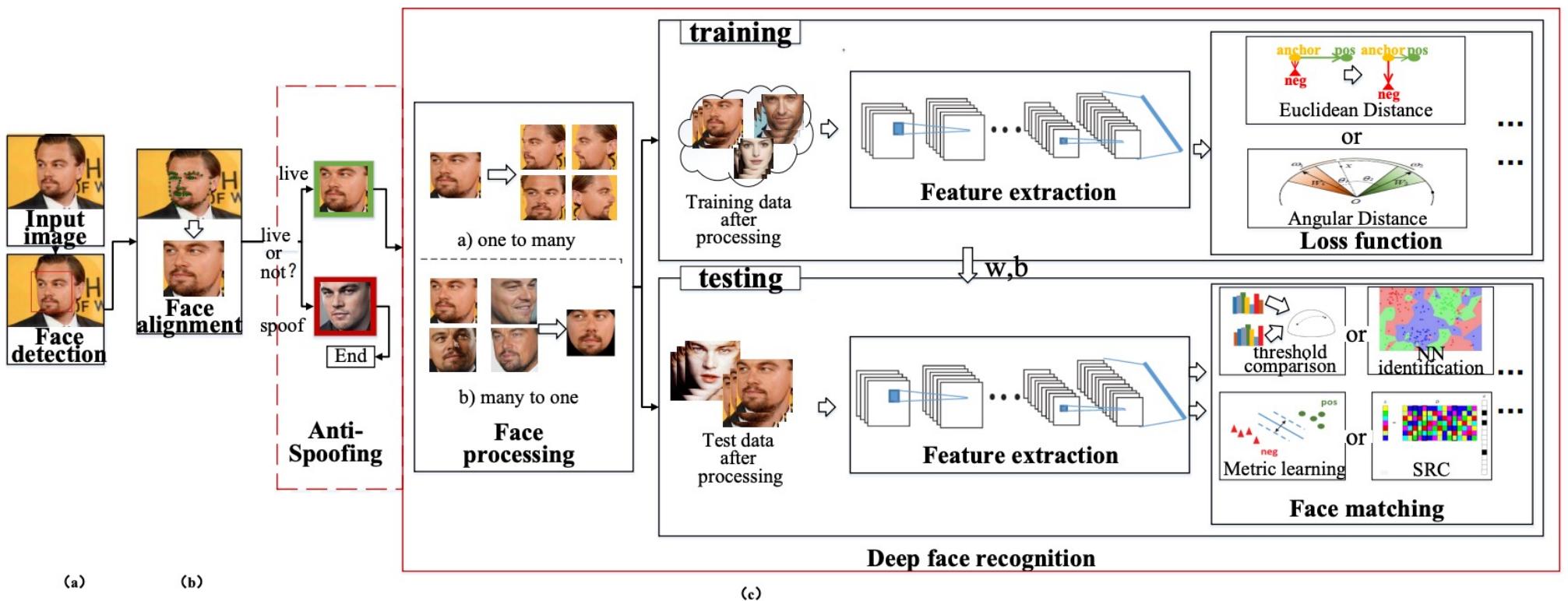
Contrastive training

- 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 verification task
- These two are often times used together, and the clear differentiation between the two are vague



<https://arxiv.org/abs/1503.03832>

Deep face verification



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

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

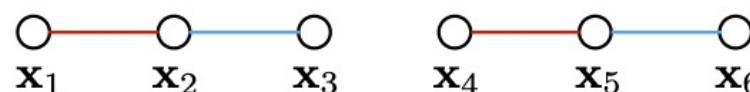


Dealing with minibatches

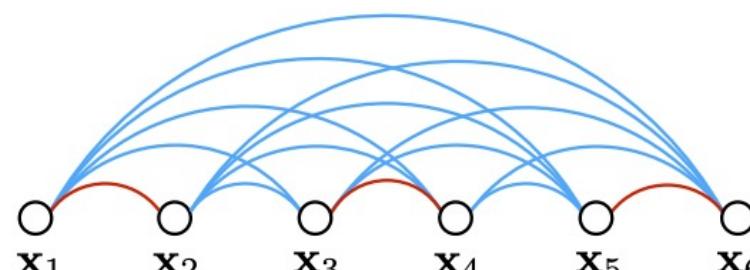
- Since we train in minibatches, most modern losses pair positive and negative samples within a minibatch for more efficient computation
 - Compute all pairwise distance within the minibatch



(a) Contrastive embedding



(b) Triplet embedding



(c) Lifted structured embedding

NCE (Noise contrastive estimation) loss

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

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 \mathbf{c} , the positive \mathbf{x} should be selected rather than the negative \mathbf{x}
- Effectively maximize mutual information between \mathbf{c} and positive \mathbf{x}

$$L_{InfoNCE} = -E[\log \frac{f(x, c)}{\sum_{x'} f(x', c)}] \quad f(x, c) = \exp(\mathbf{z}^T \mathbf{W} \mathbf{c})$$

\mathbf{z} is encoded \mathbf{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>

