COMP 652

A WHIRLWIND TOUR OF GENERATIVE MODELS

Philip Bachman

OUTLINE

Introduction

- What's the deal with generative models?
- Density estimation and learning by generating
- Cool settings for generative models

Estimating densities

- Building blocks, autoregressions, Bayesian models

Learning by generating

- GANs mostly, and maybe some RL

Branching out

- Generating functions, world models, ungenerative models

INTRODUCTION

INTRO – WHAT'S THE DEAL? (1/2)

What is a generative model?

- This isn't entirely clear
- Maximum likelihood is density estimation
- RL agents learn to generate trajectories
- GANs generate images, sound, text, etc…

Why care about generative models?

- Most problems are fundamentally about predicting and controlling
- Inference and generation are the core of prediction and control

Estimate densities or generate data?

- Density estimation focuses on analyzing observed things
- Generation is about creating new things

INTRO – WHAT'S THE DEAL? (2/2)

Uses for density estimation

- All maximum likelihood training is density estimation
- Good densities can measure surprise and detect outliers

Uses for data generation

- RL agents generating nicely distributed trajectories is great
- Text to speech, machine translation, slugdogs, etc

Strengths and weaknesses

- Both have tons of uses
- Density estimation is usually more stable,
- Learning can be more exploratory and creative

INTRO — ESTIMATING DENSITIES (1/2)

Building blocks

- Gaussian and categorical distributions
- Mixtures of Gaussians and topic models

Autoregression

- Language models, modeling sequences and vectors
- Orderless autoregression and efficient perception

Applied Autoregression

- Conditional density estimation for everything
- Generating translations, images, speech, and music

INTRO – ESTIMATING DENSITIES (2/2)

Variational Autoencoders

- Learning compatible inference and generation
- Some challenges and benefits

Extending the VAE

- Deconstructing the VAE
- Clever ways to reuse the components of VAE

Random thoughts

- Do we primarily care about generation, or about inference?
- Are priors important? Are they related to simplicity or disentanglement?

INTRO – LEARNING BY GENERATING (1/1)

Generative Adversarial Networks

- The basic GAN, a competition between critic and creator
- Challenges in the adversarial setting (they are tough!)

Extending the GAN

- Deconstructing the GAN
- Clever ways to apply the core concepts

How far have we come?

- What is the best GAN? (the answer may shock you!)

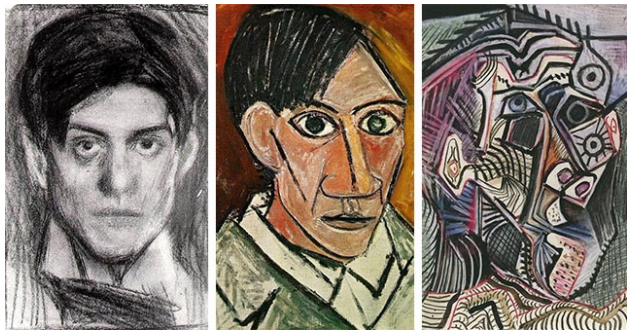
INTRO – BRANCHING OUT (1/2)

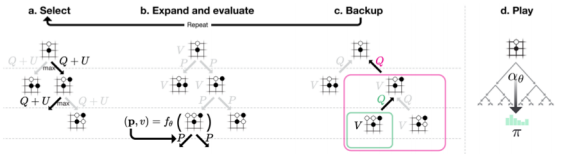
Generative models for functions

- What is a function? Can we put a distributions on it?
- Gaussian processes and VFunc

Generative models for planning in RL

- Model-based RL can be super sample-efficient.
- Can we learn models good enough to capture this?





INTRO – BRANCHING OUT (2/2)

Generative models for curiosity in RL

- Exploration is hard.
- Can we encourage an agent to visit interesting states?
- What is interesting? IDK.

Ungenerative models

- If the world generates data, can we just learn inference?
- Learning mutually informative features. A path to success?

ESTIMATING DENSITIES

BUILDING BLOCKS

Gaussian and categorical distributions

- We want to estimate the probability of a vector $x \sim D(x)$
- If x is continuous, we could try $p(x) \propto \exp(((x-\mu)^2 / \sigma^2)$
- If x is categorical, we could try $p(x = y) = t(y)$, for some table t over y

Mixtures of Gaussians and topic models

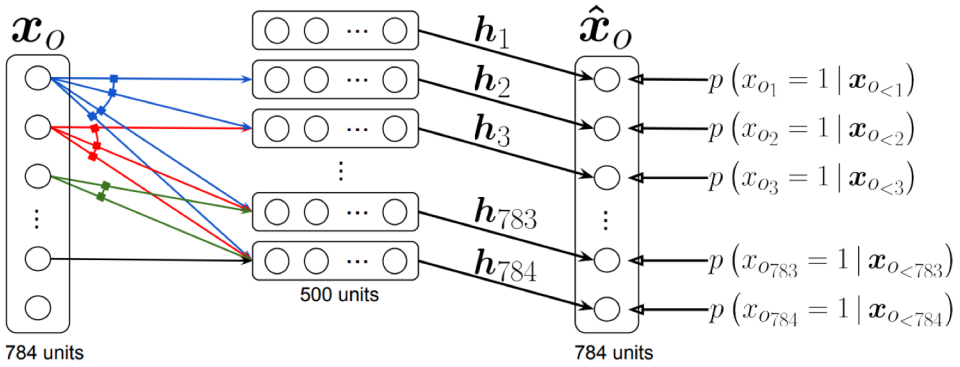
- We can extend the Gaussian model like $p(x) \propto \sum_c \exp\left((x-\mu_c)^2/\sigma_c^2\right)p(c)$
- We can extend the categorical model like $p(x = y) \propto \sum_c t_c(y)p(c)$
- Each c defines a distribution, and we average over their probabilities
- Maximum likelihood becomes non-trivial, because now we need inference
- Inference is about finding $p(c|x)$, which is called the "posterior"

These are building blocks for other models!

AUTOREGRESSION (1/4)

What is autoregression?

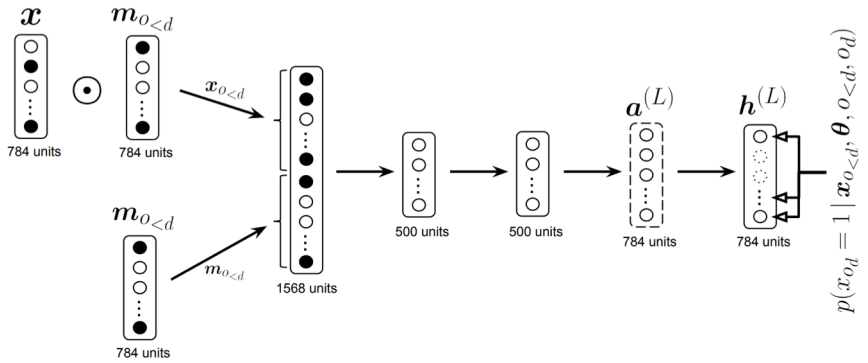
- In autoregression, we use Bayes rule to decompose a joint distribution
- We can estimate $p(x1, x2, x3)$ like $p(x3|x1, x2)p(x2|x1)p(x1)$
- This lets us build a complex distribution from many simple distributions
- E.g., we can predict a sequence of words by composing categorical distributions



AUTOREGRESSION (2/4)

