Generative Modeling

Michela Paganini
Yale University & Lawrence Berkeley National Lab

@WonderMicky
michela.paganini@yale.edu
mickypaganini.github.io

Outline

1. What is generative modeling & why care about it?

2. Overview of select generative models

Generative Modeling

- Asks question - can we build a model to approximate a *data distribution*?

- Formally we are given $x \sim p_{\text{data}}(x)$ and a finite sample from this distribution

$$X = \{x | x \sim p_{\text{data}}(x)\}, |X| = n$$

- Problem: can we find a model such that

$$p_{\text{model}}(x; \theta) \approx p_{\text{data}}(x)$$

- **Why** might this be useful?

Why care about Generative Models?

Oft over-used quote:

> “*What I cannot create, I do not understand*”
>
> -R. Feynman

Why care about Generative Models?

- Classic uses:
  - Through maximum likelihood, can fit to some interpretable parameters for a hand-designed $p_{\text{model}}(x;\theta)$
  - Learn a joint distribution with labels $p_{\text{data}}(x, y;\theta) \approx p_{\text{data}}(x, y)$ and transform to $p(y|x;\theta)$
- More interesting uses:
  - Fast-generation of compute-heavy tasks
  - Interpolation between distributions

Maximum Likelihood Estimation

- We'll focus on models that can be *made to fit* into the Maximum Likelihood Estimation (MLE) framework

- Other models exist, but MLE covers most main ones

Traditional MLE Approach

- We are given a finite sample from a data distribution

$$X = \{x | x \sim p_{\text{data}}(x)\}, |X| = n$$

- We construct a parametric model $p_{\text{model}}(x;\theta)$ for the distribution, and build a likelihood

$$\mathcal{L}(\theta; X) = \prod_{x \in X} p_{\text{model}}(x;\theta)$$

- In practice, we optimize through MCMC or other means, and obtain

$$\theta_{\text{opt}} = \arg\min_{\theta}\{-\ln \mathcal{L}(\theta; X)\}$$

Now what?

- We have obtained a model $p_{\text{model}}(x;\theta)$, we can run it in "forward mode"

- Hand designed, parametric models (usually mixtures of Gaussians, Weibull, Poisson, etc.) — limited in expressivity

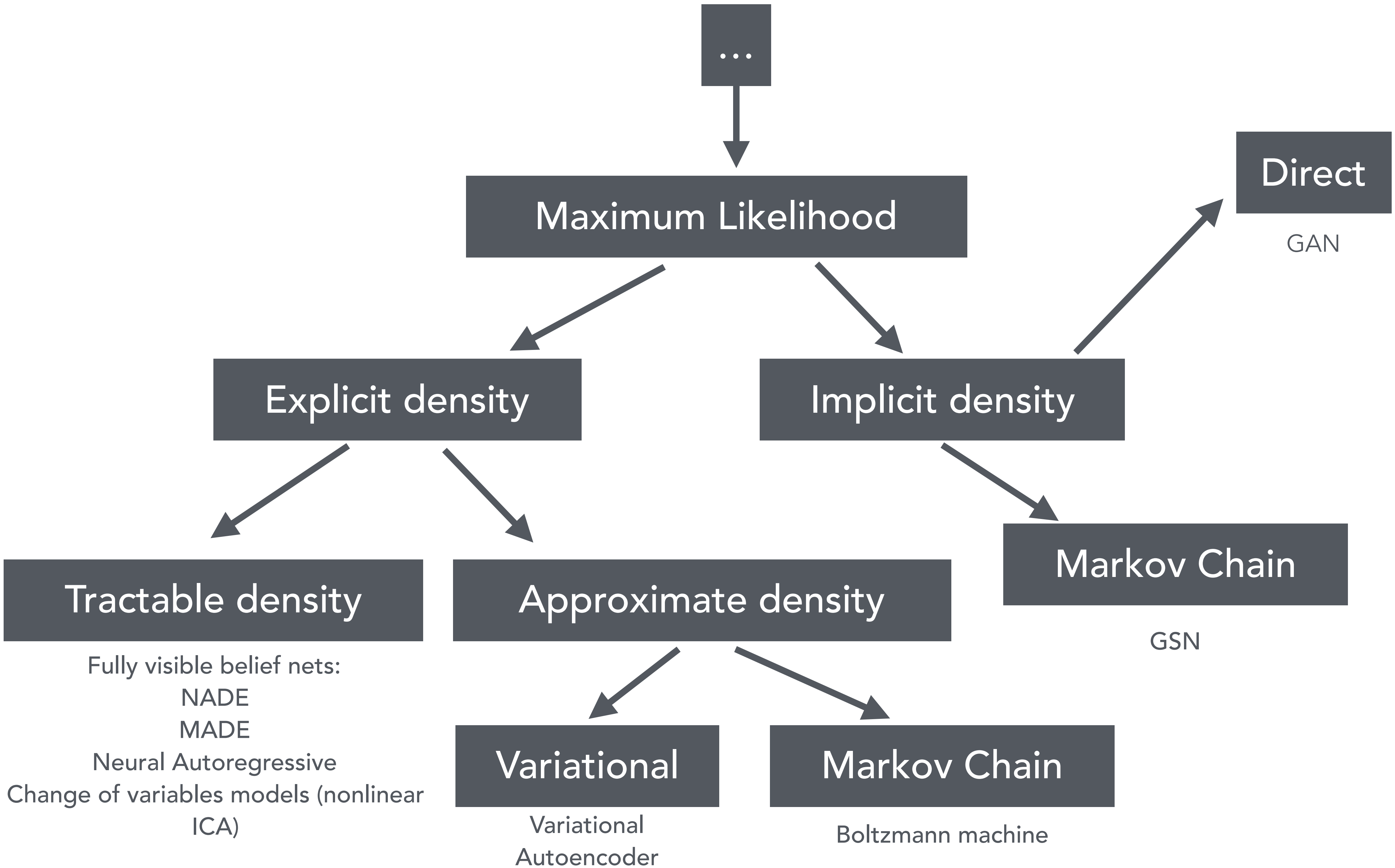
  - Modern deep models remove this issue

