# GibbsNet: Iterative Adversarial Inference for Learning Deep Graphical Models

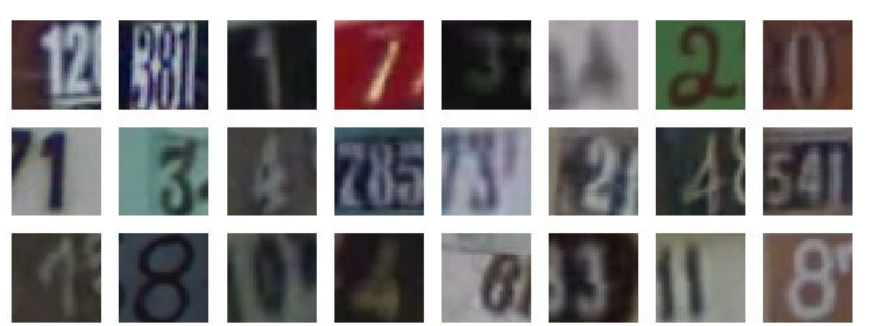
—

Alex Lamb, Devon Hjelm, Yaroslav Ganin, Joseph Paul Cohen, Aaron Courville, Yoshua Bengio

# Background: Generative Models

-Learn to model a data distribution, given samples from that distribution.

-This is a very general setup: includes classification, conditional modeling, as special cases.



Real House Number Pictures

Results from a generative model

# Generative Models with Latent Variables

-Many generative models learn a joint distribution between observed variables x, and unobserved latent variables z.

-Why do this?

-May be a much more natural place to represent the uncertainty in our distribution.

-Interpretability.

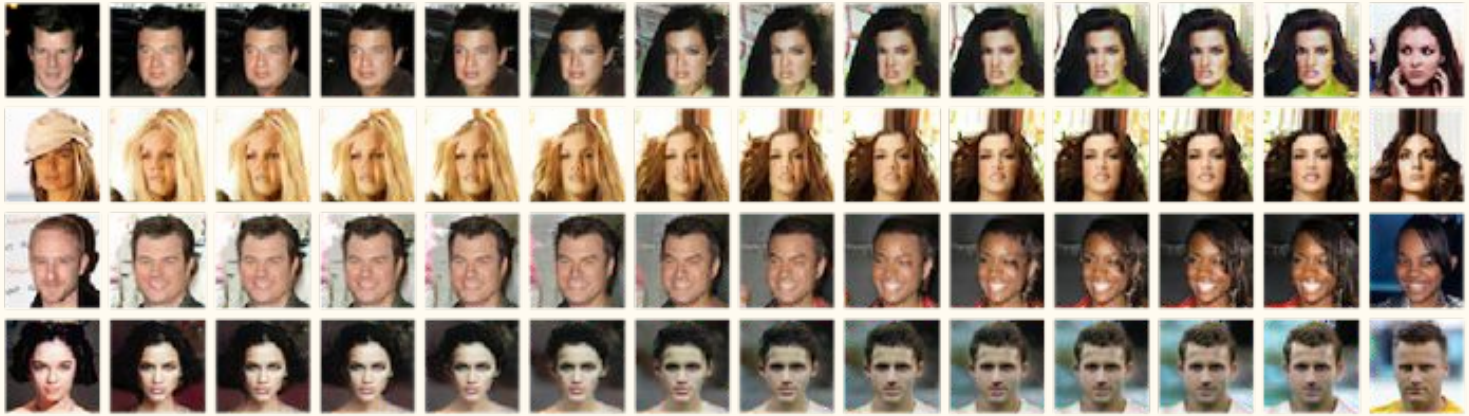
-Independence assumptions make much more sense in a latent space than in the observed space.

# Adversarially Learned Inference

-One powerful way of learning a generative model with latent variables.

