ELV: Embodied Learning and Vision

Mengye Ren

NYU

Spring 2026

elvcourse.org

Module 1:
Deep Learning for Structured Outputs

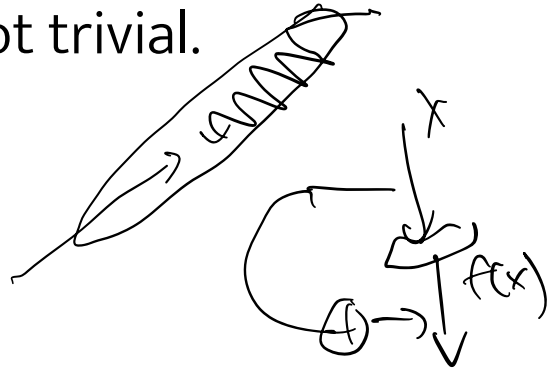
NYU

Deep Learning

- Over decades, optimizing deep neural networks was not trivial.

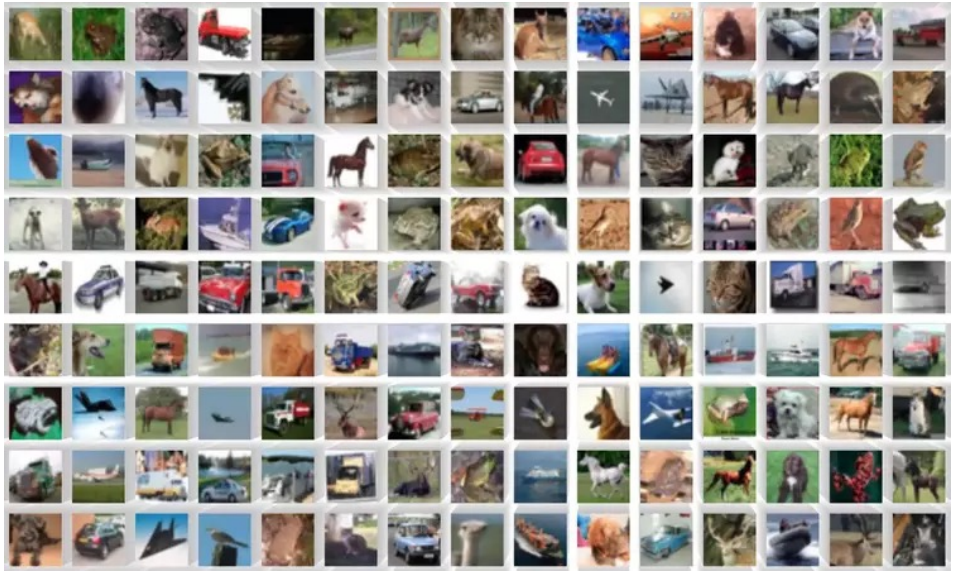
Deep Learning

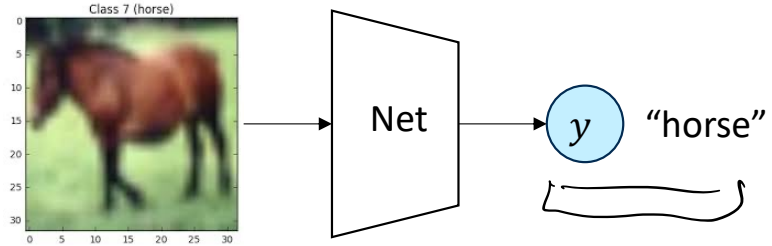
- Over decades, optimizing deep neural networks was not trivial.
- Progress came from (taken for granted nowadays):
  - Initialization
  - Optimizers (Adam, gradient conditioning)
  - Normalization (BN, LN, GN, etc.)
  - Skip connection (recurrent net, residual net, preserving gradients)
  - Regularization (dropout, noise, augmentation)
  - Attention (generalization, parallelization)



Classification

- To test how we can fit a deep neural network well, people have relied on simple benchmarks, such as image classification.

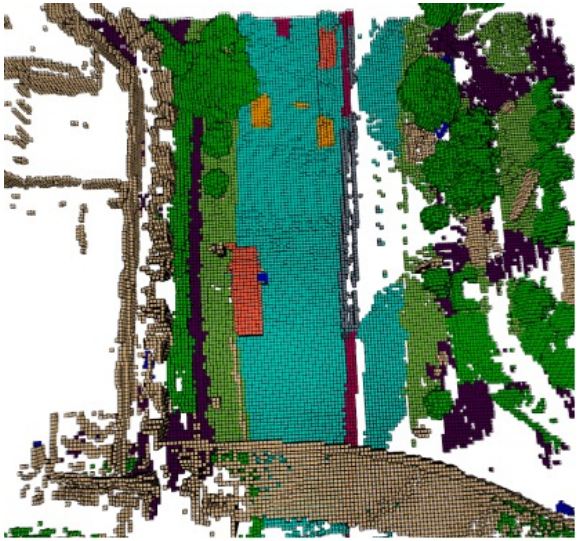




NYU

What are Structured Outputs?

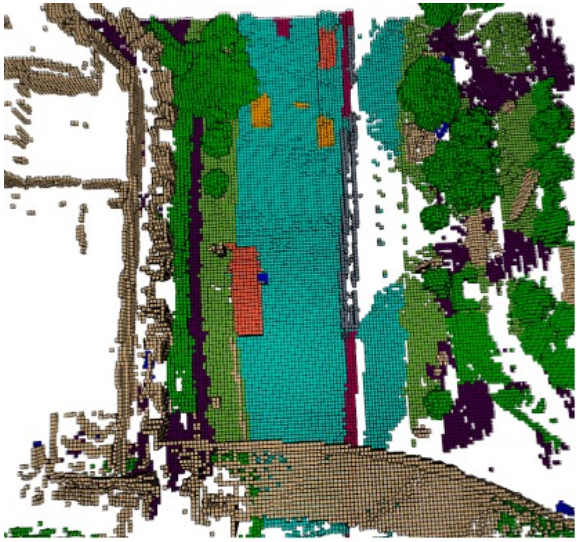
- An embodied agent needs to have a structured output space.
  - Object localization, tracking, motion, spatial segmentation, 3d occupancy
  - Trajectory, forecasting, planning



NYU

What are Structured Outputs?

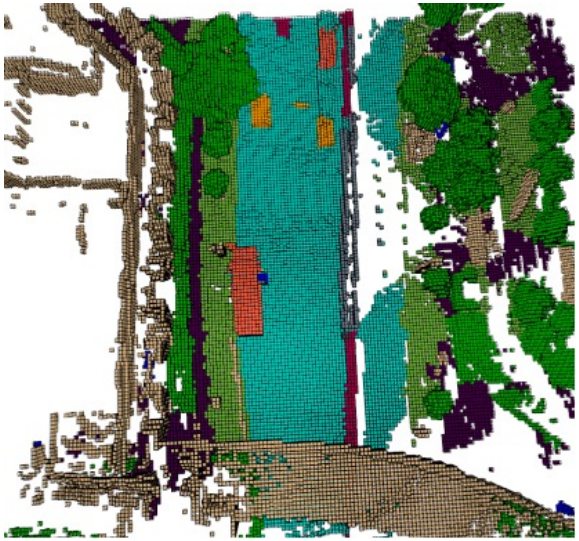
- An embodied agent needs to have a structured output space.
  - Object localization, tracking, motion, spatial segmentation, 3d occupancy
  - Trajectory, forecasting, planning
- A naïve solution is to simply have multiple output dimensions.



NYU

What are Structured Outputs?

- An embodied agent needs to have a structured output space.
  - Object localization, tracking, motion, spatial segmentation, 3d occupancy
  - Trajectory, forecasting, planning
- A naïve solution is to simply have multiple output dimensions.



- It often does not reason the joint probability

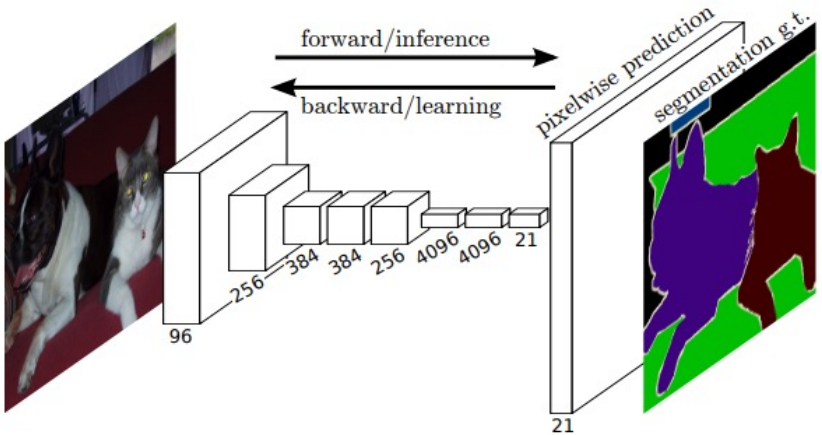
$y = Wx$ (y_1) (y_2) (y_3) . (x)

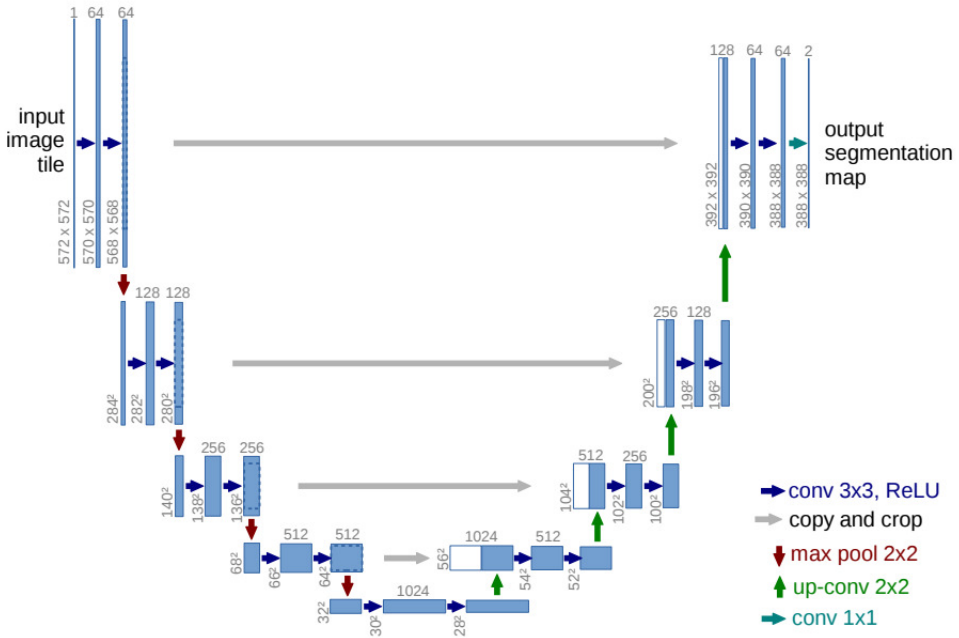
$y_i = w_i^T x$

NYU

Network Architecture for Structured Outputs

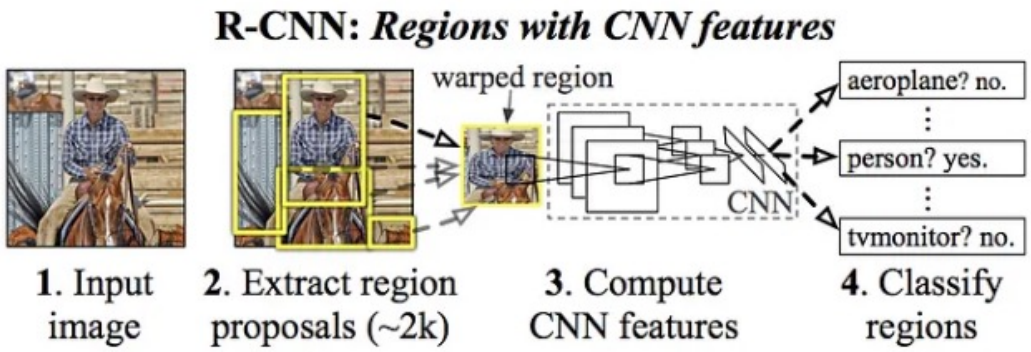
- Segmentation
- Spatial, high-resolution



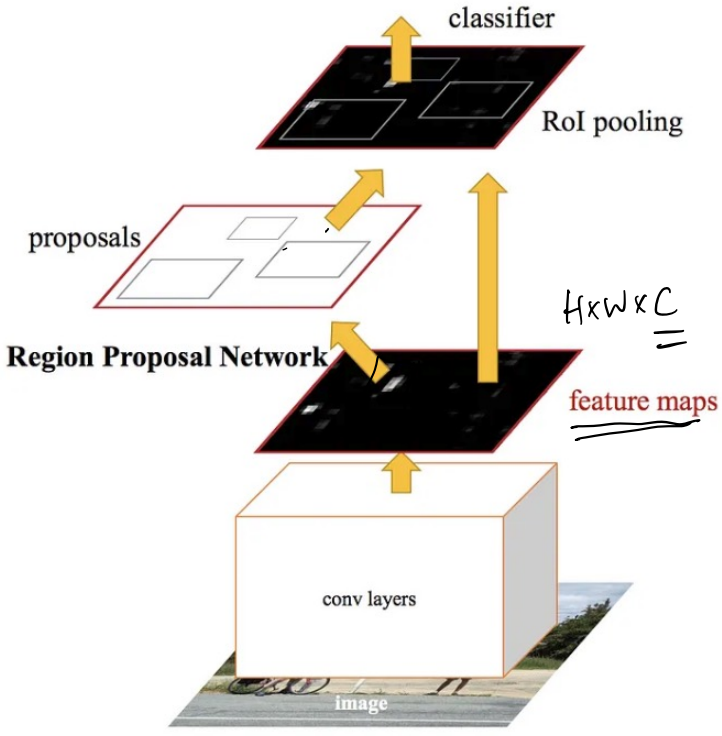


Network Architecture for Structured Outputs

- Object Detection



Girshick et al., 2013

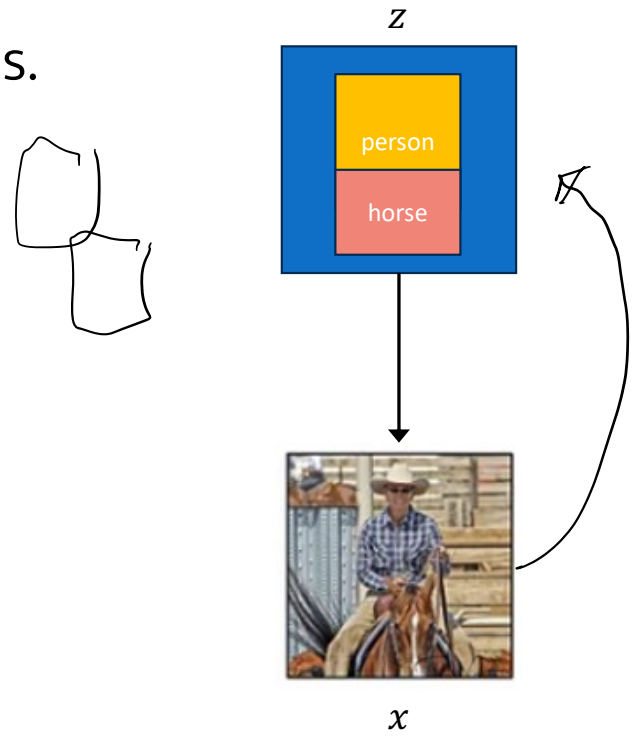


Ren et al., 2015

NYU

Object Detection as Inference

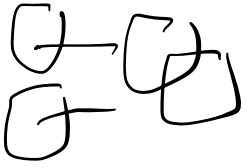
- Bounding boxes are structured latent variables.

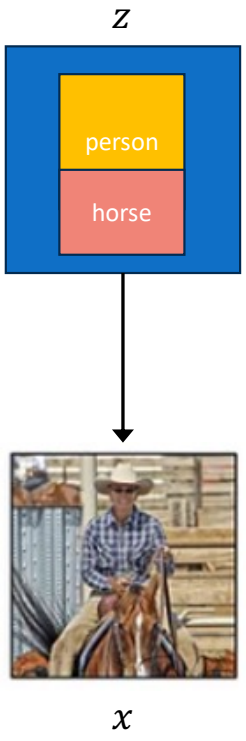


Object Detection as Inference

- Bounding boxes are structured latent variables.
- Occupancy as physical constraints.
  - One spatial 3D location can only present a single physical object.

$y = W^T x$.