- If the likelihood is explicit, then we can use this for classification or inference through Bayes

$$p(y|x) = \frac{p_{\text{model}}(x|y;\theta)p(y)}{p_{\text{model}}(x;\theta)}$$

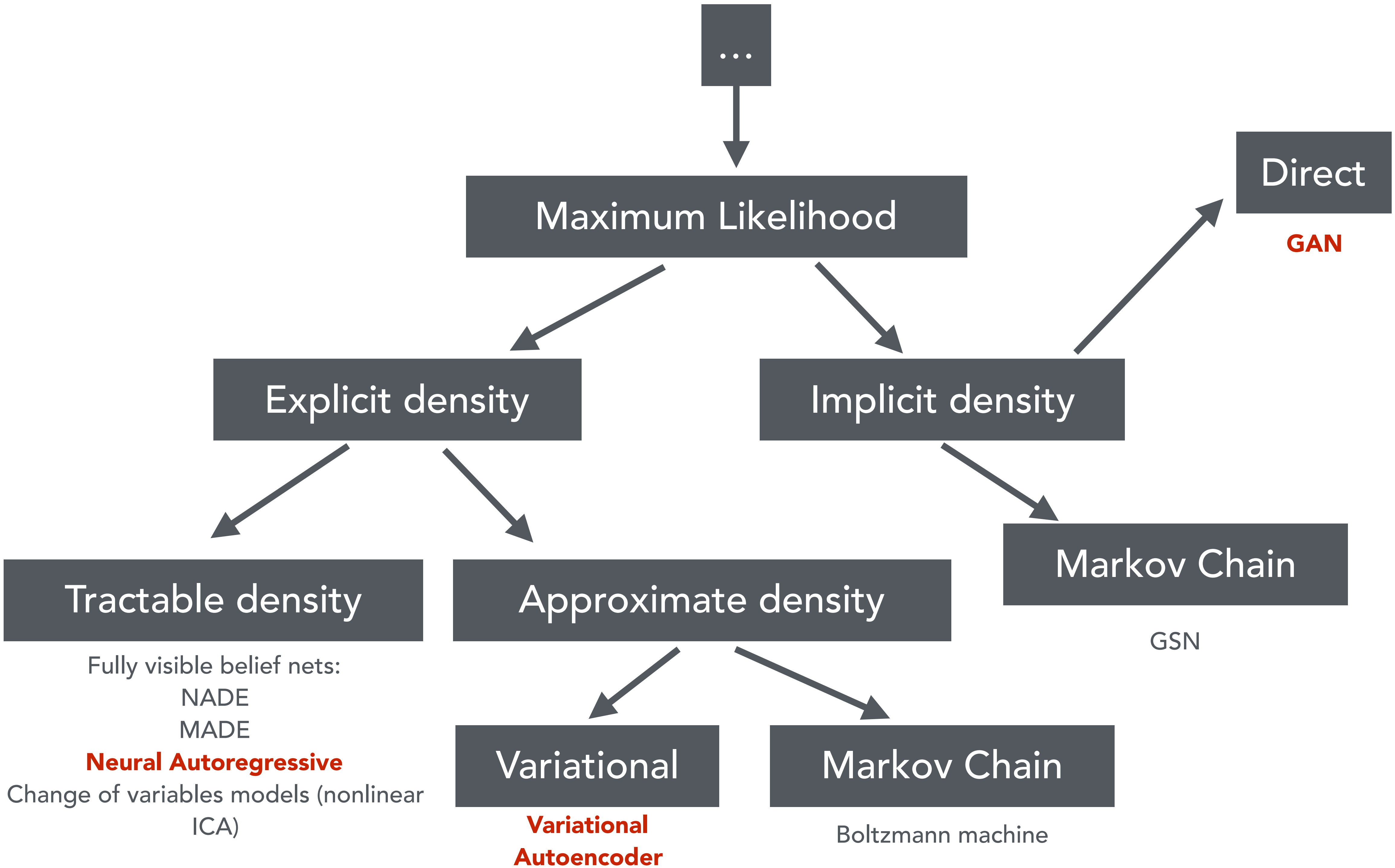
Taxonomy

Generative Model Taxonomy



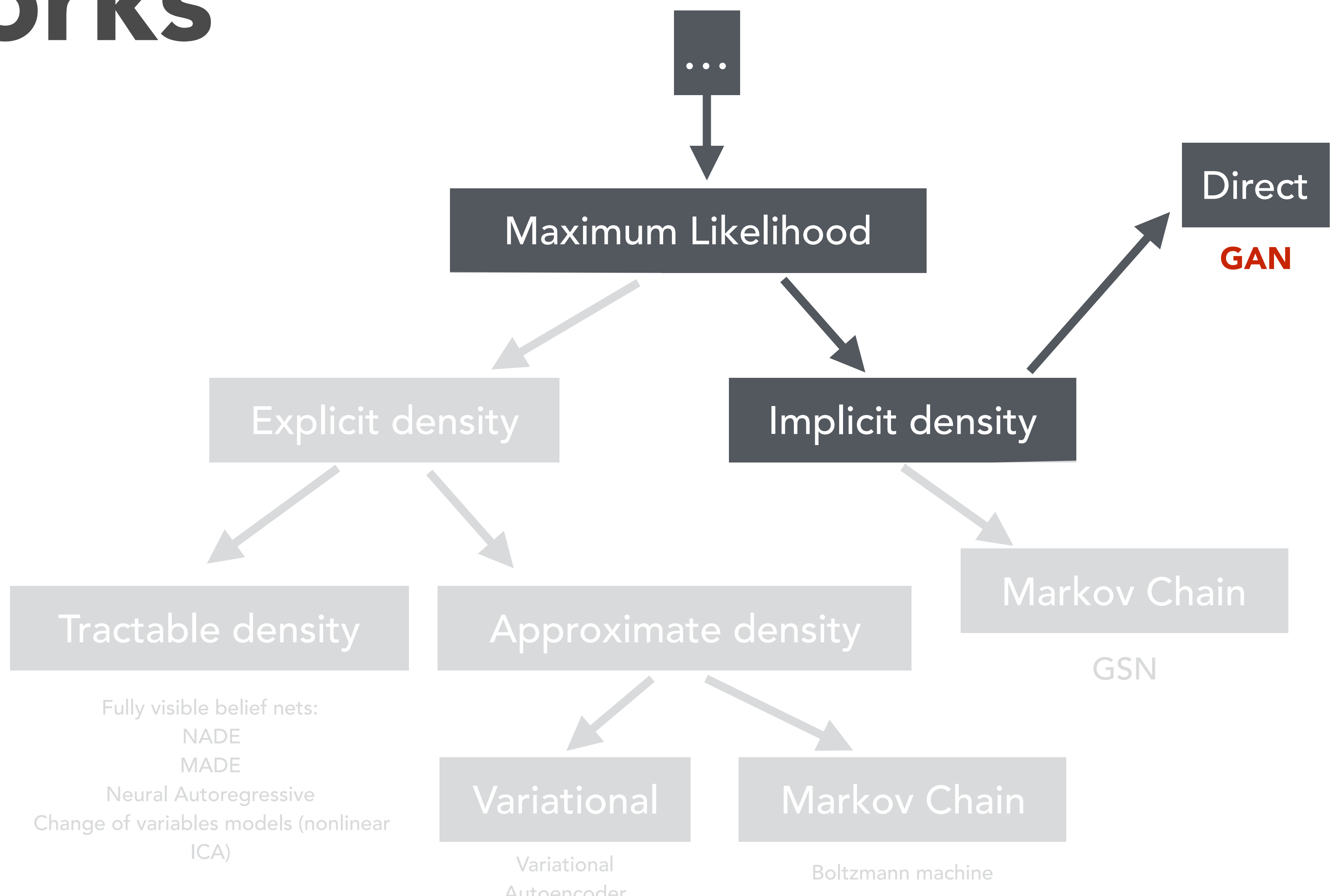
From I. Goodfellow

Generative Model Taxonomy



From I. Goodfellow

Generative Adversarial Networks



Generative Adversarial Networks

- We cast the process of building a model of the data distribution as a two-player game between a **generator** and a **discriminator**
- **Intuitively**, generator maps random noise, through a model to produce a sample, and discriminator decides whether the sample is real or not

