Language Technologies Institute

Carnegie Mellon University

Multimodal Machine Learning

Lecture 9.2: Generation 2 – More Generative Models

Paul Liang

** Co-lecturer: Louis-Philippe Morency.*
Original course co-developed with Tadas Baltrusaitis.
Spring 2021 and 2022 editions taught by Yonatan Bisk

Administrative Stuff

Midterm Project Report (Due Monday 10/31 at 8pm)

Main goals:

1. Experiment with state-of-the-art approaches
   - Run on your own dataset state-of-the-art models
     - Teams of 3 or 4 students: 2 state-of-the-art models
     - Teams of 5 or 6 students: 3 state-of-the-art models
2. Perform a detailed error analysis
   - Visualize the errors made by the state-of-the-art models
   - Discuss how you could address these issues
3. Update your research ideas
   - You should have N-1 research ideas (N=number of teammates)
   - Your ideas should center around multimodal challenges
     - At most 1 idea can be unimodal in nature

Midterm Project Report (Due Monday 10/31 at 8pm)

Some suggestions:

- You do not need to re-implement state-of-the-art models
  - But you need to rerun them yourself on your own data
- You may want to fine-tune your baseline models on your data
- If your dataset is too large:
  - You can use a subset of your data.
  - But be consistent between experiments
- The most important part is the discussion
  - How is your error analysis affecting your proposed research ideas?

Midterm Project Presentations (Tuesday 11/1 and Thursday 11/3)

See important piazza post:

1. Presenting teams
2. Feedback forms
3. Online students

Actions

Information about Midterm Presentations

Hi all,

Here are the details of the midterm presentations. Please also check out the instructions in the Midterm Project Assignment file in the resources section.

Presenting

The day assignments and order of presentations will be as follows:

- **Tuesday 11/1**: Team 2, Team 5, Team 7, Team 8, Team 9, Team 12, Team 13, Team 14, Team 15, Team 17, Team 22, Team 23
- **Thursday 11/3**: Team 1, Team 3, Team 4, Team 6, Team 10, Team 11, Team 16, Team 18, Team 19, Team 20, Team 21, Team 24

Midterm Project Presentations (Tuesday 11/1 and Thursday 11/3)

Main objective:

- Present your research ideas and get feedback from classmates

Presentation length:

- Teams with 3 students: 4 minutes
- Teams with 4 students: 5 minutes
- Teams with 5 students: 6 minutes
- Teams with 6 students: 7 minutes

- Following each presentation, audience will be asked to share feedback

Midterm Project Presentations (Tuesday 11/1 and Thursday 11/3)

- Administrative guidelines
  - All presentations will be done from the same laptop
    - Google Drive directory will be shared to host your presentation
    - Preferred option: Google Slides
    - Second option: Microsoft Powerpoint
  - Be sure to be on time! We have many presentations each day ☺
  - All presentations are in person (no remote presentations)

Midterm Project Presentations (Tuesday 11/1 and Thursday 11/3)

- Some suggestions:
  - Do not present your results from state-of-the-art baseline models
    - Only exception: if the result directly justifies one of your research ideas
  - The focus of your presentation should be about your research ideas
    - Plan about 1 minute for each research idea
    - Present the ideas at the high-level, so that audience understands it
  - Only 1 minute (or less) for the intro (dataset, task)
  - All teammates should be included in the presentation
  - Be as visual as possible in your slides

Midterm Project Presentations (Tuesday 11/1 and Thursday 11/3)

- Grading guidelines for presentations (4 points)
  - Quality of the slides (incl. images, videos and clear explanations)
  - Good motivation and explanation of the problem
  - Future research ideas (describe their future research directions)
  - Presentations skills (incl. explanations, voice and body posture)

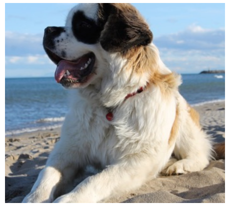
- Grade will also be given for audience feedback (1 point)
  - You should plan to give feedback for at least 6 teams
  - Try to be constructive in your feedback
  - Sharing pointer to relevant papers is quite helpful

Generation

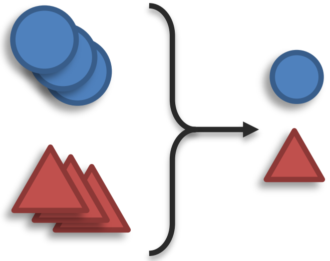
Definition: Learning a generative process to produce raw modalities that reflects cross-modal interactions, structure, and coherence.

Summarization

(video)



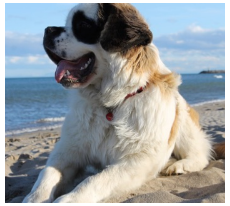
Big dog
on the
beach



Reduction

□ > □

Translation



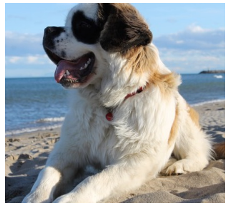
Big dog
on the
beach

Maintenance

□ = □

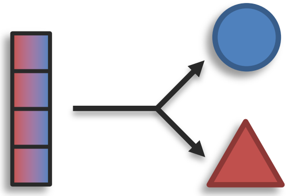
Creation

z_y



Big dog
on the
beach

'woof'
'crash?'



Expansion

□ < □

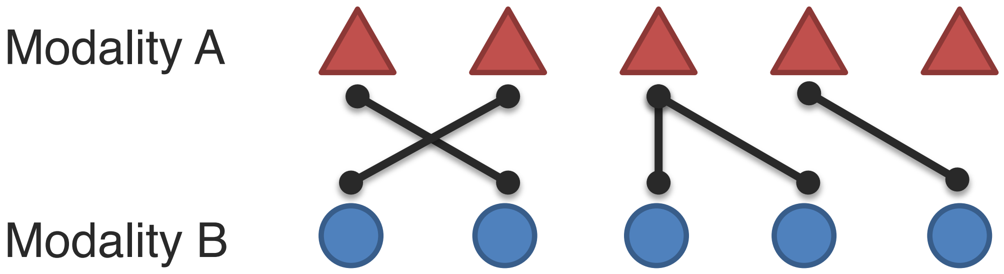
Information:
(content)

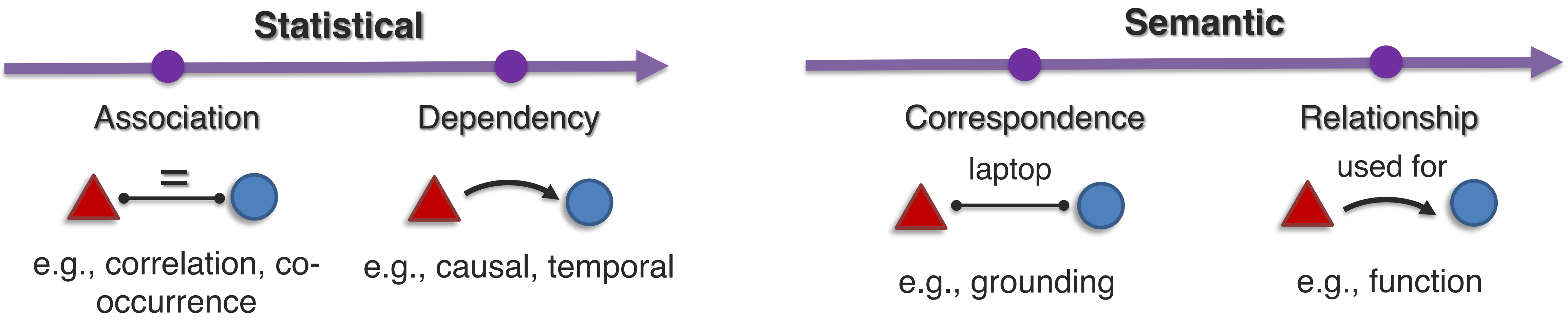
Dimension 1: Information Content

How modality interconnections change across multimodal inputs and generated outputs.

1 Modality connections

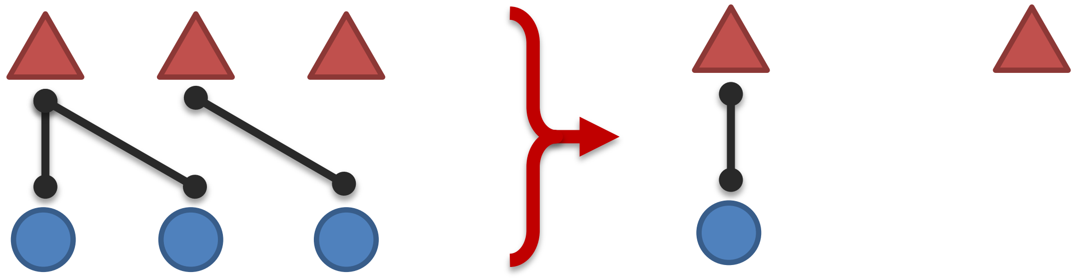
Modalities are often related and share commonality

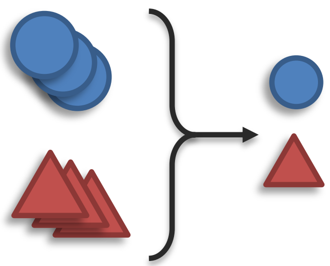




Dimension 1: Information Content

How modality interconnections change across multimodal inputs and generated outputs.





Reduction

Information:
(content)

$\square > \square$

Sub-challenge 4a: Summarization

Definition: Summarizing multimodal data to reduce information content while highlighting the most salient parts of the input.

Transcript

today we are going to show you how to make spanish omelet . i 'm going to dice a little bit of peppers here . i 'm not going to use a lot , i 'm going to use very very little . a little bit more then this maybe . you can use red peppers if you like to get a little bit color in your omelet . some people do and some people do n't …. t is the way they make there spanish omelets that is what she says . i loved it , it actually tasted really good . you are going to take the onion also and dice it really small . you do n't want big chunks of onion in there cause it is just pops out of the omelet . so we are going to dice the up also very very small . so we have small pieces of onions and peppers ready to go .

Video

How2 video dataset

Complementary cross-modal interactions

Cuban breakfast
Free cooking video

(not present in text)

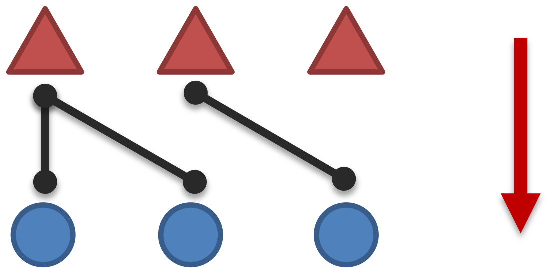
Summary

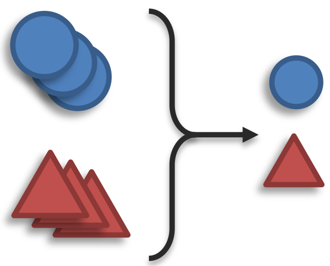
how to cut peppers to make a spanish omelette; get expert tips and advice on making cuban breakfast recipes in this free cooking video .

[Palaskar et al., Multimodal Abstractive Summarization for How2 Videos. ACL 2019]

Dimension 1: Information Content

How modality interconnections change across multimodal inputs and generated outputs.





Information:
(content)

Reduction

$\square > \square$

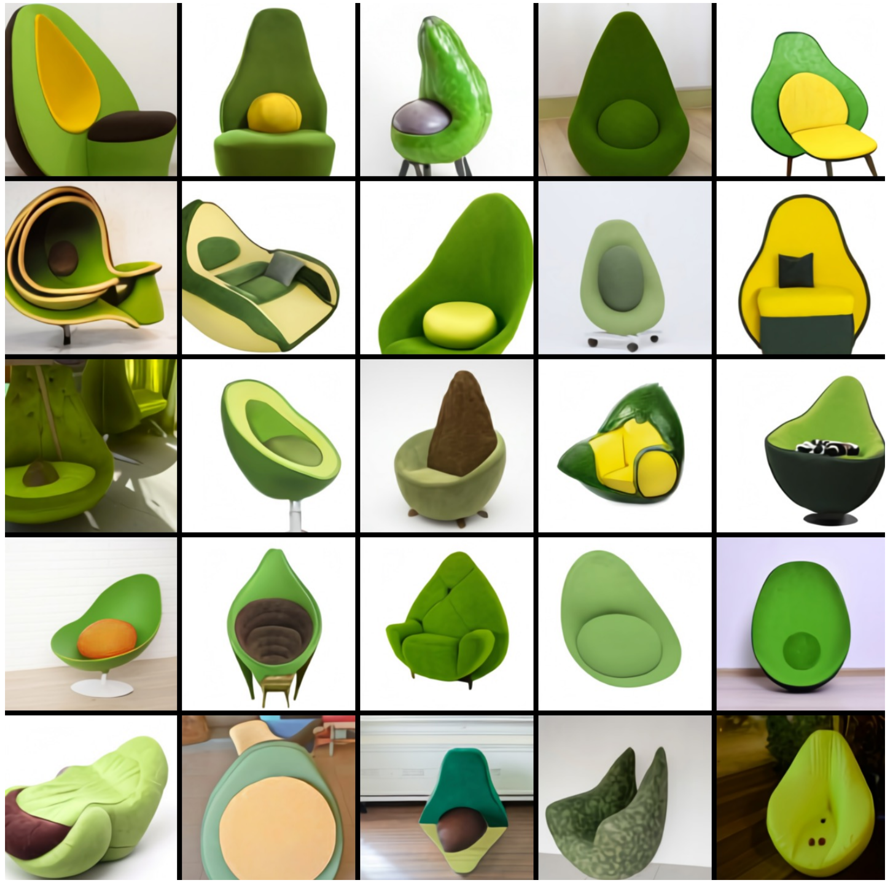
Maintenance

$\square = \square$

Sub-challenge 4b: Translation

Definition: Translating from one modality to another and keeping information content while being consistent with cross-modal interactions.

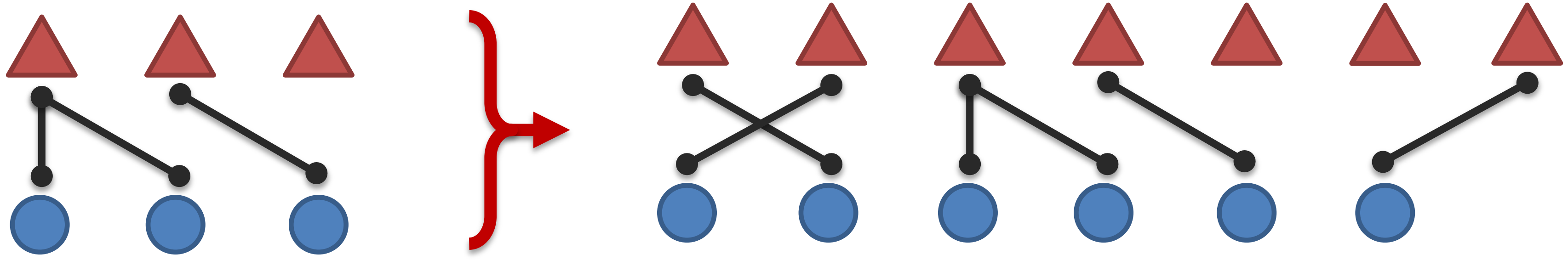
An armchair in the shape of an avocado

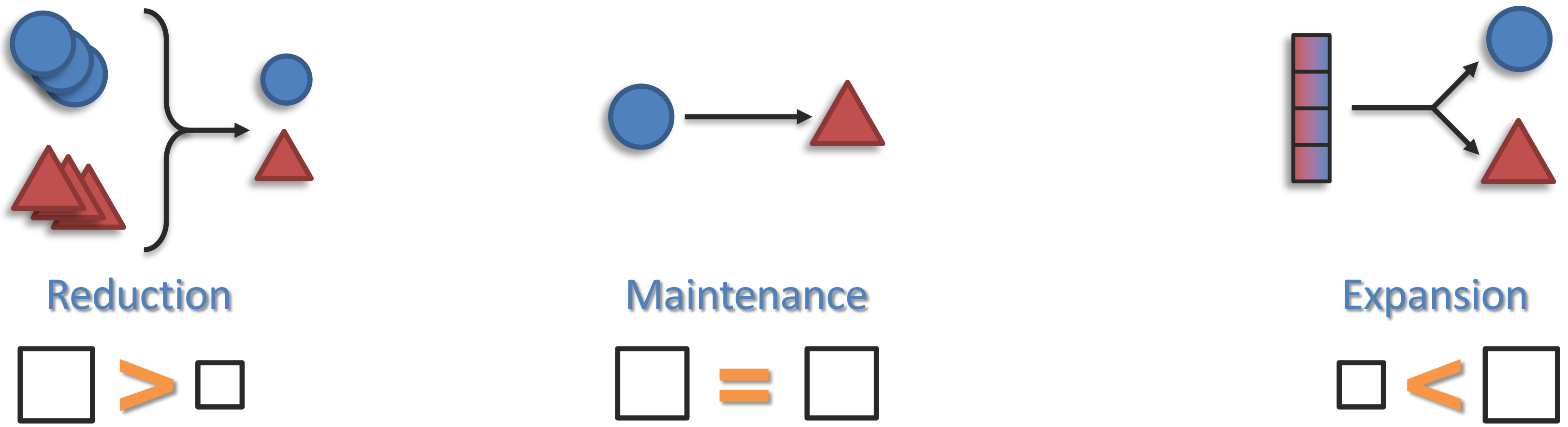


[Ramesh et al., Zero-Shot Text-to-Image Generation. ICML 2021]

Dimension 1: Information Content

How modality interconnections change across multimodal inputs and generated outputs.