Soft nearest neighbor loss

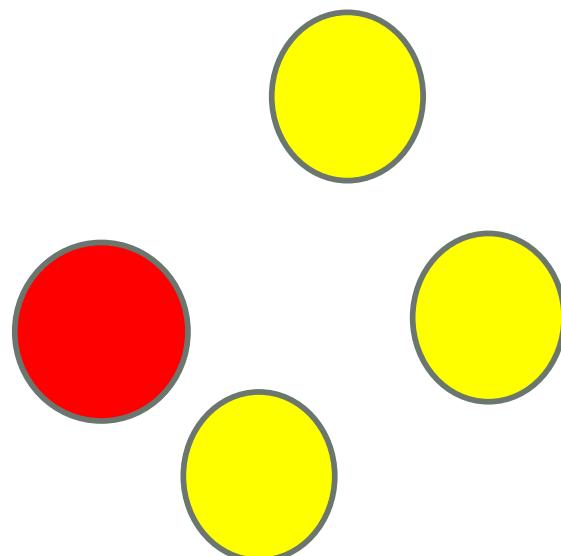
- Multiple positive and negative
- Adds temperature (either hyperparameter, or learned)

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

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

Key details to make this work

- Large batch
- Hard/semi-hard negative mining
- Augmentation on the anchor and positive (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 zeroed 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 \mathbf{W} is the weight associated with the class

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

Margin in the softmax (L-softmax)

- To classify as class 1

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

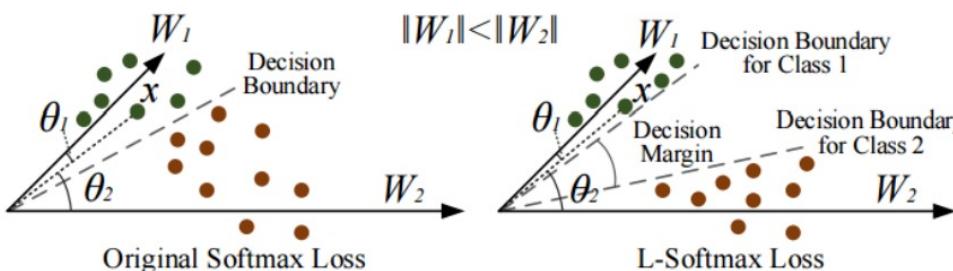
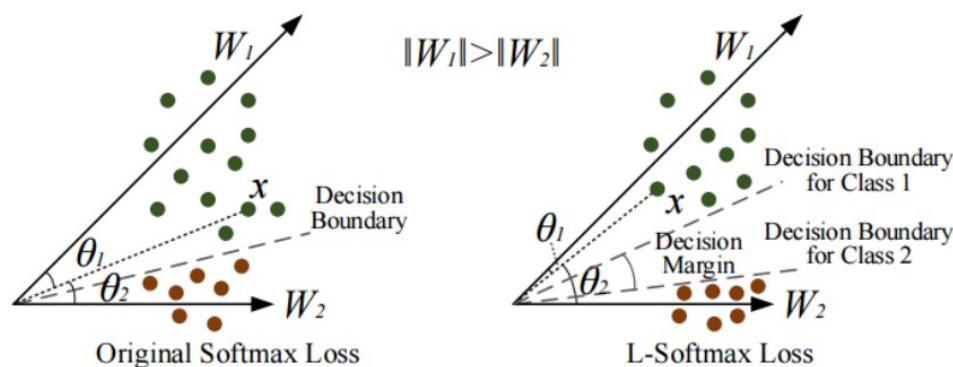
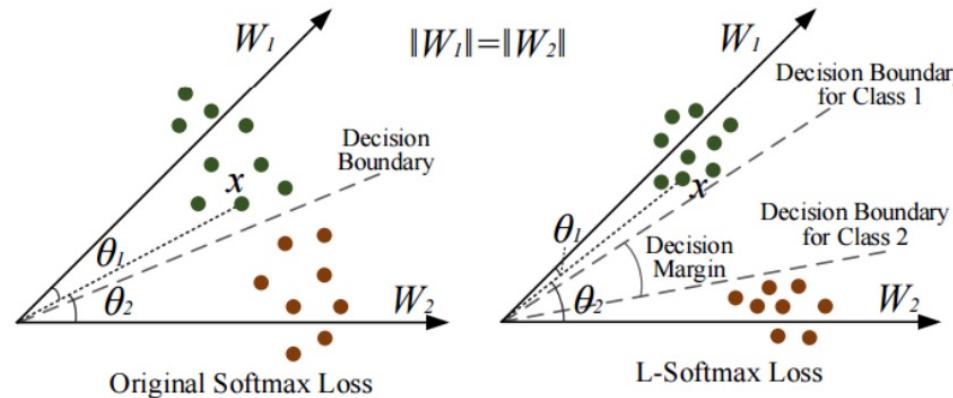
must be true