Generative Adversarial Networks

- As before, data distribution $x \sim p_{\text{data}}(x), x \in \mathcal{X}$

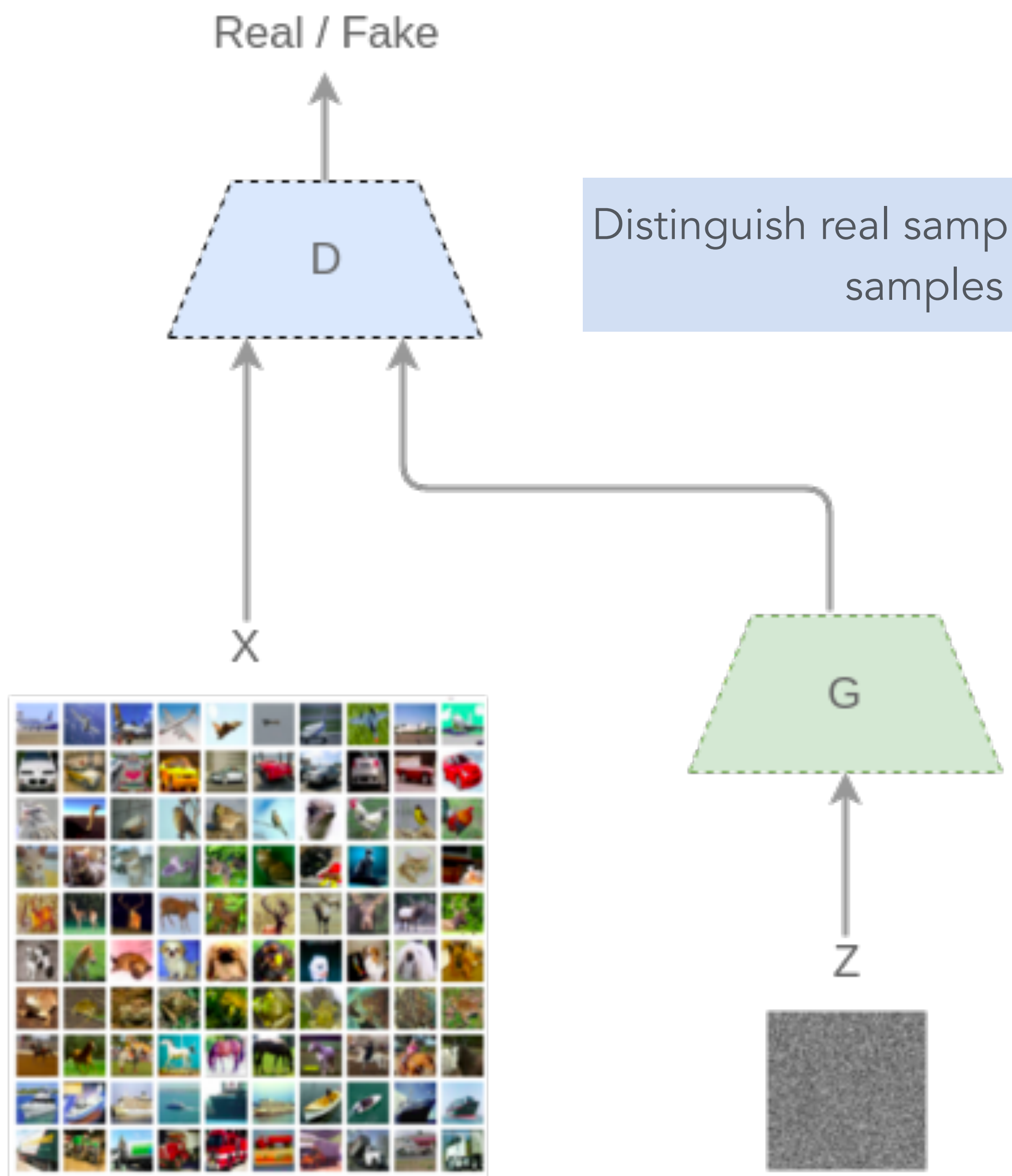
- Our **generator** has a latent prior $z \sim p_z(z), z \in \mathcal{Z}$ and maps this to sample space $G : \mathcal{Z} \longrightarrow \mathcal{X}$

- $G(\cdot; \theta_G)$ **implicitly** defines a distribution $p_{\text{model}}(x; \theta_G)$

- Our discriminator $D(\cdot; \theta_D)$ tells how fake or real a sample looks via a score $D : \mathcal{X} \longrightarrow \mathbb{R}$

Generative Adversarial Networks

$$V(D,G) = \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z;\theta_G);\theta_D))] + \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x;\theta_D)]$$