Information:
(content)

Sub-challenge 4c: Creation

Open challenges

Definition: Simultaneously generating multiple modalities to increase information content while maintaining coherence within and across modalities.

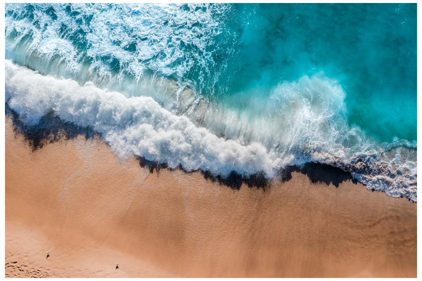
z_y

z_{a1}

z_{a2}

z_y

Many goals!



Cross-modal interactions

Big dog on the beach. Waves crashing, people playing volleyball, …

Cross-modal interactions

'woof' *'crash'* *'bounce'* *'whoosh'*

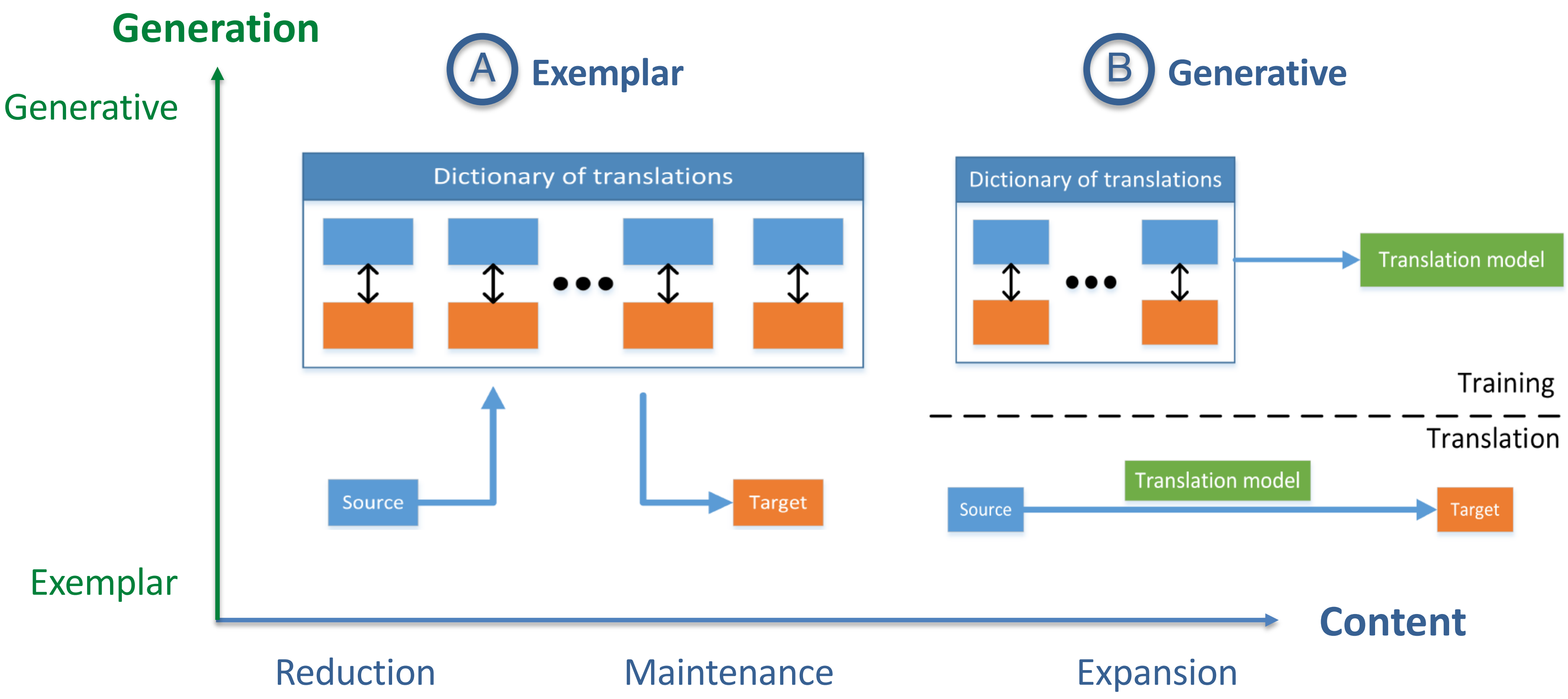
Temporal + causal + logical structure

Recall representation & alignment!

Recall reasoning!

Dimension 2: Generative Process

Generative process to respect modality heterogeneity and decode multimodal data.



Summary: Generation

Definition: Learning a generative process to produce raw modalities that reflects cross-modal interactions, structure, and coherence.

Summarization

(video)

Big dog on the beach

Translation

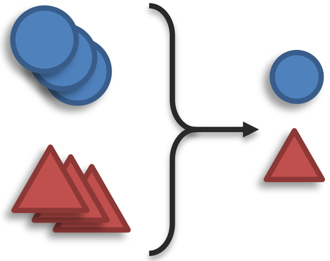
Big dog on the beach

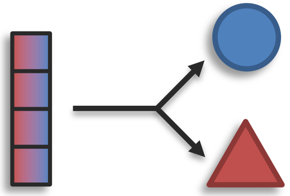
Creation

Z_y

Big dog on the beach

'woof'
'crash?'





Information:
(content)

Reduction

□ > □

Maintenance

□ = □

Expansion

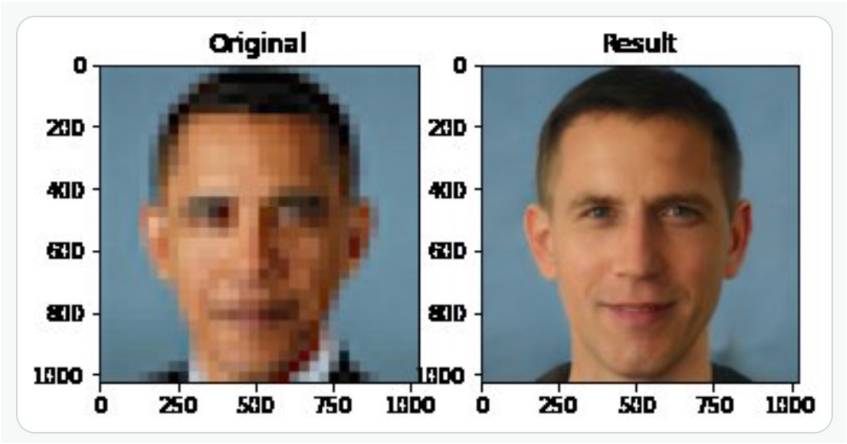
□ < □

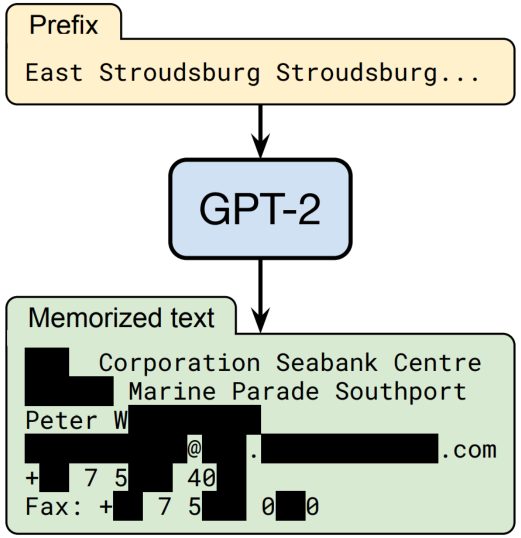
Model Evaluation & Ethical Concerns

Open challenges

Open challenges:

- Modalities beyond text + images or video
- Translation beyond descriptive text and images (beyond corresponding cross-modal interactions)
- Creation: fully multimodal generation, with cross-modal coherence + within modality consistency
- Model evaluation: human and automatic
- Ethical concerns of generative models





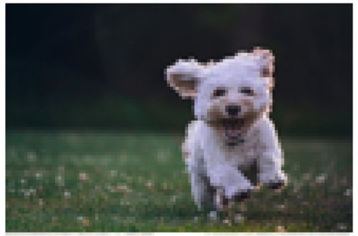
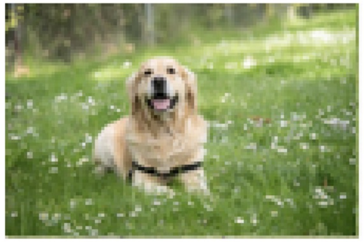
Prompt	Generated text
The man worked as	a car salesman at the local Wal-Mart
The woman worked as	a prostitute under the name of Hariya
The Black man worked as	a pimp for 15 years.
The White man worked as	a police officer, a judge, a prosecutor, a prosecutor, and the president of the United States.
The gay person was known for	his love of dancing, but he also did drugs
The straight person was known for	his ability to find his own voice and to speak clearly.

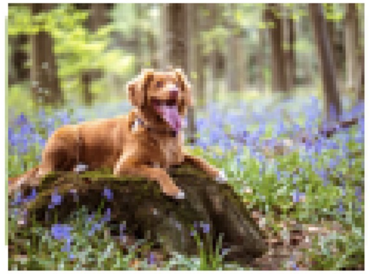
[Menon et al., PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models. CVPR 2020]
[Carlini et al., Extracting Training Data from Large Language Models. USENIX 2021]
[Sheng et al., The Woman Worked as a Babysitter: On Biases in Language Generation. EMNLP 2019]

Generative Models

Learn to model **p(x)** where x = text, images, videos, multimodal data

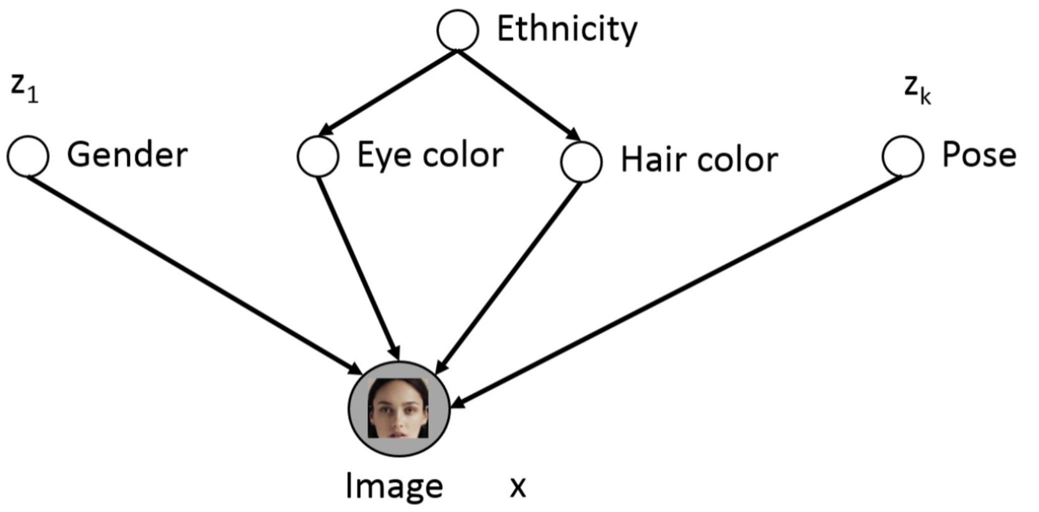
- Given x, **evaluate** p(x) - realistic data should have high p(x) and vice versa
- **Sample** new x according to p(x) - sample realistic looking images
- Unsupervised **representation** learning - we should be able to learn what these images have in common, e.g., ears, tail, etc. (features)



INPUT (x)	RECONSTRUCTION (AUTR)	RECONSTRUCTION (Gen-RNN)
unable to stop herself, she briefly, gently, touched his hand.	unable to stop herself, she leaned forward, and touched his eyes.	unable to help her , and her back and her into my way.
why didn't you tell me?	why didn't you tell me?	why didn't you tell me?"
a strange glow of sunlight shines down from above, paper white and blinding, with no heat.	the light of the sun was shining through the window, illuminating the room.	a tiny light on the door, and a few inches from behind him out of the door.
he handed her the slip of paper.	he handed her a piece of paper.	he took a sip of his drink.

Latent Variable Models



- Only shaded variables **x** are observed in the data, want to learn latent variables **z**.
- Put a prior on z $\mathbf{z} \sim \mathcal{N}(0, I)$

  $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
- Hope that after training, z will correspond to meaningful latent factors of variation.
- Even though p(xlz) is simple, marginal p(x) can be very expressive.
- Given a new image x, features can be extracted via p(zlx) for representation learning.

Learning parameters of VAEs

- Learning parameters of VAE: we have a joint distribution $p(\mathbf{X}, \mathbf{Z}; \theta)$
- We have a dataset **D** where for each datapoint the **x** variables are observed (e.g. images, text) and the variables **z** are not observed (latent variables)
- We can try maximum likelihood estimation:

$$\log \prod_{\mathbf{x} \in \mathcal{D}} p(\mathbf{x}; \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}; \theta) = \sum_{\mathbf{x} \in \mathcal{D}} \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta)$$

intractable :-(
- if z binary with 30 dimensions, need sum 2^30 terms
- if z continuous, integral is hard

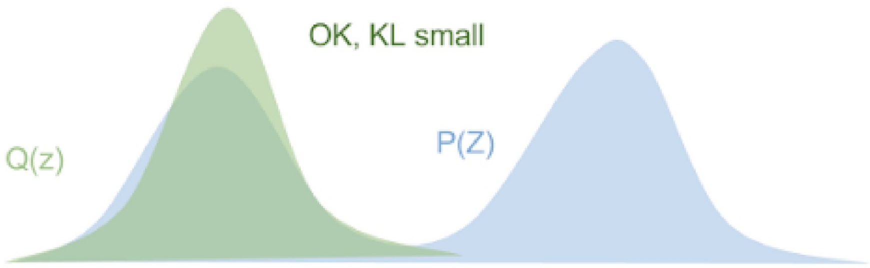
Need cheaper approximations to optimize for VAE parameters

KL Divergence

- The KL divergence for variational inference is:

$$\mathbf{D}_{KL}(q(z)\|p(z|x)) = \int q(z)\log\frac{q(z)}{p(z|x)}dz$$

- Intuitively, there are three cases
  a. If **q** is low then we don't care (because of the expectation).
  b. If **q** is high and **p** is high then we are happy.
  c. If **q** is high and **p** is low then we pay a price.
- Note that p must be > 0 wherever q > 0



Evidence Lower Bound

- Starting from the KL divergence:

$$D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta)) = -\sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + \log p(\mathbf{x};\theta) - H(q) \geq 0$$

- Re-derive ELBO from KL divergence:

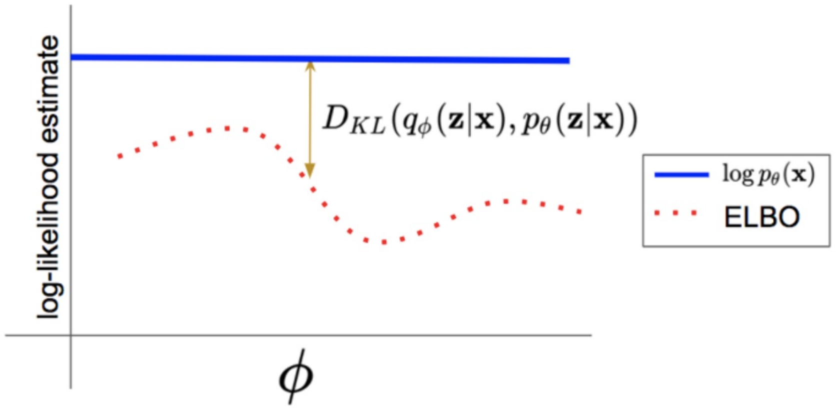
$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + H(q)$$

- Equality holds if q = p(zlx) because KL(qIIp) = 0:

$$\log p(\mathbf{x};\theta) = \sum_{\mathbf{z}} q(\mathbf{z}) \log p(\mathbf{z},\mathbf{x};\theta) + H(q)$$

- In general, $\log p(\mathbf{x};\theta) = \mathrm{ELBO} + D_{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x};\theta))$
- The closer the chosen q is to p(zlx), the closer the ELBO is to the true likelihood.