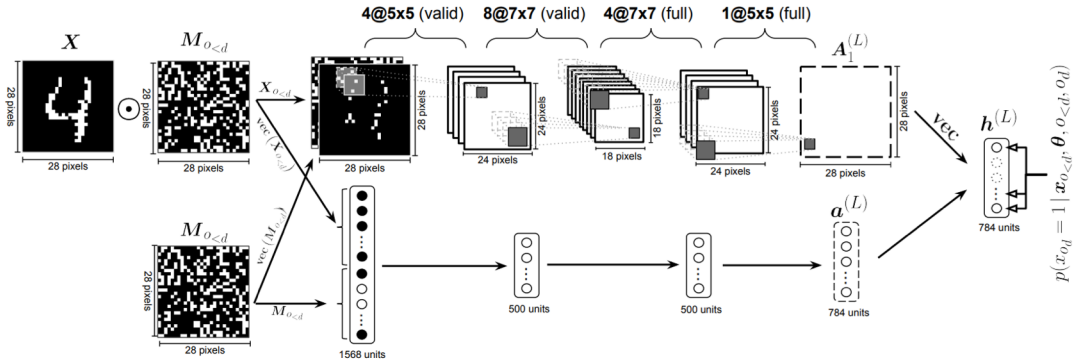
Modeling distributions over vectors

- Suppose we have a vector x, rather than a sequence $(x1, x2, x3)$
- We can assume an order over dimensions of x, then autoregressify
- Neural Autoregressive Distribution Estimation (NADE) does just this



Orderless autoregression

- It's possible to modify NADE to simultaneously work with many orderings
- Orderless autoregression is like perception
- We know where we've looked and what we've seen, then predict what we will see somewhere else



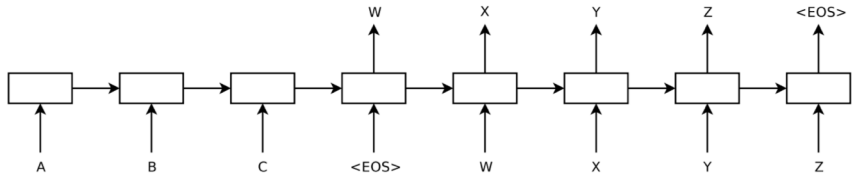
AUTOREGRESSION (3/4)

Conditional density estimation

- Many interesting applications of autoregressive models are conditional
- I.e., we have some input and want to generate some output
- We can simply define $p(x|c) = p(x_3|x_2, x_1, c)p(x_2|x_1, c)p(x_1|c)$
- This is a mixture model for $p(x)$, but with c forced by the world

Machine Translation

- This is a canonical example of conditional autoregression
- Current models are all neural nets (as far as I know)
- Seq2Seq, Transformer, etc



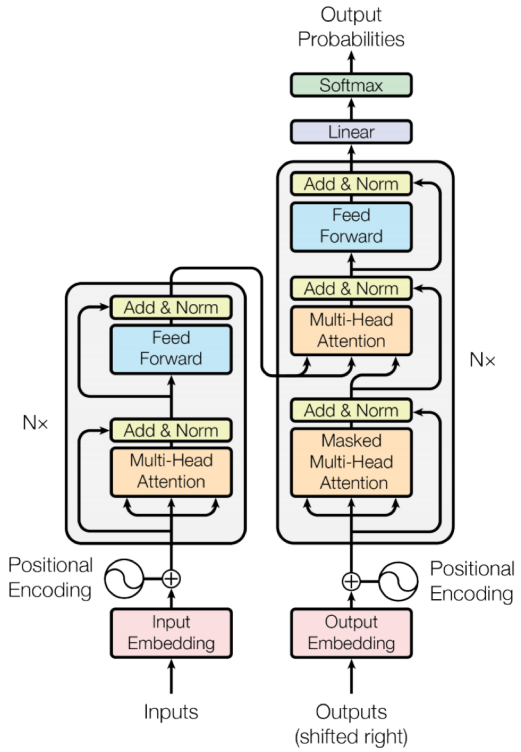
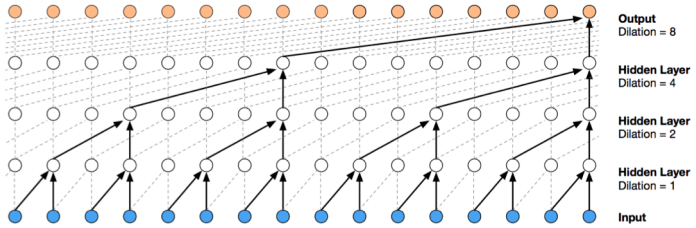


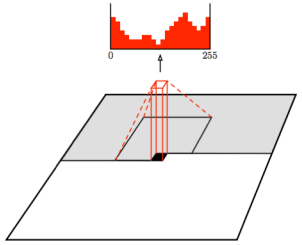
Figure 1: The Transformer - model architecture.

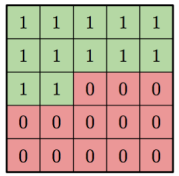
AUTOREGRESSION (4/4)

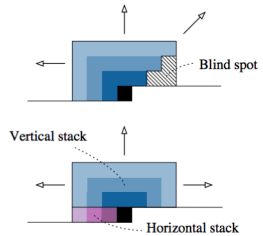
Generating high fidelity audio

- We can generate audio one sample at a time (>1khz), it's crazy
- Even minimal coherence requires long-term structure (100 samples in <0.1 seconds)
- We can use fancy masked convolutions for efficiency (generation is still tricky)