$q(x,z) = q(x)q(z|x)$ Real Data, Inference Network

$p(x,z) = p(z)p(x|z)$ Latent Prior, Generation Network



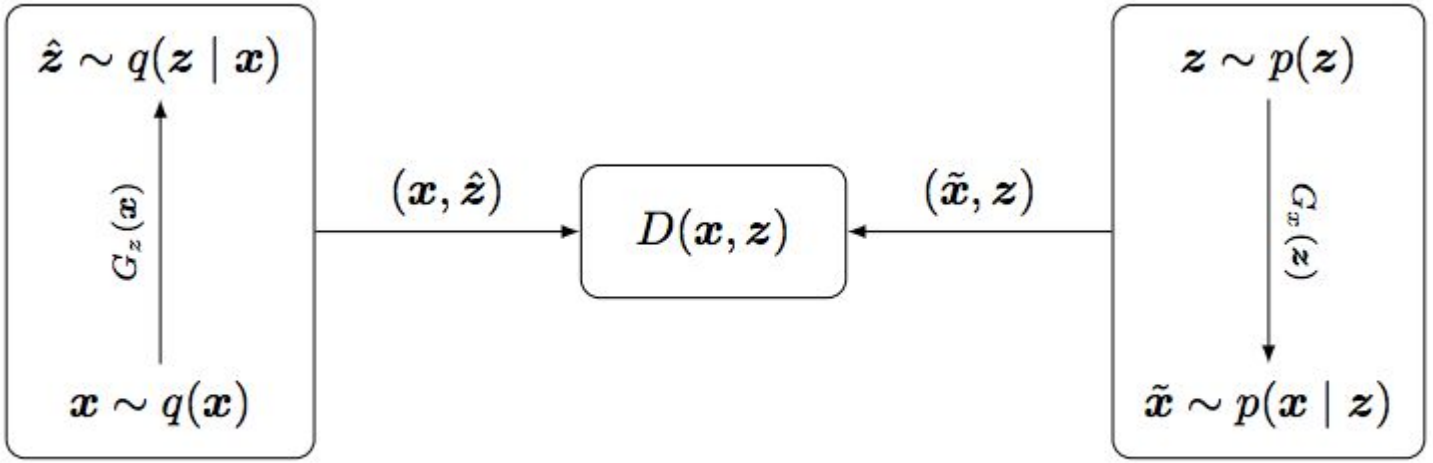
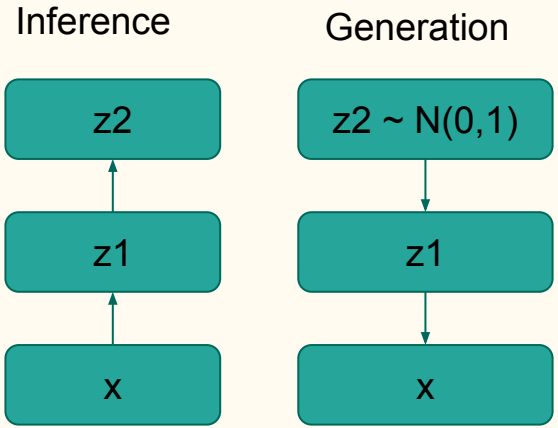


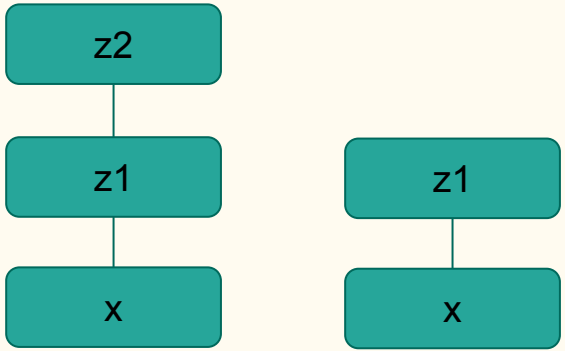

Figure 1: The adversarially learned inference (ALI) game.

# Two ideas on how to extend ALI




General Directed Graphical Models

(Hierarchical ALI or HALI)



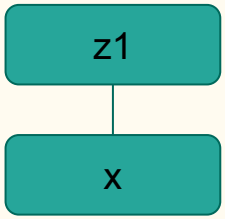

Implicit Undirected Graphical Models

(GibbsNet)

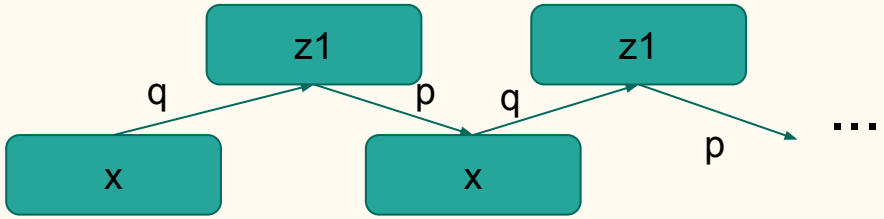
# GibbsNet - with one latent layer

-In the simplest case, corresponds to the undirected graph with a single layer of latent variables.