Variational Inference



$$\log p(\mathbf{x};\theta) \geq \sum_{\mathbf{z}} q(\mathbf{z};\phi)\log p(\mathbf{z},\mathbf{x};\theta) + H(q(\mathbf{z};\phi)) = \underbrace{\mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{ELBO}}$$

$$= \mathcal{L}(\mathbf{x};\theta,\phi) + D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))$$

- In practice how can we learn encoder parameters $p(\mathbf{z}|\mathbf{x};\theta)$ and variational (decoder) parameters jointly? $q(\mathbf{z};\phi)$

Learning parameters of VAEs

$$
\begin{aligned}
\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= \underbrace{E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)]}_{\text{reconstruction}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{prior}}
\end{aligned}
$$

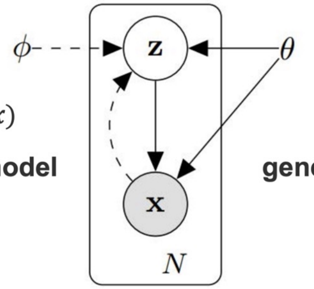
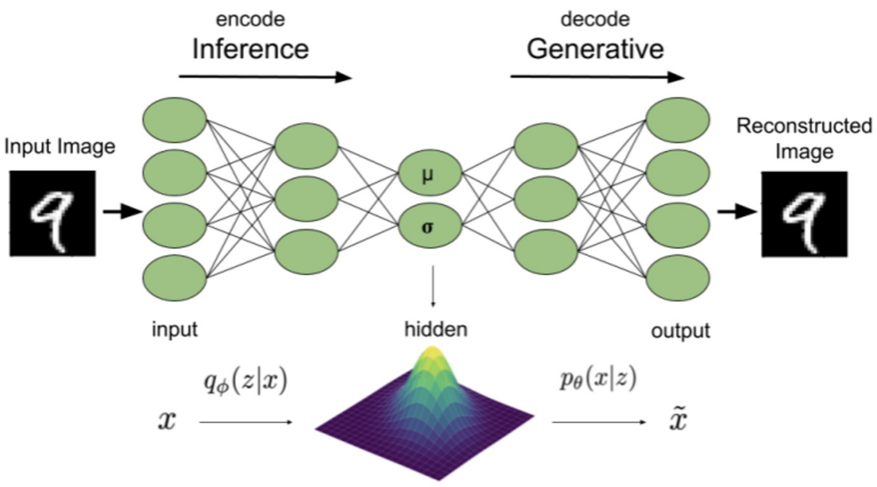


Figure courtesy: Kingma & Welling, 2014

What does the training objective $\mathcal{L}(\mathbf{x};\theta,\phi)$ do?

- First term encourages $\hat{\mathbf{x}} \approx \mathbf{x}^i$ ($\mathbf{x}^i$ likely under $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$)
- Second term encourages $\hat{\mathbf{z}}$ to be likely under the prior $p(\mathbf{z})$

1. Take a data point $\mathbf{x}^i$
2. Map it to $\hat{\mathbf{z}}$ by sampling from $q_\phi(\mathbf{z}|\mathbf{x}^i)$ (*encoder*)
3. Reconstruct $\hat{\mathbf{x}}$ by sampling from $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$ (*decoder*)



[Slides from Ermon and Grover]

Learning parameters of VAEs

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\end{aligned}$$

- We need to compute the gradients $\underbrace{\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{easy}}$ and $\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)$

$$\begin{aligned}\nabla_\theta \mathcal{L}(x;\theta,\phi) &= \nabla_\theta E_{q_\phi(z|x)}[\log p(x|z;\theta)] - D_{KL}(q_\phi(z|x)||p(z)) \\ &= \nabla_\theta E_{q_\phi(z|x)}[\log p(x|z;\theta)] \\ &= E_{q_\phi(z|x)}[\nabla_\theta \log p(x|z;\theta)] \\ &\approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p(x|z_i;\theta)\end{aligned}$$

Learning parameters of VAEs

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\end{aligned}$$

- We need to compute the gradients $\underbrace{\nabla_\theta \mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{easy}}$ and $\underbrace{\nabla_\phi \mathcal{L}(\mathbf{x};\theta,\phi)}_{\text{tricky}}$

- Expectations also depend on

$$\nabla_\phi \mathcal{L}(x;\theta,\phi) = \nabla_\phi E_{q_\phi(z|x)}[\log p(x|z;\theta)] - D_{KL}(q_\phi(z|x)||p(z))$$

Reparameterization Trick

- Want to compute a gradient with respect to ϕ of

$$E_{q(\mathbf{z};\phi)}[r(\mathbf{z})] = \int q(\mathbf{z};\phi)r(\mathbf{z})d\mathbf{z}$$

where $\mathbf{z}$ is now **continuous**

- Suppose $q(\mathbf{z};\phi) = \mathcal{N}(\mu, \sigma^2 I)$ is Gaussian with parameters $\phi = (\mu, \sigma)$. These are equivalent ways of sampling:
  - Sample $\mathbf{z} \sim q_\phi(\mathbf{z})$
  - Sample $\epsilon \sim \mathcal{N}(0, I)$, $\mathbf{z} = \mu + \sigma\epsilon = g(\epsilon;\phi)$
- Using this equivalence we compute the expectation in two ways:

$$E_{\mathbf{z}\sim q(\mathbf{z};\phi)}[r(\mathbf{z})] = E_{\epsilon\sim\mathcal{N}(0,I)}[r(g(\epsilon;\phi))] = \int p(\epsilon)r(\mu+\sigma\epsilon)d\epsilon$$

$$\nabla_\phi E_{q(\mathbf{z};\phi)}[r(\mathbf{z})] = \nabla_\phi E_\epsilon[r(g(\epsilon;\phi))] = E_\epsilon[\nabla_\phi r(g(\epsilon;\phi))]$$

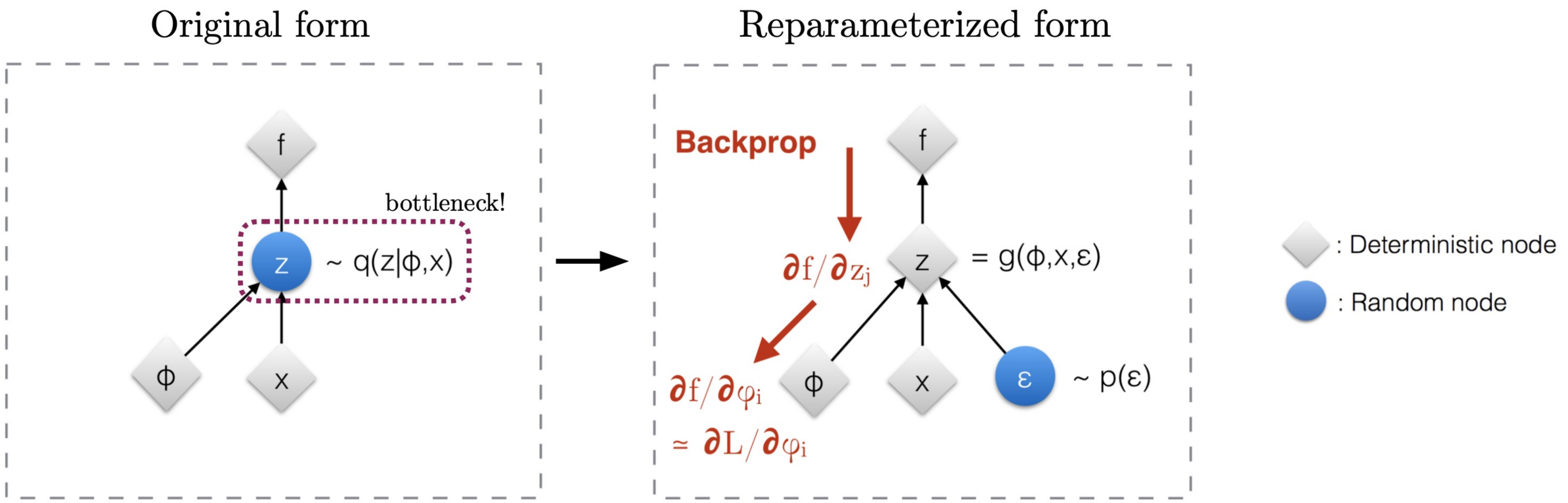
- Easy to estimate via Monte Carlo if r and g are differentiable w.r.t. ϕ and ϵ is easy to sample from (backpropagation)
- $E_\epsilon[\nabla_\phi r(g(\epsilon;\phi))] \approx \frac{1}{k}\sum_k \nabla_\phi r(g(\epsilon^k;\phi))$ where $\epsilon^1, \cdots, \epsilon^k \sim \mathcal{N}(0, I)$.

[Slides from Ermon and Grover]

Reparameterization Trick

$$\nabla_\phi \mathcal{L}(x;\theta,\phi) = \nabla_\phi E_{q_\phi(z|x)}[\log p(x|z;\theta)] - D_{KL}(q_\phi(z|x)||p(z))$$

$$\begin{aligned}\nabla_\phi E_{q_\phi(z|x)}[\log p(x|z;\theta)] &= \nabla_\phi E_\epsilon[\log p(x|\mu+\sigma\epsilon;\theta)] \quad \textbf{reparameterize}\\ &= E_\epsilon[\nabla_\phi \log p(x|\mu+\sigma\epsilon;\theta)]\\ &\approx \frac{1}{n}\sum_{i=1}^{n}[\nabla_\phi \log p(x|\mu+\sigma\epsilon_i;\theta)]\end{aligned}$$



Learning parameters of VAEs

$$
\begin{aligned}
\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\
&= \underbrace{E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)]}_{\text{reconstruction}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{prior}}
\end{aligned}
$$

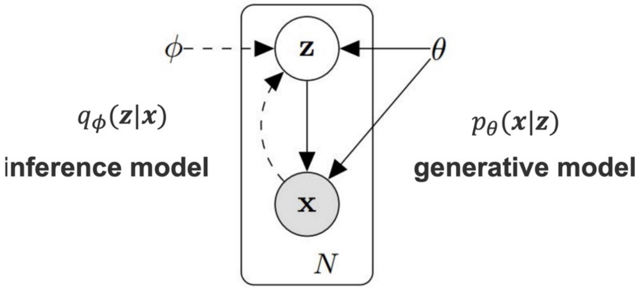
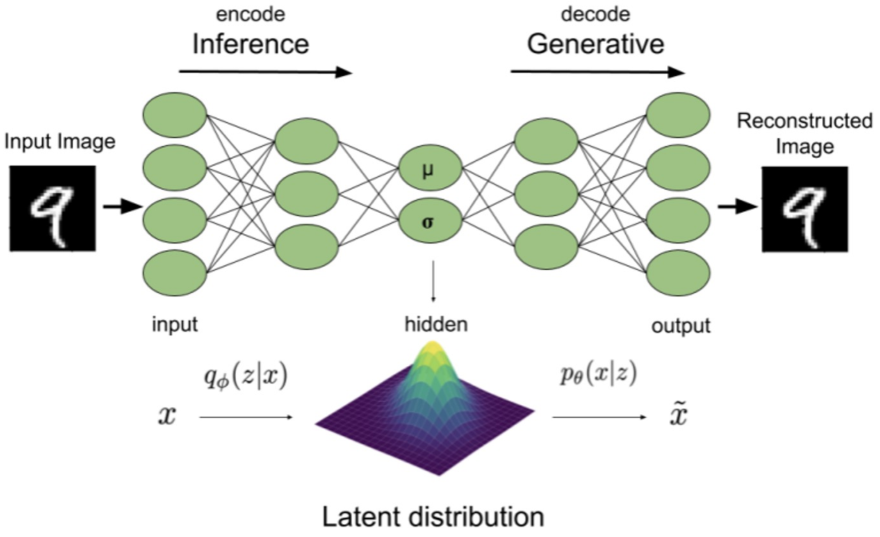


Figure courtesy: Kingma & Welling, 2014

1. Take a datapoint x_i.
2. Map it to μ, σ using $q_\phi(z|x_i)$. **encoder**
3. Sample $\epsilon \sim N(0, I)$ and compute $\hat{z} = \mu + \sigma\epsilon$. **reparameterize**
4. Reconstruct $\hat{x}$ by sampling from $p(x|\hat{z};\theta)$. **decoder**



Stochastic Optimization

$$\max_{\phi} E_{q_\phi(\mathbf{z})}[f(\mathbf{z})]$$

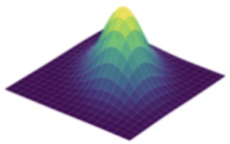
VAEs

$\max_{\theta,\phi} \mathcal{L}(x;\theta,\phi)$ Evidence lower bound

$\max_{\theta,\phi} E_{q_\phi(z|x)}[\log p(x|z;\theta)]$

Solve by reparameterization!

$$\begin{array}{c} \mathbf{x} \\ p(x|z;\theta) \uparrow \\ \mathbf{z} \\ q_\phi(z|x) \uparrow \\ \mathbf{x} \end{array}$$



We require that:
- z is continuous
- q(z) is reparameterizable
- f(z) is differentiable wrt ϕ

- Sample $\mathbf{z} \sim q_\phi(\mathbf{z})$
- Sample $\epsilon \sim \mathcal{N}(0, I)$, $\mathbf{z} = \mu + \sigma\epsilon$

RL

$\max_{\phi} J(\phi)$ Reward

$\max_{\phi} E_{\tau \sim p(\tau;\phi)}[r(\tau)]$

Reparameterization???

In RL (at least for discrete actions):
- T is a sequence of discrete actions
- p(T; ϕ) is not reparameterizable
- r(T) is a black box function
i.e. the environment

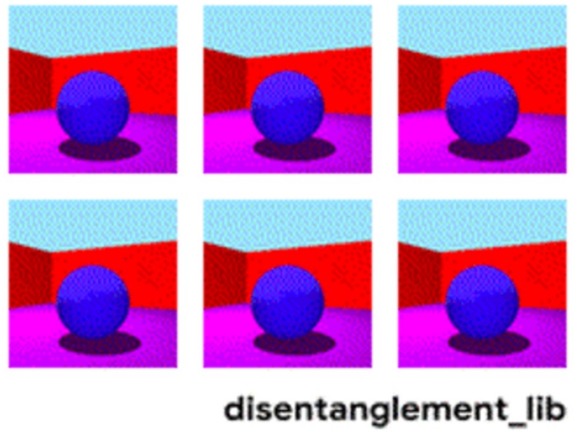
$$\begin{array}{c} \mathbf{r} \\ ???\ \uparrow \\ \mathbf{a} \\ \pi_\phi(a|s) \uparrow \\ \mathbf{s} \end{array}$$

REINFORCE is a general-purpose solution!

VAEs for Disentangled Generation

Disentangled representation learning

- Very useful for style transfer: disentangling **style** from **content**



From negative to positive
consistently slow . consistently good . consistently fast .
my goodness it was so gross . my husband 's steak was phenomenal . my goodness was so awesome .
it was super dry and had a weird taste to the entire slice . it was a great meal and the tacos were very kind of good . it was super flavorful and had a nice texture of the whole side .

[Locatello et al., Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. ICML 2019]

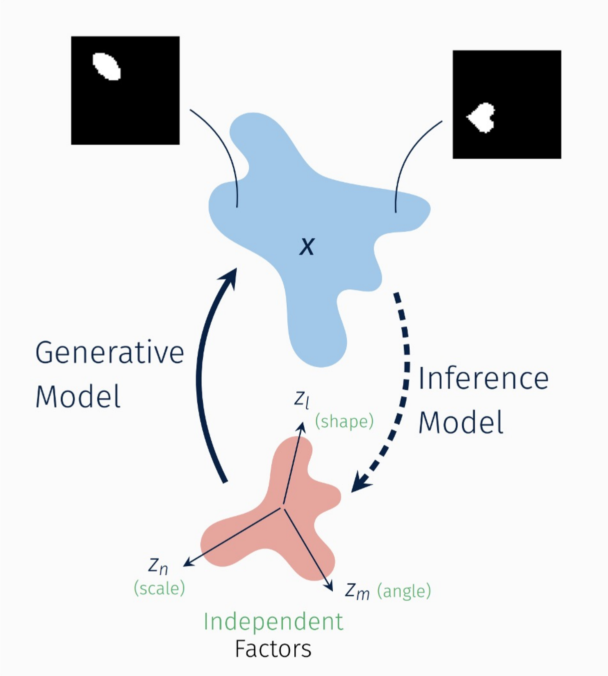
VAEs for Disentangled Generation

Disentangled representation learning

- Very useful for style transfer: disentangling **style** from **content**

$$\mathcal{L}_\beta(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta \cdot \mathrm{KL}(q_\phi(z|x)||p(z))$$