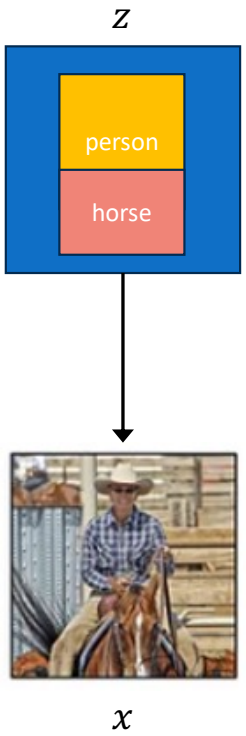




NYU

Object Detection as Inference

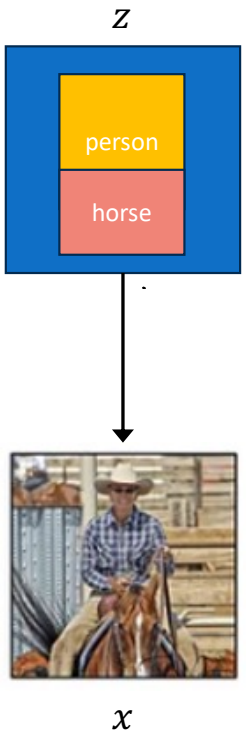
- Bounding boxes are structured latent variables.
- Occupancy as physical constraints.
  - One spatial 3D location can only present a single physical object.
- Object co-occurrence in the scene
  - Context



NYU

Object Detection as Inference

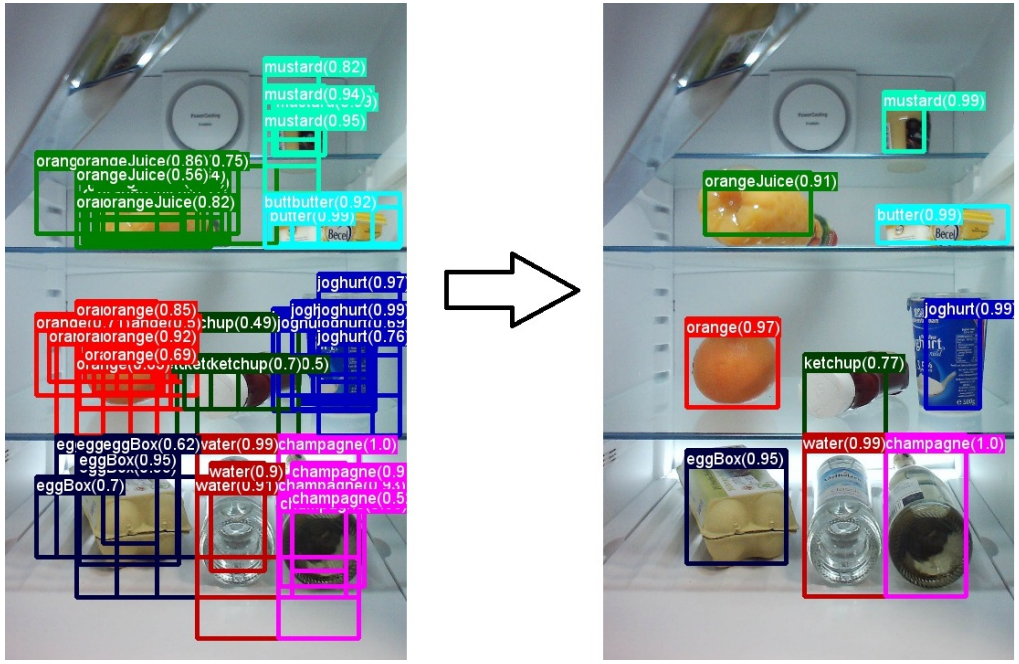
- Bounding boxes are structured latent variables.
- Occupancy as physical constraints.
  - One spatial 3D location can only present a single physical object.
- Object co-occurrence in the scene
  - Context
- The role of the network is to perform “inference” on the latent variables.



NYU

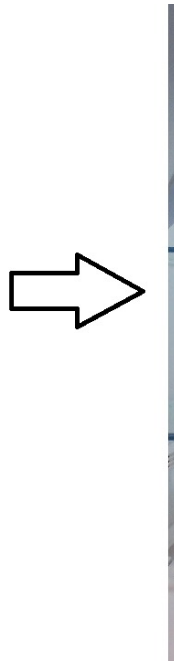
Non-Maximal Suppression

- Box Proposals: Samples of boxes that may come from a single object (latent)



Non-Maximal Suppression

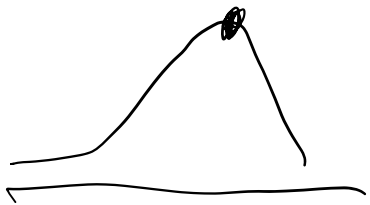
- Box Proposals: Samples of boxes that may come from a single object (latent)
- Each with a confidence score, density representation of the inferred latent distribution $q(z|x)$.

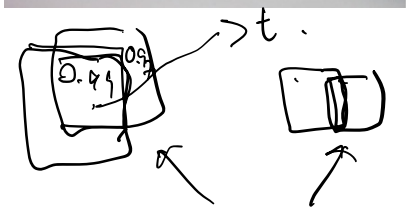


NYU

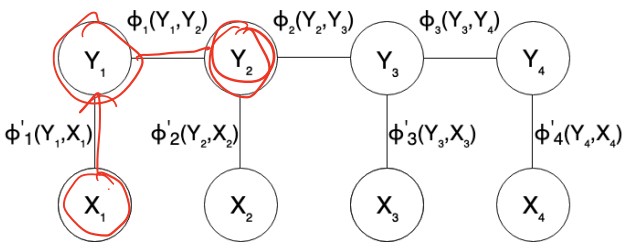
Non-Maximal Suppression

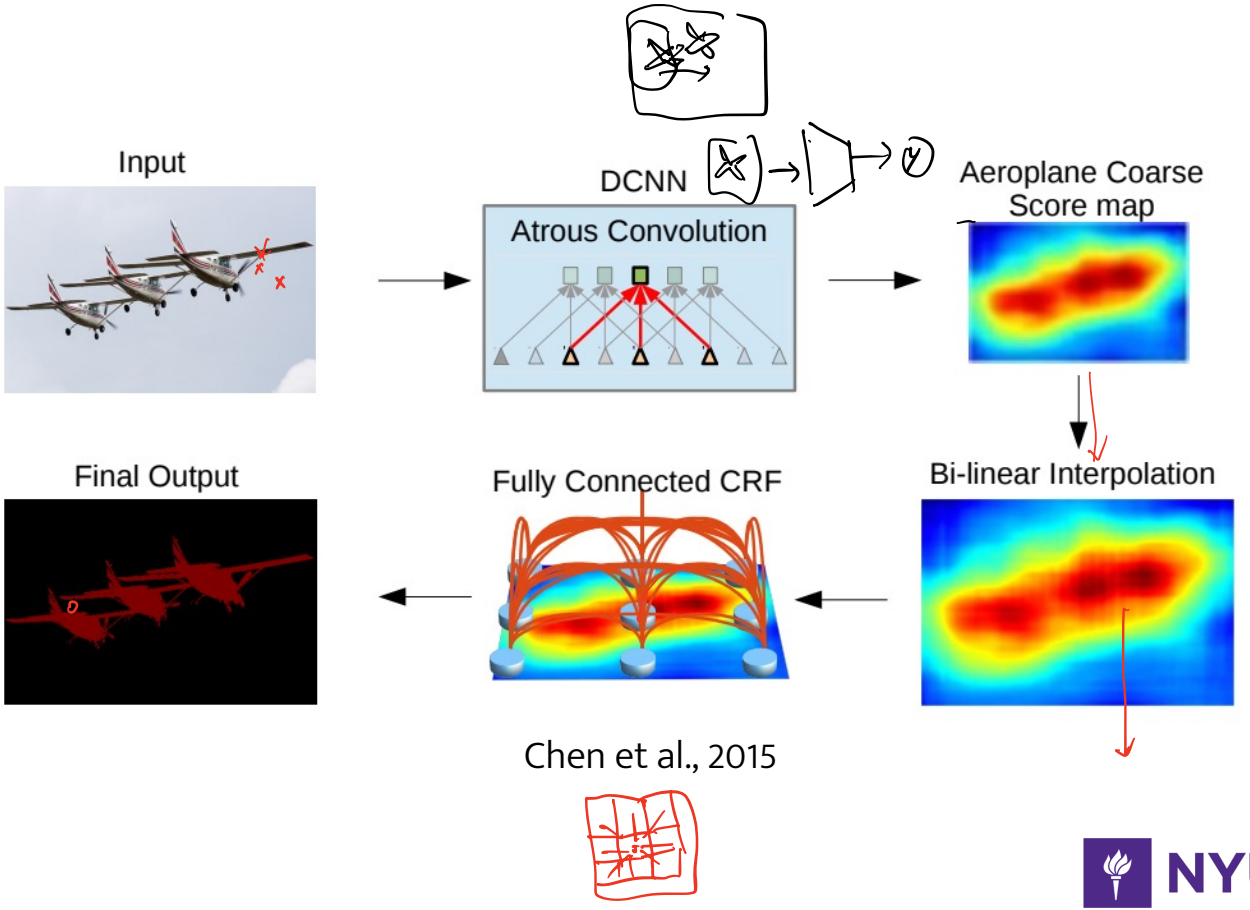
- Box Proposals: Samples of boxes that may come from a single object (latent)
- Each with a confidence score, density representation of the inferred latent distribution $q(z|x)$.
- MAP: take the mode of the distribution





Segmentation as CRF Inference





Chen et al., 2015

NYU

Inference Problem

- Knows $p(x|z)$.

NYU

Inference Problem

- Knows $p(x|z)$.
- Wants to know $p(z|x)$. Bayes rule.

$$p(z|x;\theta) = \frac{p(x,z;\theta)}{\int_z p(x,z;\theta)}.$$

Inference Problem

- Knows $p(x|z)$.
- Wants to know $p(z|x)$. Bayes rule.

$$p(z|x;\theta) = \frac{p(x,z;\theta)}{\int_z p(x,z;\theta)}.$$

- Brute force

all possible latent state.

Inference Problem

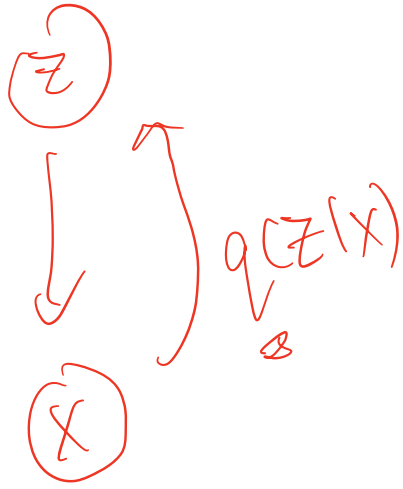
- Knows $p(x|z)$.
- Wants to know $p(z|x)$. Bayes rule.

$$p(z|x;\theta) = \frac{p(x,z;\theta)}{\int_z p(x,z;\theta)}.$$

- Brute force
- Message passing, sum-product, belief propagation (DP)

NYU

Inference Problem



- Knows $p(x|z)$.
- Wants to know $p(z|x)$. Bayes rule.

$$p(z|x;\theta) = \frac{p(x,z;\theta)}{\int_z p(x,z;\theta)}.$$

- Brute force
- Message passing, sum-product, belief propagation (DP)
- Variational inference, mean field

NYU

Inference Problem

- Knows $p(x|z)$.
- Wants to know $p(z|x)$. Bayes rule.

$$p(z|x;\theta) = \frac{p(x,z;\theta)}{\int_z p(x,z;\theta)}.$$

- Brute force
- Message passing, sum-product, belief propagation (DP)
- Variational inference, mean field
- Stochastic sampling, MCMC

NYU

Variational Inference

- Jensens inequality to get ELBO.

$$\log p(x) = \log \int_z p(x, z)$$

$$= \log \int_z p(x, z) \frac{q(z)}{q(z)}$$

Variational Inference

- Jensens inequality to get ELBO.

$$\log p(x) = \log \int_z p(x, z)$$
$$= \log \int_z p(x, z) \frac{q(z)}{q(z)}$$
$$= \log \left(\mathbb{E}_q \frac{p(x, z)}{q(z)} \right)$$

Variational Inference

- Jensens inequality to get ELBO.

$$
\begin{aligned}
\log p(x) &= \log \int_z p(x,z) \\
&= \log \int_z p(x,z) \frac{q(z)}{q(z)} \\
&= \log \left(\mathbb{E}_q \frac{p(x,z)}{q(z)} \right) \\
&\geq \mathbb{E}_q \log \frac{p(x,z)}{q(z)} \\
&= \mathbb{E}_q \log p(x,z) - \mathbb{E}_q \log q(z) = \mathcal{L}.
\end{aligned}
$$

Mean-Field Inference

- If there are many latent variables, we can assume factorization (local variational approximation):

$$q(z_1, \ldots, z_m) = \prod_{j=1}^{m} q(z_j).$$

Mean-Field Inference

- If there are many latent variables, we can assume factorization (local variational approximation):

$$q(z_1, \ldots, z_m) = \prod_{j=1}^{m} q(z_j).$$

$$\mathcal{L} = \log p(x) + \sum_{j=1}^{m} \mathbb{E}_{q(z_j)} \log p(z_j | z_{-j}, x) - \mathbb{E}_{q(z_j)} \log(q(z_j)).$$

Inference Operations

- CRF with pairwise energy. Use x as labels.

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j),$$

unary. binary.

[Krähenbühl & Koltun, 2012]

Inference Operations

- CRF with pairwise energy. Use x as labels.

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j),$$

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \underbrace{\sum_{m=1}^{K} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)}_{k(\mathbf{f}_i, \mathbf{f}_j)}$$

[Krähenbühl & Koltun, 2012]

NYU

Inference Operations

- CRF with pairwise energy. Use x as labels.

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j),$$

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \underbrace{\sum_{m=1}^{K} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)}_{k(\mathbf{f}_i, \mathbf{f}_j)}$$

$$k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) = \exp\left(-\tfrac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^{\mathrm{T}} \Lambda^{(m)} (\mathbf{f}_i - \mathbf{f}_j)\right)$$

$$w^{(1)} \underbrace{\exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right)}_{\text{appearance kernel}} + w^{(2)} \underbrace{\exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right)}_{\text{smoothness kernel}}.$$

$$\mu(x_i, x_j) = [x_i \neq x_j]$$

[Krähenbühl & Koltun, 2012]

NYU

Inference in Fully Connected CRF

- Iterative mean-field inference.