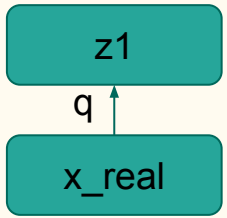
-Sampling from the model is done by blocked gibbs sampling (hence the name).




Undirected Graph

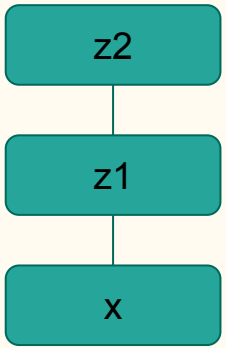



Sampling procedure

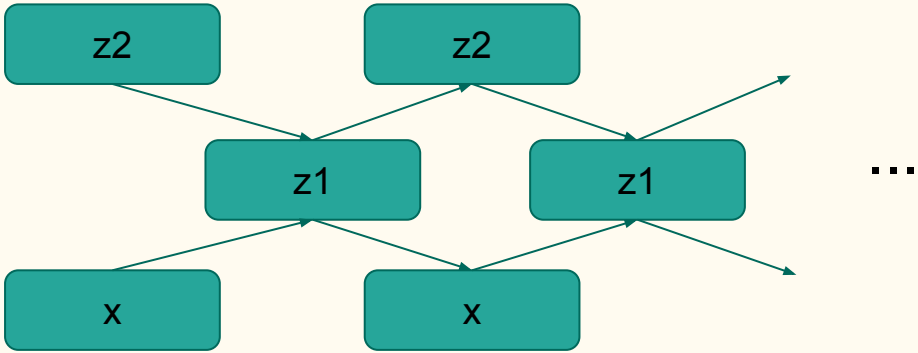



Inference Procedure

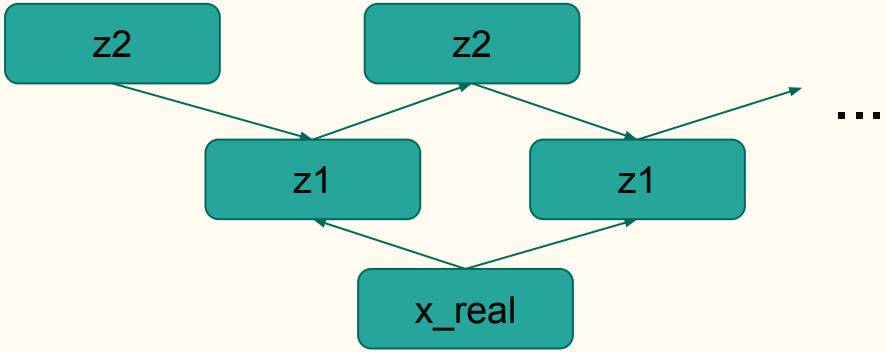
# GibbsNet with two latent layers




Undirected Graph




Sampling procedure




Inference Procedure

# GibbsNet

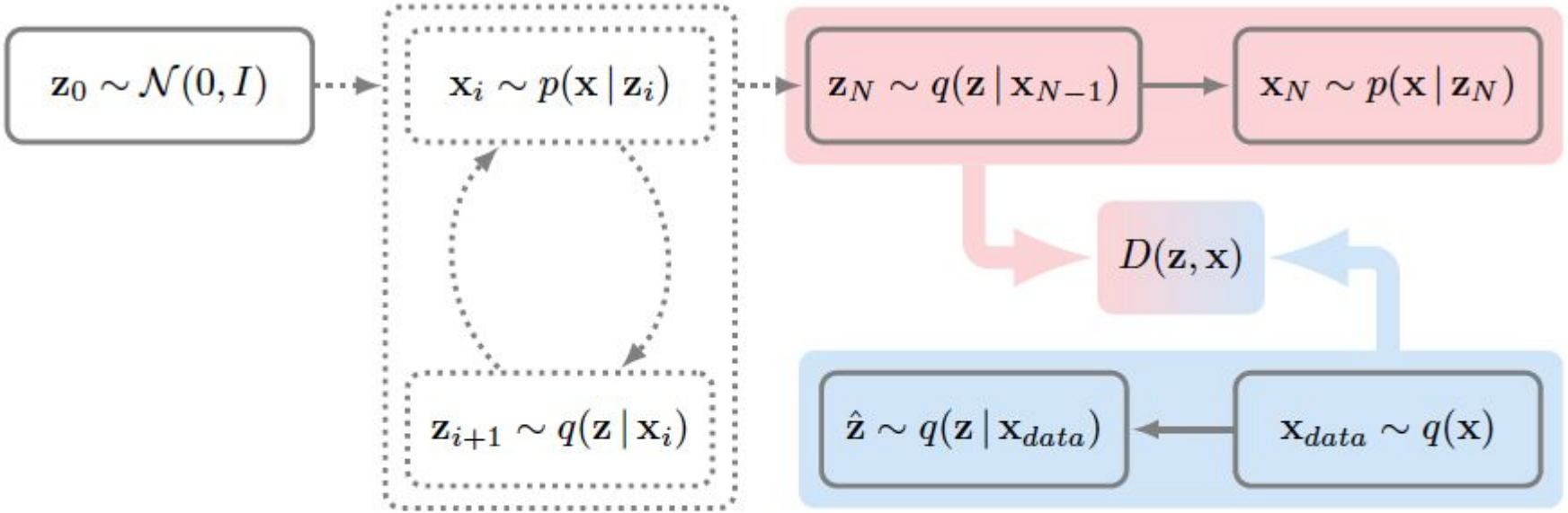