Distinguish real samples from fake samples

Transform noise into a realistic sample

$$\min_G \max_D V(D,G)$$

Real data

GANs in Context

- GANs have shown *empirical* promise in learning complicated, high dimensional physical realizations

Figure 7: 128x128 pixel images generated by SN-GANs trained on ILSVRC2012 dataset. The inception score is 21.1±.35.

arxiv:1802.05957

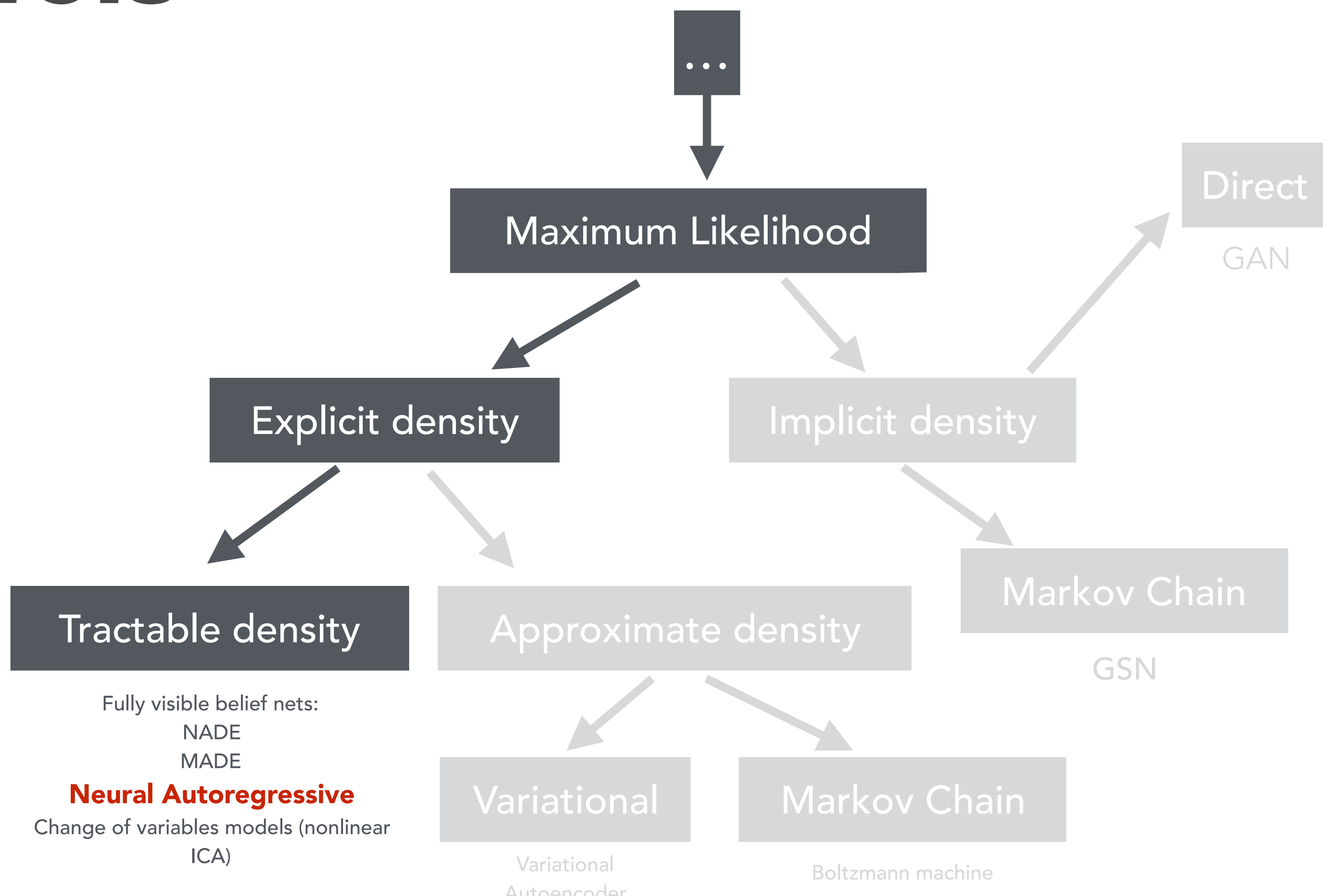

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

arxiv:1710.10196

- Lack full theoretical understanding of *why* they work

- Can sample from, but not evaluate the likelihood (implicit model)

Neural Autoregressive Models



Why Autoregressive?