Algorithm 1 Mean field in fully connected CRFs

Initialize Q ▷ $Q_i(x_i) \leftarrow \frac{1}{Z_i}\exp\{-\phi_u(x_i)\}$

while not converged **do** ▷ See Section 6 for convergence analysis

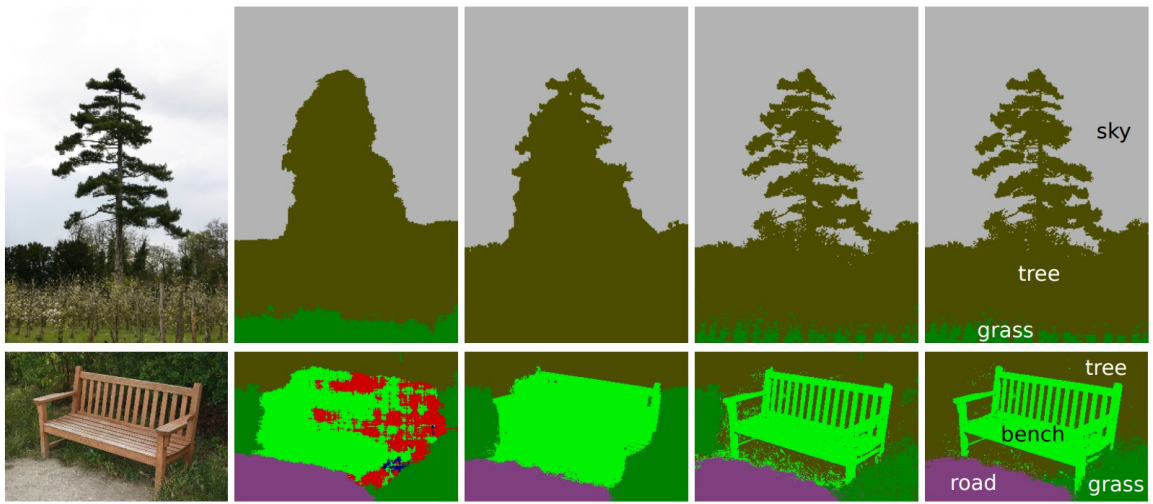
$\tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l)$ for all m ▷ **Message passing** from all X_j to all X_i

$\hat{Q}_i(x_i) \leftarrow \sum_{l \in \mathcal{L}} \mu^{(m)}(x_i, l) \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l)$ ▷ **Compatibility transform**

$Q_i(x_i) \leftarrow \exp\{-\psi_u(x_i) - \hat{Q}_i(x_i)\}$ ▷ **Local update**

normalize $Q_i(x_i)$

end while

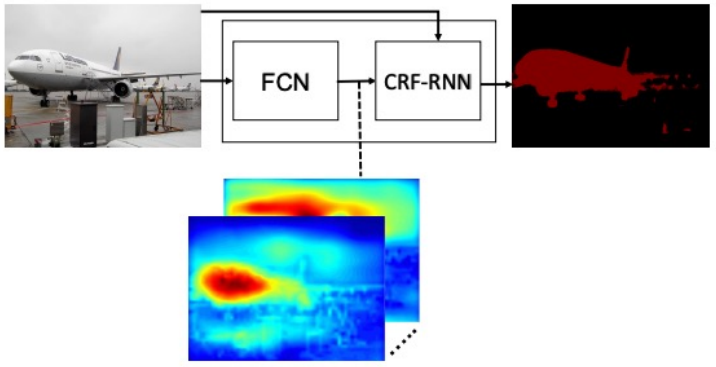


(a) Image (b) Unary classifiers (c) Robust P^n CRF (d) Fully connected CRF, MCMC inference, 36 hrs (e) Fully connected CRF, our approach, 0.2 seconds

[Krähenbühl & Koltun, 2012]

NYU

CRFs as RNNs



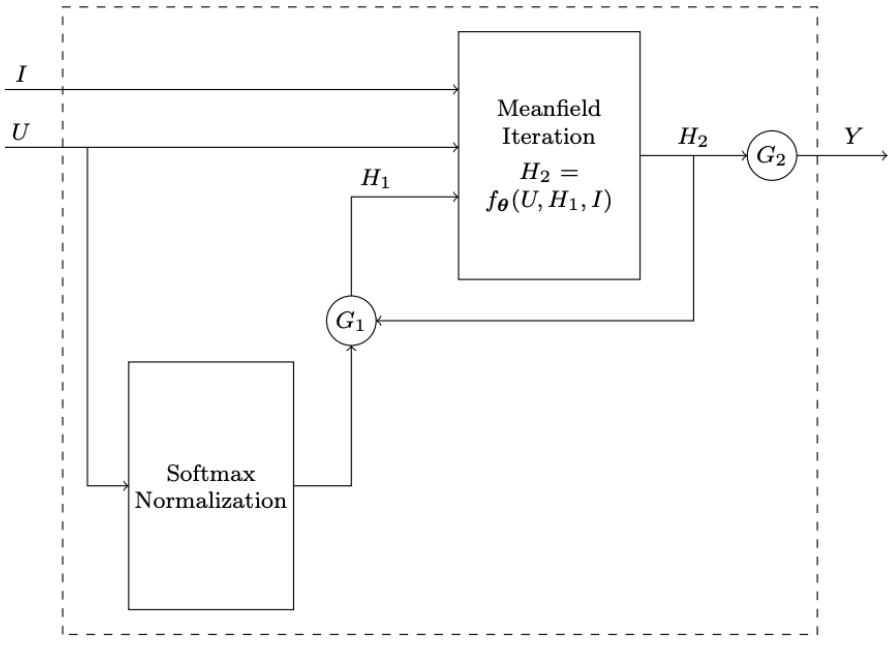


Figure 2. **The CRF-RNN Network.** We formulate the iterat mean-field algorithm as a Recurrent Neural Network (RNN). G ing functions G_1 and G_2 are fixed as described in the text.

$Q_i(l) \leftarrow \frac{1}{Z_i} \exp\left(U_i(l)\right)$ for all i ▷ Initialization

while not converged **do**

$\tilde{Q}_i^{(m)}(l) \leftarrow \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l)$ for all m

▷ Message Passing

$\check{Q}_i(l) \leftarrow \sum_m w^{(m)} \tilde{Q}_i^{(m)}(l)$

▷ Weighting Filter Outputs

$\hat{Q}_i(l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \check{Q}_i(l')$

▷ Compatibility Transform

$\breve{Q}_i(l) \leftarrow U_i(l) - \hat{Q}_i(l)$

▷ Adding Unary Potentials

$Q_i \leftarrow \frac{1}{Z_i} \exp\left(\breve{Q}_i(l)\right)$

▷ Normalizing

end while

[Zheng et al., 2015]

NYU

Summary

- Perception of high dimensional objects can be viewed as inferring latent variables with probabilistic distributions.

NYU

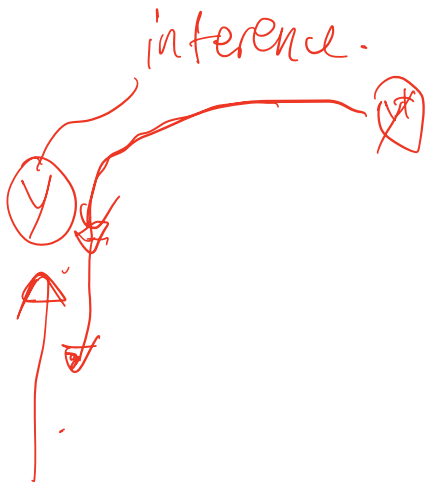
Summary

- Perception of high dimensional objects can be viewed as inferring latent variables with probabilistic distributions.
- We can impose structure.

NYU

Summary

- Perception of high dimensional objects can be viewed as inferring latent variables with probabilistic distributions.
- We can impose structure.
- We can learn through the inference process.
  - Taking the inference process into account.
  - Learning representations that matter.



NYU

Some Nuances

- If the process is deterministic or unimodal, standard deep networks may work.

NYU

Some Nuances

- If the process is deterministic or unimodal, standard deep networks may work.
- Network forward propagation vs. relaxed probabilistic inference.

NYU

Some Nuances

- If the process is deterministic or unimodal, standard deep networks may work.
- Network forward propagation vs. relaxed probabilistic inference.
- Having a stronger prior has the potential to be more data efficient.

NYU

Some Nuances

- If the process is deterministic or unimodal, standard deep networks may work.
- Network forward propagation vs. relaxed probabilistic inference.
- Having a stronger prior has the potential to be more data efficient.
- And you will need structured / generative learning when there are multiple modes.
  - E.g. Motion generation and planning: there can be multiple future trajectories

NYU

Deep Generative Models

- Autoregressive models

Deep Generative Models

- Autoregressive models
- Energy-based models
  - CRF
  - Max-Margin

NYU

Deep Generative Models

- Autoregressive models
- Energy-based models
  - CRF
  - Max-Margin
- Normalizing flow

NYU

Deep Generative Models

- Autoregressive models
- Energy-based models
  - CRF
  - Max-Margin
- Normalizing flow
- Diffusion models

NYU

Autoregressive Modeling

- Another type of output is autoregressive modeling.

NYU

Autoregressive Modeling

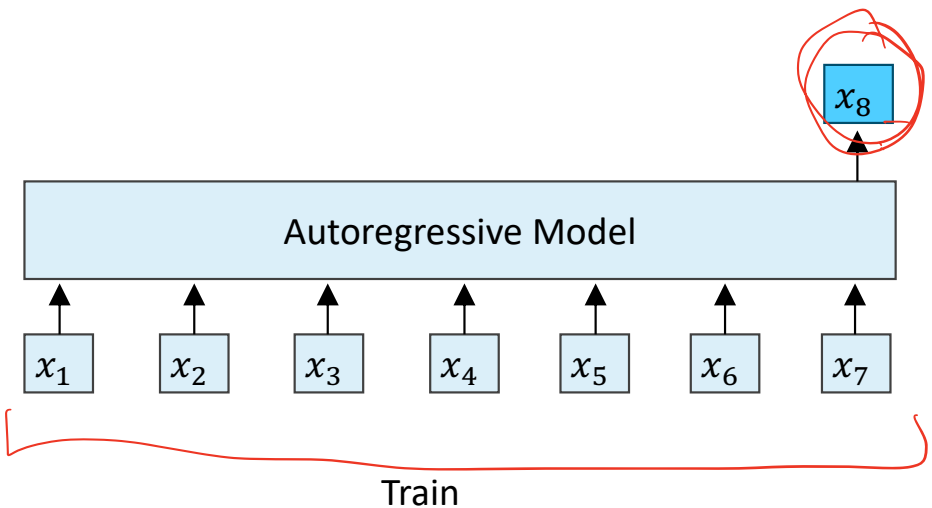
- Another type of output is autoregressive modeling.
- Example: Object detection/segmentation.

Autoregressive Modeling

- Another type of output is autoregressive modeling.
- Example: Object detection/segmentation.
- Intuition: Our visual attention focus on one object at a time.

Teacher Forcing

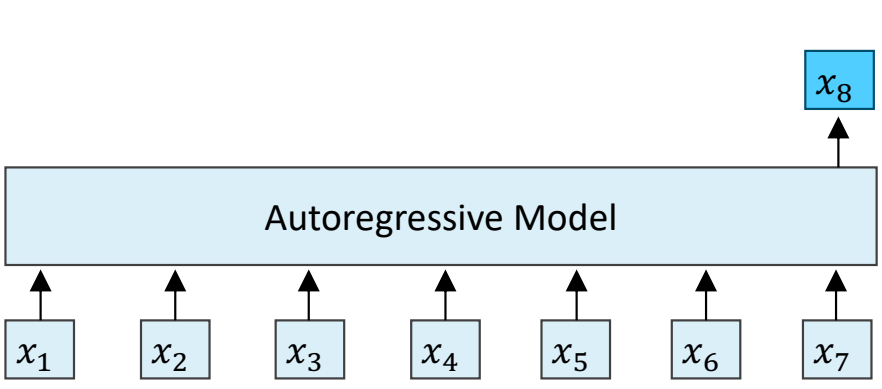
- Teacher Forcing: Pretend that you know the whole sequence $\{y_1 \ldots y_t\}$ at training time, and train for y_{t+1}.



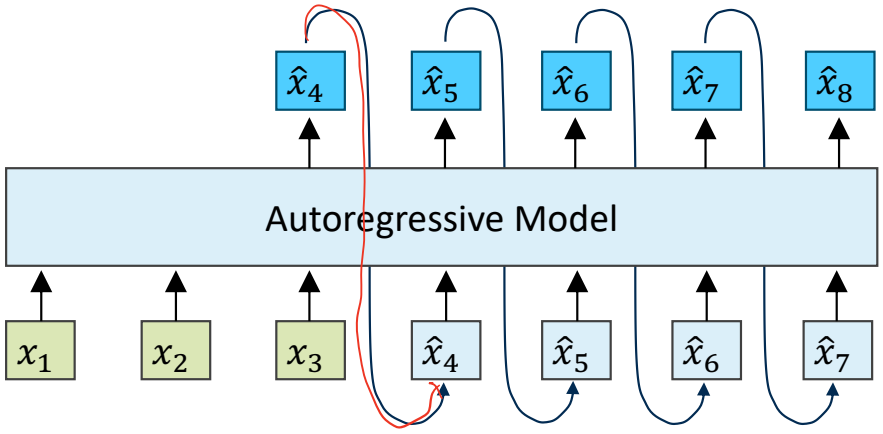
NYU

Teacher Forcing

- Teacher Forcing: Pretend that you know the whole sequence $\{y_1 \ldots y_t\}$ at training time, and train for y_{t+1}.



Train

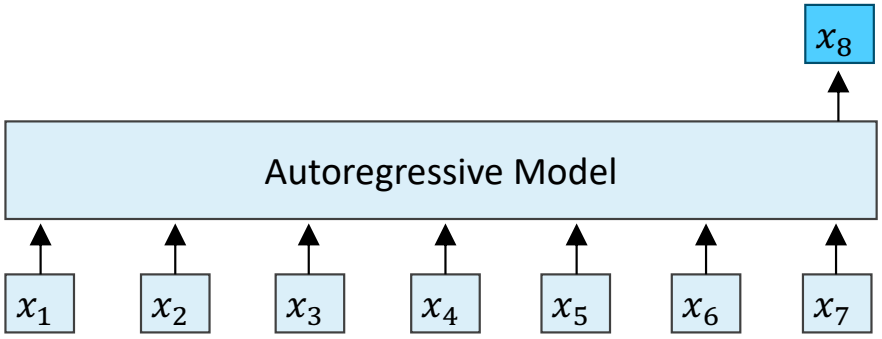


Test

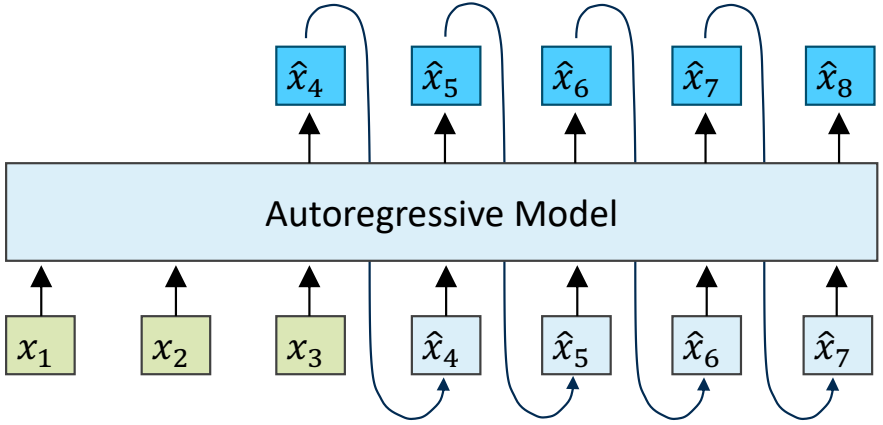
NYU

Teacher Forcing

- Teacher Forcing: Pretend that you know the whole sequence $\{y_1 \ldots y_t\}$ at training time, and train for y_{t+1}.
- Problem: You have to know the ordering.



Train



Test

NYU

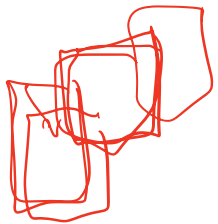
More on Ordering

- There are many set to set problems.

NYU

More on Ordering

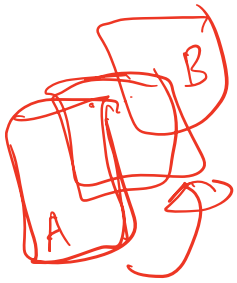
- There are many set to set problems.
- E.g. Detection, segmentation, generating multiple objects, clustering

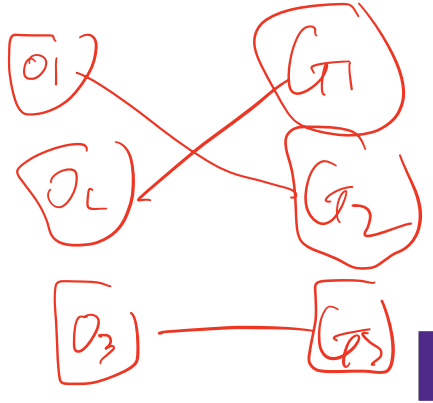


NYU

More on Ordering

- There are many set to set problems.
- E.g. Detection, segmentation, generating multiple objects, clustering
- Input does not follow a particular order. Loss function should not favor a particular order.
  - Attention: The attention operation is order-invariant.
  - Matching: The teacher can match the next input based on the closest matching ground truth instances.