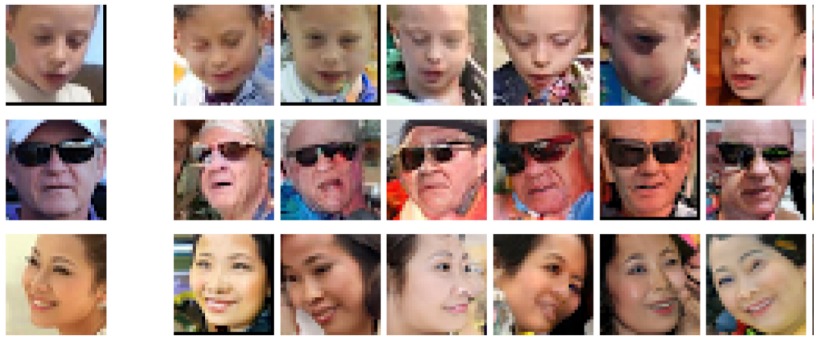






Generating images

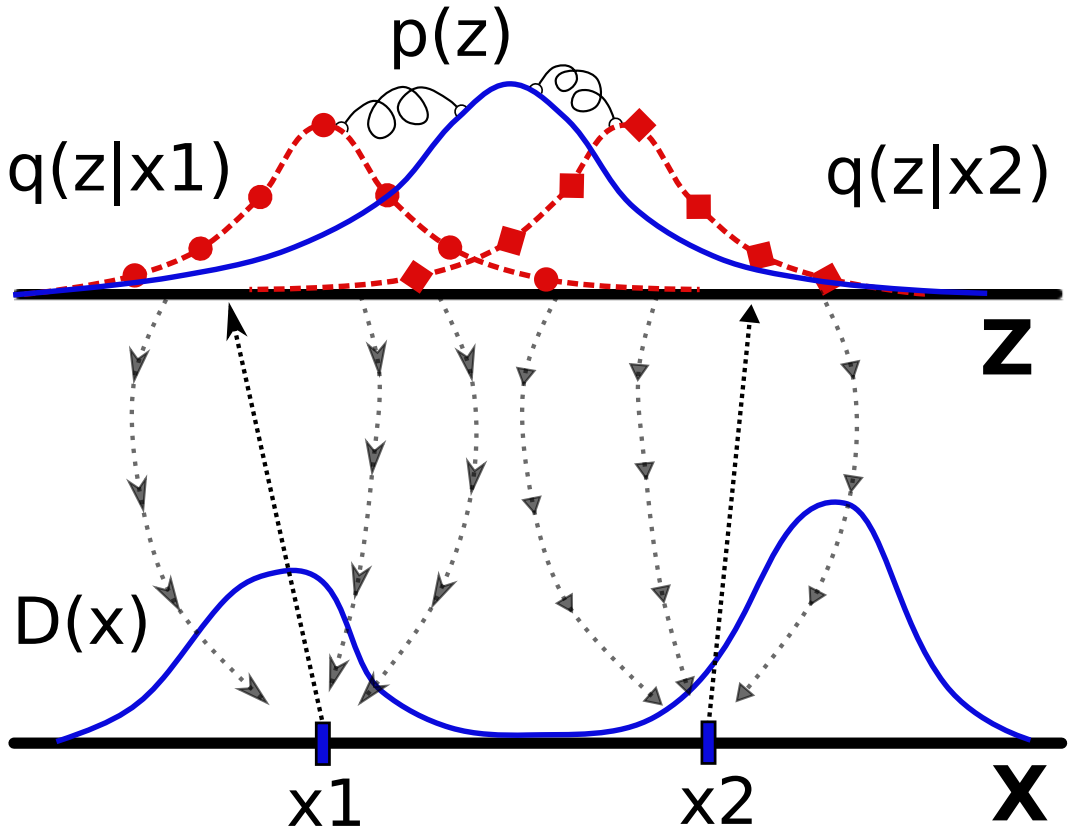
- We can generate images one pixel at a time, it's crazy
- Reasonable coherence requires global spatial structure
- We can use fancy masked convolutions for efficiency



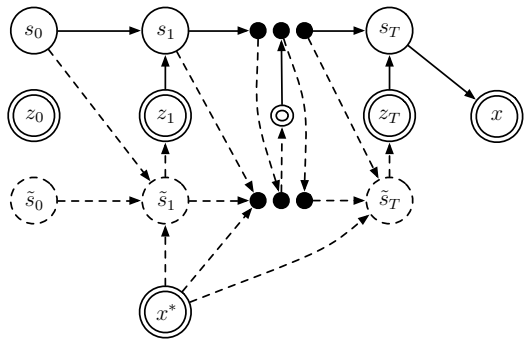
BAYES – VAE RECAP

Variational Autoencoders

- Estimate $p(x)$ using a mixture of very many simple models
- E.g., $p(x) = \sum_z p(x|z)p(z)$, where z covers all 64d fp32 vectors
- Estimating the posterior $p(z|x)$ is tricky, so we use approximate inference
- We optimize the "ELBO" – $\log p(x) \geq \sum_z q(z|x) \log p(x|z) + KL(q(z|x), p(z))$

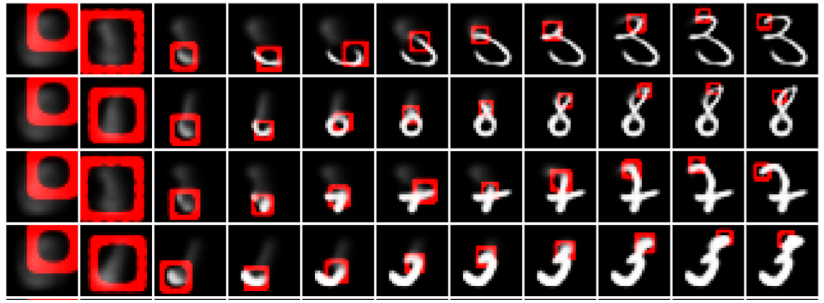


BAYES – BEYOND THE VAE (1/2)



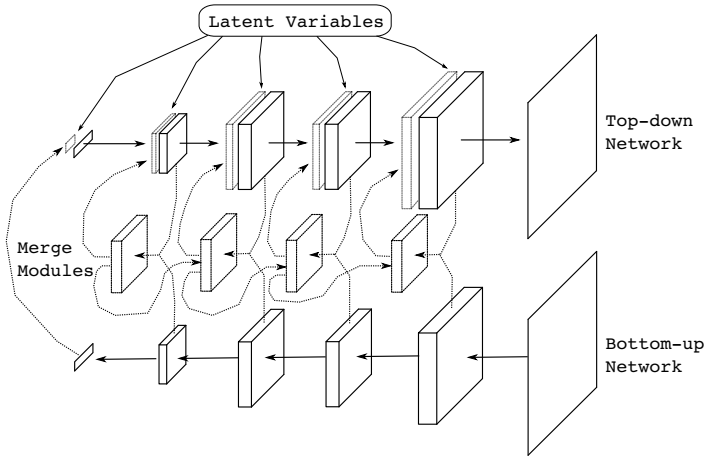
Algorithm 1 GenTrainLoop1(x^*)

Set s_0, $\tilde{s}_0$, and c_0 from constants.
Compute $\text{nll}_0 = -\log p(x^*|c_0)$.
Set kl_0 to 0.
for $t = 1$ to T **do**
  Update $\tilde{s}_t \leftarrow f_\phi(\tilde{s}_{t-1}, g_\phi(s_{t-1}, c_{t-1}, x^*))$.
  Sample $z_t \sim q_\phi(z_t|\tilde{s}_t)$.
  Update $s_t \leftarrow f_\theta(s_{t-1}, z_t)$.
  Update $c_t \leftarrow c_{t-1} + \omega_\theta(s_t)$ (or $\omega_\theta(s_t)$).
  Compute $\text{kl}_t = \text{KL}(q_\phi(z_t|\tilde{s}_t) \,||\, p_\theta(z_t))$.
  Compute $\text{nll}_t = -\log p(x^*|c_t)$.
end for
return $c_{0:T}$, $\text{nll}_{0:T}$, and $\text{kl}_{0:T}$.



Multi-step and hierarchical VAEs

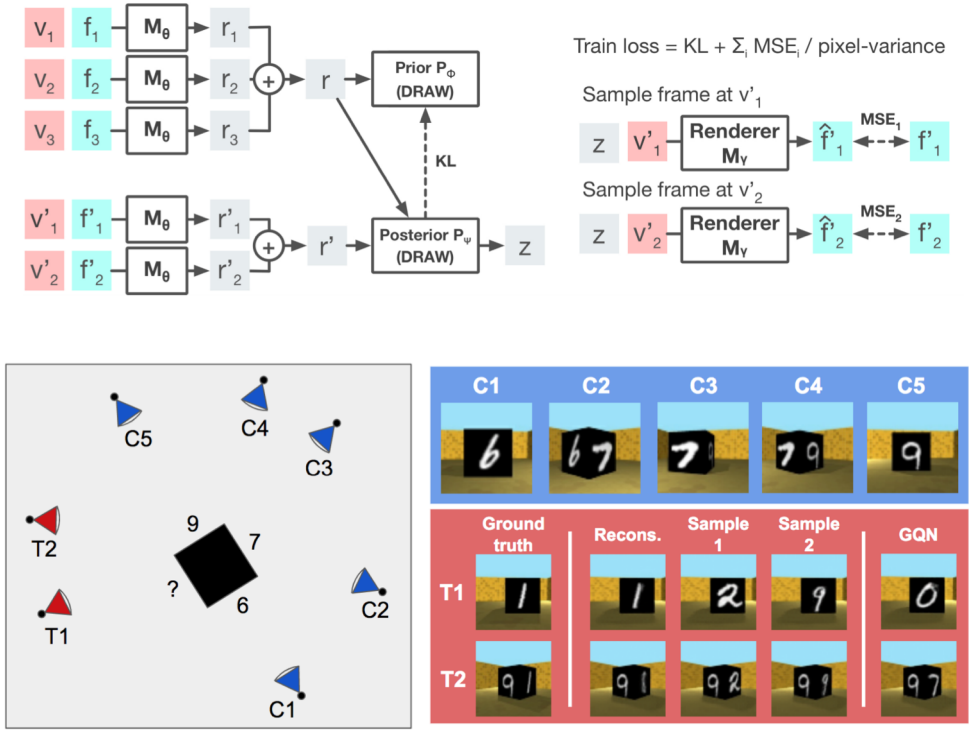
- We can make a VAE which constructs x over multiple steps
- We can iteratively refine x, or the steps can be strictly in z space