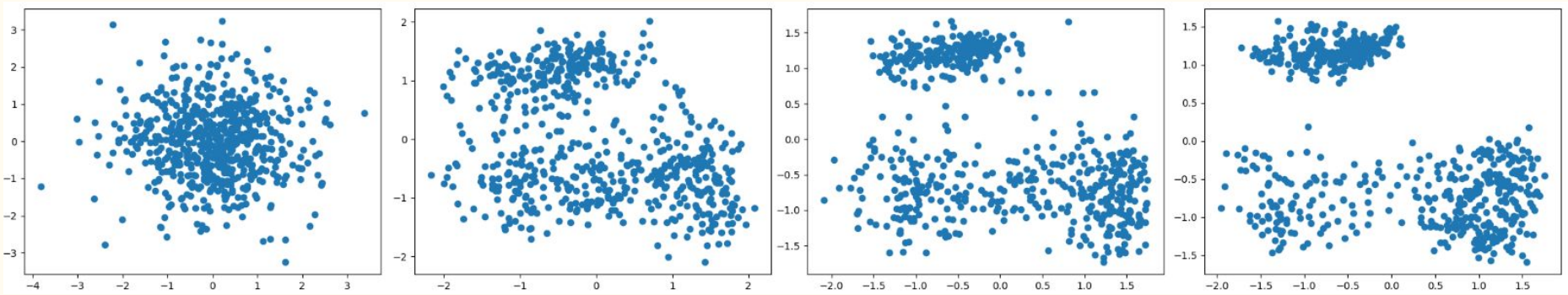



Figure 1: Diagram illustrating the training procedure for GibbsNet. The **unclamped chain** (dashed box) starts with a sample from an isotropic Gaussian distribution $\mathcal{N}(0, I)$ and runs for $N$ steps. The last step (iteration $N$) shown as a solid pink box is then compared with a single step from the **clamped chain** (solid blue box) using joint discriminator $D$.