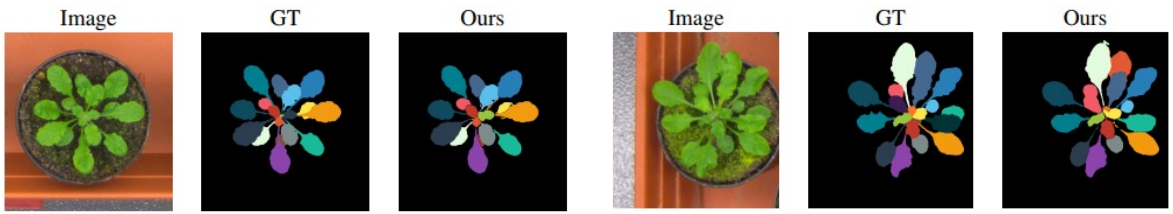


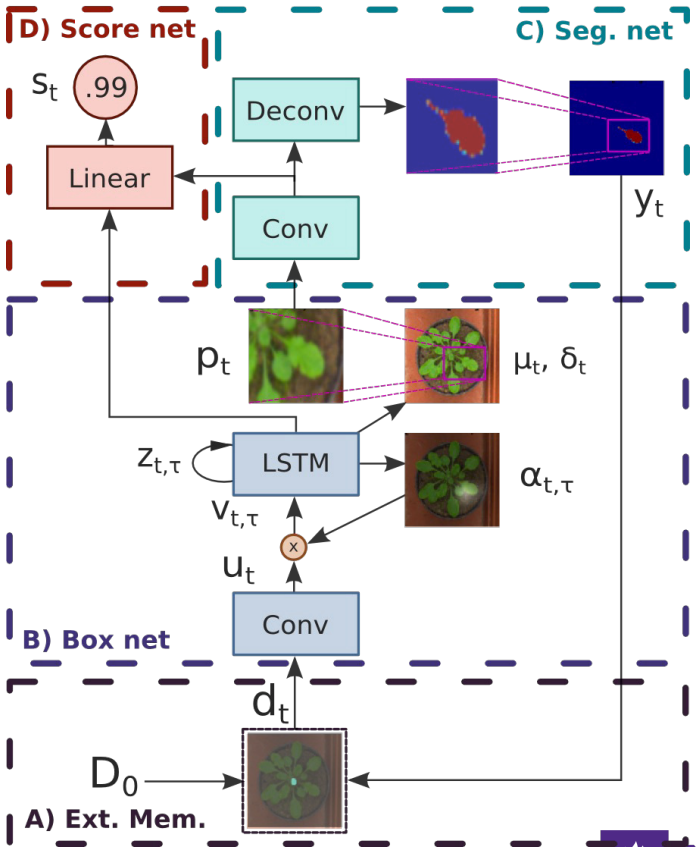


NYU

Instance Segmentation using Attention

- Attending to one region at a time.
- Zooming in for segmentation.
- End-to-end differentiable box proposals and external memory.
- State-of-the-art on leave segmentation problems for many years.





NYU

Summary

- Autoregressive models
  - Exact likelihood (instead of a lower bound or approximation)
  - Conditional probability may be easier to model
  - Need an ordering
  - Can take a long time to decode

NYU

Summary

- Autoregressive models
  - Exact likelihood (instead of a lower bound or approximation)
  - Conditional probability may be easier to model
  - Need an ordering
  - Can take a long time to decode
- Autoregressive models aren't just for language.
  - Planning
  - Set prediction problems
  - Perception

NYU

Summary

- Autoregressive models
  - Exact likelihood (instead of a lower bound or approximation)
  - Conditional probability may be easier to model
  - Need an ordering
  - Can take a long time to decode
- Autoregressive models aren't just for language.
  - Planning
  - Set prediction problems
  - Perception
- Generating the next variable is a different objective from learning a good representation of the whole sequence.

NYU

Summary

- Autoregressive models
  - Exact likelihood (instead of a lower bound or approximation)
  - Conditional probability may be easier to model
  - Need an ordering
  - Can take a long time to decode
- Autoregressive models aren't just for language.
  - Planning
  - Set prediction problems
  - Perception
- Generating the next variable is a different objective from learning a good representation of the whole sequence.
- No hierarchical planning.

Energy-Based Modeling

- The goal is to learn a distribution $p(x)$ to model the set of examples.

NYU

Energy-Based Modeling

- The goal is to learn a distribution $p(x)$ to model the set of examples.
- Assuming Boltzmann distribution:

$$p(x;\theta) = \frac{1}{Z(\theta)} \exp(-E(x;\theta))$$

$$Z(\theta) = \int_X \exp(-E(x;\theta))$$

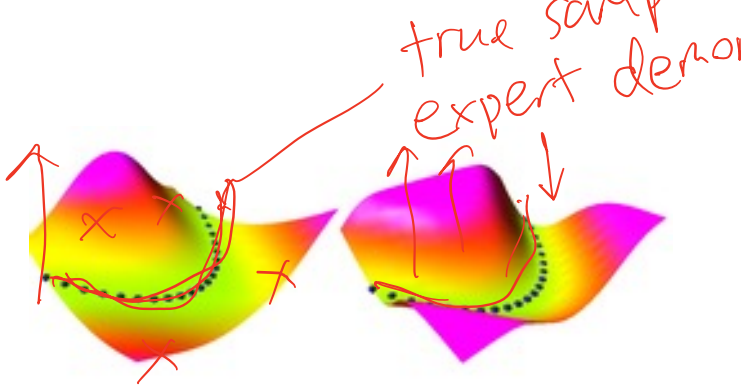
network.

Energy-Based Modeling

- The goal is to learn a distribution $p(x)$ to model the set of examples.
- Assuming Boltzmann distribution:

$$p(x;\theta) = \frac{1}{Z(\theta)} \exp(-E(x;\theta))$$

$$Z(\theta) = \int_X \exp(-E(x;\theta))$$



Picture from LeCun

- Maximizing the likelihood:

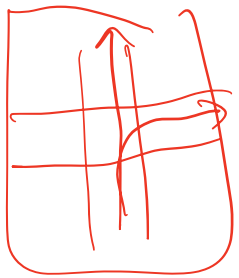
$$\frac{\partial}{\partial\theta} \log p(x;\theta) = \mathbb{E}_{x' \sim p(x)} \left[\frac{\partial E(x';\theta)}{\partial\theta} \right] - \frac{\partial E(x;\theta)}{\partial\theta}.$$

real data.

Structured Prediction

- EBM can be easily adapted to model the joint distribution of x and y.
- Requires optimization of y at inference time.

$$\arg\min_y E(x, y)$$



traj : y.

visual scene : x.

Optimization

- Computing $\mathbb{E}_{x' \sim p(x)} \left[\frac{\partial E(x';\theta)}{\partial \theta} \right]$ is non-trivial. We need to sample from p(x).

Song & Kingma. How to Train Your Energy-Based Models.

NYU

Optimization

- Computing $\mathbb{E}_{x' \sim p(x)} \left[\frac{\partial E(x';\theta)}{\partial \theta} \right]$ is non-trivial. We need to sample from p(x).
- But we only know $E(x)$!

Song & Kingma. How to Train Your Energy-Based Models.

NYU

Optimization

- Computing $\mathbb{E}_{x' \sim p(x)} \left[\frac{\partial E(x';\theta)}{\partial \theta} \right]$ is non-trivial. We need to sample from p(x).
- But we only know $E(x)$!
- Through MCMC samplers: MH, Langevin, HMC, Gibbs.

Song & Kingma. How to Train Your Energy-Based Models.

NYU

Optimization

- Computing $\mathbb{E}_{x' \sim p(x)} \left[\frac{\partial E(x';\theta)}{\partial \theta} \right]$ is non-trivial. We need to sample from p(x).
- But we only know $E(x)$!
- Through MCMC samplers: MH, Langevin, HMC, Gibbs.
- Approximations: Using truncated steps (Contrastive Divergence)

Song & Kingma. How to Train Your Energy-Based Models.

NYU

Optimization

- Computing $\mathbb{E}_{x' \sim p(x)} \left[\frac{\partial E(x';\theta)}{\partial \theta} \right]$ is non-trivial. We need to sample from p(x).
- But we only know $E(x)$!
- Through MCMC samplers: MH, Langevin, HMC, Gibbs.
- Approximations: Using truncated steps (Contrastive Divergence)
- Score Matching: Tries to model the score function $\nabla_x \log p_\theta(x)$
  - If we know the gradient, we can improve the samples.
  - Want the model gradient to match with that from the data distribution
  - Closely related to diffusion models

Song & Kingma. How to Train Your Energy-Based Models.

NYU

Max-Entropy Inverse RL

- Want to model the reward function of trajectories.
- Given a reward function, the policy distribution can be modeled as a Boltzmann distribution.
- Given expert trajectories, we can maximize the likelihood under such distribution induced by the learned reward

$$p(\tau|\mathcal{O}_{1:T},\psi) \propto \cancel{p(\tau)} \exp\left(\sum_t r_\psi(\mathbf{s}_t,\mathbf{a}_t)\right)$$

can ignore (independent of ψ)

maximum likelihood learning: $$\max_\psi \frac{1}{N}\sum_{i=1}^{N} \log p(\tau_i|\mathcal{O}_{1:T},\psi) = \max_\psi \frac{1}{N}\sum_{i=1}^{N} r_\psi(\tau_i) - \log Z$$

Slide credit: Sergey Levine

NYU

Max-Entropy Inverse RL

$$\max_{\psi} \frac{1}{N} \sum_{i=1}^{N} r_\psi(\tau_i) - \log Z \qquad Z = \int p(\tau) \exp(r_\psi(\tau)) d\tau$$

$$\nabla_\psi \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \underbrace{\frac{1}{Z} \int p(\tau) \exp(r_\psi(\tau)) \nabla_\psi r_\psi(\tau) d\tau}_{p(\tau|\mathcal{O}_{1:T}, \psi)}$$

current policy.

$$\nabla_\psi \mathcal{L} = E_{\tau \sim \pi^\star(\tau)}[\nabla_\psi r_\psi(\tau_i)] - E_{\tau \sim p(\tau|\mathcal{O}_{1:T}, \psi)}[\nabla_\psi r_\psi(\tau)]$$

estimate with expert samples

soft optimal policy under current reward

Slide credit: Sergey Levine

Ziebart et al. Maximum Entropy Inverse Reinforcement Learning. AAAI 2008.

NYU

Max-Margin Learning

- In addition to probabilistic learning, we can also apply the max-margin framework.

Max-Margin Learning

- In addition to probabilistic learning, we can also apply the max-margin framework.
- Review: SSVM vs. CRF.

Max-Margin Learning

- In addition to probabilistic learning, we can also apply the max-margin framework.
- Review: SSVM vs. CRF.

$$\arg\min_\theta \sum_{n=1}^{N} \max[0, m + E(x_n;\theta) - E(x^*;\theta)] + \lambda||\theta||_2^2.$$

n^{th} demo

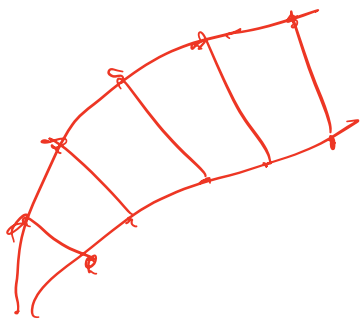
best output from model.

Max-Margin Learning

- In addition to probabilistic learning, we can also apply the max-margin framework.
- Review: SSVM vs. CRF.

$$\arg\min_\theta \sum_{n=1}^{N} \max[0, m + E(x_n; \theta) - E(x^*; \theta)] + \lambda ||\theta||_2^2.$$

- Margin can be difference in trajectories.



NYU

Max-Margin Learning

- In addition to probabilistic learning, we can also apply the max-margin framework.
- Review: SSVM vs. CRF.

$$\arg\min_\theta \sum_{n=1}^{N} \max[0, m + E(x_n; \theta) - E(x^*; \theta)] + \lambda ||\theta||_2^2.$$

- Margin can be difference in trajectories.
- Non-probabilistic

NYU

Max-Margin Learning

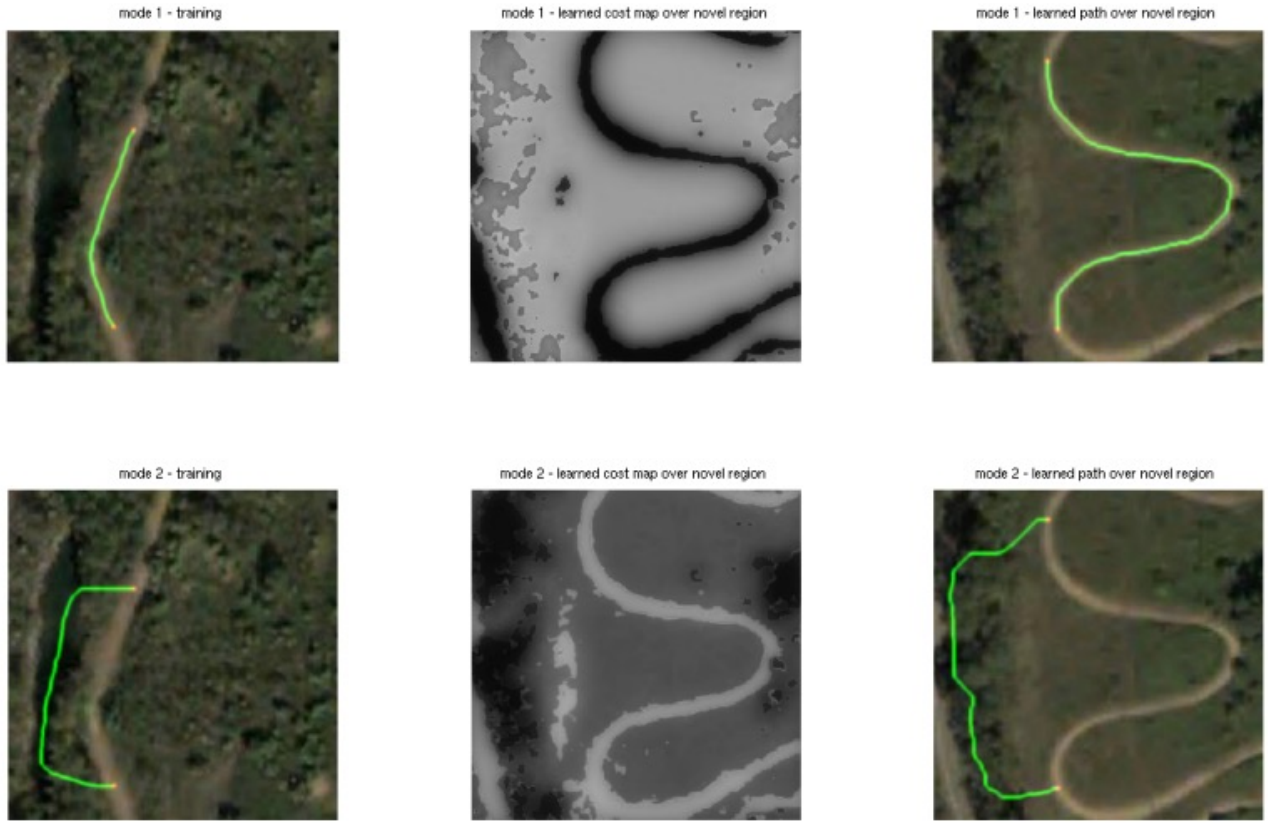
- In addition to probabilistic learning, we can also apply the max-margin framework.
- Review: SSVM vs. CRF.

$$\arg\min_\theta \sum_{n=1}^{N} \max[0, m + E(x_n; \theta) - E(x^*; \theta)] + \lambda ||\theta||_2^2.$$

- Margin can be difference in trajectories.
- Non-probabilistic
- Still need to run optimization to find the best x^*

Samples

Max-Margin Planning

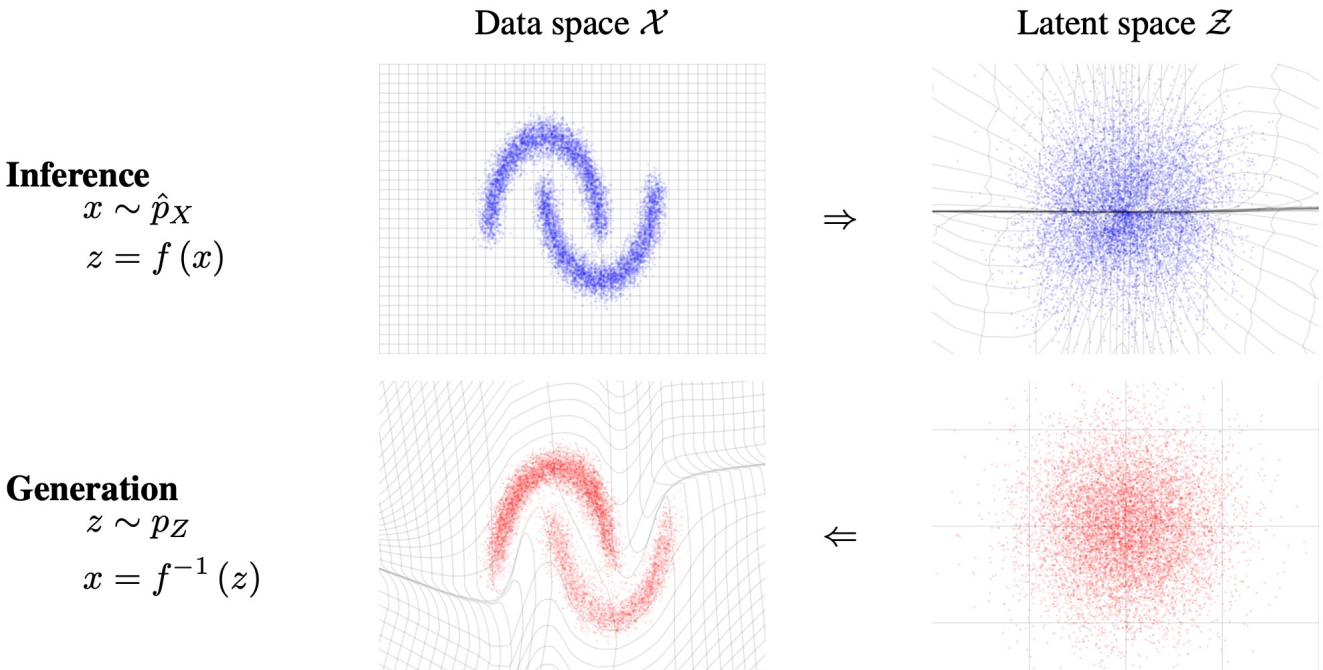


Ratliff et al. Maximum Margin Planning. ICML 2006.