BAYES – BEYOND THE VAE (2/2)

Generative Query Networks

- This combines VAEs with a little bit of metalearning
- Observe a scene from several perspectives
- Infer a posterior $q(z|x)$ in which z describes information shared across views of the scene
- Given a new perspective, estimate what we would see there
- I.e. $p(obs|pov) = \sum_z \log p(obs|pov, z) q(z|prior\ obs\ and\ povs)$+ KL term



BAYES – EXTRA THOUGHTS

Do we care more about generation, or inference?

- Generation can be useful for data augmentation, or model-based RL
- Imagination in RL is like model-based experience augmentation…
- Inference helps better understand what is actually happening
- Inference maintains and updates beliefs about how/why we saw what we saw

What is the significance of the prior?

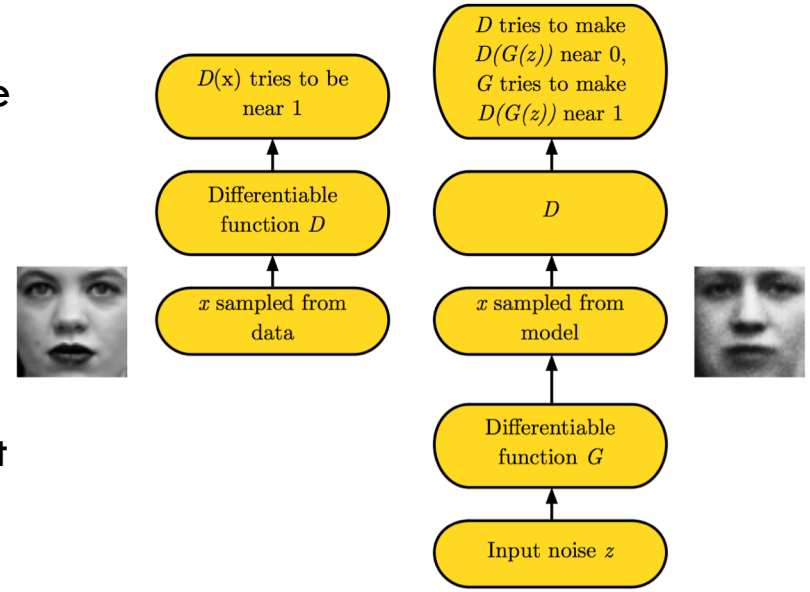
- A simple prior does not imply a simple posterior
- E.g., independent Gaussian priors don't induce disentanglement
- Priors should be chosen to facilitate inference and make models work

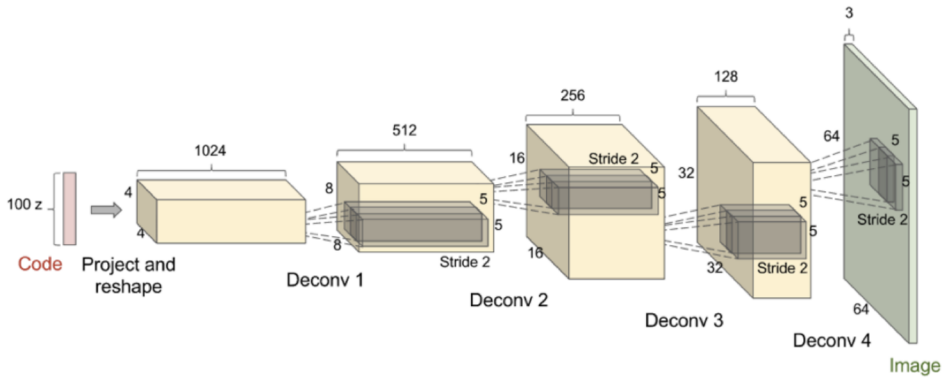
LEARNING BY GENERATING

GANS – GETTING STARTED

Generative Adversarial Networks

- **Idea:** train a model G to generate real(istic) data by training to generate data that tricks another model D into thinking the generated data is real
- This is like the complement of maximum likelihood – i.e., train the model to make all generated data real, rather than generate all real data
- The general principle can be reapplied in all kinds of ways
- The secret is detecting and eliminating the presence of certain information by measuring whether decisions can be made which require it

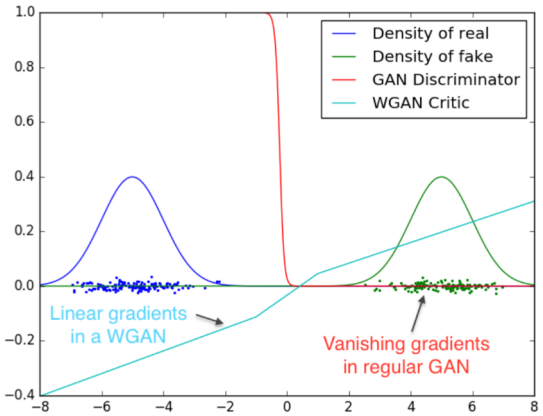




GANS – SOME KNOWN CHALLENGES

Degenerate divergences and WGAN

- If the real and fake distributions do not overlap…
- The discriminator can become too sharp
- The generator does not receive informative feedback
- Fix by restricting power of discriminator



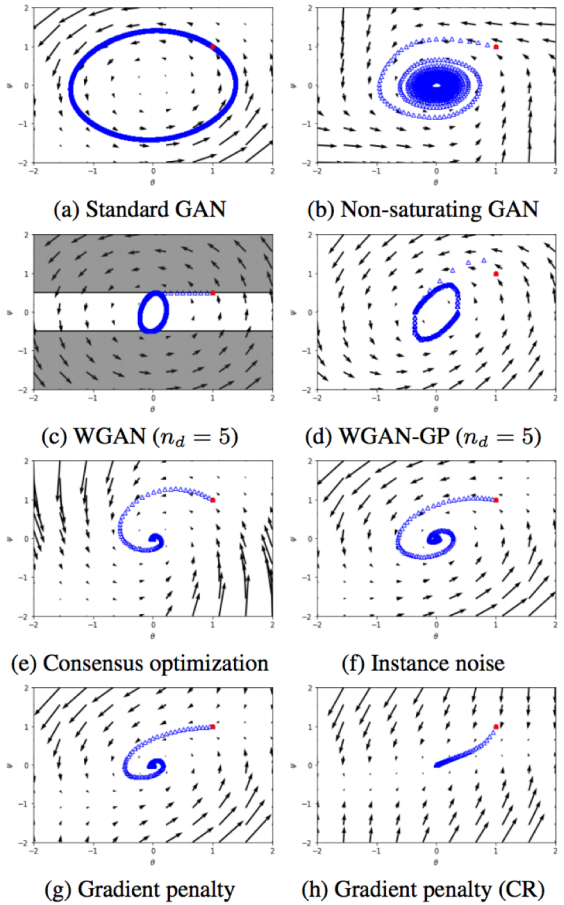
Loopy training and G<>D imbalance

- The generator and discriminator tend to "chase" each other
- We can regularize the discriminator