# What does the Unclamped Chain do?



# Partially Clamped Chains (only at Test Time!)

# More Partially Clamped Chains (Only test time)

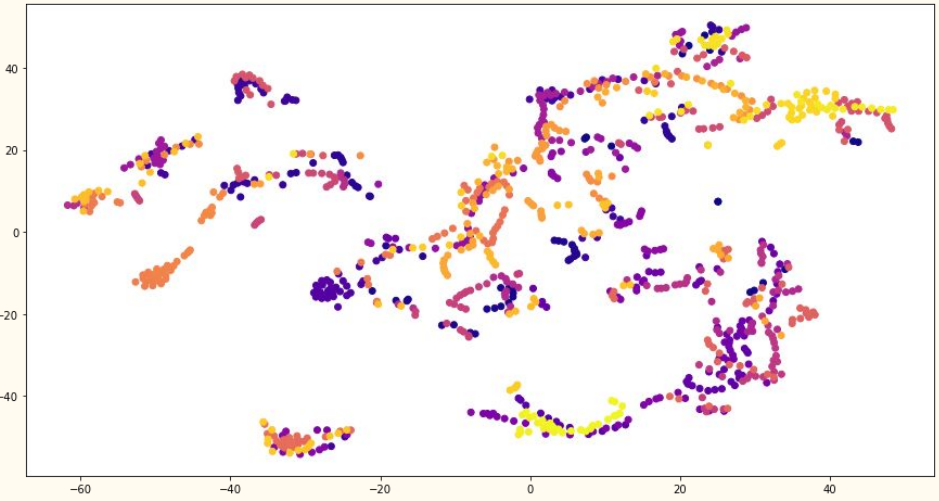
# Comparisons with ALI

“Therefore love moderately, long love does so.  Too swift arrives as tardy as too slow” - Shakespeare