- As before, we have $x \sim p_{\text{data}}(x)$ with $x \in \mathbb{R}^d$
- Since x is a vector, we can factorize **per dimension**

$$p_{\text{data}}(x) = q(x_1)q(x_2|x_1)q(x_3|x_2, x_1)...q(x_d|x_{d-1}, \ldots, x_1)$$

$$p_{\text{data}}(x) = q(x_1)\prod_{i=2}^{d} q(x_i|x_{i-1}, \ldots, x_1)$$

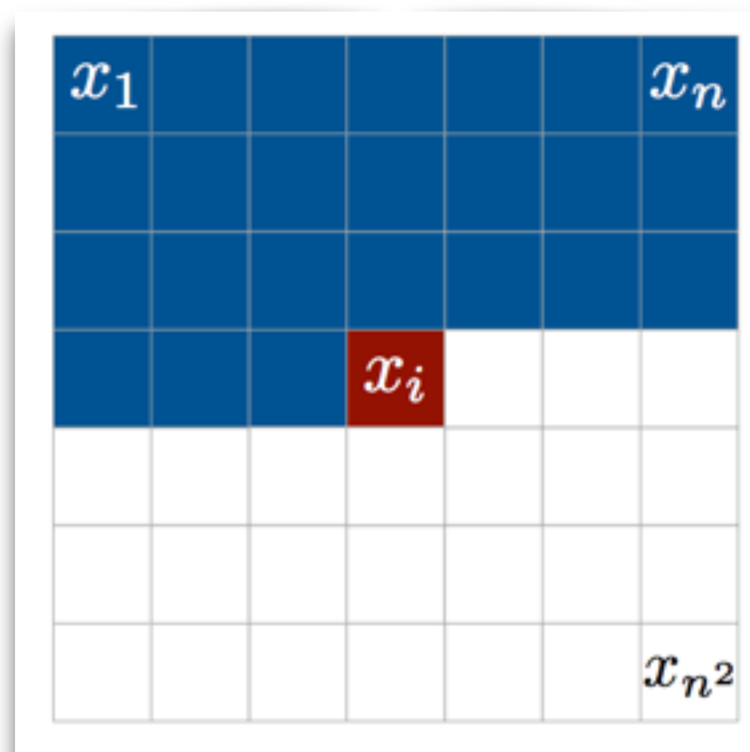
- We can now model $q(x_i|x_{i-1}, \ldots, x_1)$ with a neural network!

Pixel{RNN, CNN, CNN++}

$q(x_i|x_{i-1}, \ldots, x_1)$ predicts out distribution of individual pixels conditional on all pixels up and to the left

Figure 3: Class-Conditional samples from the Conditional PixelCNN.

arxiv:1606.05328



arxiv:1601.06759

Wave{Net, RNN}

$q(x_i|x_{i-1},...,x_1)$ predicts out speech signal value at current time step conditioned on previous speech waveform

Figure 1: A second of generated speech.