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

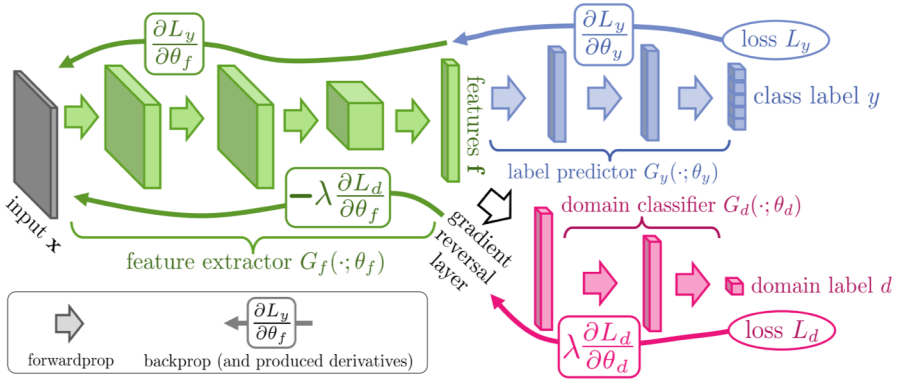
while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

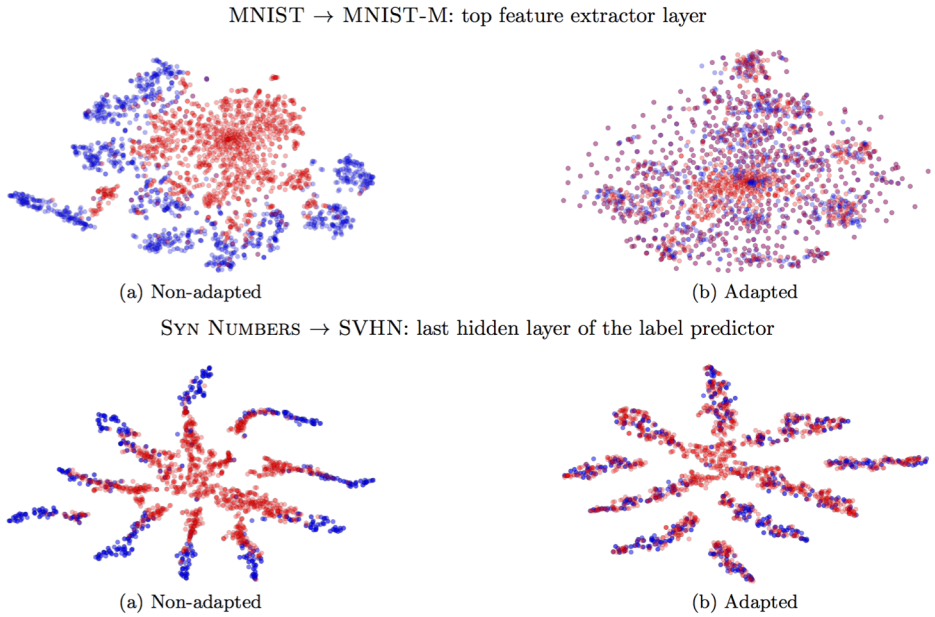


GANS – DOMAIN ADAPTION

Adversarial Domain Adaptation

- Given a lot of labeled data for domain A and a lot of unlabeled data for domain B
- When A and B have similar class structure…
- Train a classifier on domain A using features f(a)
- Train f so that the distribution of f(b) matches that of f(a)
- Features f(b) may be good input to a classifier over domain B

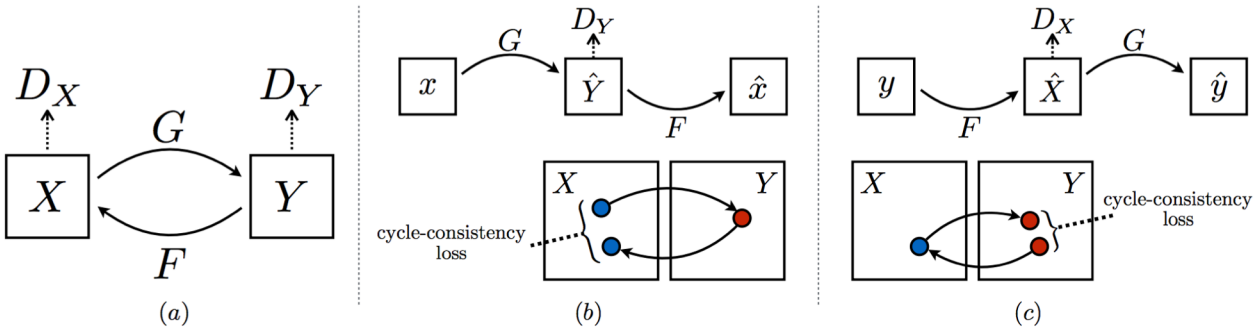


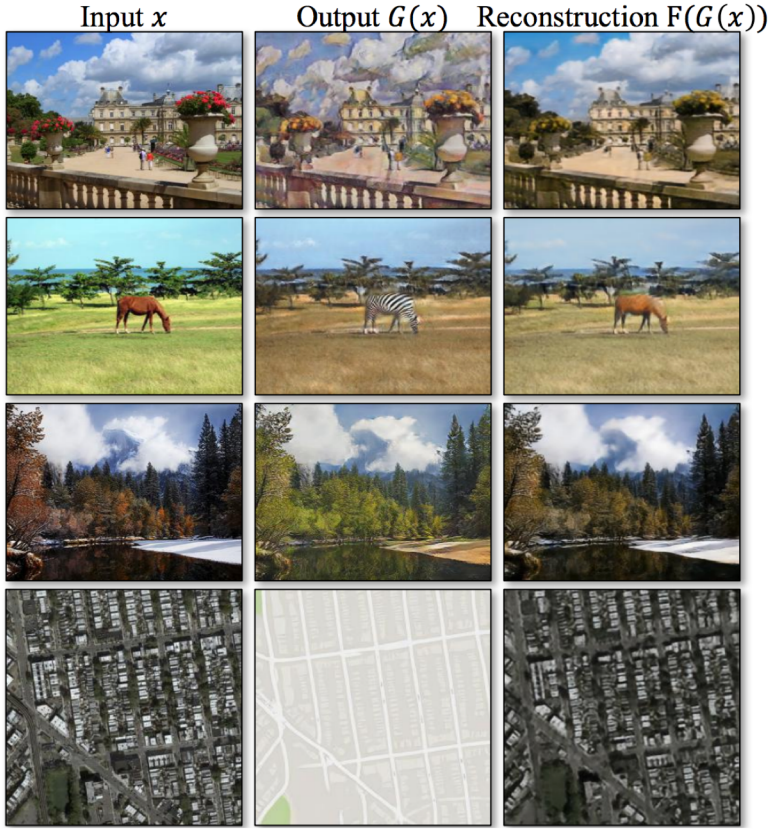


GANS – UNSUPERVISED TRANSLATION

CycleGANs

- Given domains A and B with a lot of unlabeled data
- Train a mapping f(A) > B and a mapping g(B) > A
- Train f(A) and g(B) so the marginal distributions of their outputs match the observed distributions in their output domains
- What do we get from this?