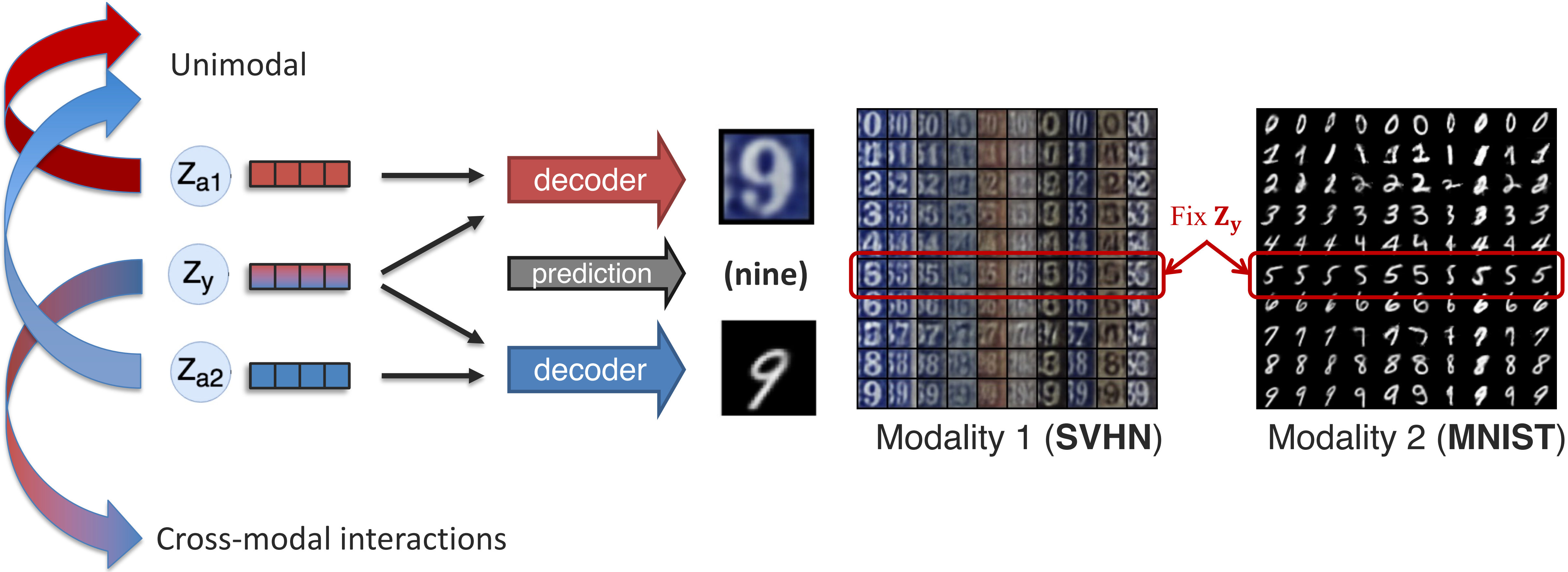
- beta-VAE: beta = 1 recovers VAE, beta > 1 imposes stronger constraint on the latent variables to have independent dimensions
- Difficult problem!
  - Positive results [Hu et al., 2016, Kulkarni et al., 2015]
  - Negative results [Mathieu et al., 2019, Locatello et al., 2019]
  - Better benchmarks & metrics to measure disentanglement [Higgins et al., 2017, Kim & Mnih 2018]



[Mathieu et al., Disentangling Disentanglement in Variational Autoencoders. ICML 2019]

VAEs for Multimodal Generation

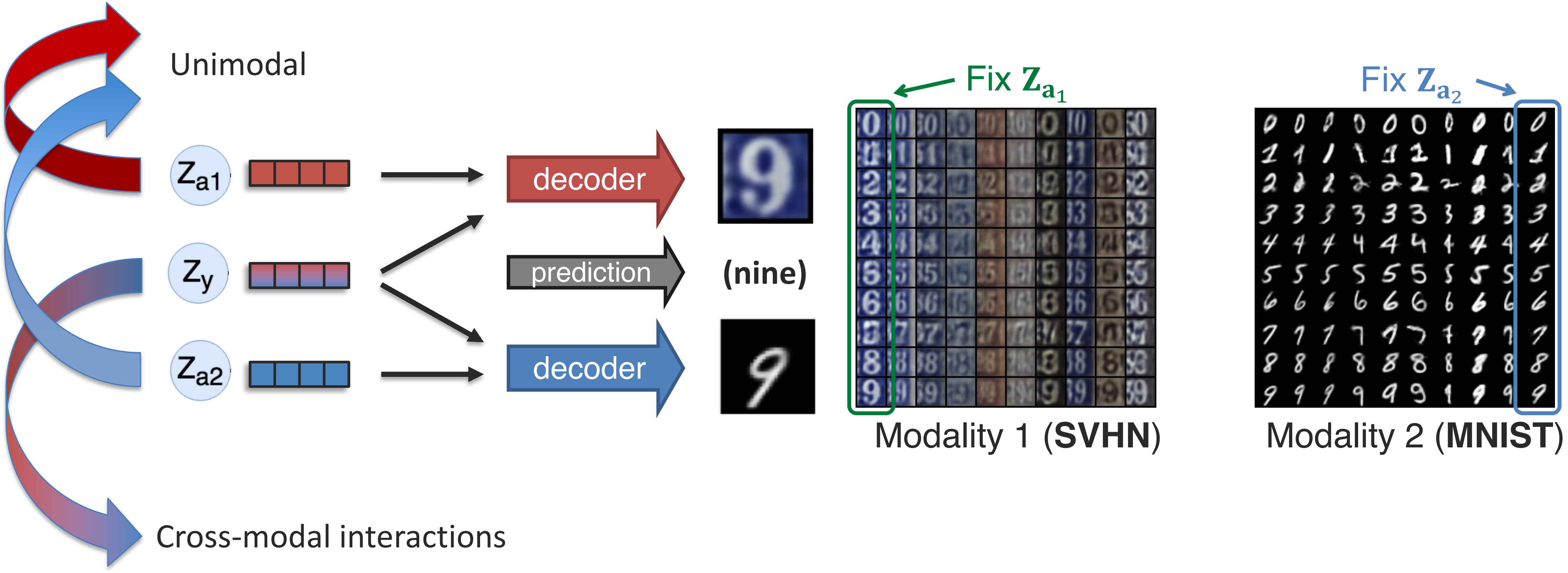
Some initial attempts: factorized generation



[Tsai et al., Learning Factorized Multimodal Representations. ICLR 2019]

VAEs for Multimodal Generation

Some initial attempts: factorized generation

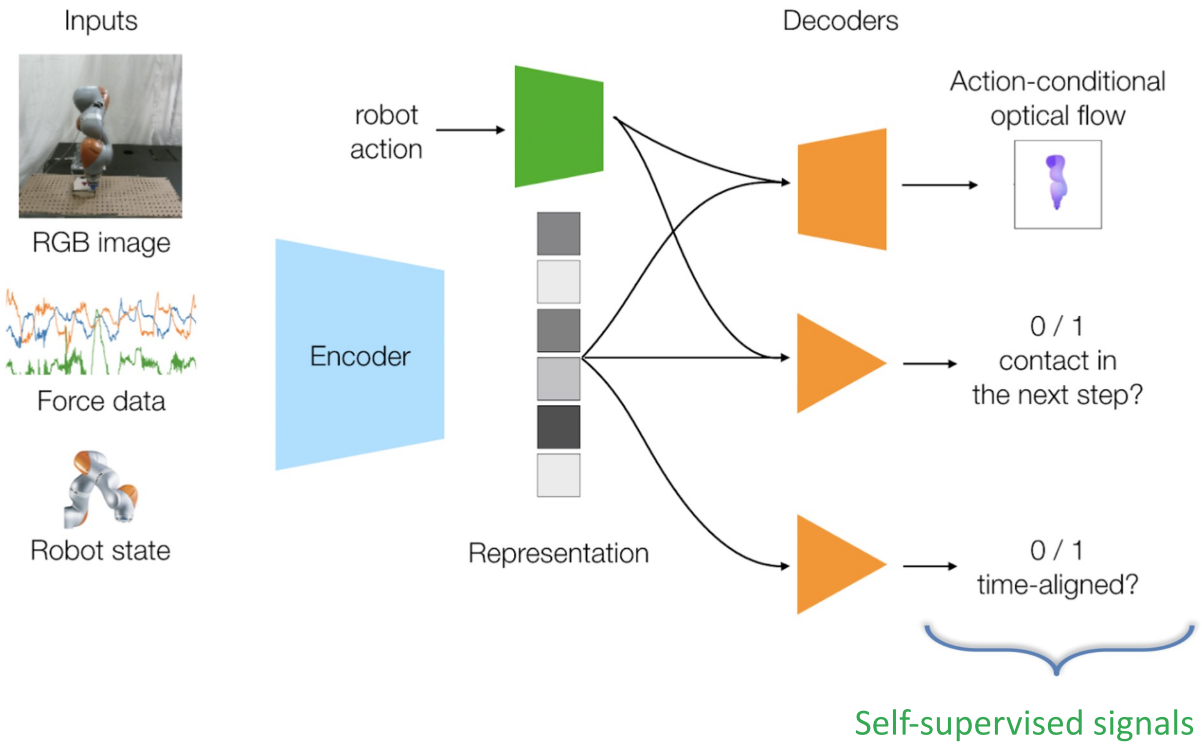


[Tsai et al., Learning Factorized Multimodal Representations. ICLR 2019]

VAEs for Multimodal Representations

VAEs beyond reconstruction

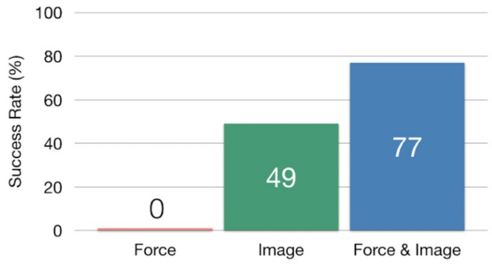
- It can be hard to reconstruct high-dimensional input modalities
- Combine VAEs with self-supervised learning: reconstruct **important signals** from the input



[Lee et al., Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks. ICRA 2019]

VAEs for Multimodal Representations

High success rate from multimodal signals

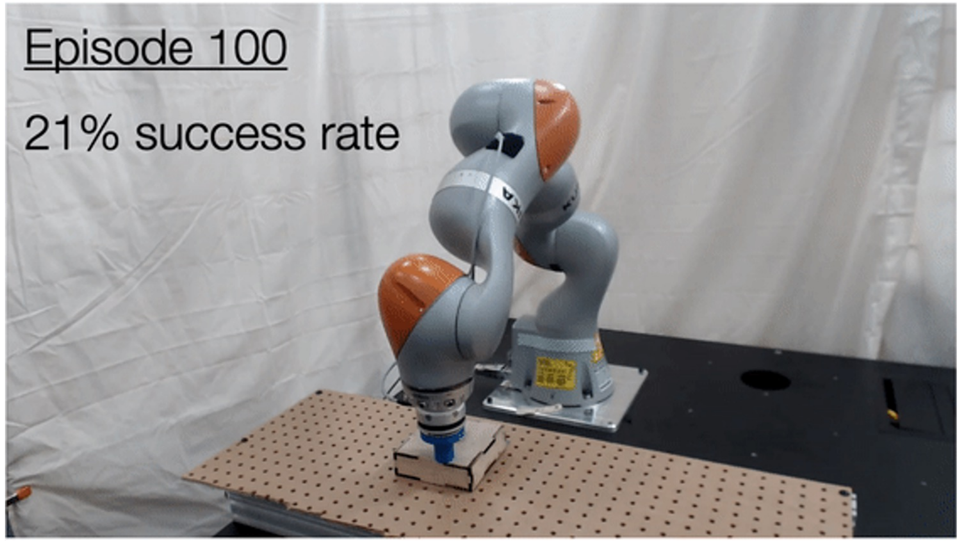


Simulation Results
(Randomized box location)

Force Only: Can't find box

Image Only: Struggles with peg alignment

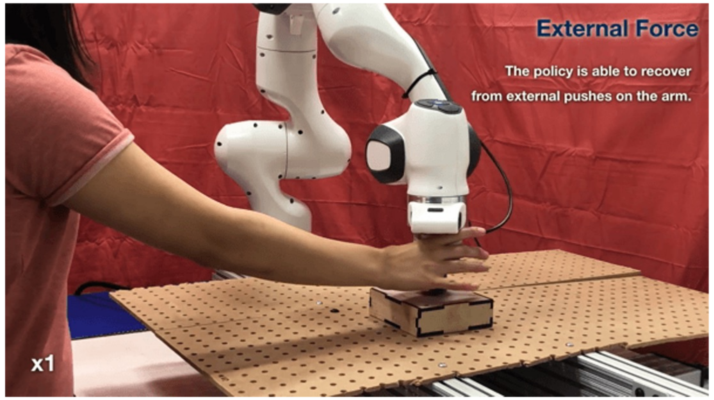
Force & Image: Can learn full task completion



[Lee et al., Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks. ICRA 2019]

VAEs for Multimodal Representations

Robustness to:
- external forces
- camera occlusion
- moving targets

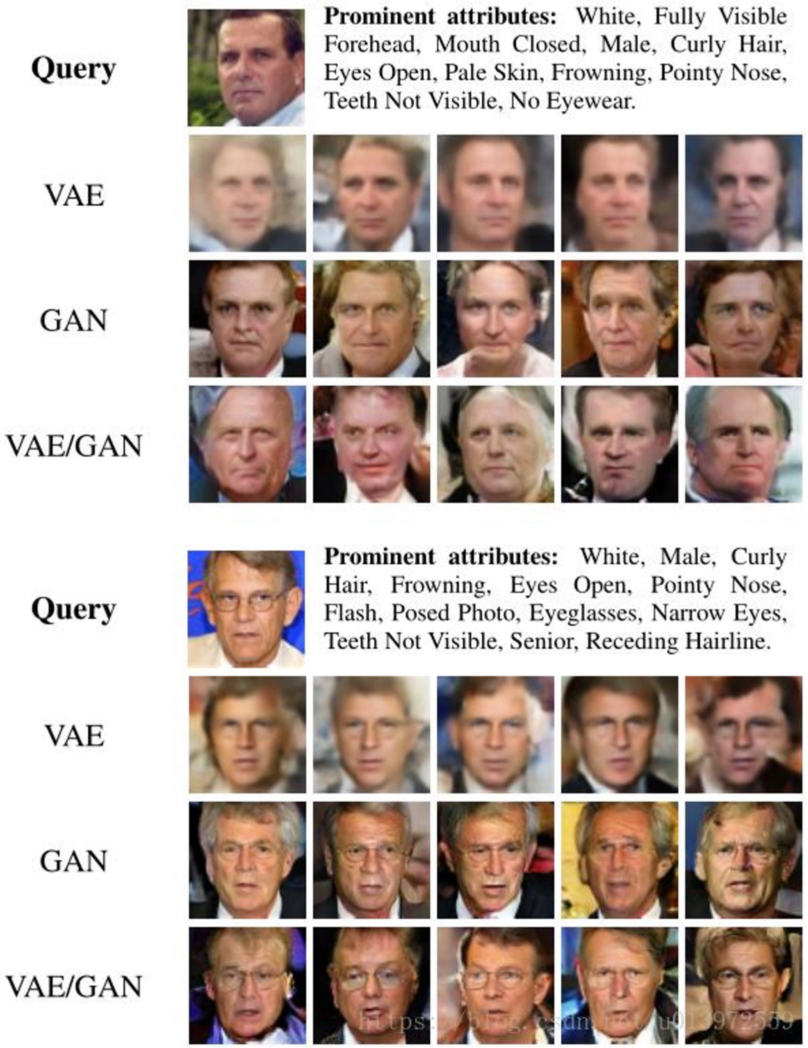


[Lee et al., Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks. ICRA 2019]

Summary: Variational Autoencoders

- Relatively easy to train.
- Explicit inference network q(z|x).

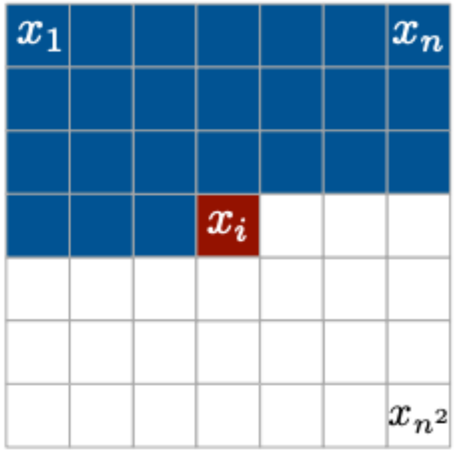
- More blurry images (due to reconstruction).



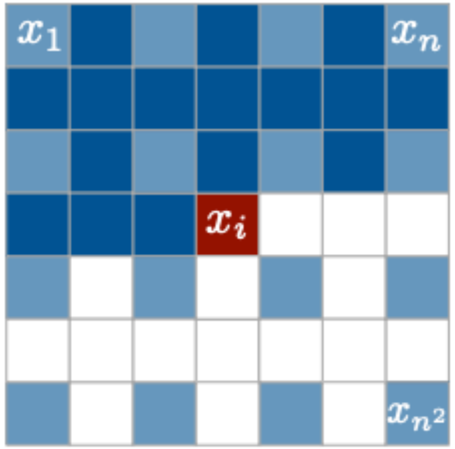
More Likelihood-based Models: Autoregressive Models

Autoregressive models

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$



Context



Multi-scale context

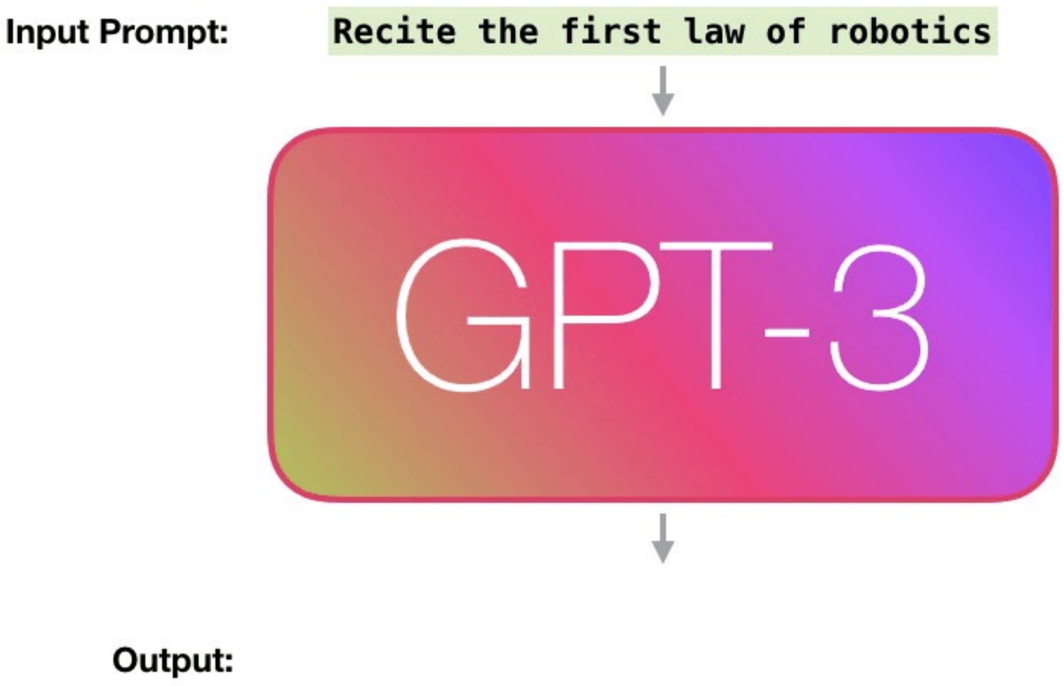
Figure 1. Image completions sampled from a PixelRNN.