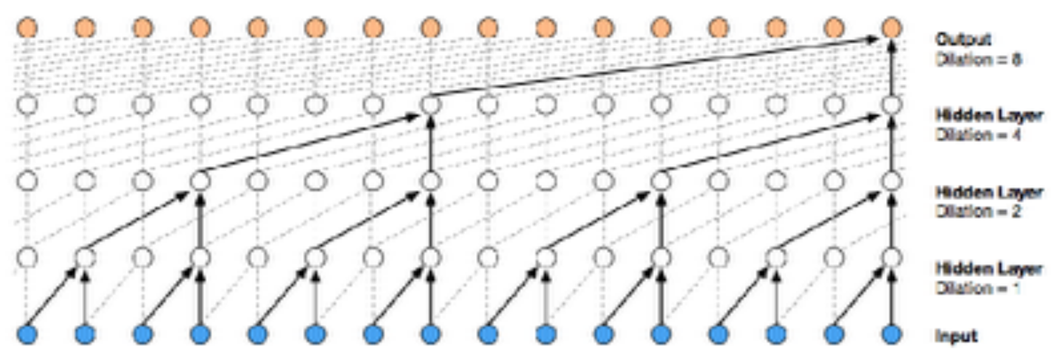


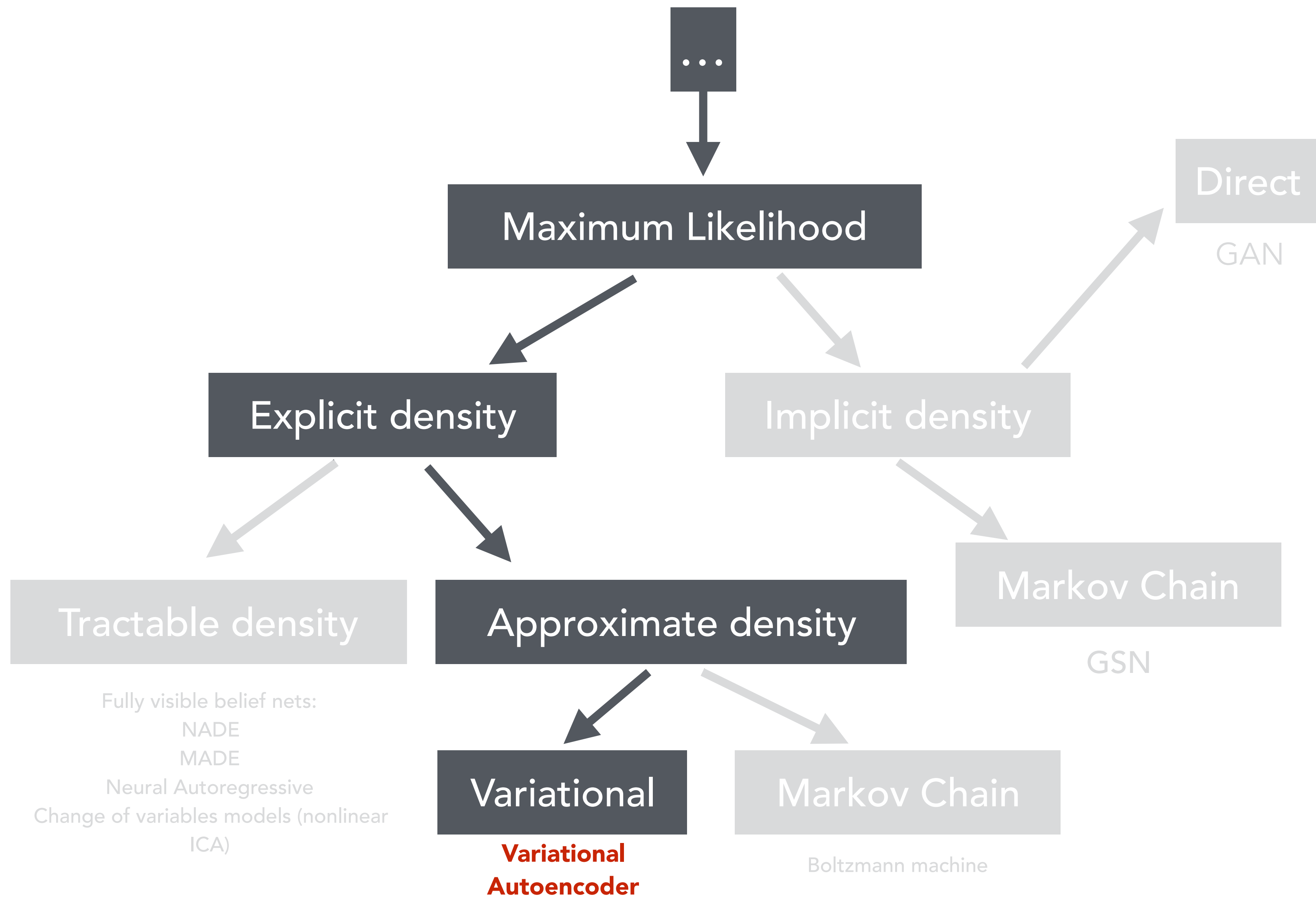
Figure 3: Visualization of a stack of *dilated* causal convolutional layers.

arxiv:1609.03499

Neural Autoregressive Models in Context

- State of the art in temporal signal generation (WaveNet / WaveRNN)
- Autoregressive models admit a **tractable and explicit likelihood**, and can assign a probability to a sample
- Often is expensive to generate samples from distribution

Variational Autoencoders



Anatomy of a Variational Autoencoder (VAE)

- Encoder takes a data point and maps it to a latent code via a neural network, often called an *information bottleneck*
- Decoder takes a latent code and maps it to a sample via a neural network
- For illustrative purposes, assume we are working with binary images

VAE Encoders

- From a sample x, map it (stochastically) to a latent vector z via $q_\theta(z|x)$ (our posterior)
- We have a prior $p(z)$ on $z \sim q_\theta(z|x)$, usually normal
- Our neural net outputs the parameters of the distribution to sample our latent space (assume normal), so we can sample $z \sim q_\theta(z|x)$

VAE Decoders

- Parameterized the generated reconstruction as a neural net $r_\omega(\tilde{x}|z)$

- From a latent code z, we require the reconstructed sample $r_\omega(\tilde{x}|z)$ to be close to the data used to obtain the latent code, x

- We penalize the reconstructions for being wrong, using the binary cross entropy, $\text{BCE}(x, r_\omega(\tilde{x}|z))$

Training a VAE

Want to minimize:

$$\mathbb{E}_{z\sim q_\theta(z|x),x\sim p_{\mathsf{data}}(x)}[\mathsf{BCE}(x, r_\omega(\tilde{x}|z))] +$$
$$\mathsf{KL}(q_\theta(z|x)\|p(z))$$

Training a VAE

Want to minimize:

$$\mathbb{E}_{z \sim q_\theta(z|x), x \sim p_{\text{data}}(x)}[\text{BCE}(x, r_\omega(\tilde{x}|z))] + \text{KL}(q_\theta(z|x) \| p(z))$$