GANS – SIMPLE TRICKS, IMPRESSIVE RESULTS

BigGAN (honesty in naming)

- Idea: maybe we can stabilize GAN training by using bigger batches?
- Result: yes, and it works as well as most of the fancy stuff

(a) 128×128 (b) 256×256 (c) 512×512 (d)

Batch	Ch.	Param (M)	Shared	Hier.	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	✗	✗	✗	1000	15.30	58.77(±1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03(±1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85(±3.83)
2048	96	173.5	✗	✗	✗	295(±18)	9.54(±0.62)	92.98(±4.27)
2048	96	160.6	✓	✗	✗	185(±11)	9.18(±0.13)	94.94(±1.32)
2048	96	158.3	✓	✓	✗	152(±7)	8.73(±0.45)	98.76(±2.84)
2048	96	158.3	✓	✓	✓	165(±13)	8.51(±0.32)	99.31(±2.10)
2048	64	71.3	✓	✓	✓	371(±7)	10.48(±0.10)	86.90(±0.61)

BRANCHING OUT

GENERATING FUNCTIONS – VFUNC (1/4)

Representing and Manipulating Distributions over Functions

- **Motivation:** knowing uncertainty can be useful in various scenarios
- **Idea:** represent uncertainty in function space rather than parameter space
- **Method:** build a deep, directed generative model for functions that acts like a VAE
  - We can maximize mutual information or minimize KL from observed data (InfoGAN vs. VAE)

$$H(f) = H(z) - H(z|f) + H(f|z)$$
$$H(f) \geq H(z) + \mathop{\mathbb{E}}_{(f,z)\sim p(f,z)}[\log q(z|f)] + H(f|z)$$

MaxEnt – density estimation

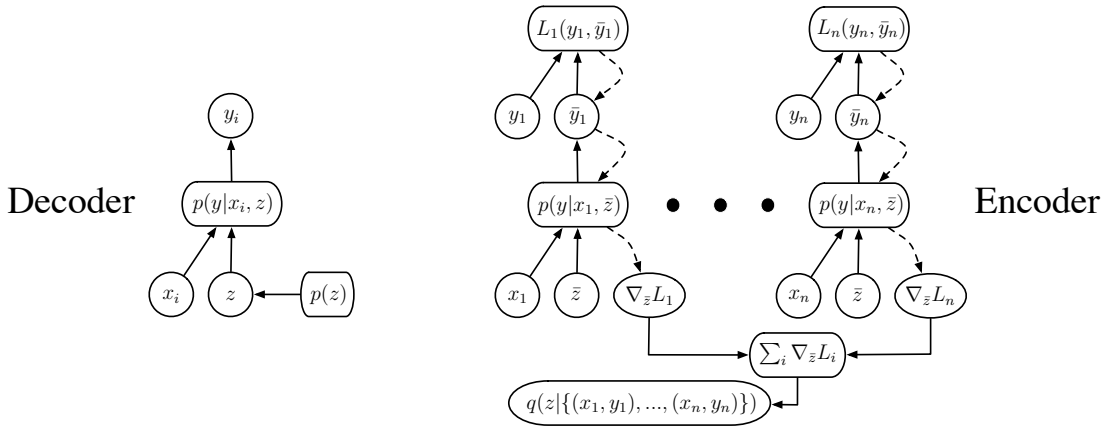
$$\mathop{\mathbb{E}}_{(z_1,\ldots,z_k)\sim p(z)}\left[\mathop{\mathbb{E}}_{f\sim p(f|z=z_1)}[\log q(z=z_1|f,z_1,\ldots,z_k)]\right]$$

MaxEnt – dynamic discretization

$$\mathop{\mathbb{E}}_{f\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{z\sim q(z|f)}[\log p(f|z)] - \mathrm{KL}(q(z|f)\,||\,p(z))\right]$$

MaxLL – standard ELBO like VAE

GENERATING FUNCTIONS – VFUNC (2/4)



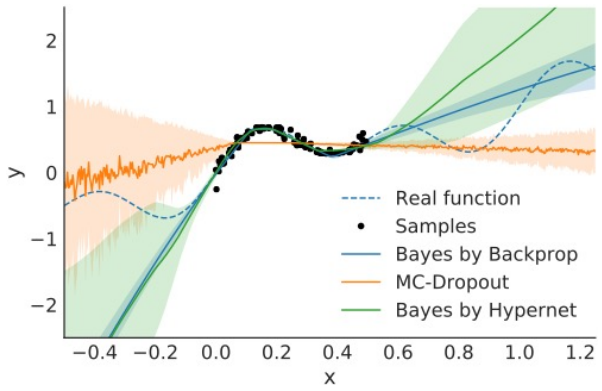
- Sample functions by sampling z and evaluating conditional network p(y | x, z)
- Encoder performs a "function diff"
- Gradient of diff helps predict z
- Density estimator q(z | f), or learn an "RKHS" for DDB

Algorithm RL: optimize $p(\pi)$ for Entropy and Task

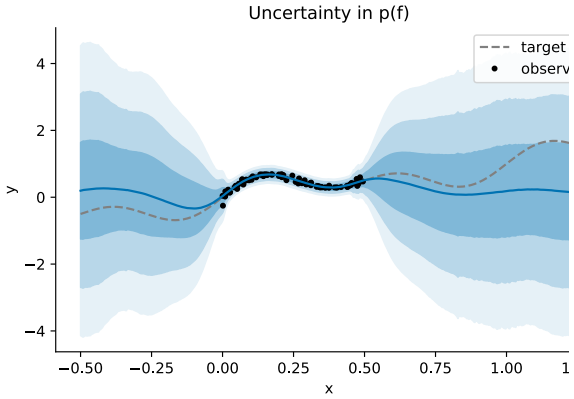
 # – We optimize a weighted sum of reward and entropy
 # – We maximize entropy in the space of mappings from
 # states to action probabilities (not states to actions)
 # – We add noise to $\pi(a|s, z)$ to prevent degenerate $H(\pi|z)$
while training **do**
 # optimize for task using policy gradient
$z_\tau \leftarrow z \sim p(z)$
$\tau \leftarrow \texttt{ROLLOUT}(\pi(a|s, z_\tau))$ *# $\tau = \{(s_1, a_1), ..., (s_t, a_t)\}$*
$\delta_R = \texttt{REWARD}(\tau) \cdot \nabla_\theta \log \pi(\tau|z_\tau)$
 # optimize for entropy using variational bound
$z_H \leftarrow z \sim p(z)$
$S_H \leftarrow \{s_0, ..., s_n\} \sim \texttt{REPLAY\_BUFFER}$
$\pi_H(S_H) \leftarrow \{\pi(\cdot|s_0, z_H), ..., \pi(\cdot|s_n, z_H)\}$
$\delta_H = \nabla_\theta \log q(z_H|\pi_H(S_H))$
 # update model parameters...
$\theta \leftarrow \theta + \alpha(\delta_R + \lambda_H \delta_H)$
end while