NYU

Learning Latent Representations

- z may contain useful high-level compressed information of the input.



Variational Autoencoders (VAE)

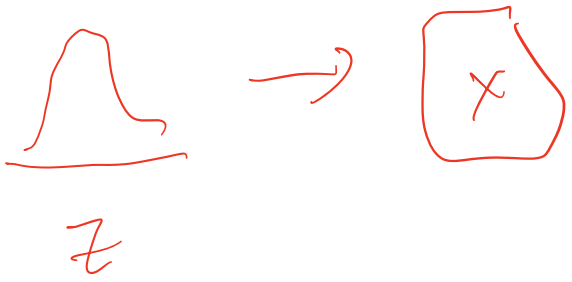
- z from a simple prior distribution.
- Optimize ELBO.

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

NYU

Variational Autoencoders (VAE)

- z from a simple prior distribution.
- Optimize ELBO.



$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

$$= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

Variational Autoencoders (VAE)

- z from a simple prior distribution.
- Optimize ELBO.

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

$$= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$

$$\geq -\mathbb{D}_{\mathrm{KL}}[q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})] + \mathbb{E}_q[\log p_\theta(\mathbf{x}|\mathbf{z})]$$

NYU

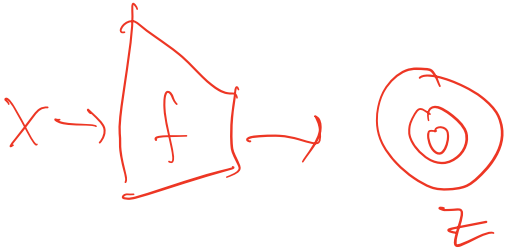
VAE Problems

- Hard to transform the inputs to the prior distribution with a network.
- The coarser the approximation of q, the worse the lower bound.

$$
\begin{aligned}
ELBO &= \int_z q(z) \log \frac{p(x,z)}{q(z)} \\
&= \mathbb{E}_{q(z)} \log \frac{p(z|x)p(x)}{q(z)}
\end{aligned}
$$

NYU

VAE Problems



- Hard to transform the inputs to the prior distribution with a network.
- The coarser the approximation of q, the worse the lower bound.

$$
\begin{aligned}
ELBO &= \int_z q(z) \log \frac{p(x,z)}{q(z)} \\
&= \mathbb{E}_{q(z)} \log \frac{p(z|x)p(x)}{q(z)} \\
&= -\mathbb{E}_{q(z)} \log \frac{q(z)}{p(z|x)} + \mathbb{E}_{q(z)} \log p(x) \\
&= -KL(q(z)||p(z|x)) + \log p(x)
\end{aligned}
$$

true posterior.

Normalizing Flows

- Requires a bijection between the input and output.

NYU

Normalizing Flows

- Requires a bijection between the input and output.
- Needs to maintain dimensionality.

NYU

Normalizing Flows

- Requires a bijection between the input and output.
- Needs to maintain dimensionality.

$$\max_{\theta} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \log p_{\text{NF}}(\boldsymbol{x}; \theta) = \log p_0(f_\theta(\boldsymbol{x}); \theta) + \log \left(\left| \det \left(\frac{\partial f_\theta(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right| \right)$$

Normalizing Flows

- Requires a bijection between the input and output.
- Needs to maintain dimensionality.
- Needs an efficient way to estimate determinant of Jacobian (typically O(n^3)).

$$\max_{\theta} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}} \log p_{\text{NF}}(\boldsymbol{x}; \theta) = \log p_0(f_\theta(\boldsymbol{x}); \theta) + \log \left(\left| \det \left(\frac{\partial f_\theta(\boldsymbol{x})}{\partial \boldsymbol{x}} \right) \right| \right)$$

NYU

Adding Constraints in Transformation

- Rezende & Mohamed (2015):

$$f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^\top \boldsymbol{z} + b).$$

Rezende & Mohamed. Variational Inference with Normalizing Flows. ICML 2015.

NYU

Adding Constraints in Transformation

- Rezende & Mohamed (2015):

$$f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^\top \boldsymbol{z} + b).$$

scalar

parameter.

- Determinant of Jacobian:

$$\psi(\boldsymbol{z}) = h'(\boldsymbol{w}^\top z + b)\boldsymbol{w}.$$

Rezende & Mohamed. Variational Inference with Normalizing Flows. ICML 2015.

Adding Constraints in Transformation

- Rezende & Mohamed (2015):

$$f(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^\top \boldsymbol{z} + b).$$

- Determinant of Jacobian:

$$\psi(\boldsymbol{z}) = h'(\boldsymbol{w}^\top z + b)\boldsymbol{w}.$$

$$\left|\det \frac{\partial f}{\partial \boldsymbol{z}}\right| = |\det(I + \boldsymbol{u}\psi(\boldsymbol{z})^\top)| = |1 + \boldsymbol{u}^\top \psi(\boldsymbol{z})|.$$

Rezende & Mohamed. Variational Inference with Normalizing Flows. ICML 2015.

Masked Autoregressive Flow (MAF)

- Another way to make Jacobian determinant computable -> Lower triangular matrix.

Papamakarios et al. Masked Autoregressive Flow for Density Estimation. NIPS 2017.

NYU

Masked Autoregressive Flow (MAF)

- Another way to make Jacobian determinant computable -> Lower triangular matrix.
- Consider the autoregression of generating one dimension at a time.

$$x_i = \mu_i(x_{1:i-1}) + \sigma_i(x_{1:i-1}) z_i, \quad z_i = \frac{x_i - \mu_i(x_{1:i-1})}{\sigma_i(x_{1:i-1})}.$$

Papamakarios et al. Masked Autoregressive Flow for Density Estimation. NIPS 2017.

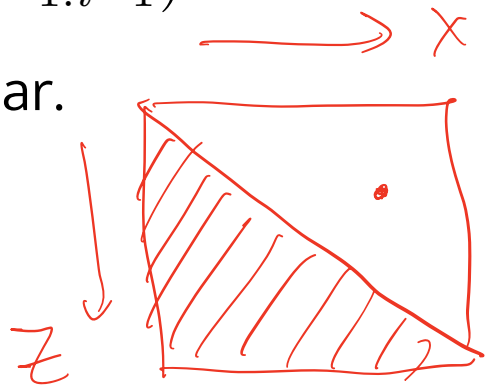
NYU

Masked Autoregressive Flow (MAF)

- Another way to make Jacobian determinant computable -> Lower triangular matrix.
- Consider the autoregression of generating one dimension at a time.

$$x_i = \mu_i(x_{1:i-1}) + \sigma_i(x_{1:i-1})z_i, \quad z_i = \frac{x_i - \mu_i(x_{1:i-1})}{\sigma_i(x_{1:i-1})}.$$

- z_i does not depend on x_j for any $j > i$. Lower triangular.



Papamakarios et al. Masked Autoregressive Flow for Density Estimation. NIPS 2017.

NYU

Masked Autoregressive Flow (MAF)

- Another way to make Jacobian determinant computable -> Lower triangular matrix.
- Consider the autoregression of generating one dimension at a time.

$$x_i = \mu_i(x_{1:i-1}) + \sigma_i(x_{1:i-1})z_i, \quad z_i = \frac{x_i - \mu_i(x_{1:i-1})}{\sigma_i(x_{1:i-1})}.$$

- z_i does not depend on x_j for any j > i. Lower triangular.

$$\log|\det J| = \sum_{i=1}^{D} \log\left|\frac{1}{\sigma_i(x_{1:i-1})}\right| = -\sum_{i=1}^{D} \log \sigma_i(x_{1:i-1})$$

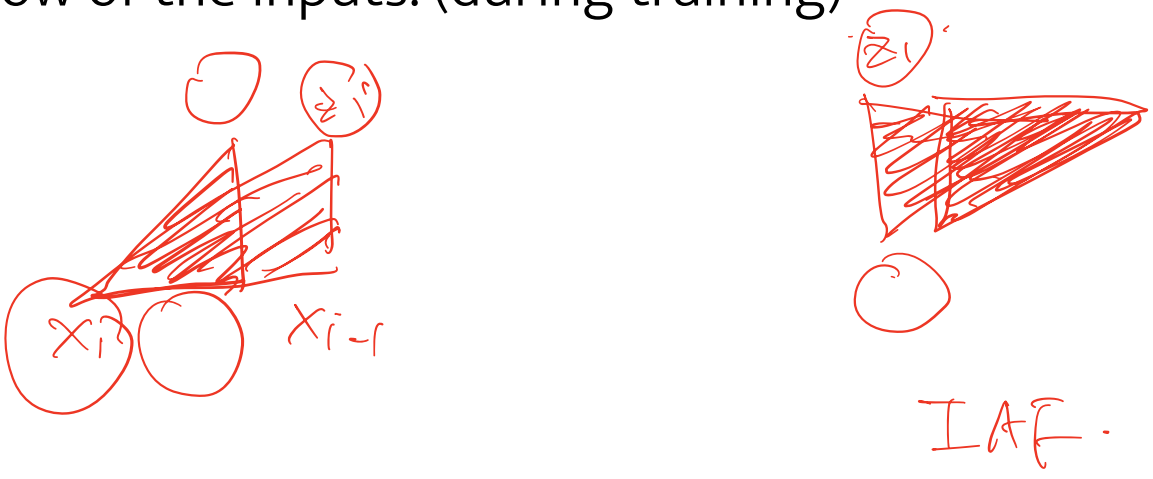
$$L = \sum_i \frac{1}{2}||z_i||_2^2 - \sum_i \log \sigma_i.$$

Papamakarios et al. Masked Autoregressive Flow for Density Estimation. NIPS 2017.

NYU

MAF vs IAF

- MAF inference can be parallelized by directly passing different window of the inputs. (during training)



Kingma et al. Improving Variational Inference with Inverse Autoregressive Flow. NIPS 2016.

NYU

MAF vs IAF

- MAF inference can be parallelized by directly passing different window of the inputs. (during training)
- Sampling needs to be sequential: One input dimension at a time.

Kingma et al. Improving Variational Inference with Inverse Autoregressive Flow. NIPS 2016.

NYU

MAF vs IAF

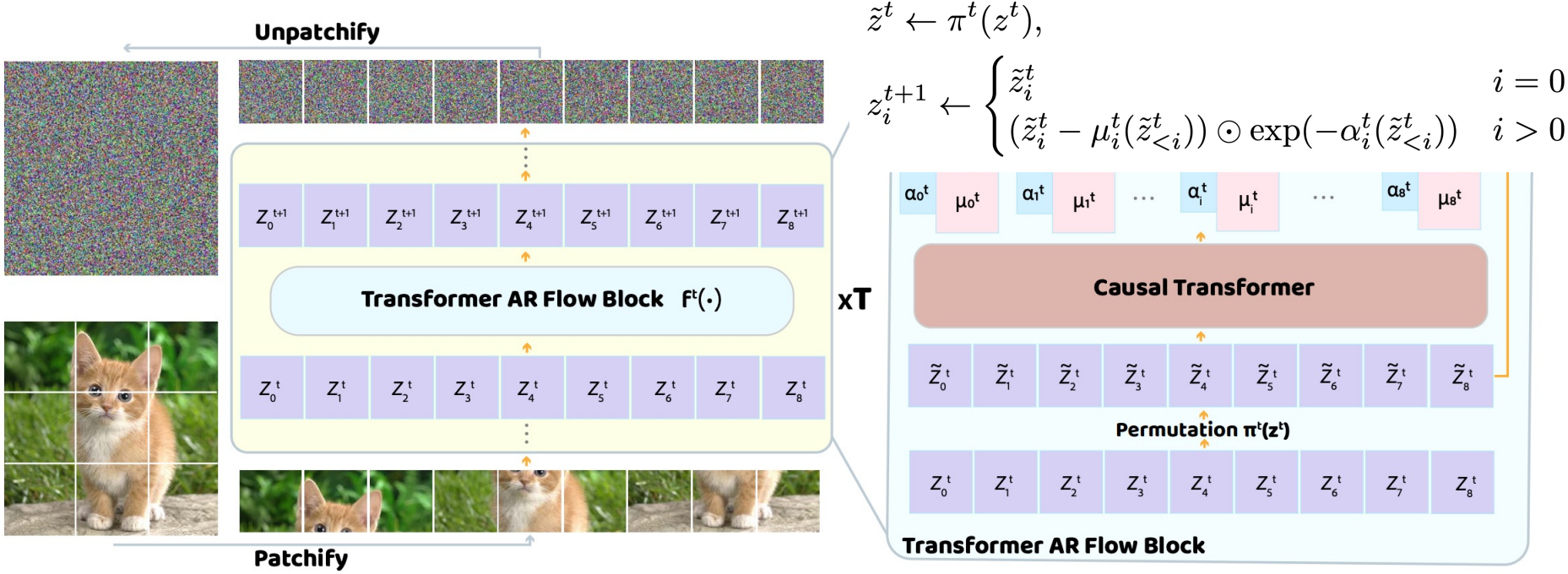
- MAF inference can be parallelized by directly passing different window of the inputs. (during training)
- Sampling needs to be sequential: One input dimension at a time.
- Inverse Autoregressive Flow (IAF) does the reverse. Parallel sample, sequential inference (slow training).

Kingma et al. Improving Variational Inference with Inverse Autoregressive Flow. NIPS 2016.

NYU

Block Autoregressive Flows

- Using blocks, permutations, and element-wise operations



$$\tilde{z}^t \leftarrow \pi^t(z^t),$$

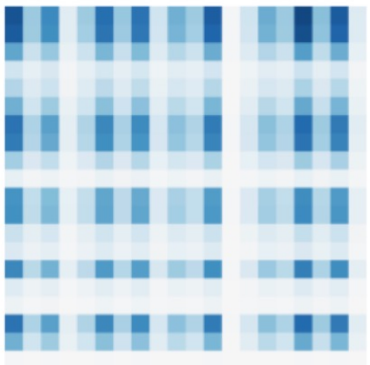
$$z_i^{t+1} \leftarrow \begin{cases} \tilde{z}_i^t & i = 0 \\ (\tilde{z}_i^t - \mu_i^t(\tilde{z}_{<i}^t)) \odot \exp(-\alpha_i^t(\tilde{z}_{<i}^t)) & i > 0 \end{cases}$$

Zhai et al. Normalizing Flows are Capable Generative Models. ICML 2025.

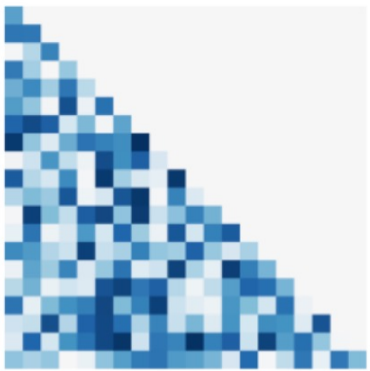
NYU

Full-Rank Jacobian

- Is it possible to make the architecture more flexible?



Low-rank



Auto-regressive

Full-rank

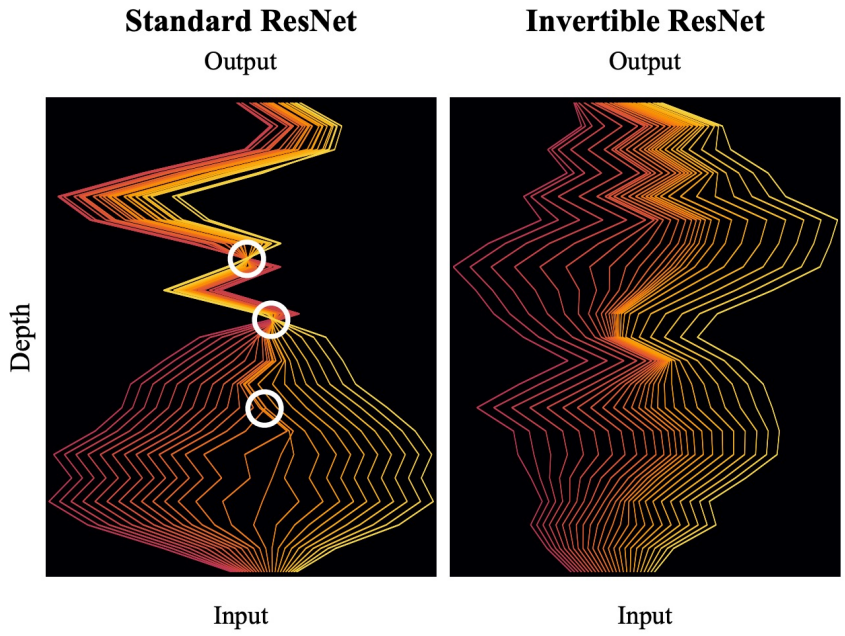
Image Credit

Behrmann et al. Invertible Residual Networks. ICML 2019.