For each datapoint and it's corresponding NN-generated code

Training a VAE

Want to minimize:

Make sure the decoder can reconstruct the sample faithfully

$$\mathbb{E}_{z\sim q_\theta(z|x),x\sim p_{\text{data}}(x)}[\mathsf{BCE}(x, r_\omega(\tilde{x}|z))] + \mathsf{KL}(q_\theta(z|x)\|p(z))$$

For each datapoint and it's corresponding NN-generated code

Training a VAE

Want to minimize:

Make sure the decoder can reconstruct the sample faithfully

$$\mathbb{E}_{z \sim q_\theta(z|x), x \sim p_{\text{data}}(x)}[\text{BCE}(x, r_\omega(\tilde{x}|z))] + \text{KL}(q_\theta(z|x) \| p(z))$$

For each datapoint and it's corresponding NN-generated code

But don't make our encoder output codes that don't look like our prior (Normal)

Training a VAE

Want to minimize:

Make sure the decoder can reconstruct the sample faithfully

$$\mathbb{E}_{z \sim q_\theta(z|x), x \sim p_{\text{data}}(x)}[\text{BCE}(x, r_\omega(\tilde{x}|z))] + \text{KL}(q_\theta(z|x) \| p(z))$$

For each datapoint and it's corresponding NN-generated code

But don't make our encoder output codes that don't look like our prior (Normal)

Minimize this over the parameters of our encoder NN θ and our decoder network ω

VAEs in Context

- Though not a pure generative model, sampling from prior and running through the decoder NN is a generative model!

- Very popular, can modify simplistic framework shown today to enforce interpretable latent spaces, other desirable properties

- They are **bidirectional**, i.e., I can not only sample from them, but encode samples (great for unsupervised learning)

Conclusion

- Today:
  - High level generative modeling taxonomy
  - Detail into GANs, Neural Autoregressive Models, and VAEs
- Generative have many interesting applications in HEP

Thanks!

Backup

Proof of Vanilla GAN formulation

- How can we jointly optimize G and D?

- Construct a two-person zero-sum minimax game with a value V

$$V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x;\theta_D)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z;\theta_G);\theta_D))]$$

- We have an inner maximization by D and an outer minimization by G

$$\min_G \max_D V(D,G)$$

Theoretical Guarantees

- From original paper, know that $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x;\theta_G)}$
- Define generator solving for infinite capacity discriminator, $C(G) = V(D^*, G)$
- We can rewrite value as

$$C(G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x;\theta_G)}] + \mathbb{E}_{x \sim p_{\text{model}}(x;\theta_G)}[\log \frac{p_{\text{model}}(x;\theta_G)}{p_{\text{data}}(x) + p_{\text{model}}(x;\theta_G)}]$$

- Simplifying notation, and applying some algebra

$$C(G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log \frac{p_{\text{data}}}{p_{\text{data}} + p_{\text{model}}}] + \mathbb{E}_{x \sim p_{\text{model}}}[\log \frac{p_{\text{model}}}{p_{\text{data}} + p_{\text{model}}}]$$

$$C(G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log \frac{p_{\text{data}}}{p_{\text{data}}/2 + p_{\text{model}}/2}] + \mathbb{E}_{x \sim p_{\text{model}}}[\log \frac{p_{\text{model}}}{p_{\text{data}}/2 + p_{\text{model}}/2}] - \log(4)$$

- But we recognize this as a summation of two KL-divergences

$$C(G) = D_{\text{KL}}\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_{\text{model}}}{2}\right) + D_{\text{KL}}\left(p_{\text{model}} \middle\| \frac{p_{\text{data}} + p_{\text{model}}}{2}\right) - \log(4)$$

- And can combine these into the Jenson-Shannon divergence

$$C(G) = 2 \cdot \text{JSD}\left(p_{\text{data}} \| p_{\text{model}}\right) - \log(4)$$

- This yields a unique global minimum precisely when

$$p_{\text{model}} = p_{\text{data}} \implies C(G) = -\log(4)$$

Theoretical Guarantees

- TL;DR from the previous proof is as follows
- If D and G are allowed to come from the space of all continuous functions, then we have:
    - Unique equilibrium $(\theta_G^{\text{opt}}, \theta_D^{\text{opt}})$
    - The discriminator admits a flat posterior, i.e.,

$$D(x; \theta_D^{\text{opt}}) = 1/2 \qquad D(G(z; \theta_G^{\text{opt}}); \theta_D^{\text{opt}}) = 1/2$$

$$\forall x \sim p_{\text{data}}(x) \qquad \forall z \sim p_z(z)$$

    - The implicit distribution defined by the generator exactly recovers the data distribution $p_{\text{model}}(x; \theta_G^{\text{opt}}) = p_{\text{data}}(x)$