- We introduce an angular margin m

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

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

Angular margin effect



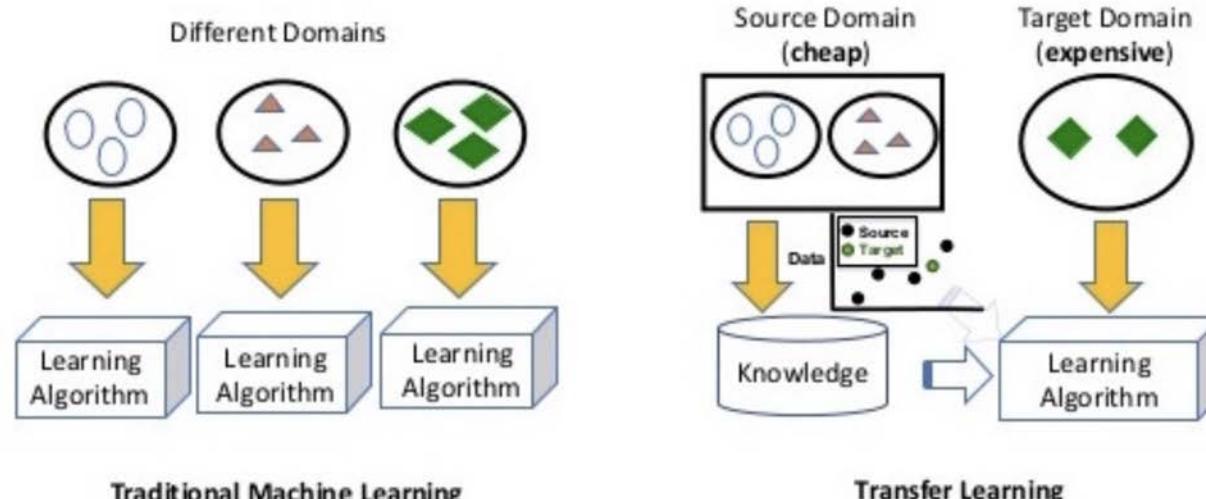
Other related tasks

Domain adaptation (domain-shift)

Classify sentiment on book reviews -> Classify sentiment on restaurant reviews

With labels in the target domain (supervised domain adaptation)

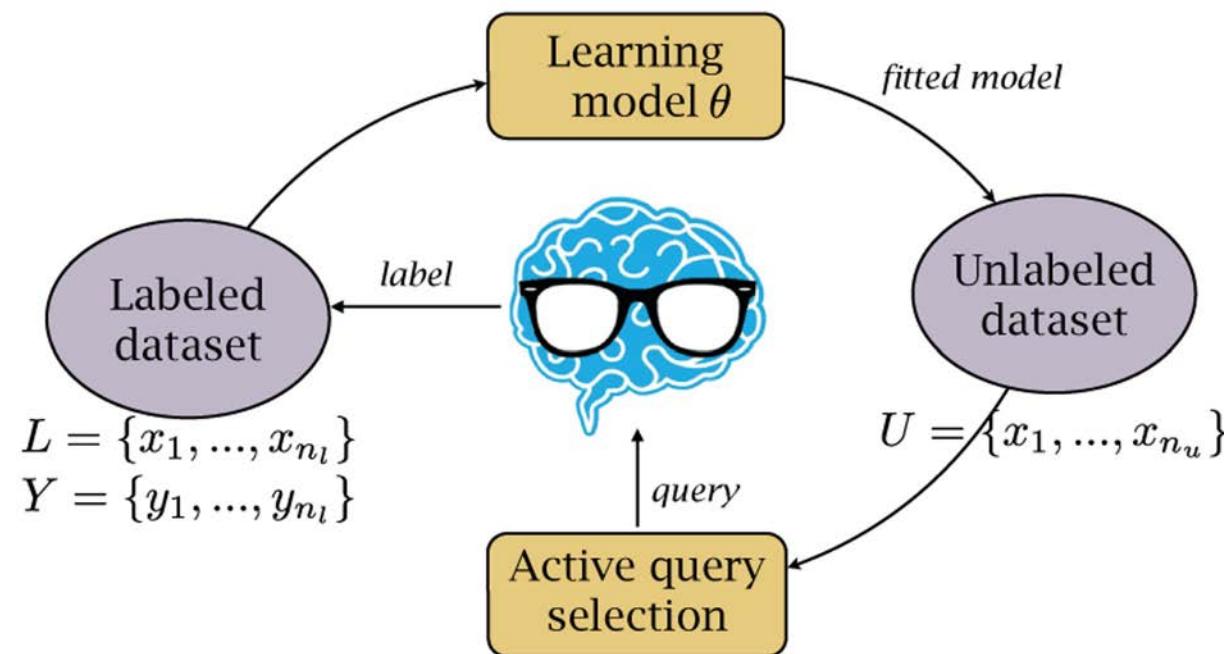
Without labels in the target domain (unsupervised domain adaptation)



Other related tasks

Active learning

Find interesting unlabeled data for additional labeling



Summary

Noise and augmentation

- Network noise and data augmentation

Pull positive and push negative

- Different variants

Further reading

- Contrastive loss

<https://lilianweng.github.io/posts/2021-05-31-contrastive>

- Self-supervise

<https://lilianweng.github.io/posts/2019-11-10-self-supervised/>