NYU

Invertibility of ResNet

$$x_{t+1} \leftarrow x_t + g_{\theta_t}(x_t)$$



Behrmann et al. Invertible Residual Networks. ICML 2019.

NYU

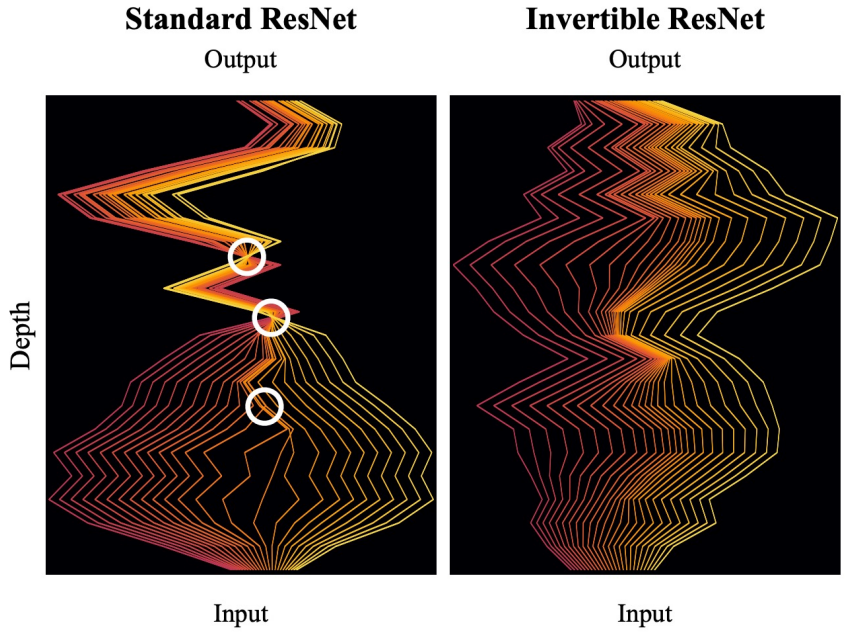
Invertibility of ResNet

$$x_{t+1} \leftarrow x_t + g_{\theta_t}(x_t)$$

- The inverse can be recovered from fixed point iteration if Lip(g) < 1.

$$x_t^0 := x_{t+1}$$

$$x_t^{k+1} := x_{t+1} - g_{\theta_t}(x_t^k)$$



Behrmann et al. Invertible Residual Networks. ICML 2019.

NYU

Invertibility of ResNet

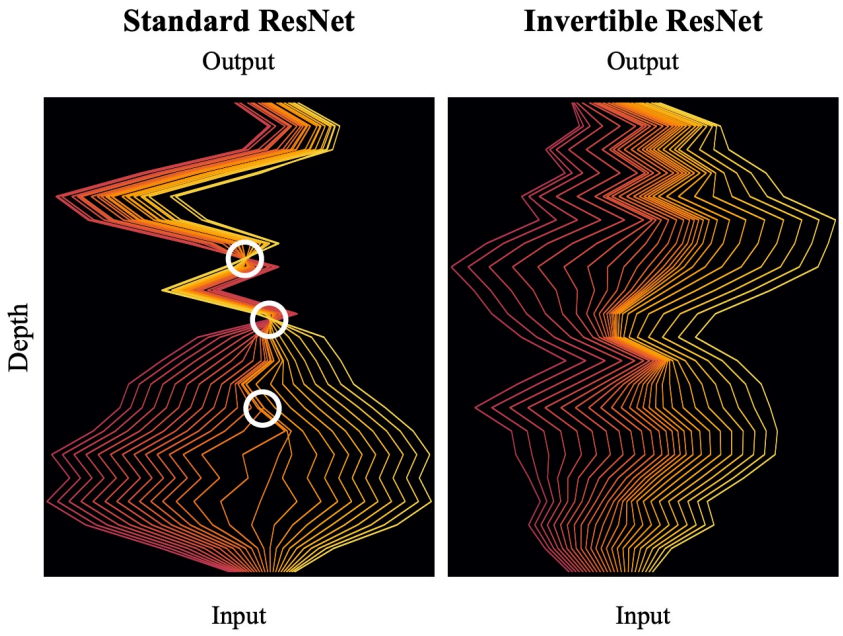
$$x_{t+1} \leftarrow x_t + g_{\theta_t}(x_t)$$

- The inverse can be recovered from fixed point iteration if Lip(g) < 1.

$$x_t^0 := x_{t+1}$$

$$x_t^{k+1} := x_{t+1} - g_{\theta_t}(x_t^k)$$

- MAF uses lower triangular Jacobian structure while iResNet uses approximation.



Behrmann et al. Invertible Residual Networks. ICML 2019.

NYU

Estimating Jacobian Determinant

Grathwohl et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. ICML 2019.
Behrmann et al. Invertible Residual Networks. ICML 2019.

NYU

Estimating Jacobian Determinant

- Using power series to estimate det J.

$$\log|\det J_F| = \operatorname{tr}(\log J_F).$$

$$\operatorname{tr}(\log(I + J_g)) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\operatorname{tr}(J_g^k)}{k}.$$

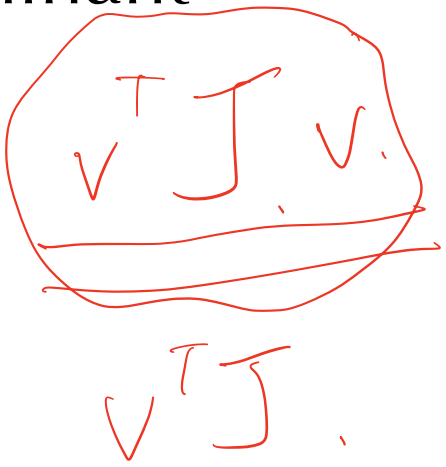
Grathwohl et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. ICML 2019.
Behrmann et al. Invertible Residual Networks. ICML 2019.

Estimating Jacobian Determinant

- Using power series to estimate det J.

$$\log|\det J_F| = \operatorname{tr}(\log J_F).$$

$$\operatorname{tr}(\log(I + J_g)) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\operatorname{tr}(J_g^k)}{k}.$$



- To compute the trace without the full Jacobian. Use a stochastic estimate:

$$\operatorname{tr}(A) = \mathop{\mathbb{E}}_{v}[v^\top A v].$$

Grathwohl et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. ICML 2019.
Behrmann et al. Invertible Residual Networks. ICML 2019.

NYU

Estimating Jacobian Determinant

- Using power series to estimate det J.

$$\log|\det J_F| = \operatorname{tr}(\log J_F).$$

$$\operatorname{tr}(\log(I + J_g)) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\operatorname{tr}(J_g^k)}{k}.$$

- To compute the trace without the full Jacobian. Use a stochastic estimate:

$$\operatorname{tr}(A) = \mathop{\mathbb{E}}_{v}[v^\top A v].$$

Draw v from $\mathcal{N}(0, I)$
$w^T := v^T$
$\ln\det := 0$
for $k = 1$ **to** n **do**
  $w^T := w^T J_g$ (vector-Jacobian product)
  $\ln\det := \ln\det + (-1)^{k+1} w^T v/k$
end for

Grathwohl et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. ICML 2019.
Behrmann et al. Invertible Residual Networks. ICML 2019.

NYU

Summary: Normalizing Flow

- Requires invertibility and tractability on Jacobian determinant
- Constraint on the functional form
- Or estimation of Jacobian / iterative inverse (numerical approximation).

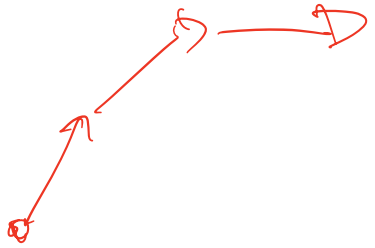
power series + stochastic estimate.

fixed.

Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$



Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$

CNF

- Can be written in continuous time (continuous normalizing flow).

$$x_t = \phi_t(x_0) = x_0 + \int_0^t \delta u_s(x_s)\, ds.$$

NYU

Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$

- Can be written in continuous time (continuous normalizing flow).

$$x_t = \phi_t(x_0) = x_0 + \int_0^t \delta u_s(x_s)\ ds.$$

- Integration: Euler, or directly run an ODE solver

NYU

Inference as Integration

- Residual Flow: $\phi(x) = x + \delta u(x)$

$$\frac{\phi(x) - x}{\delta} = u(x)$$

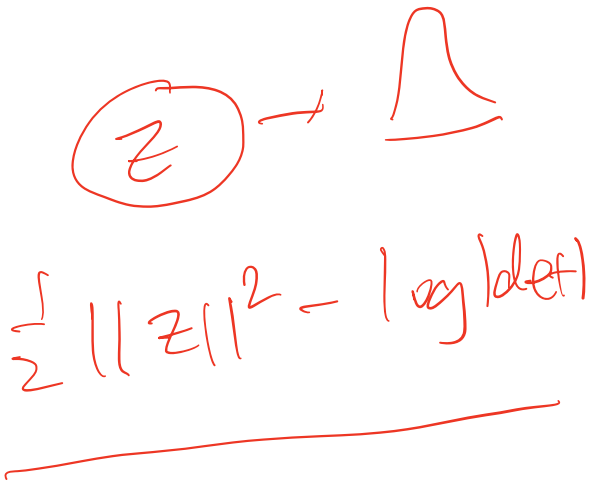
- Can be written in continuous time (continuous normalizing flow).

$$x_t = \phi_t(x_0) = x_0 + \int_0^t \delta u_s(x_s)\ ds.$$

- Integration: Euler, or directly run an ODE solver
- Backprop solving an ODE without storing intermediate states

Flow Matching

- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.

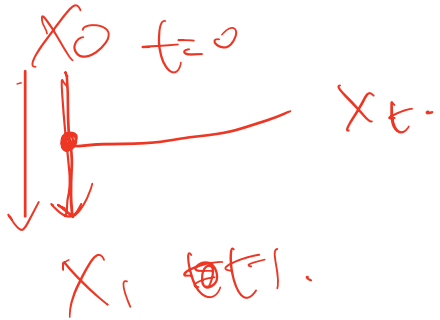


Lipman et al. Flow Matching for Generative Modeling. ICLR 2023.

NYU

Flow Matching

- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.
- Given an original input x_1 and a noise x_0, $x_t = (1-t)x_0 + tx_1$



Flow Matching

- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.
- Given an original input x_1 and a noise x_0, $x_t = (1-t)x_0 + tx_1$
- Want to fit a velocity $v_t(x)$ to match $u_t = x_1 - x_0$

Lipman et al. Flow Matching for Generative Modeling. ICLR 2023.

NYU

Flow Matching

- Taking integrals during training is cumbersome, and the learning signal from the top latent is weak.
- Given an original input x_1 and a noise x_0, $x_t = (1-t)x_0 + tx_1$
- Want to fit a velocity $v_t(x)$ to match $u_t = x_1 - x_0$
- Flow Matching learning objective:

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t,x_0,x_1}[||v(x_t,t) - u_t(x_0,x_1)||^2]$$

Lipman et al. Flow Matching for Generative Modeling. ICLR 2023.

NYU

Diffusion Models

- A popular generative model formulation.

NYU

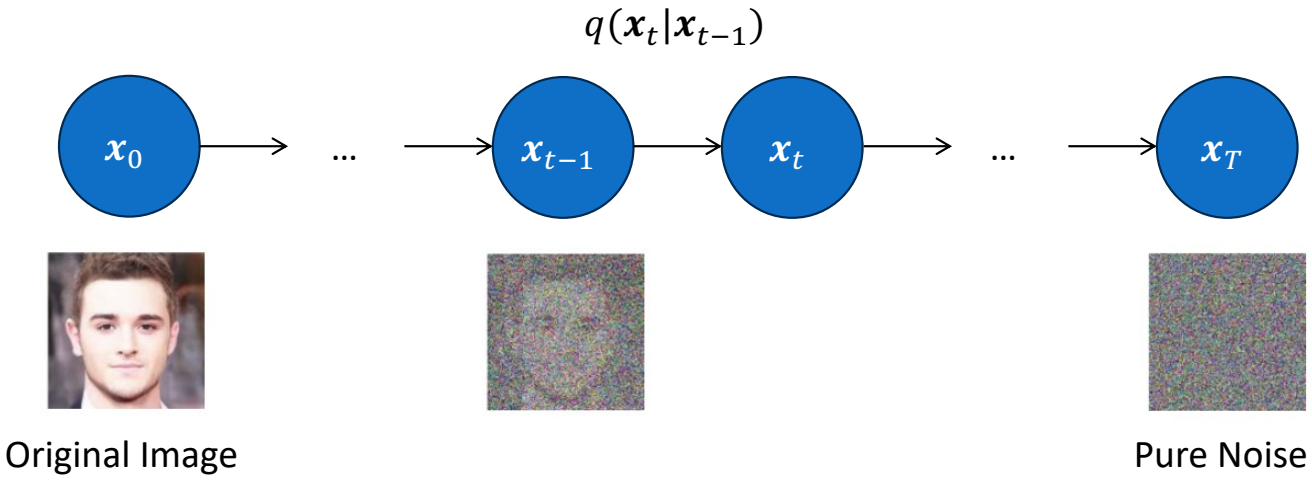
Diffusion Models

- A popular generative model formulation.
- The intuition is to iteratively denoise from Gaussian random noises into an image.

NYU

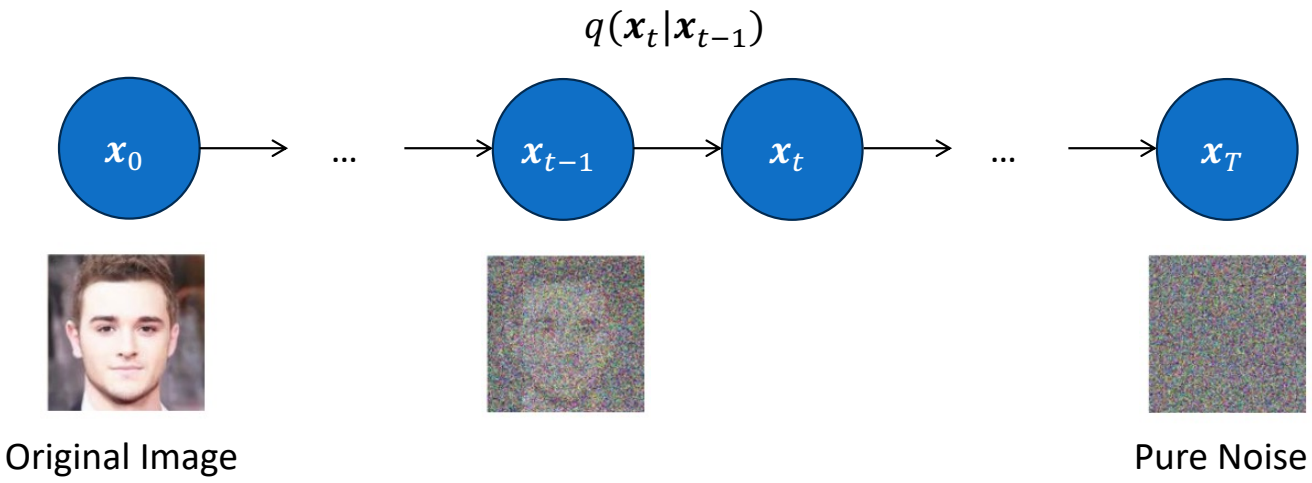
Diffusion Models

- A popular generative model formulation.
- The intuition is to iteratively denoise from Gaussian random noises into an image.