RL training loop

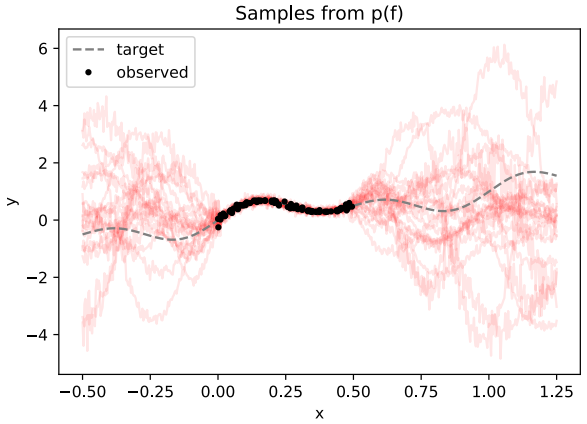
GENERATING FUNCTIONS – VFUNC (3/4)



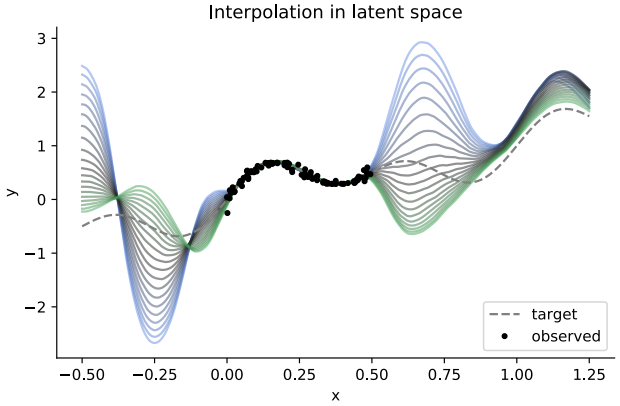
Baselines



Uncertainty



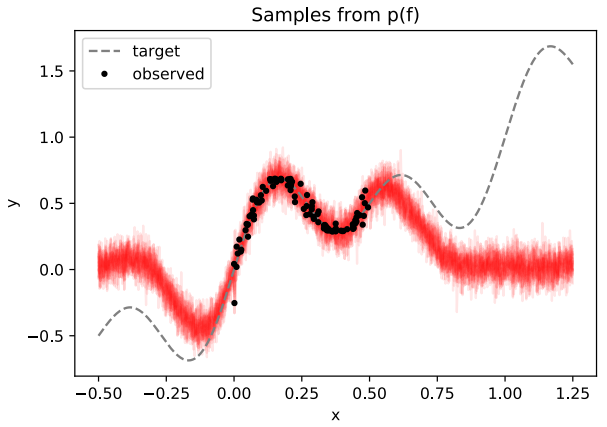
Samples



Interpolation

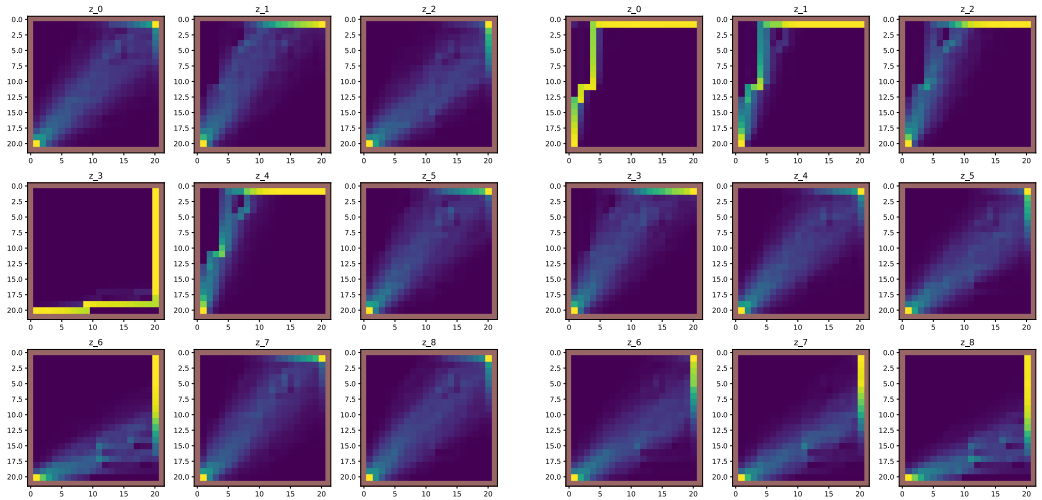
- Find diverse functions that explain the data
- Functions map inputs to specific y values
- Standard supervised loss (e.g. L2) for task
- Sample unlabeled data for entropy

- Uniform prior is weak
- We use a GP prior



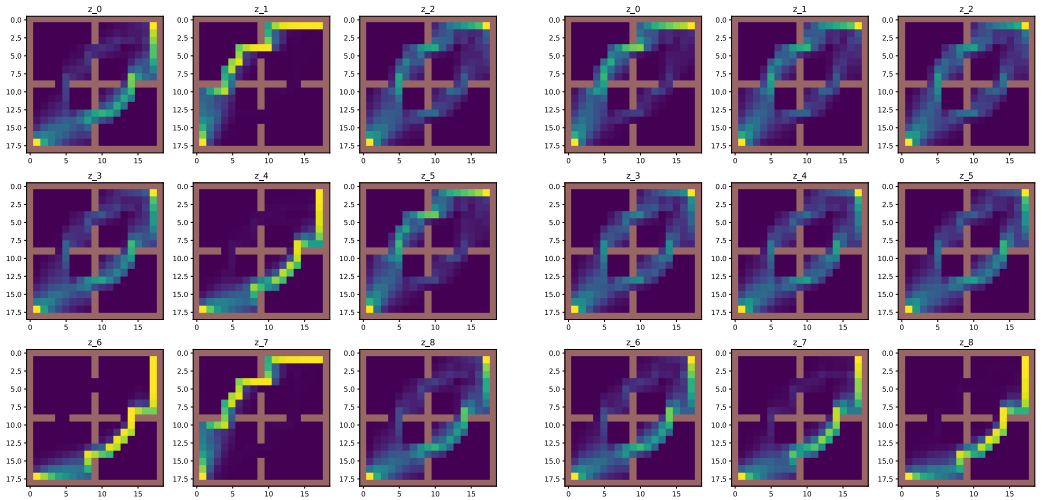
Samples -- no maxent

GENERATING FUNCTIONS – VFUNC (4/4)



Open World

- Simple discrete pathfinding problem
- Model should learn many shortest paths
- Policy gradient for task
- Experience replay for entropy



Four-Rooms

- Functions map states to action probabilities
- Sticking in modes seems to be an issue
- One-hot states make generalization tricky

RL – CURIOSITY AND MODELS

Exploration in Reinforcement Learning

- With sparse reward, it's hard to learn
- We can reward an agent for visiting interesting states
- Defining interesting is non-trivial