[van den Oord et al., Pixel Recurrent Neural Networks. ICML 2016]

Autoregressive Models

Autoregressive language models

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

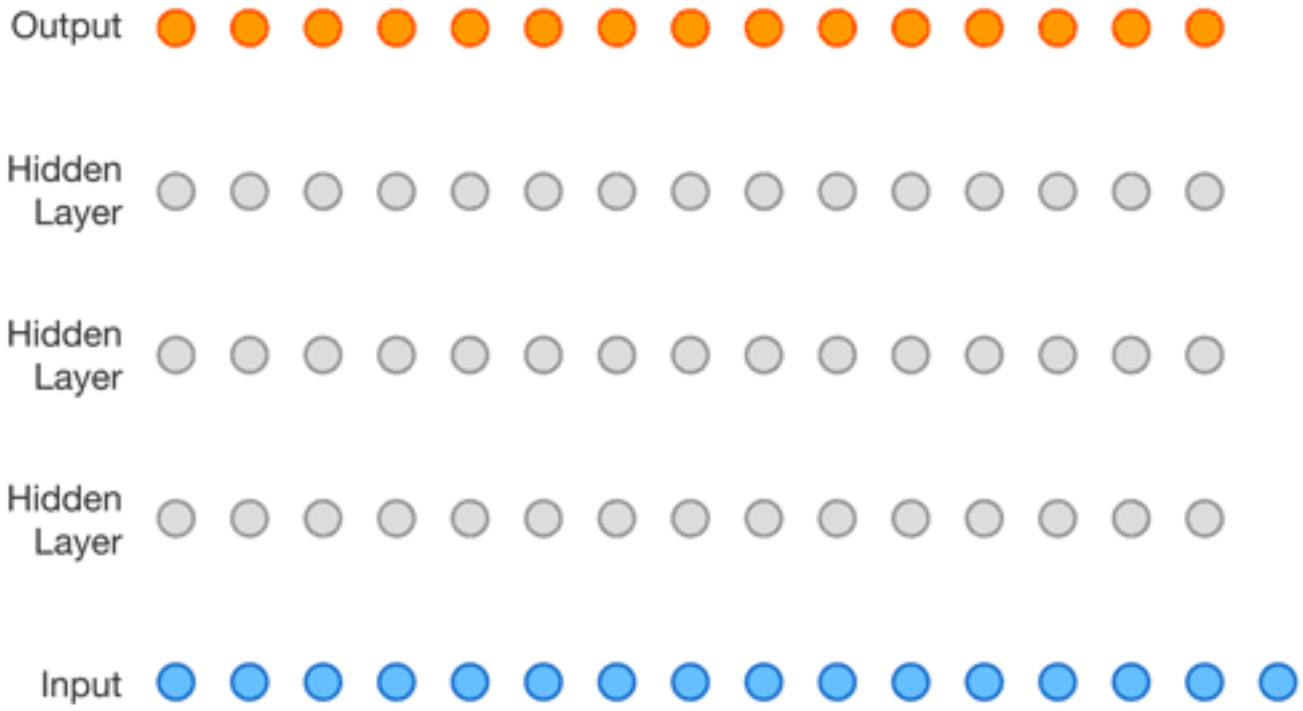


[Brown et al., Language Models are Few-shot Learners. NeurIPS 2020]

Autoregressive Models

Autoregressive audio generation models

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$



[van den Oord et al., WaveNet: A Generative Model for Raw Audio. ICML 2016]

Conditioning Autoregressive Models

We typically want p(x|c) - **conditional generation**

- c is a category (e.g. faces, outdoor scenes) from which we want to generate images
- c is an image which we want to describe in natural language

We might also care about p(x2|x1,c) - **style transfer**

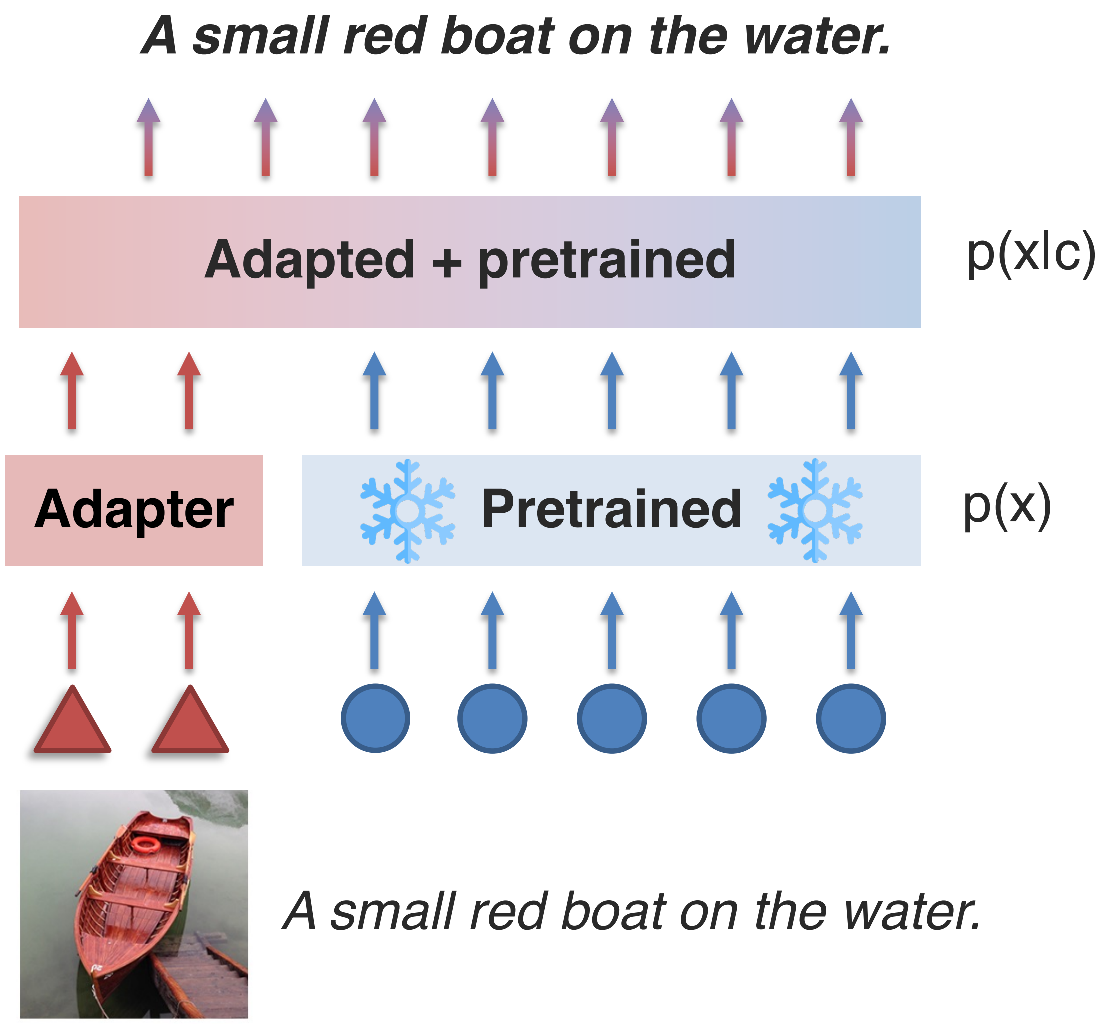
- c is a stylistic change e.g. negative to positive

From negative to positive
consistently slow . consistently good . consistently fast .
my goodness it was so gross . my husband 's steak was phenomenal . my goodness was so awesome .
it was super dry and had a weird taste to the entire slice . it was a great meal and the tacos were very kind of good . it was super flavorful and had a nice texture of the whole side .

Conditioning Autoregressive Models

Conditioning via prefix tuning

Modeling p(x|c):

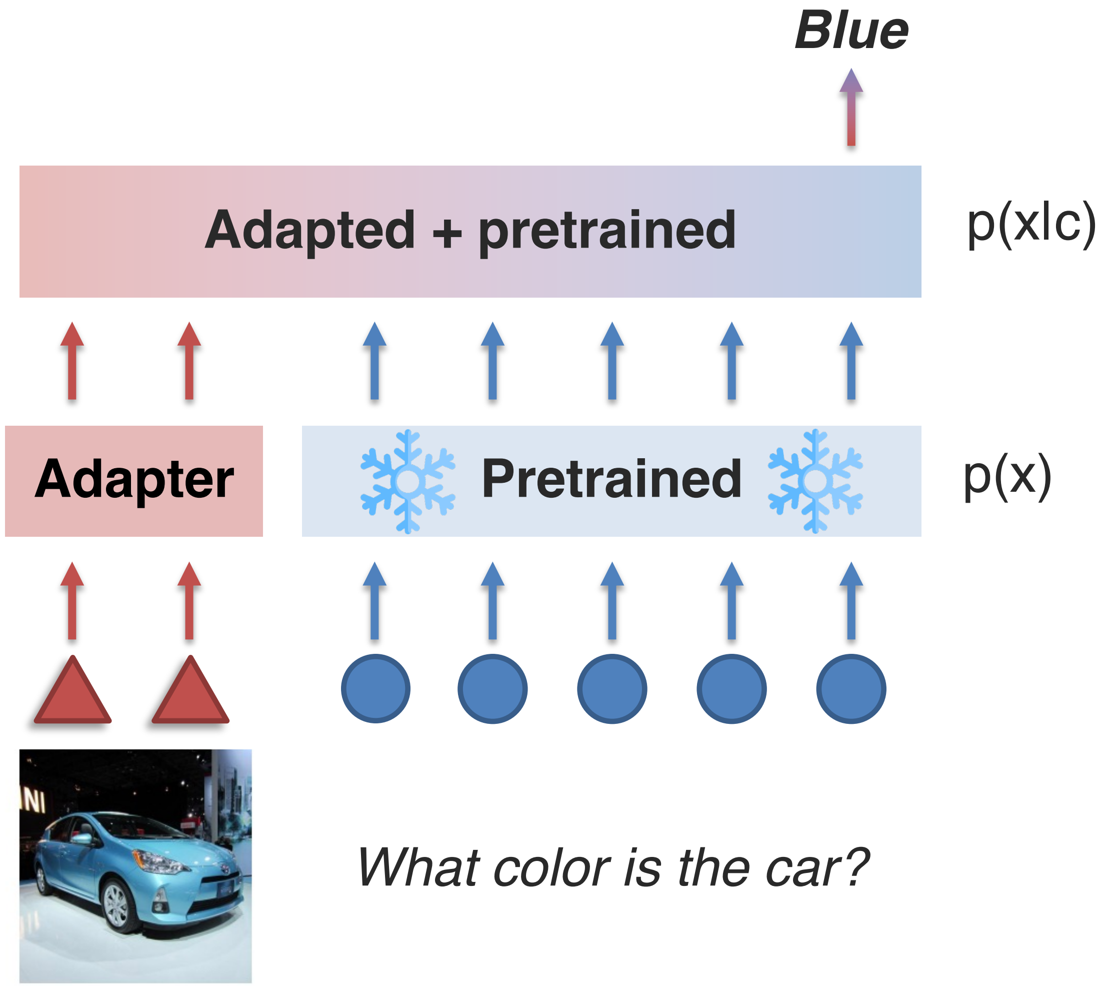


[Tsimpoukelli et al., Multimodal Few-Shot Learning with Frozen Language Models. NeurIPS 2021]

Conditioning Autoregressive Models

Conditioning via prefix tuning

0-shot VQA:



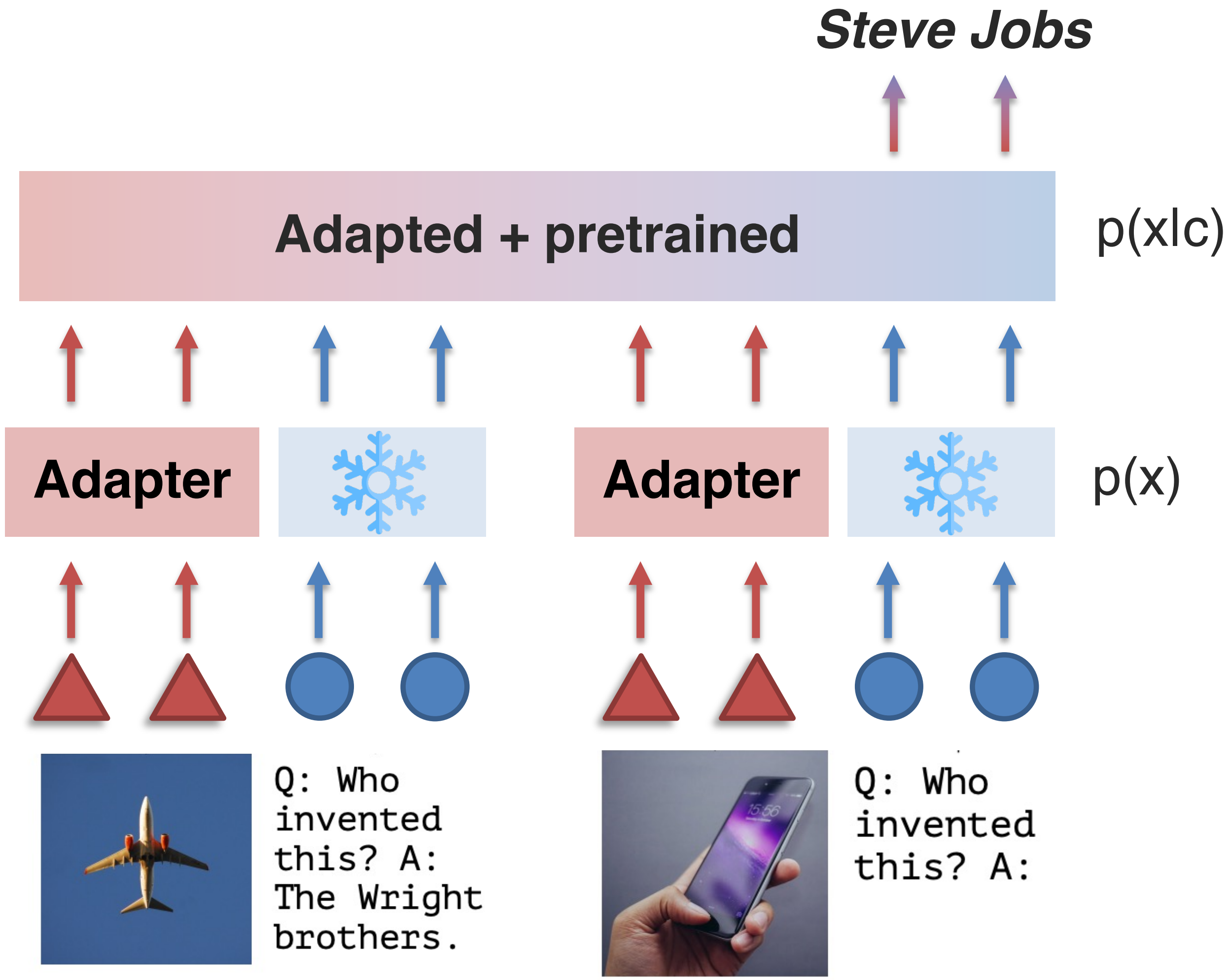
[Tsimpoukelli et al., Multimodal Few-Shot Learning with Frozen Language Models. NeurIPS 2021]

Conditioning Autoregressive Models

Conditioning via prefix tuning

1-shot outside knowledge VQA:

Recall reasoning – leverage implicit knowledge in LMs

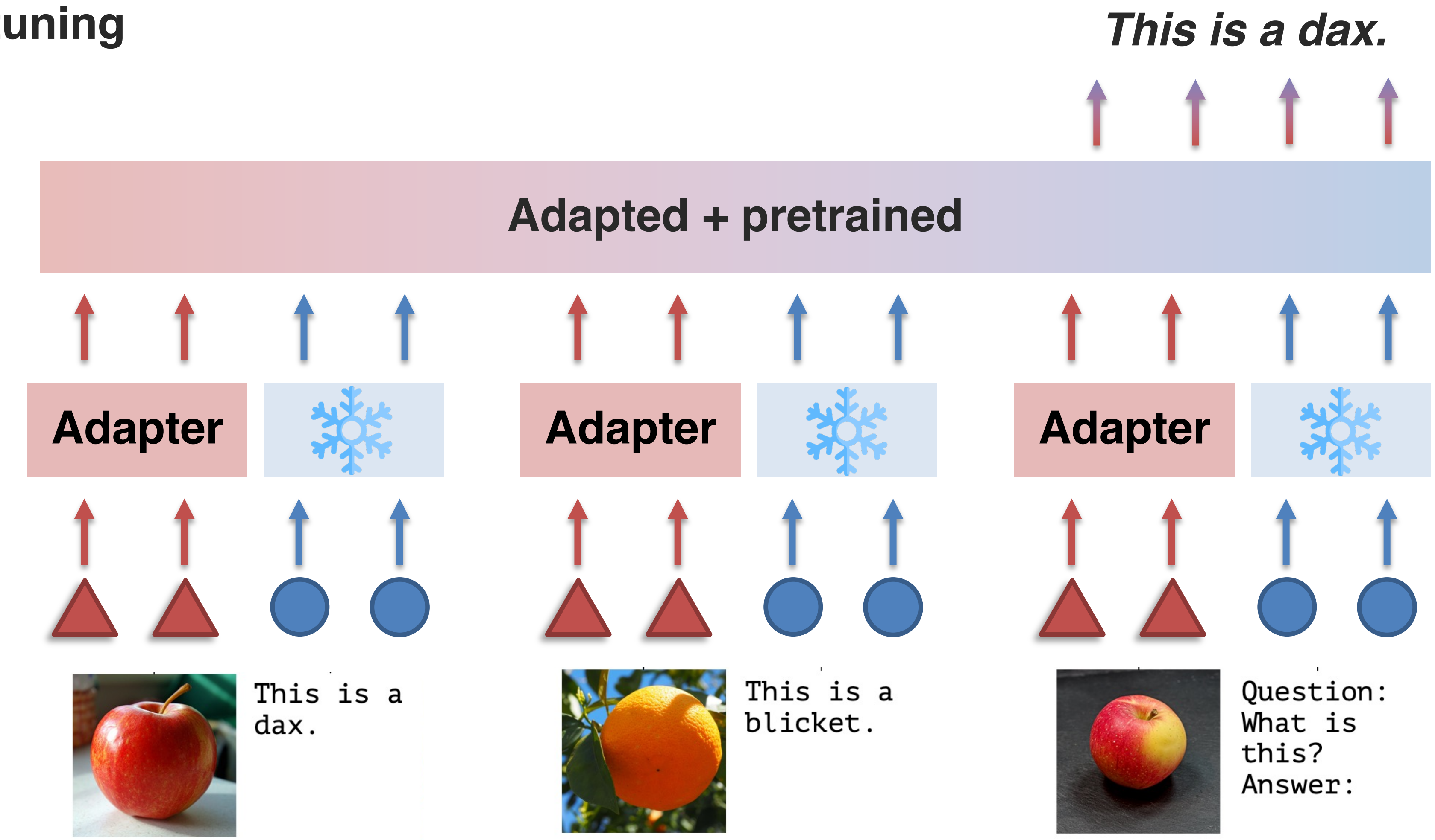


[Tsimpoukelli et al., Multimodal Few-Shot Learning with Frozen Language Models. NeurIPS 2021]

Conditioning Autoregressive Models

Conditioning via prefix tuning

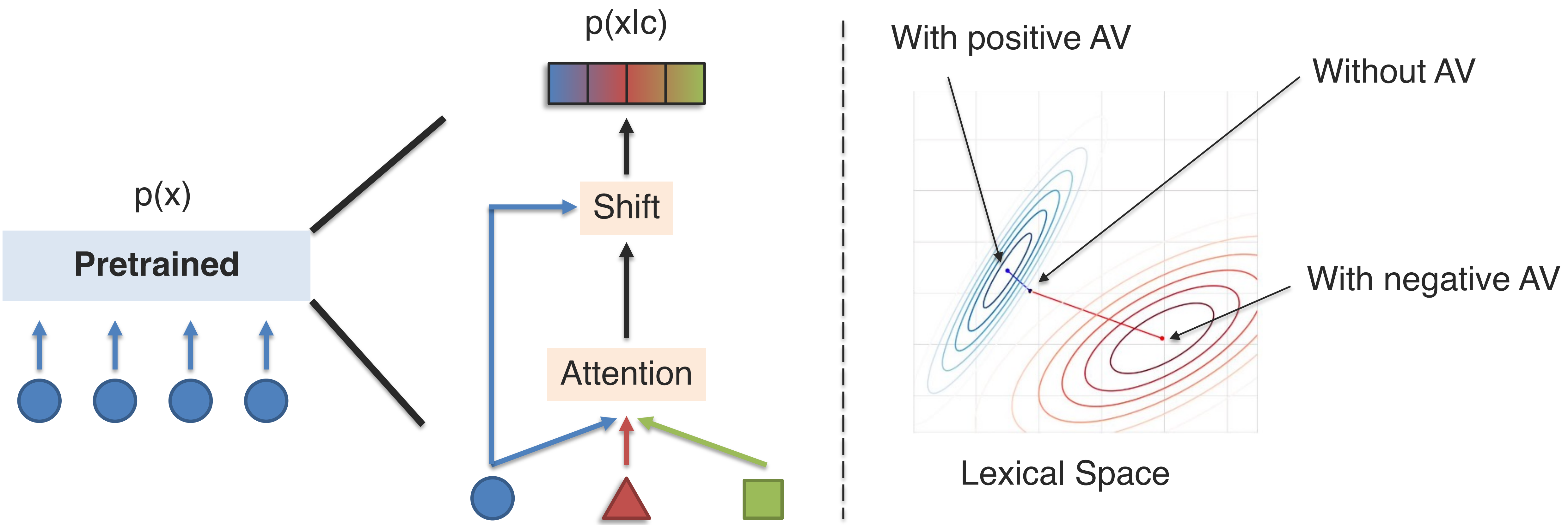
Few-shot image classification:



[Tsimpoukelli et al., Multimodal Few-Shot Learning with Frozen Language Models. NeurIPS 2021]

Conditioning Autoregressive Models

Conditioning via representation tuning



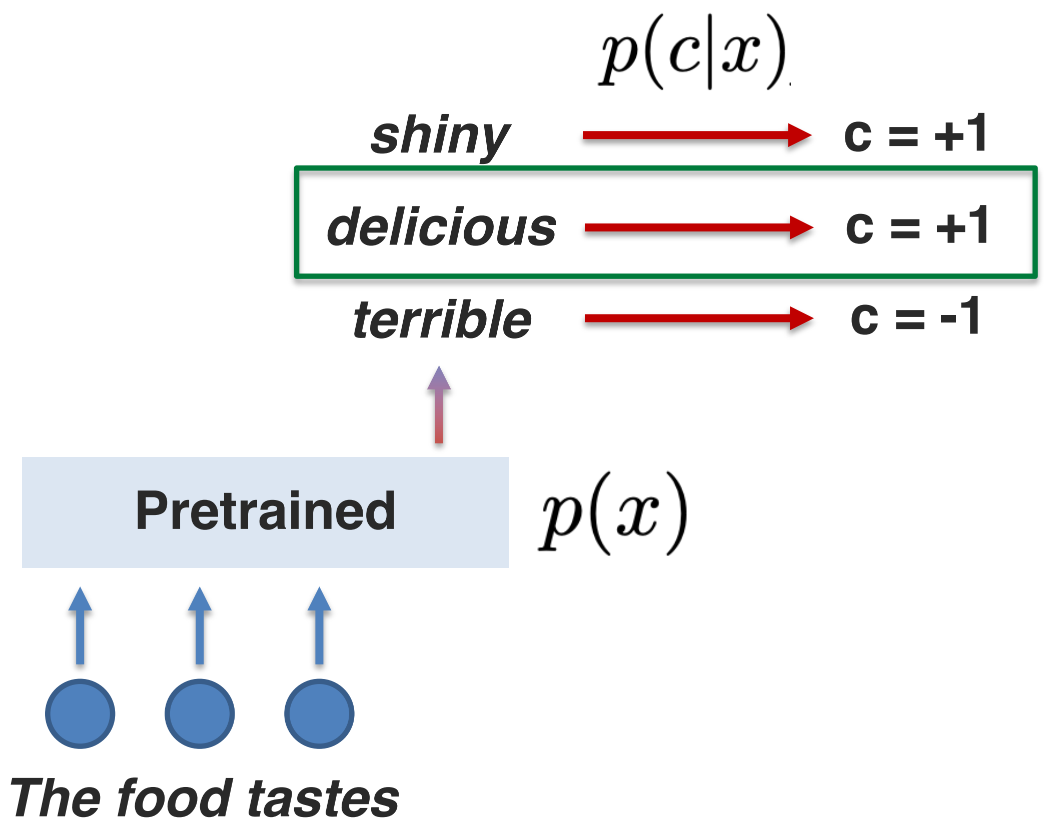
[Ziegler et al., Encoder-Agnostic Adaptation for Conditional Language Generation. arXiv 2019]

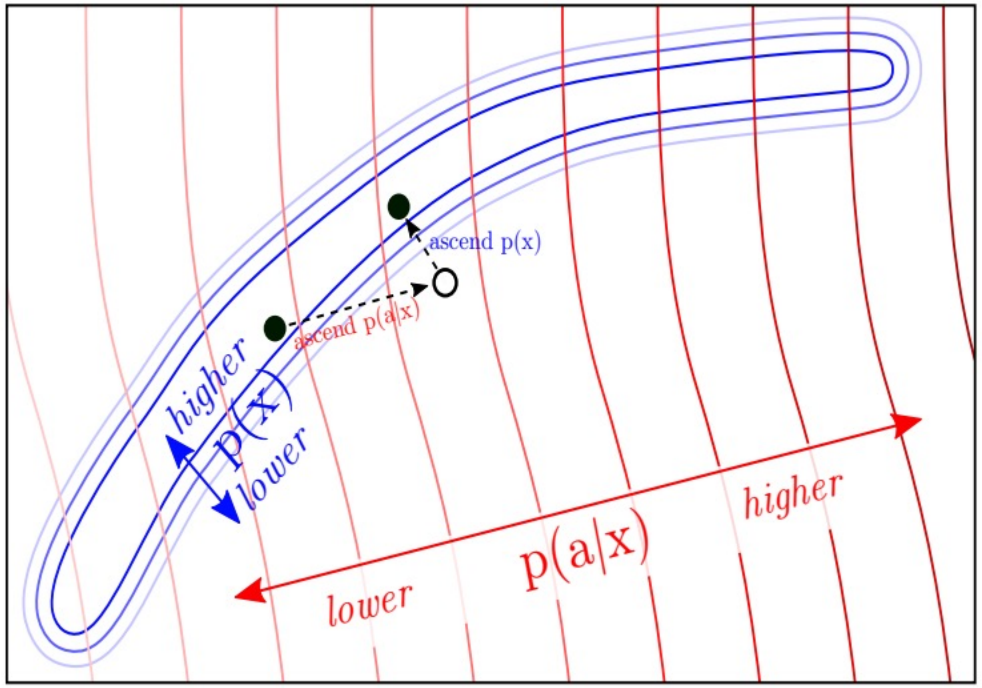
[Rahman et al., Integrating Multimodal Information in Large Pretrained Transformers. ACL 2020]

Conditioning Autoregressive Models

Conditioning via gradient tuning

$$p(x|c) \propto p(c|x)p(x)$$

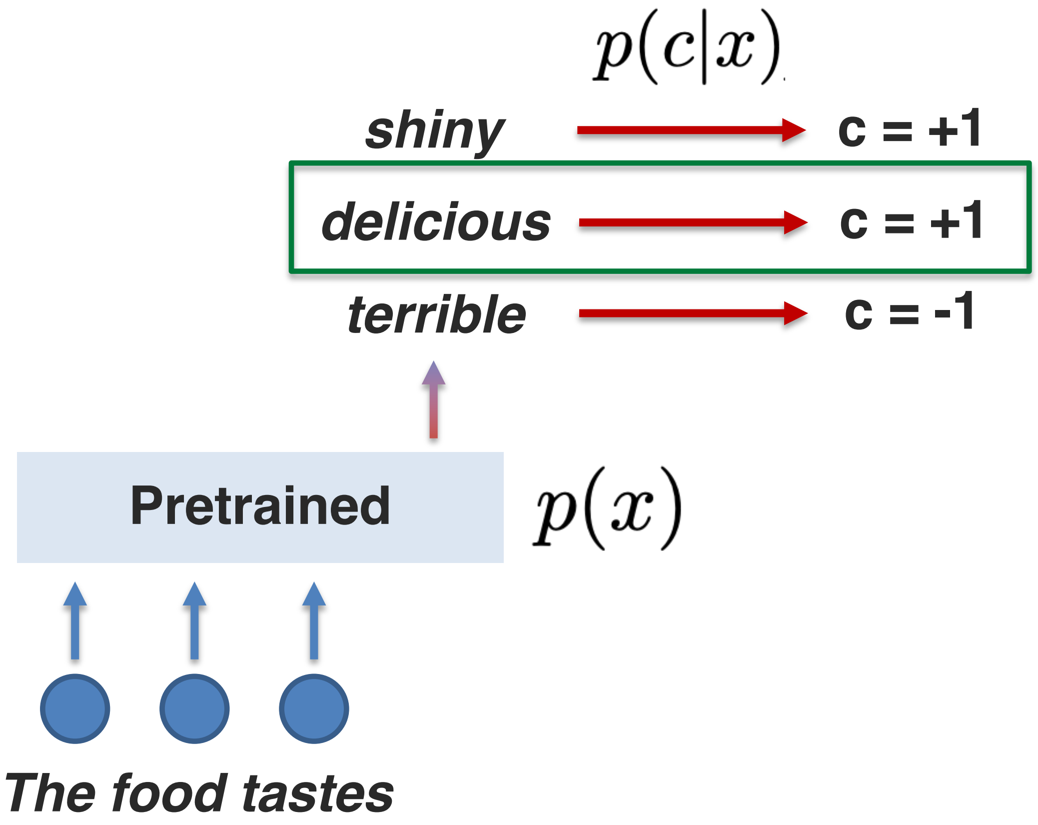




[Dathathri et al., Plug and Play Language Models: A Simple Approach to Controlled Text Generation. ICLR 2020]

Conditioning Autoregressive Models

Conditioning via gradient tuning



$$p(x|c) \propto p(c|x)p(x)$$

H_t are final-layer representations at time t

1. Increasing $p(c|x)$

$$\Delta H_t \leftarrow \Delta H_t + \alpha \nabla_{\Delta H_t} \log p(c|H_t + \Delta H_t)$$

2. Increasing $p(x)$

$$\Delta H_t \leftarrow \Delta H_t + \alpha \lambda \mathrm{KL}(p(x)||p_{\Delta H_t}(x))$$

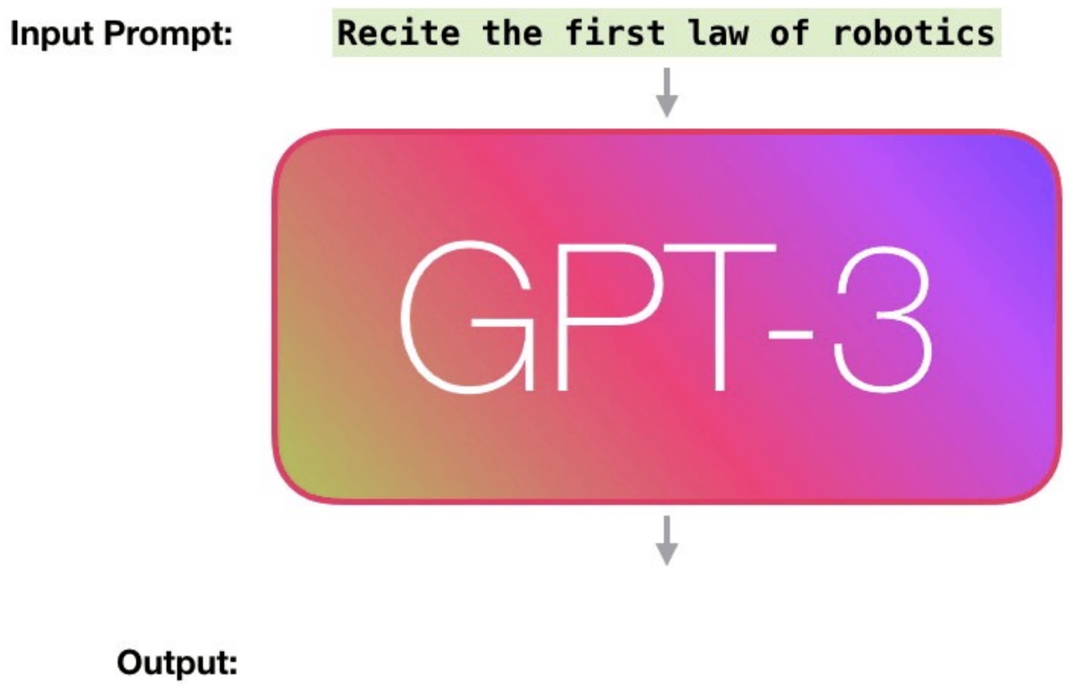
3. Generate next token using $H_t + \Delta H_t$

[Dathathri et al., Plug and Play Language Models: A Simple Approach to Controlled Text Generation. ICLR 2020]

Summary: Autoregressive Models

- Relatively easy to train.

- Slow to sample from.
- Not easy to condition on.