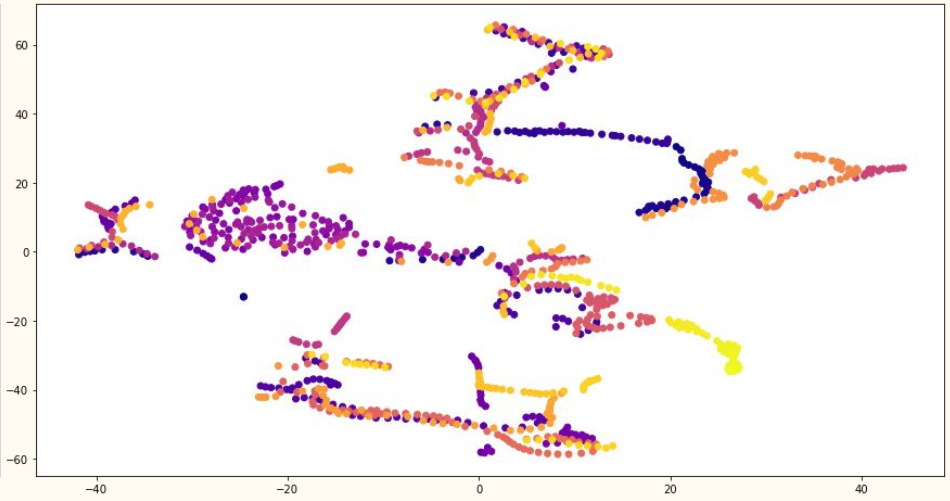
ALI

GibbsNet

# ALI vs. GibbsNet, Unclamped Chain T-SNEs

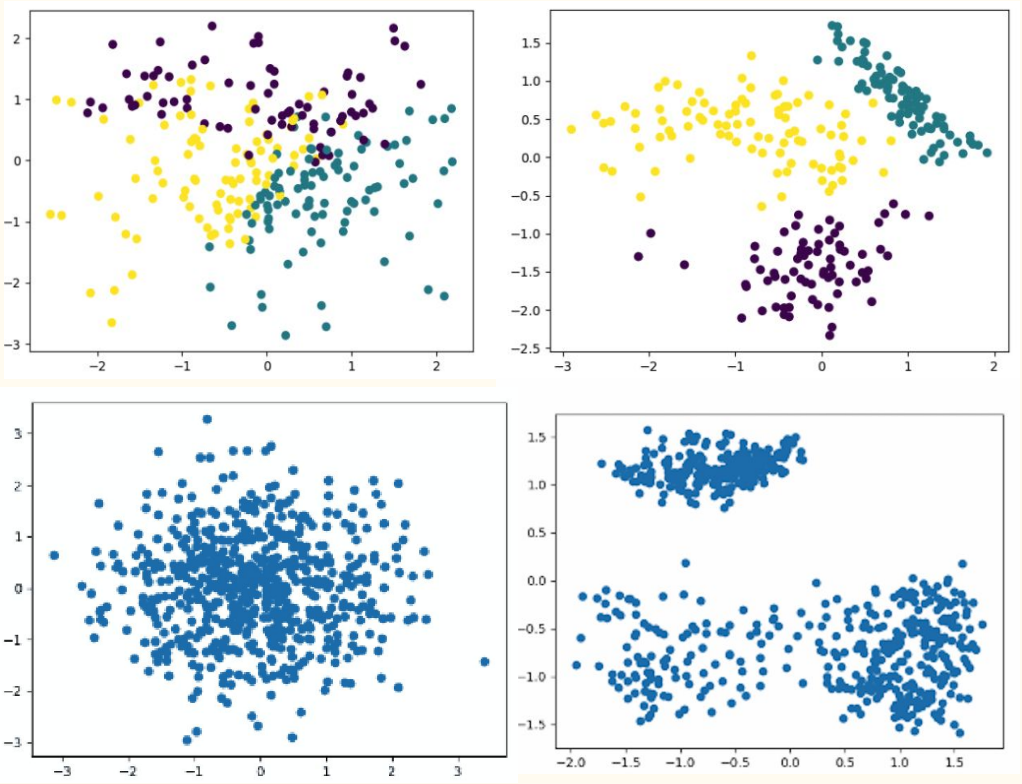



ALI




GibbsNet

# ALI vs. GibbsNet Latent Space



# What about Partial Clamping at train time?

-Have not experimented with this yet, but it should be possible.

-Idea is that our clamped chain will now only be partially clamped on the x's, and it will also be an iterative procedure.

-This is a pretty interesting capability, because it means that we can have arbitrary missing features in our dataset during training, which can be totally different than what's missing during testing.

-We might also be able to do cool stuff like constraint based inpainting. For example, what if we never have access to samples $x \sim p(x)$ but instead have $y \sim f(p(x))$, we know f, and we want to model p(x)? Had problems like this at amazon.

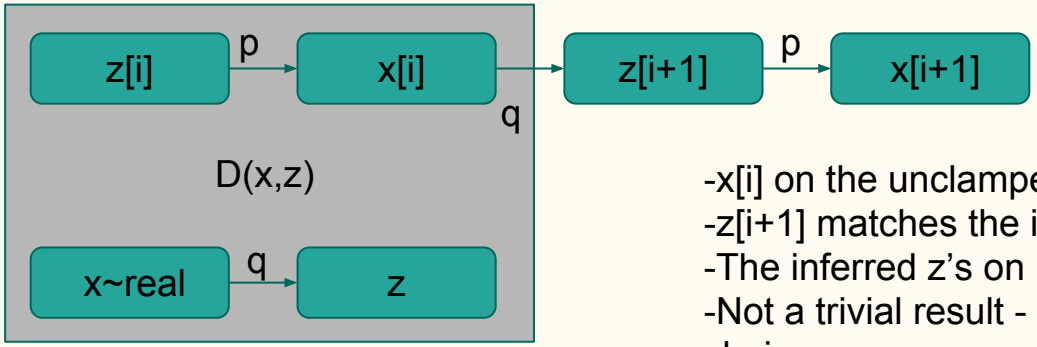
# Theoretical Properties of GibbsNet

-Assume that our generative process fools an optimal discriminator (common foundation for analyzing GANs) after N steps.

-Two main results (mostly by Yaroslav and Yoshua):

-We're guaranteed to have our unclamped chain get to data dist in N steps.

-More surprising: guaranteed to stay on the data manifold for all steps i $>$ N!




-x[i] on the unclamped chain matches real data.
-z[i+1] matches the inferred z's on real data.
-The inferred z's on real data will transition to real data.
-Not a trivial result - only holds with parameter sharing between chains.

# Why is GibbsNet Appealing?

-Our inference and generation procedures now share parameters, which could make our model more compact and more stable than a similarly deep generator without this parameter sharing.

-Encoder network uses gradient through the decoder network for training - in this way it's more like variational autoencoders than ALI.

-Sometimes specifying undirected graph is more appealing than specifying directed graph.

-More eclectic: interesting connection to Heidegger's existential phenomenology!