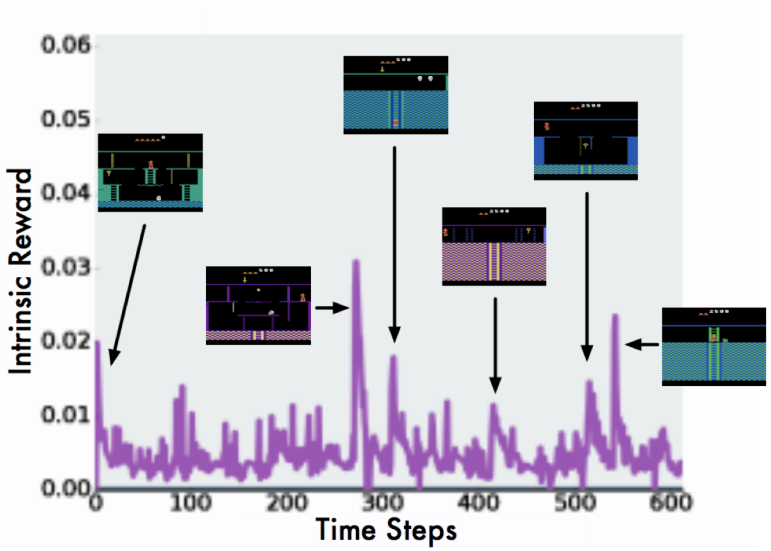
Count-based Exploration

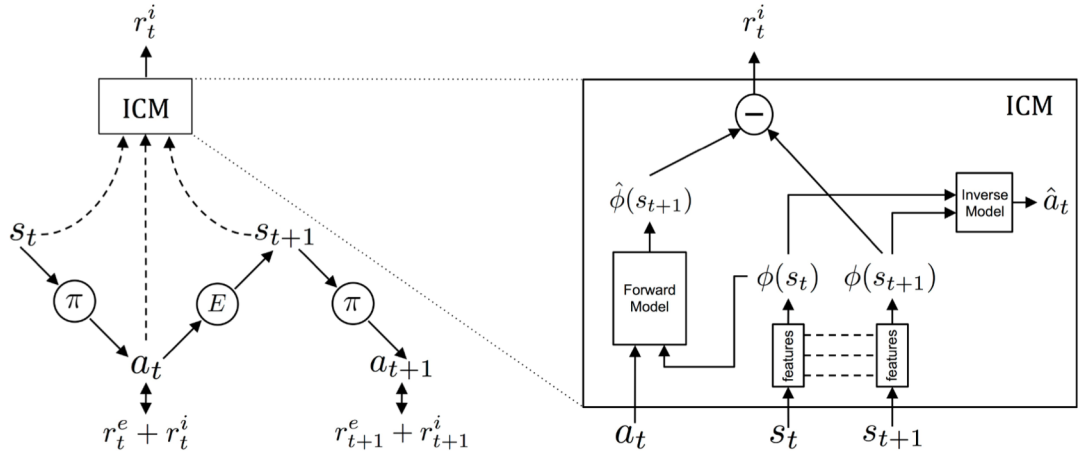
- Call states interesting if they haven't been visited often
- Measuring this is tricky – use density estimates or magic

Dynamics-based Exploration

- Call state+action pairs interesting if they defy expectations
- Learning good dynamics models is hard, maybe too hard
- Learn dynamics in abstract state space if possible

Don't get stuck watching a noisy TV…

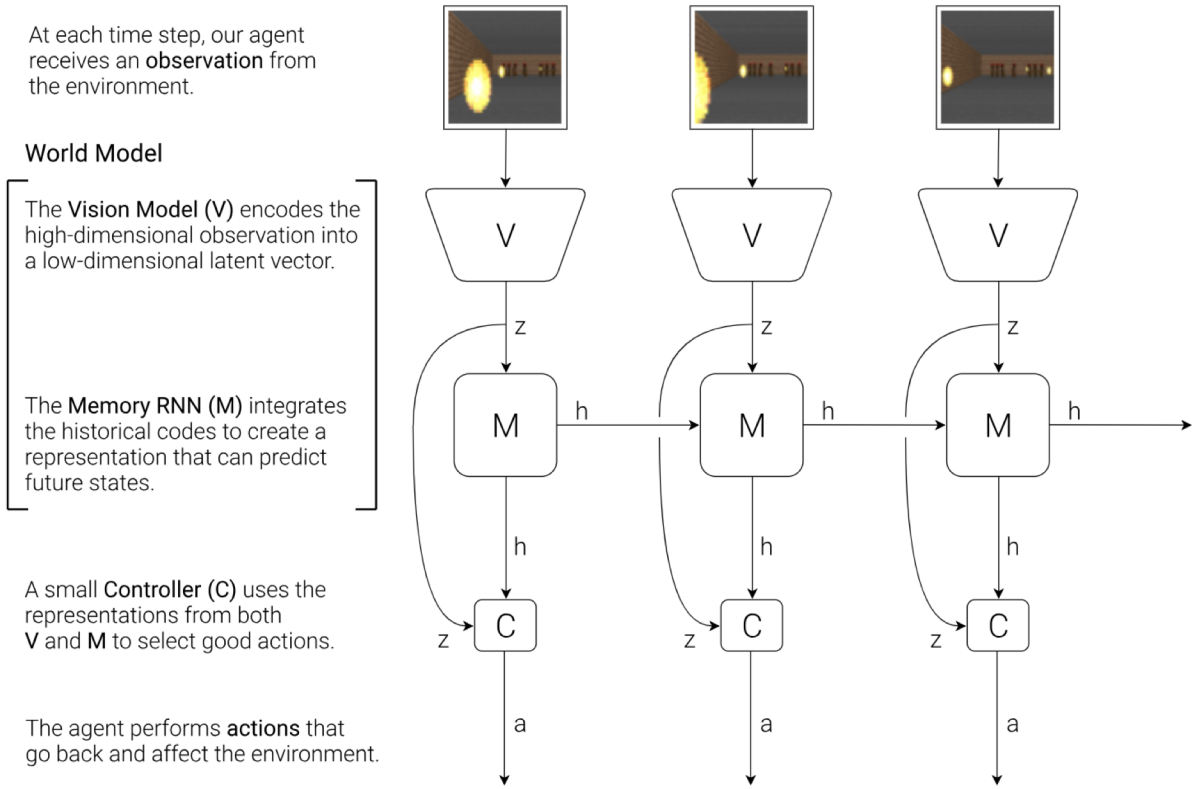




RL – PLANNING BY IMAGINING

Planning with world models

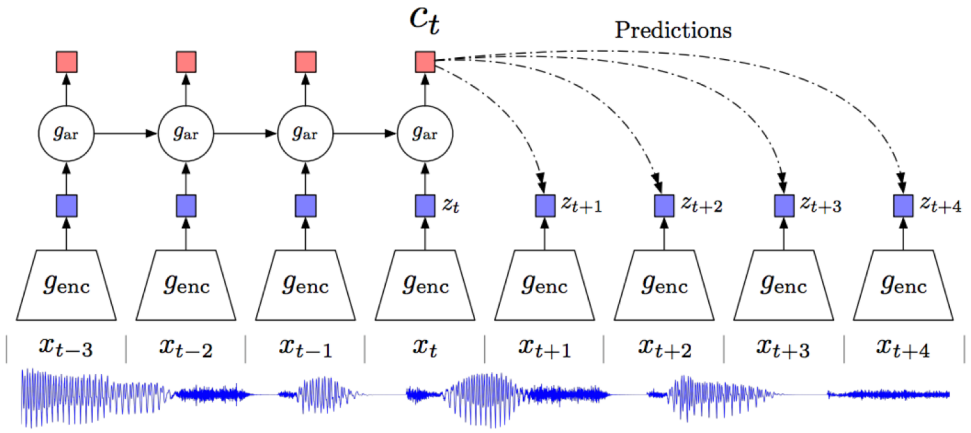
- With a powerful enough model…
- The agent can learn in imagined futures
- The model needs to be good enough
- Can we efficiently learn a model adequate for planning?
- Maybe…



UNGENERATIVE MODELS

Self-supervised learning and mutual information maximization

- Can we learn aspects of the generating process without generating data?
- E.g., learn features that can be extracted now, such that we can predict what features will be extracted 5 time steps from now
- Or, which features will be extracted if we look somewhere else or do something else
- Major gotcha – we must prevent features from collapsing!



THE END

Questions?