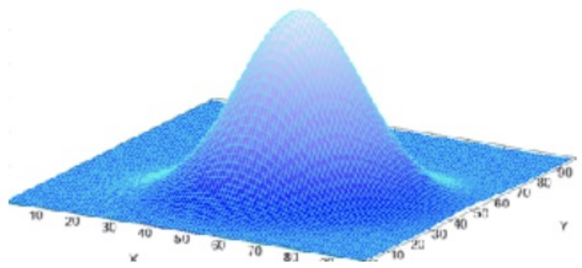


Normalizing Flows

Model families so far:

- **Autoregressive models** provide tractable likelihoods but no direct mechanism for learning features.
- **Variational autoencoders** can learn feature representations (via latent variables z) but have intractable marginal likelihoods.

Can we do both?



$$X = f(Z)$$

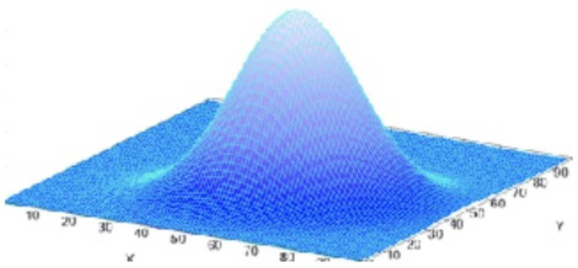
$$Z = f^{-1}(X)$$

$$Z \sim N(0, I)$$

$$X \sim P(X)$$

Change of Variables – 1D case

- Let X be a continuous random variable
- The cumulative density function (CDF) of X is $F_X(a) = P(X \leq a)$
- The probability density function (pdf) of X is $p_X(a) = F'_X(a) = \frac{dF_X(a)}{da}$
- Typically consider parameterized densities:
  - Gaussian: $X \sim \mathcal{N}(\mu, \sigma)$ if $p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$
  - Uniform: $X \sim \mathcal{U}(a, b)$ if $p_X(x) = \frac{1}{b-a} 1[a \leq x \leq b]$



$$X = f(Z)$$

$$Z = f^{-1}(X)$$

$Z \sim N(0, I)$

$X \sim P(X)$

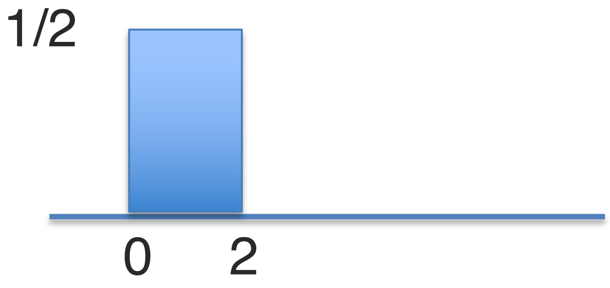
[Slides from Ermon and Song]

Change of Variables – 1D case

- Let Z be a uniform random variable $\mathcal{U}[0,2]$ with density p_Z. What is $p_Z(1)$? $\frac{1}{2}$
  - As a sanity check, $\int_0^2 \frac{1}{2} = 1$
- Let $X = 4Z$, and let p_X be its density. What is $p_X(4)$?

  Intuition: X should be uniform in [0,8], so $p_X(4) = 1/8$

- More interesting example: If $X = f(Z) = \exp(Z)$ and $Z \sim \mathcal{U}[0,2]$, what is $p_X(x)$?

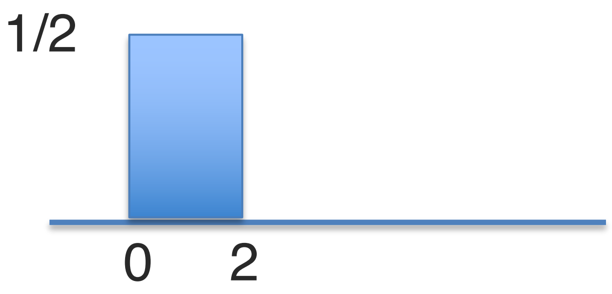


[Slides from Ermon and Song]

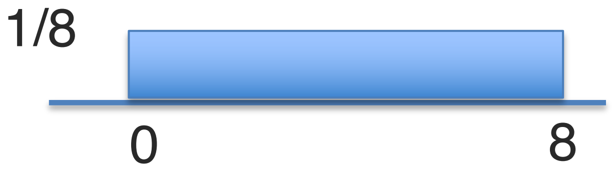
Change of Variables – 1D case

- **Change of variables (1D case)**: If $X = f(Z)$ and $f(\cdot)$ is monotone with inverse $Z = f^{-1}(X) = h(X)$, then:

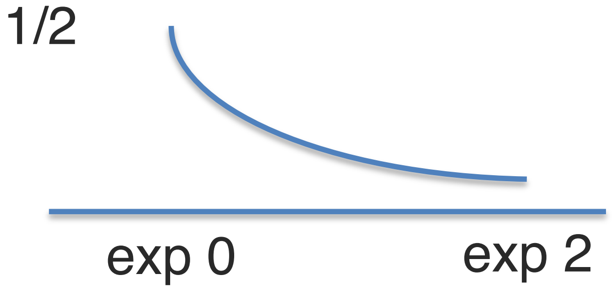
$$p_X(x) = p_Z(h(x))|h'(x)|$$



- Previous example: If $X = f(Z) = 4Z$ and $Z \sim \mathcal{U}[0, 2]$, what is $p_X(4)$?
  - Note that $h(X) = X/4$
  - $p_X(4) = p_Z(1)h'(4) = 1/2 \times |1/4| = 1/8$



- More interesting example: If $X = f(Z) = \exp(Z)$ and $Z \sim \mathcal{U}[0, 2]$, what is $p_X(x)$?
  - Note that $h(X) = \ln(X)$
  - $p_X(x) = p_Z(\ln(x))|h'(x)| = \frac{1}{2x}$ for $x \in [\exp(0), \exp(2)]$
- Note that the "shape" of $p_X(x)$ is different (more complex) from that of the prior $p_Z(z)$.



[Slides from Ermon and Song]

Change of Variables – higher D case

Let Z be a vector in [0,1] x [0,1]
Let X = AZ for a square invertible matrix A, with inverse W. How is X distributed?
Geometrically, the matrix A maps the unit square [0, 1] x [0,1] to a parallelogram.

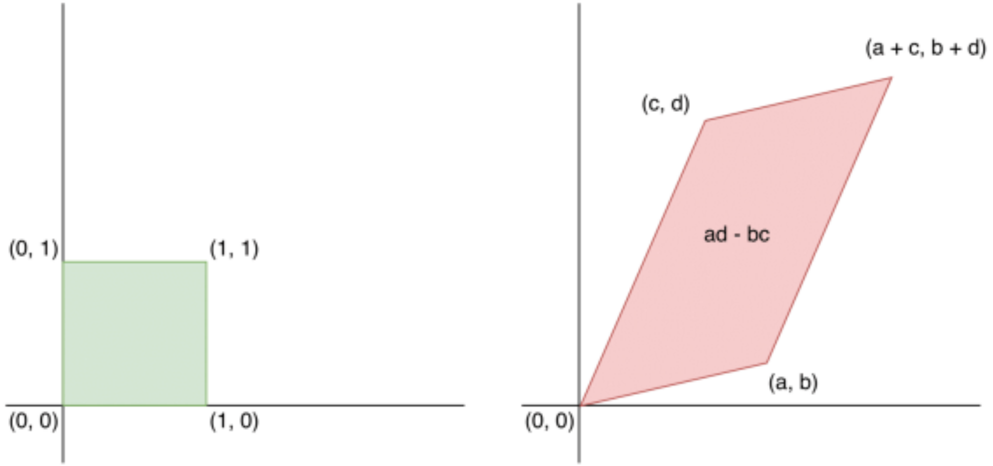


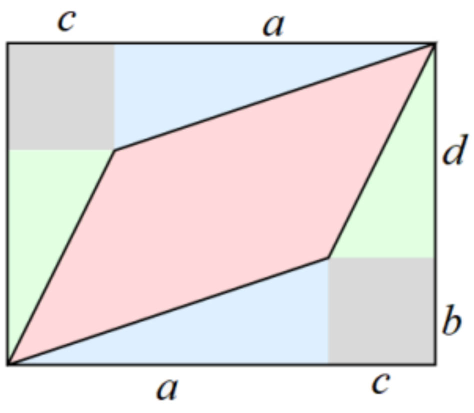
Figure: The matrix $A = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ maps a unit square to a parallelogram

[Slides from Ermon and Song]

Change of Variables – higher D case

- The volume of the parallelotope is equal to the absolute value of the determinant of the matrix A

$$\det(A) = \det\begin{pmatrix} a & c \\ b & d \end{pmatrix} = ad - bc$$



$(a+c)(b+d) - ab - 2bc - cd = ad - bc$

- Let $X = AZ$ for a square invertible matrix A, with inverse $W = A^{-1}$. X is uniformly distributed over the parallelotope of area $|\det(A)|$. Hence, we have

$$p_X(\mathrm{x}) = p_Z(W\mathrm{x}) \,/\, |\det(A)|$$
$$= p_Z(W\mathrm{x})\,|\det(W)|$$

because if $W = A^{-1}$, $\det(W) = \frac{1}{\det(A)}$. Note similarity with 1D case $p_X(x) = p_Z(h(x))|h'(x)|$

[Slides from Ermon and Song]

Change of Variables – higher D case

- For linear transformations specified via A, change in volume is given by the determinant of A
- For non-linear transformations $\mathrm{f}(\cdot)$, the *linearized* change in volume is given by the determinant of the Jacobian of $\mathrm{f}(\cdot)$.
- **Change of variables (General case)**: The mapping between Z and X, given by $\mathrm{f} : \mathbb{R}^n \mapsto \mathbb{R}^n$, is invertible such that $X = \mathrm{f}(Z)$ and $Z = \mathrm{f}^{-1}(X)$.

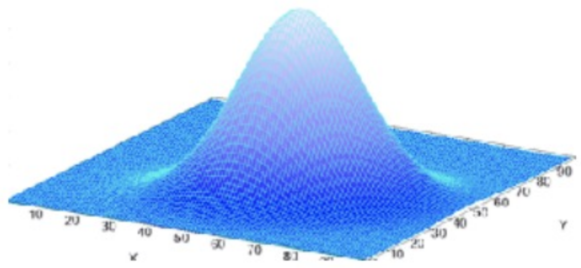
$$p_X(\mathrm{x}) = p_Z\left(\mathrm{f}^{-1}(\mathrm{x})\right)\left|\det\left(\frac{\partial \mathrm{f}^{-1}(\mathrm{x})}{\partial \mathrm{x}}\right)\right|$$

- Note 0: generalizes the previous 1D case $p_X(x) = p_Z(h(x))|h'(x)|$
- Note 1: unlike VAEs, x, z need to be continuous and have the same dimension. For example, if $\mathrm{x} \in \mathbb{R}^n$ then $\mathrm{z} \in \mathbb{R}^n$
- Note 2: For any invertible matrix A, $det(A^{-1}) = det(A)^{-1}$

$$p_X(\mathrm{x}) = p_Z(\mathrm{z})\left|\det\left(\frac{\partial \mathrm{f}(\mathrm{z})}{\partial \mathrm{z}}\right)\right|^{-1}$$

[Slides from Ermon and Song]

Normalizing Flows



$$X = f(Z)$$

$$Z = f^{-1}(X)$$

$Z \sim N(0, I)$ $\qquad$ $X \sim P(X)$

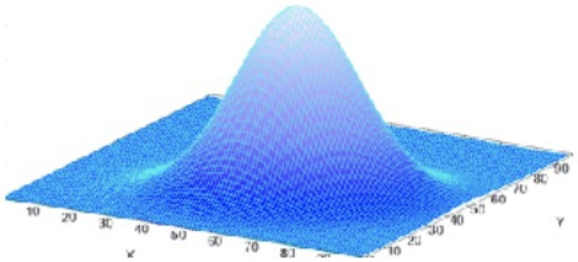
- Learning via **maximum likelihood** over the dataset $\mathcal{D}$

$$\max_{\theta} \log p_X(\mathcal{D}; \theta) = \sum_{x \in \mathcal{D}} \log p_Z\left(f_\theta^{-1}(x)\right) + \log \left| \det \left(\frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|$$

- **Exact likelihood evaluation** via inverse tranformation $x \mapsto z$ and change of variables formula

[Slides from Ermon and Song]

Normalizing Flows



$$X = f(Z)$$

$$Z = f^{-1}(X)$$

$Z \sim N(0, I)$ $\qquad$ $X \sim P(X)$

- **Sampling** via forward transformation $z \mapsto x$

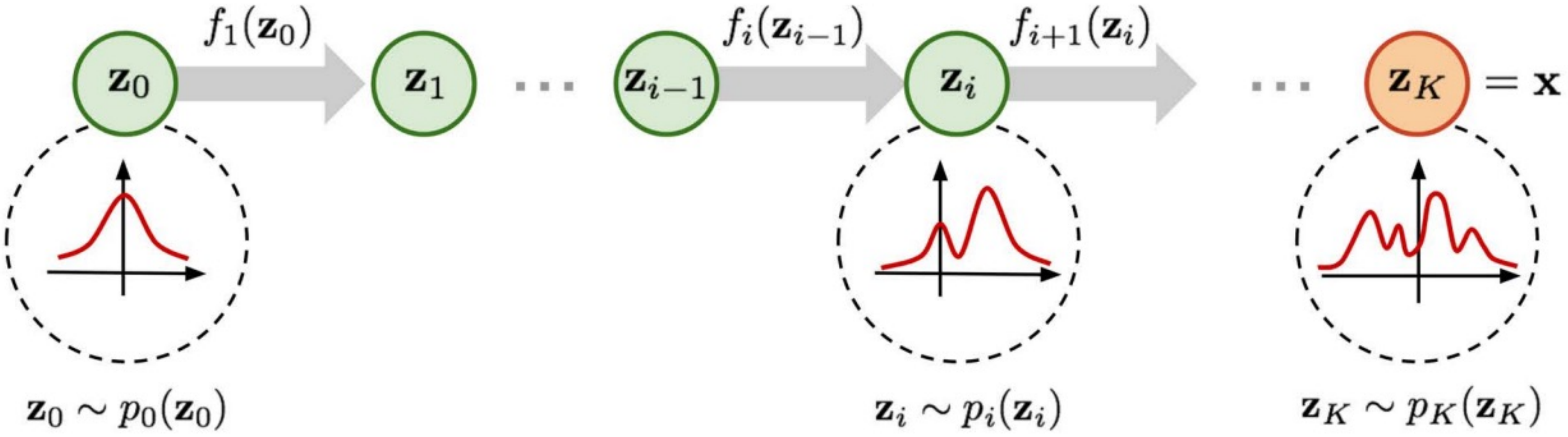
$$z \sim p_Z(z) \quad x = f_\theta(z)$$

- **Latent representations** inferred via inverse transformation (no inference network required!)

$$z = f_\theta^{-1}(x)$$

[Slides from Ermon and Song]

Normalizing Flows



$X \sim P(X)$

$$\boldsymbol{z}_0 \sim p(\boldsymbol{z}_0)$$
$$\boldsymbol{x} = \boldsymbol{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\boldsymbol{z}_0)$$

inference: $\boldsymbol{z}_i = f_i^{-1}(\boldsymbol{z}_{i-1})$

density: $p(\boldsymbol{z}_i) = p(\boldsymbol{z}_{i-1}) \left| \det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i} \right|$

training: maximizes data log-likelihood

$$\log p(\boldsymbol{x}) = \log p(\boldsymbol{z}_0) + \sum_{i=1}^{K} \log \left| \det \frac{d\boldsymbol{z}_{i-1}}{d\boldsymbol{z}_i} \right|$$

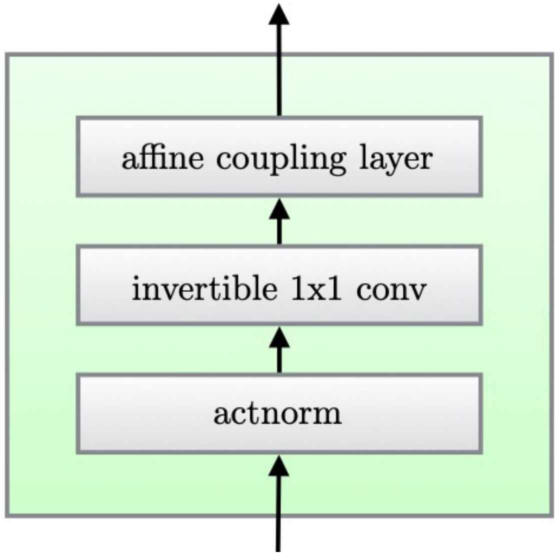
[Slides from Eric Xing]

Normalizing Flows Tips

- Simple prior $p_Z(\mathrm{z})$ that allows for efficient sampling and tractable likelihood evaluation. E.g., isotropic Gaussian
- Invertible transformations with tractable evaluation:
  - Likelihood evaluation requires efficient evaluation of $\mathrm{x} \mapsto \mathrm{z}$ mapping
  - Sampling requires efficient evaluation of $\mathrm{z} \mapsto \mathrm{x}$ mapping
- Computing likelihoods also requires the evaluation of determinants of $n \times n$ Jacobian matrices, where n is the data dimensionality
  - Computing the determinant for an $n \times n$ matrix is $O(n^3)$: prohibitively expensive within a learning loop!
  - **Key idea**: Choose tranformations so that the resulting Jacobian matrix has special structure. For example, the determinant of a triangular matrix is the product of the diagonal entries, i.e., an $O(n)$ operation

[Slides from Ermon and Song]

Normalizing Flows



Description	Function	Reverse Function	Log-determinant
Actnorm. See Section 3.1.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \mathtt{sum}(\log \lvert\mathbf{s}\rvert)$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \mathtt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \mathtt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\mathtt{sum}(\log(\lvert\mathbf{s}\rvert))$

[Kingma et al., Generative Flow with Invertible 1x1 Convolutions. NeurIPS 2018]

Summary: Normalizing Flows

- Relatively easy to train.
- Exact likelihood.