Diffusion Models

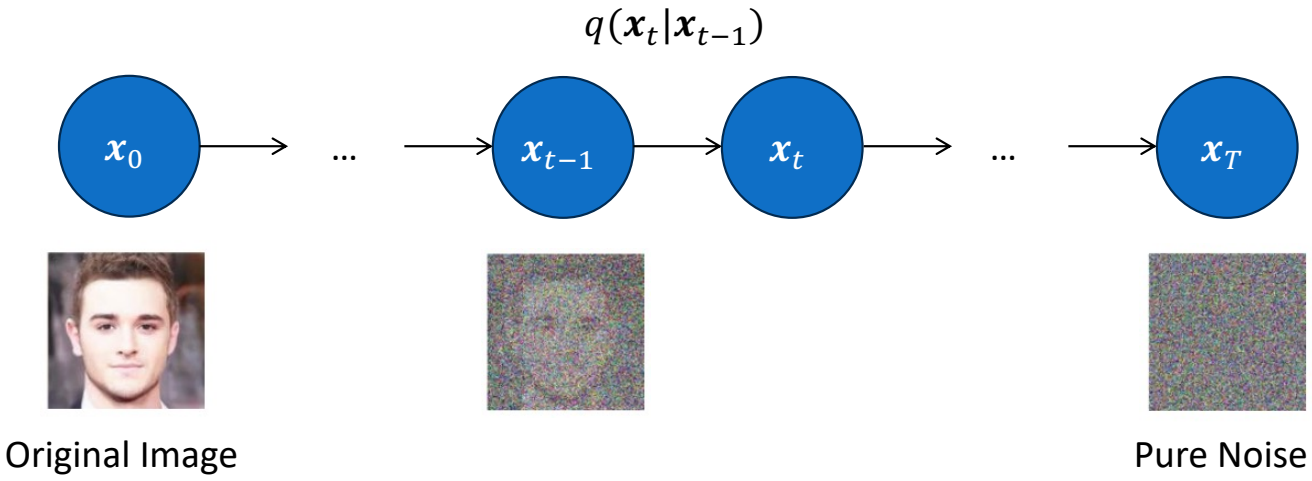
- Forward process: $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$.



NYU

Diffusion Models

- Forward process: $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$.
- You can also write: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t,\ \epsilon_t \sim \mathcal{N}(0, I)$.



Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$

NYU

Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$
- Write x_t as a function of x_0 with larger noises:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I).$$

NYU

Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$
- Write x_t as a function of x_0 with larger noises:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I).$$

- Cumulative schedule: $\alpha_t = 1 - \beta_t.$ $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$

NYU

Properties of the Forward Process

- Forward process: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t$
- Write x_t as a function of x_0 with larger noises:

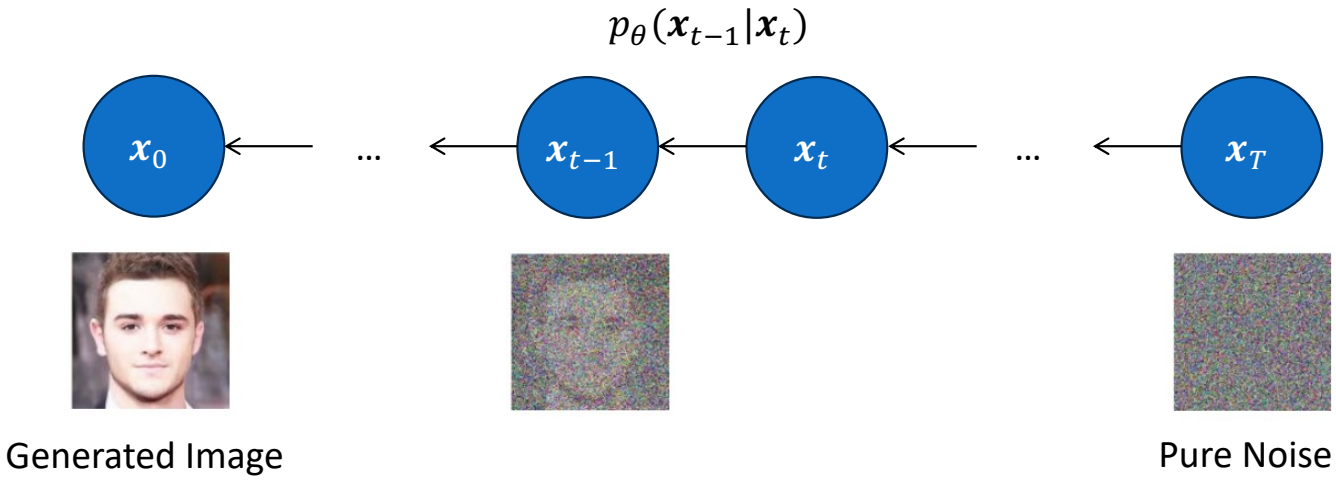
$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I).$$

- Cumulative schedule: $\alpha_t = 1 - \beta_t.\ \ \bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$
- $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon.$

NYU

Reverse Process

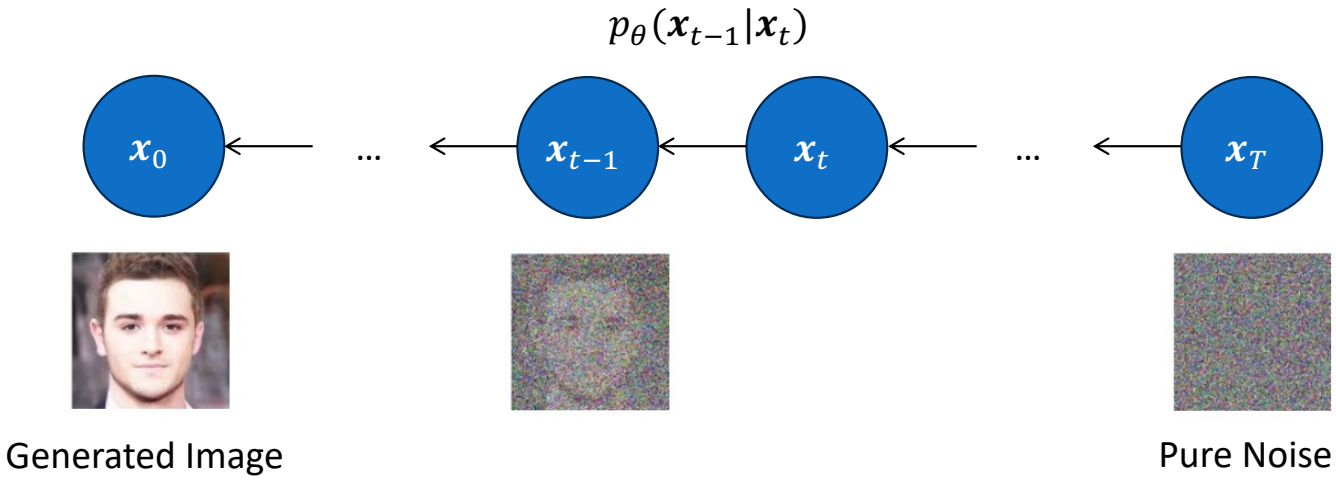
- A generative model wants to predict x_0 from x_T.



NYU

Reverse Process

- A generative model wants to predict x_0 from x_T.
- The reverse process transition is also Gaussian distributed. But we don't know what the transition will be like just by looking at the noisy image!

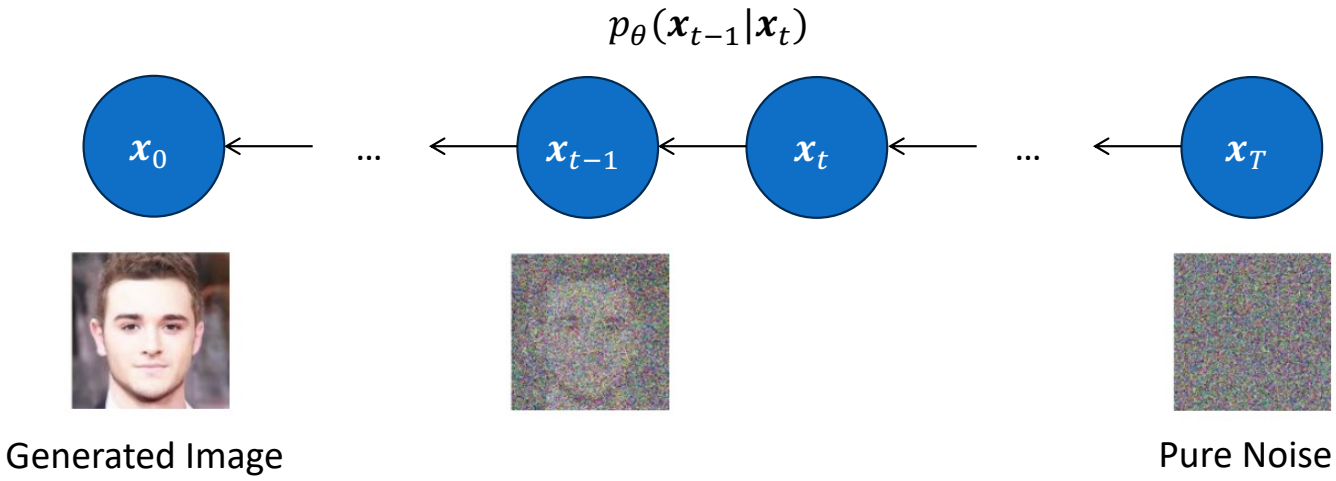


NYU

Reverse Process

- So, we need to learn a "model":

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)).$$

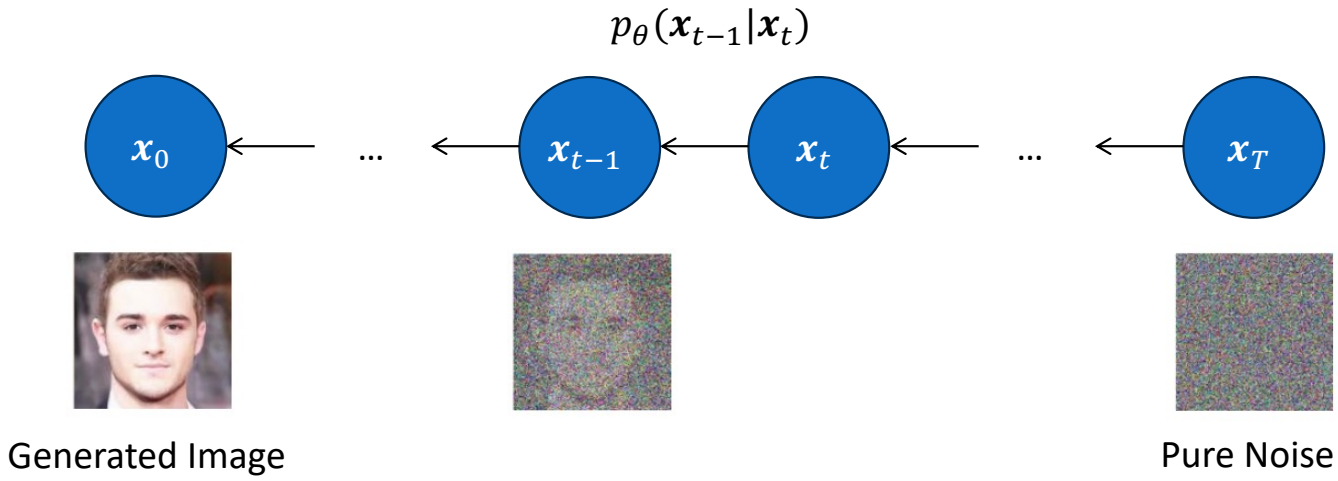


Reverse Process

- So, we need to learn a "model":

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)).$$

- μ_θ is the denoising vector.



Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.

NYU

Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.
- Bayes rule:

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}.$$

NYU

Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.
- Bayes rule:

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}.$$

- But we don't know the marginal distribution $q(x_{t-1})$. We only know q_T and $q(x_t|x_{t-1})$.

NYU

Reverse Process

- Compute μ_θ? Derive $p(x_{t-1}|x_t)$.
- Bayes rule:

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}.$$

- But we don't know the marginal distribution $q(x_{t-1})$. We only know q_T and $q(x_t|x_{t-1})$.
- Solution: Condition on the original input x_0:

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

NYU

Reverse Process

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

NYU

Reverse Process

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t, \tilde{\beta} I).$$

NYU

Reverse Process

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t, \tilde{\beta} I).$$

$$\tilde{\mu}_t = \frac{\sqrt{\alpha_t}\bar{\beta}_{t-1}}{\bar{\beta}_t} x_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{\bar{\beta}_t} x_0 = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon).$$

NYU

Reverse Process

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|x_0)}{q(x_t|x_0)}.$$

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t, \tilde{\beta} I).$$

$$\tilde{\mu}_t = \frac{\sqrt{\alpha_t}\bar{\beta}_{t-1}}{\bar{\beta}_t} x_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{\bar{\beta}_t} x_0 = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon).$$

- Want: train up a μ_θ to match with $\tilde{\mu}_t$.

NYU

Training

- Sometimes it is more common to predict the denoising vector ϵ instead of μ.

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

NYU

Training

- Sometimes it is more common to predict the denoising vector ϵ instead of μ.
- Randomly pick at a time step and predict the difference between the noisy and the original.

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

NYU

Training

- Sometimes it is more common to predict the denoising vector ϵ instead of μ.
- Randomly pick at a time step and predict the difference between the noisy and the original.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon),$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

NYU

Sampling

- How do we sample an image?

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

NYU

Sampling

- How do we sample an image?
- We know μ_θ which will help us transition from x_t to x_{t-1}.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

NYU

Sampling

- How do we sample an image?
- We know μ_θ which will help us transition from x_t to x_{t-1}.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t)).$$

- Sample from $\mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2)$. σ_t can either be β_t or $\tilde{\beta}_t$ derived from the posterior.

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

NYU

Deep Generative Models Summary

Guided Diffusion

- We can add guidance on the diffusion updates at inference time.

Classifier Guidance / External Score Model

$$x_{t-1} \leftarrow \text{sample from } \mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$$

$$\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$$
$$x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$$

Dhariwal & Nichol. Diffusion Models Beat GANs on Image Synthesis. NeurIPS 2021.
Ho & Salimans. Classifier-Free Diffusion Guidance. 2022.

NYU

Guided Diffusion

- We can add guidance on the diffusion updates at inference time.

Classifier Guidance / External Score Model

$$x_{t-1} \leftarrow \text{sample from } \mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$$

$$\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$$
$$x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$$

- We also can train a conditional diffusion model.

repeat
- $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
- $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
- $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
- $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
- Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model

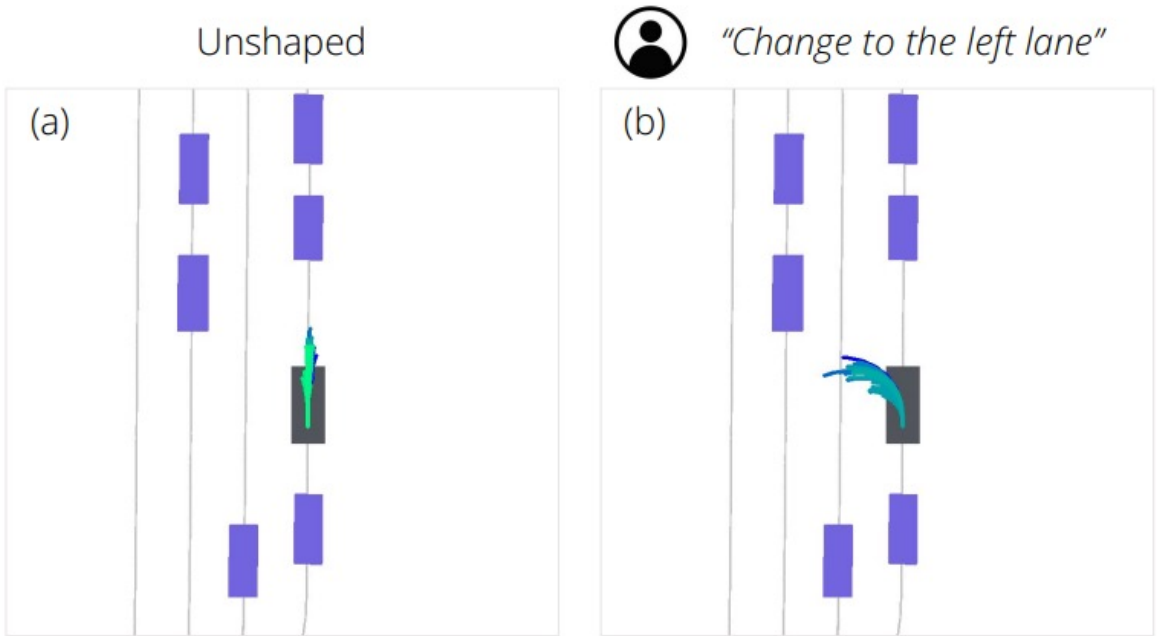
until converged

Dhariwal & Nichol. Diffusion Models Beat GANs on Image Synthesis. NeurIPS 2021.
Ho & Salimans. Classifier-Free Diffusion Guidance. 2022.