# Heidegger’s theory

-Two modes of being: ready-to-hand and present-at-hand.

Ready-to-hand: when actively working on a task, the task itself is experienced rather than objects in the world.

Present-at-hand: when equipment breaks down, the world becomes divided between subjects and objects

-Idea: maybe ready-to-hand is kind of like the negative phase and present-at-hand is like the positive phase?

# Heidegger Quotes

The peculiarity of what is proximally to hand is that, in its readiness-to-hand, it must, as it were, withdraw ... that with which we concern ourselves primarily is the work
Pure presence at hand announces itself in such equipment, but only to withdraw to the readiness-in-hand with which one concerns oneself — that is to say, of the sort of thing we find when we put it back into repair

Essentially ahead of itself, [being-in-the-world] has projected itself upon its potentiality-for-Being before going on to any mere consideration of itself. In its projection it reveals itself as something which has been thrown. It has been thrownly abandoned to the ‘world’, and falls into it concernfully

# Four Basic Modes of Being

-Dreaming: nothing is clamped, and the initial states are essentially random

-Ready-to-hand: the high-level is clamped, lower-levels initialized from the present

-Daydreaming: nothing is clamped, but the lowest-level is rendered impotent upon the higher levels, high-level initialized from present.

-Present-at-hand: lowest-level is clamped, and the higher-level is probably rendered impotent to allow for fast inference.

# Four Basic Modes of Being

| | Low Level | Middle Level | High Level |
|---|---|---|---|
| Present-at-Hand | Clamped | Free | Free, Somewhat Impotent |
| Ready-to-Hand | Free | Free | Clamped |
| Daydreaming | Clamped, Impotent | Free | Free |
| Dreaming | Free | Free | Free |

# Four modes for variables

-Clamped vs. Unclamped.

-When clamped a variable is not sampled but rather takes a fixed value.

-Potent vs. Impotent:

-A variable doesn’t affect sampling for adjacent variables if it’s impotent.

-This isn’t standard in graphical models.

-For example, when in a lecture and zoned-out - fantasizing about stuff, the lower-level contents are clamped but they are impotent as they do not force the higher level contents to reflect upon them unless they constitute an “emergency”.

# Speculative Theory of Functionality

-Dreaming is more useful than daydreaming, but dreaming is “expensive” because it takes you out of the world completely.

-So daydreaming is just the “next best thing”, which the brain obtains cheaply.

-GibbsNet = pulling present-at-hand and dreaming together.

-Issue: impotence should easily be detected by the joint discriminator. So impotent units need to be hidden from the discriminator.

# Future Work (1) - RBMs and DBMs

-Can we use GibbsNet to train RBMs and DBMs? Conceptually it's straightforward - hard to get working in practice.

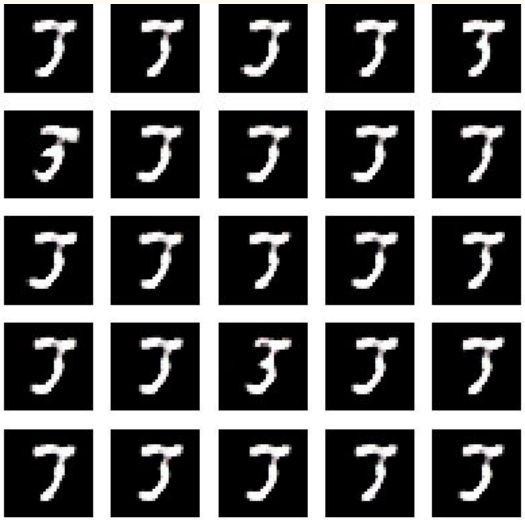
-RBM:

$$p(x|z) = \text{bernoulli}(\text{sigmoid}(Wz + b))$$

$$q(z|x) = \text{bernoulli}(\text{sigmoid}(Wx + c))$$

-Consistent by construction, but also limits the form a lot.

-Doesn't seem to work in practice. Collapses.



# Future Work (2)

-Having a more complex graphical model, and use local discriminators over different neighborhoods.  Does this still learn long-term dependencies?

-Having a graph with a z with spatial structure, such that the bottleneck is removed in the P-chain.

-Instead of parameterizing the transition operator with networks, can we have general networks for the energy and derive transition operators from that?