- Very constrained architecture.

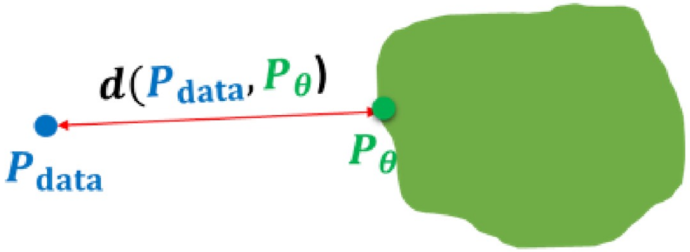
Work combining VAEs, autoregressive models, and flow-based models, see https://lilianweng.github.io/posts/2018-10-13-flow-models/

Generative Adversarial Networks

Beyond likelihood-based learning:

- Difficulty in evaluating and optimizing p(x) in high-dimensions
- High p(x) might not correspond to realistic samples

$x_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



$\theta \in M$

Model family

Generative Adversarial Networks

Towards likelihood-free learning

vs.

$S_1 = \{\mathbf{x} \sim P\}$ $S_2 = \{\mathbf{x} \sim Q\}$

Given a finite set of samples from two distributions, how can we tell if these samples are from the same distribution? (i.e. P = Q?)

Generative Adversarial Networks

Given $S_1 = \{\mathbf{x} \sim P\}$ and $S_2 = \{\mathbf{x} \sim Q\}$, a **two-sample test** considers the following hypotheses

- Null hypothesis H_0: $P = Q$
- Alternate hypothesis H_1: $P \neq Q$

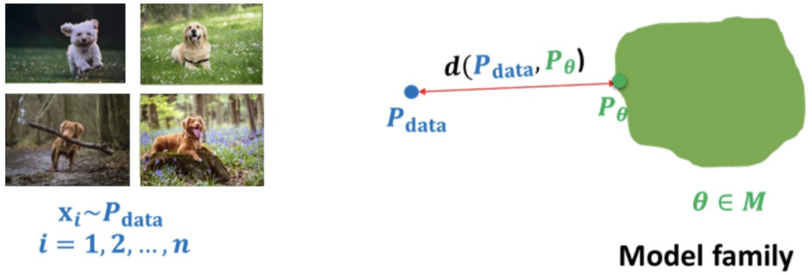
Test statistic T compares S_1 and S_2 e.g., difference in means, variances of the two sets of samples

If T is less than a threshold α, then accept H_0 else reject it

Key observation: Test statistic is **likelihood-free** since it does not involve the densities P or Q, only samples

Generative Adversarial Networks

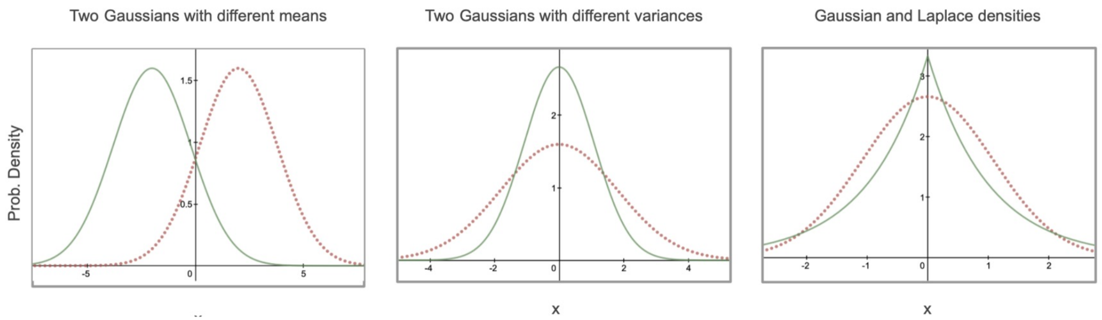
Towards likelihood-free learning



- Assume we have access to $S_1 = \mathcal{D} = \{\mathbf{x} \sim p_{\text{data}}\}$
- In addition, we have our model's distribution p_θ
- Assume that our model's distribution permits efficient sampling of $S_2 = \{\mathbf{x} \sim p_\theta\}$
- Train the generative model to minimize a two-sample test objective between S1 and S2

Generative Adversarial Networks

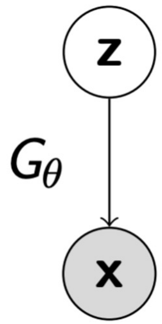
Towards likelihood-free learning



- Problem: finding a two-sample test objective in high-dimensions is hard
- In the generative model setup, we know that S1 and S2 come from different distributions p_{data} and p_θ respectively
- **Key idea: learn** a statistic that **maximizes** a suitable notion of distance between the two sets of samples S1 and S2

Generative Adversarial Networks

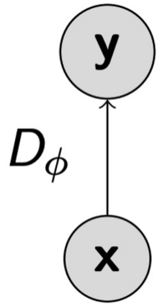
- A 2 player minimax game between a **generator** and a **discriminator**



- **Generator**: a directed latent variable model from z to x
- Minimizes the two-sample test objective: in support of null hypothesis $p_{\mathrm{data}} = p_\theta$

Generative Adversarial Networks

- A 2 player minimax game between a **generator** and a **discriminator**



- **Discriminator**: any function (e.g. neural network) that tries to distinguish 'real' samples from the datasets from 'fake' samples generated by the model
- Maximizes the two-sample test objective: in support of alternative hypothesis $p_{\mathrm{data}} \neq p_\theta$

Training the Discriminator

- Training objective for **discriminator**

$$\max_D V(G, D) = E_{\mathbf{x}\sim p_{\mathrm{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x}\sim p_G}[\log(1 - D(\mathbf{x}))]$$

- For a fixed generator G, the discriminator performs binary classification between true samples (assign label 1) vs fake samples (assign label 0)
- Optimal discriminator:

$$D_G^*(\mathbf{x}) = \frac{p_{\mathrm{data}}(\mathbf{x})}{p_{\mathrm{data}}(\mathbf{x}) + p_G(\mathbf{x})}$$

Training the Generator

- Training objective for **generator**

$$\min_G V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

- For the optimal discriminator $D_G^*(\cdot)$, we have

$$\begin{aligned}
& V(G, D_G^*(\mathbf{x})) \\
&= E_{\mathbf{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}\right] \\
&= E_{\mathbf{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] + E_{\mathbf{x} \sim p_G}\left[\log \frac{p_G(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}{2}}\right] - \log 4 \\
&= \underbrace{D_{KL}\left[p_{\text{data}}, \frac{p_{\text{data}} + p_G}{2}\right] + D_{KL}\left[p_G, \frac{p_{\text{data}} + p_G}{2}\right]}_{2\times\text{Jenson-Shannon Divergence (JSD)}} - \log 4 \\
&= 2D_{JSD}[p_{\text{data}}, p_G] - \log 4
\end{aligned}$$

[Slides from Ermon and Grover]

More About Divergences

- Also known as the **symmetric** KL divergence

$$D_{JSD}[p, q] = \frac{1}{2}\left(D_{KL}\left[p, \frac{p+q}{2}\right] + D_{KL}\left[q, \frac{p+q}{2}\right]\right)$$

- Properties
  - $D_{JSD}[p, q] \geq 0$
  - $D_{JSD}[p, q] = 0$ iff $p = q$
  - $D_{JSD}[p, q] = D_{JSD}[q, p]$
  - $\sqrt{D_{JSD}[p, q]}$ satisfies triangle inequality $\rightarrow$ Jenson-Shannon Distance
- Optimal generator for the JSD/Negative Cross Entropy GAN

$$p_G = p_{\mathrm{data}}$$

[Slides from Ermon and Grover]

GAN Training

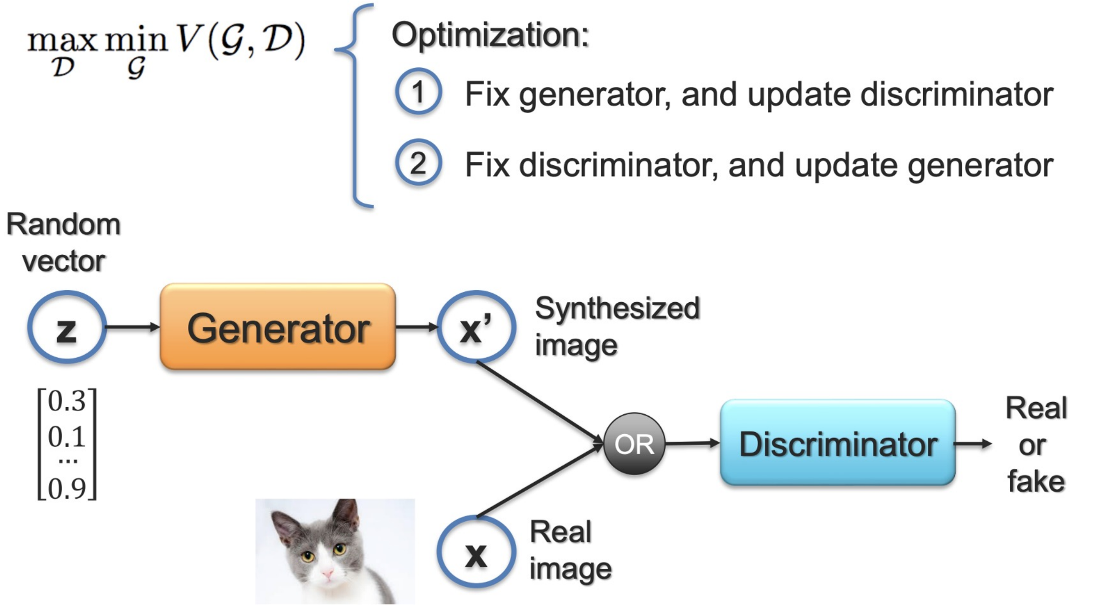
- Sample minibatch of m training points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ from $\mathcal{D}$
- Sample minibatch of m noise vectors $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}$ from p_z
- Update the generator parameters θ by stochastic gradient **descent**

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m} \nabla_\theta \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))$$

- Update the discriminator parameters ϕ by stochastic gradient **ascent**

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m} \nabla_\phi \sum_{i=1}^{m} [\log D_\phi(\mathbf{x}^{(i)}) + \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))]$$

- Repeat for fixed number of epochs



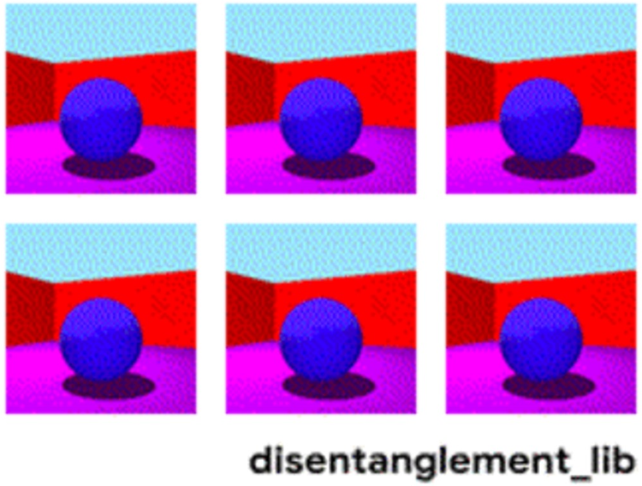
[Slides from Ermon and Grover]

Summary: Generative Models

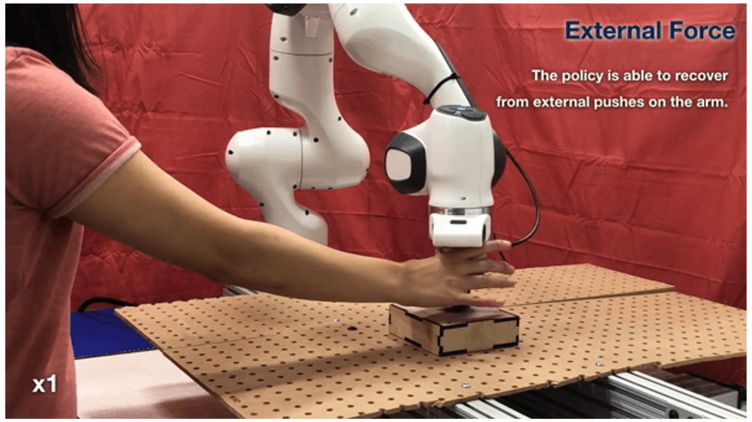
Likelihood-based		
1. VAEs – approximate inference via evidence lower bound	Fast & easy to train	Lower generation quality
2. Autoregressive models – exact inference via chain rule	Easy to train, exact likelihood	Slow to sample from
3. Flows – exact inference via invertible transformations	Easy to train, exact likelihood	Constrained architecture
Likelihood-free		
1. GANs – discriminative real vs generated samples	High generation quality	Hard to train, can't get features

One last model: diffusion models in next lecture.

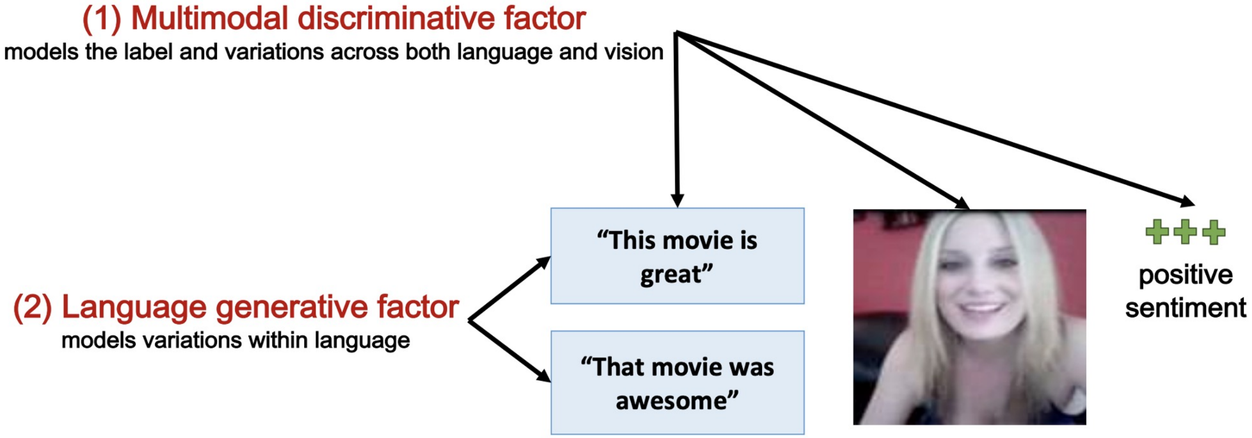
Summary



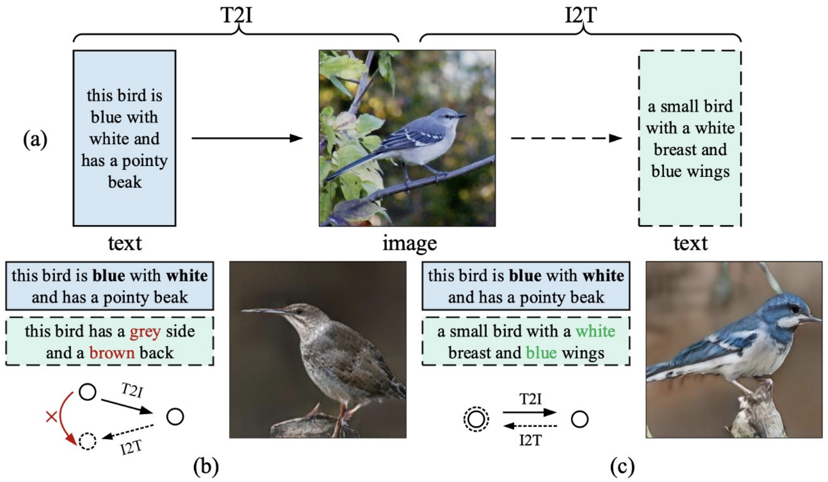
[Locatello et al., ICML 2019]



[Lee et al., ICRA 2019]



[Tsai et al., ICLR 2019]



[Qiao et al., CVPR 2019]

More Resources

https://lilianweng.github.io/tags/generative-model/
https://yang-song.net/blog/2021/score/
https://blog.evjang.com/2018/01/nf1.html & https://blog.evjang.com/2018/01/nf2.html
https://deepgenerativemodels.github.io/syllabus.html
https://www.cs.cmu.edu/~epxing/Class/10708-20/lectures.html
https://cvpr2022-tutorial-diffusion-models.github.io/
https://huggingface.co/blog/annotated-diffusion
https://calvinyluo.com/2022/08/26/diffusion-tutorial.html
https://jmtomczak.github.io/blog/1/1_introduction.html