NYU

Test-Time Guidance / Adaptation

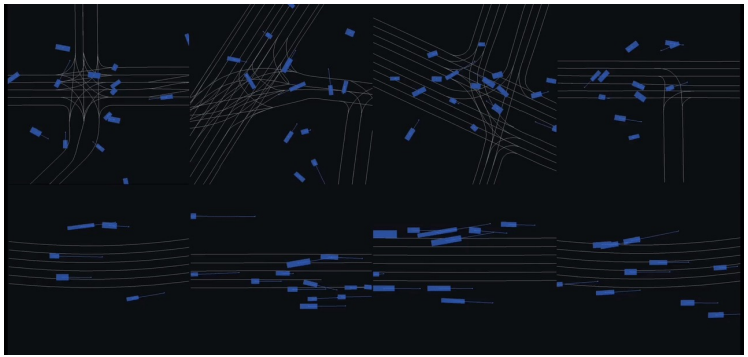
- Diffusion can be combined / guided at test time.

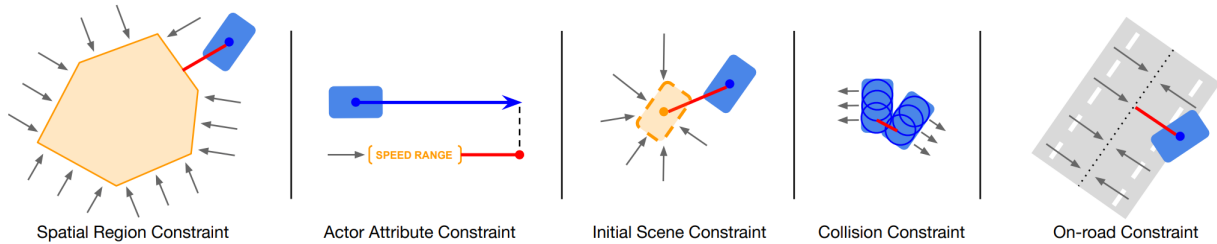


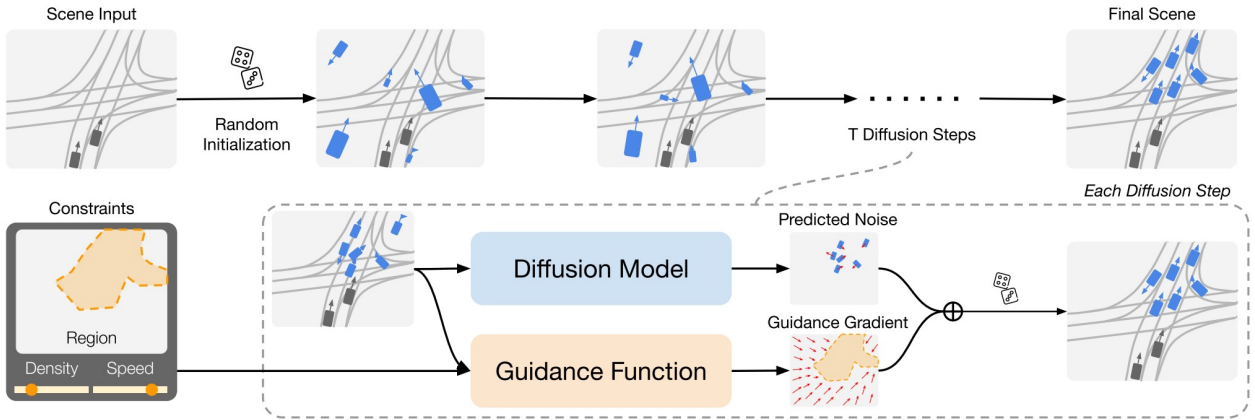
Yang et al. Diffusion-ES: Gradient-free Planning with Diffusion for Autonomous and Instruction-guided Driving. CVPR 2024.

NYU

Generating Simulation Scenes



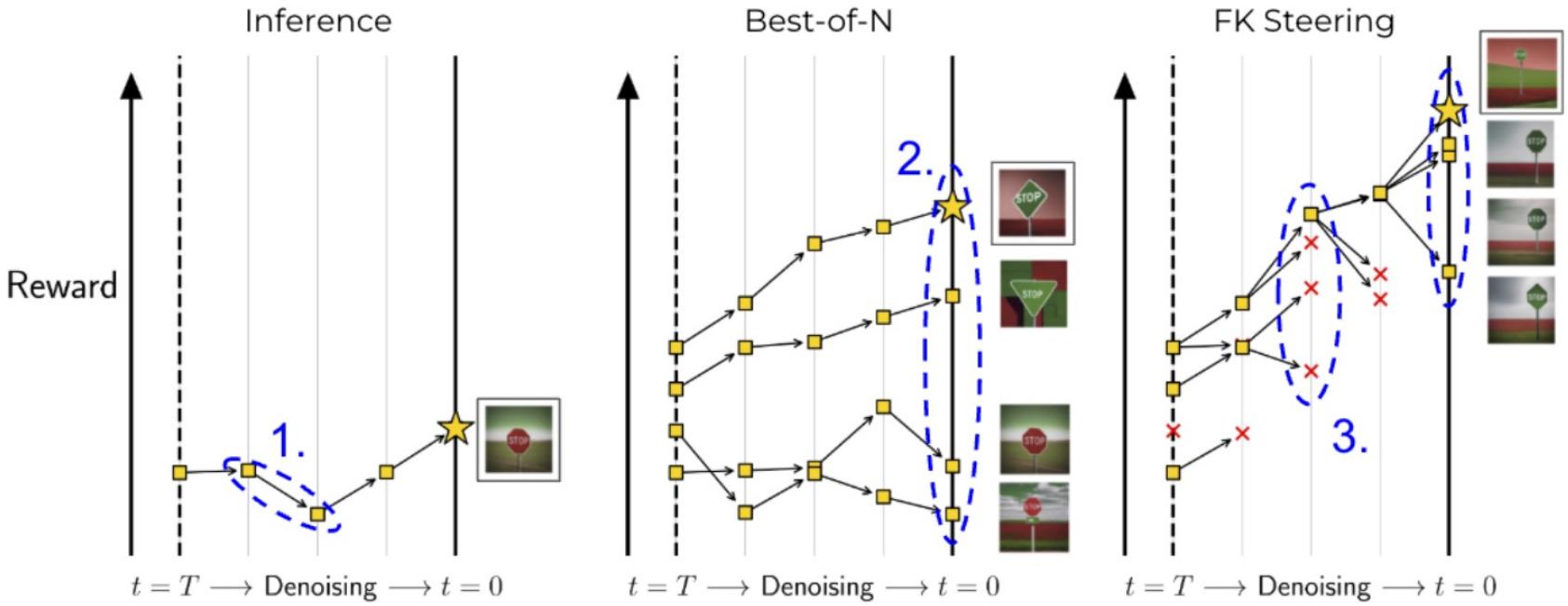




Lu et al. SceneControl: Diffusion for Controllable Traffic Scene Generation. ICRA 2024.

NYU

Non-Differentiable Rewards?



Prompt: *"a green stop sign in a red field"*

(1) Iteratively de-noise $x_T \rightarrow x_{T-1} \rightarrow \ldots \rightarrow x_0$.
(2) Generate multiple samples (*particles*).
(3) Resample promising particles at *intermediate* steps.

Singhal et al. A general framework for inference-time scaling and steering of diffusion models. ICML 2025.

NYU

Diffusion Models for Detection

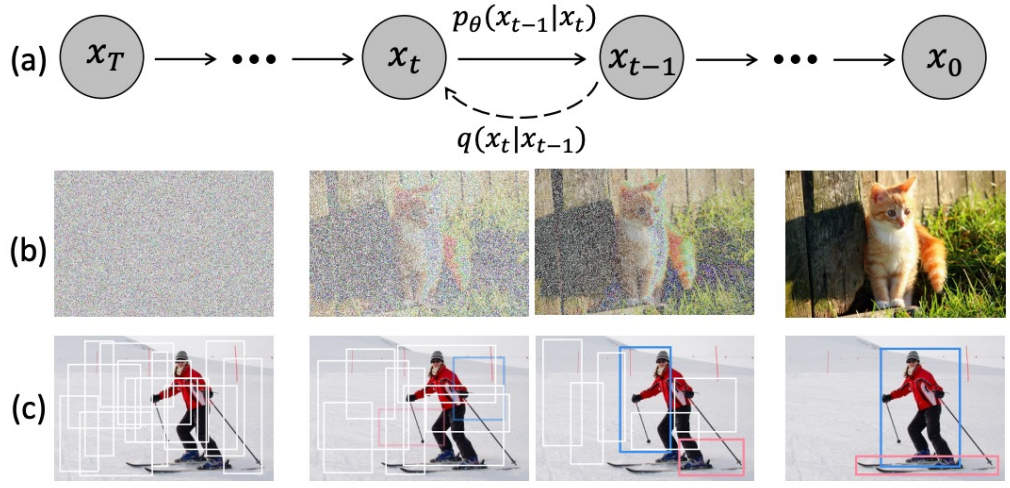
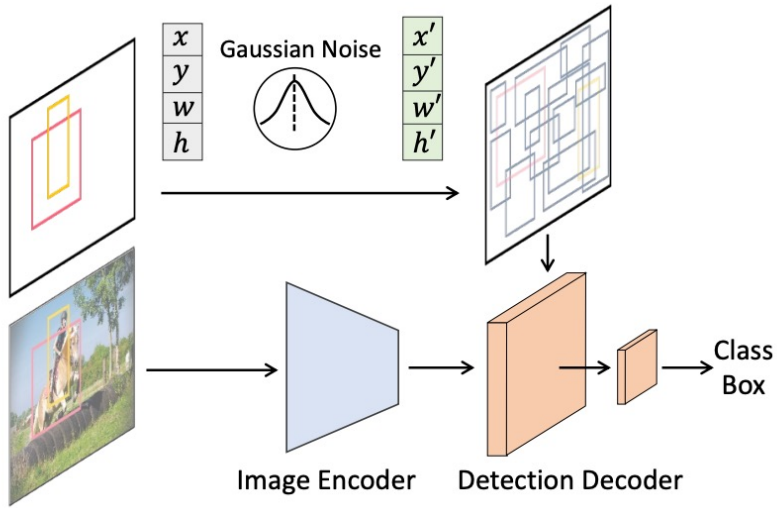


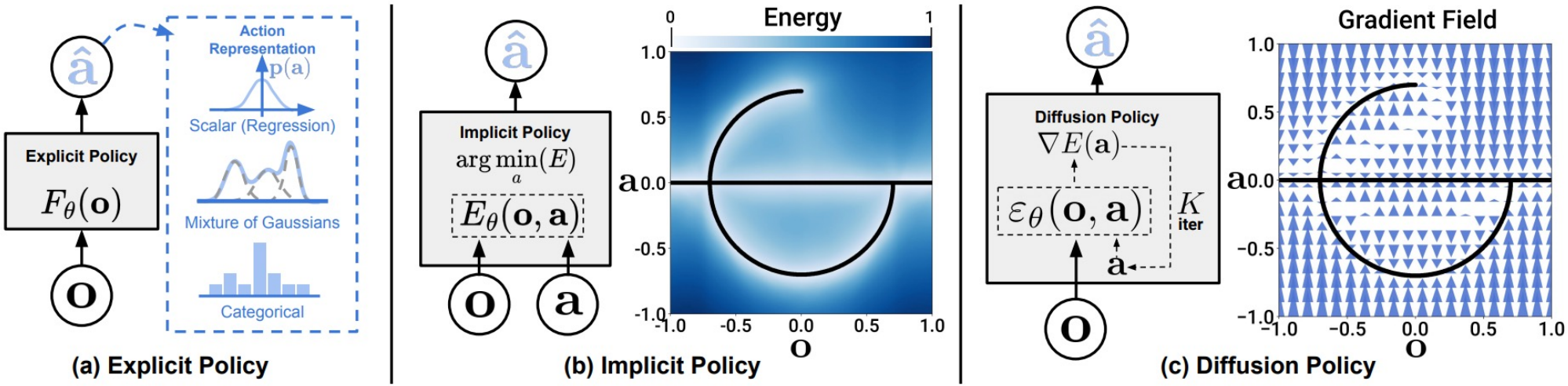
Figure 1. **Diffusion model for object detection**. (a) A diffusion model where q is the diffusion process and p_θ is the reverse process. (b) Diffusion model for image generation task. (c) We propose to formulate object detection as a denoising diffusion process from noisy boxes to object boxes.



Chen et al. DiffusionDet: Diffusion Model for Object Detection. ICCV 2023.

NYU

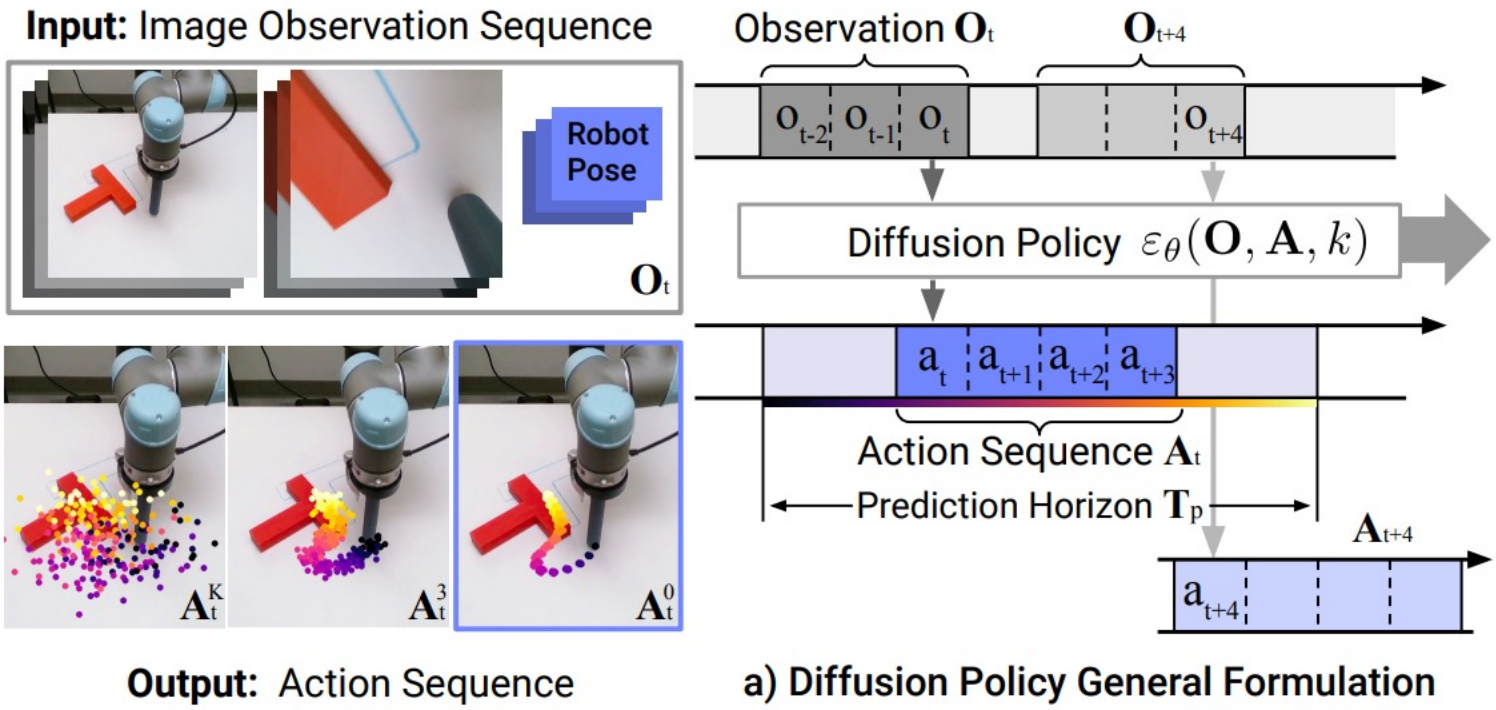
Planning and Control



Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023.

NYU

Diffusion for Planning and Control



Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. RSS 2023.

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about deep generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Normalizing flow
  - Diffusion

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about deep generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Normalizing flow
  - Diffusion
- Understand relations, pros and cons.

NYU

Summary: DL for Structured Outputs

- Expanding the output dimension has limitations.
- Requires us thinking about deep generative models.
  - Graphical models
  - Autoregressive
  - Energy-based
  - Normalizing flow
  - Diffusion
- Understand relations, pros and cons.
- Application in embodied environments.

NYU

What's Next

- Tutorial on HPC
- Next week: 3D vision, mapping

NYU