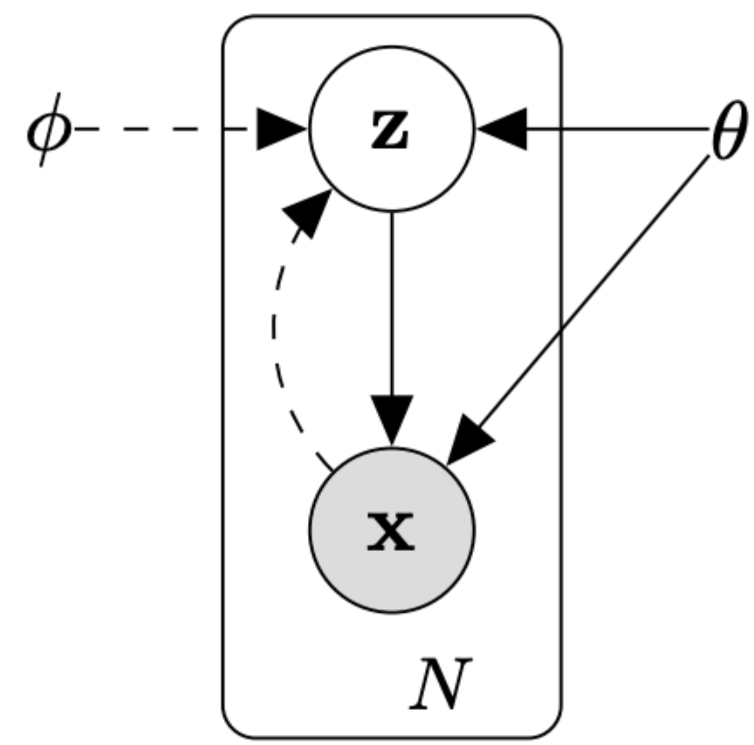
STA414/2104

Statistical Methods for Machine Learning II

Murat A. Erdogdu & David Duvenaud

Department of Computer Science
Department of Statistical Sciences

Variational Autoencoders



UNIVERSITY OF TORONTO

Today

- Recap PPCA, extend to non-linear, non-Gaussian
- Recap variational inference, extend to per-datapoint latent variables
- Train a neural network to help us optimize the ELBO for each datapoint faster
- Extension to time-series models

Variational Inference

- Optimize a tractable distribution $q(z|x)$ to match $p(z|x)$
- Main difficulty: Measure difference between $q(z|x)$ and $p(z|x)$ using only cheap operations.
- By assumption, we can't sample from $p(z|x)$ or evaluate its normalized density. We can:
  - Sample from $q(z|x)$ and evaluate its density
  - Evaluate density $p(x, z)$ (or unnormalized $p(z|x)$)

Variational Inference

- Directly optimize the parameters phi of an approximate distribution q(z|x, phi) to match p(z|x, theta)

- What if there is a local latent variable per-datapoint, and some global parameters? e.g. Bayesian PCA, generative image models, topic models

- Directly optimize the parameters phi_i of each approximate distribution q(z_i|x_i, phi_i) to match p(z_i|x_i, theta)

ADVI Algorithm:

- Loop:
  - 1. Sample z ~ q(z|x, phi)
  - 2. Compute gradient w.r.t phi of
    log p(z, x) - log q(z | x_i, phi)
  - Update phi with gradient
- Eventually gives phi = local argmin KL(q(z|x)||p(z|x))
- In this setting, only one set of latent params z, as in a Bayesian neural net

ADVI with per-data latents:

Loop:

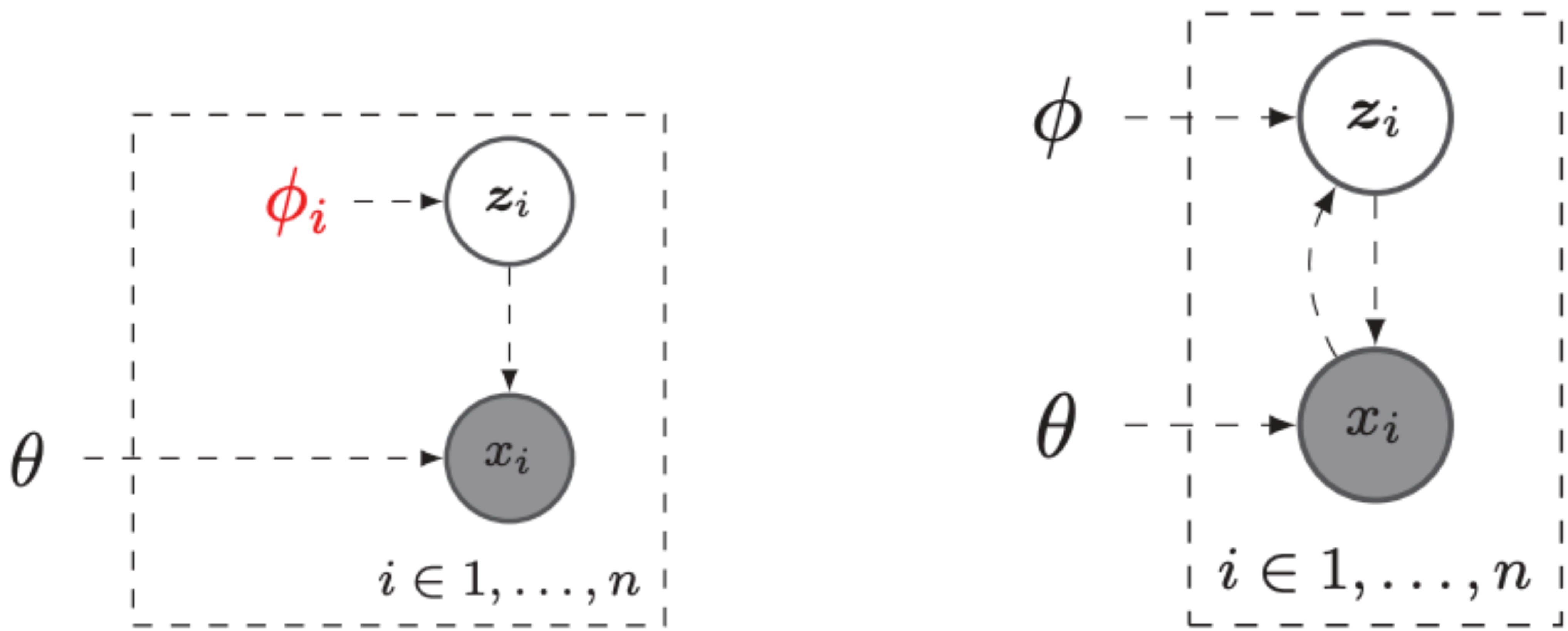
1. Sample x_i from dataset
2. phi_i = argmin KL(q(z_i|x_i, phi_i)|| p(z_i|x_i, theta))
3. Sample z ~ q(z_i|x_i, phi_i)
4. Get gradient w.r.t. theta of
   log p(z_i, x_i | theta) - log q(z_i | x_i, phi_i)
5. Update theta with gradient

Variational Autoencoder:

Loop:

1. Sample x_i from dataset
2. phi_i = neural_net_predict(x, phi_r)
3. Sample z ~ q(z_i|x_i, phi_i)
4. Get gradient w.r.t. theta and phi_r of
   log p(z_i, x_i | theta) - log q(z_i | x_i, phi_i)
5. Update theta and phi_r with gradient

In graphical notation:



(a) SVI for iid observations. (b) VAE for iid observations.

Multi-Level Variational Autoencoder: Learning Disentangled Representations from Grouped Observations. Ryota Tomioka, Sebastian Nowozin. 2018

Consequences of using encoder

- Gradient updates of theta is like M-step
  phi_i = nn_predict(x, phi_r) is approximate E-step
  Gradient updates of phi_r improves E-step

- Don't need to re-optimize phi_i each time theta changes - much faster

- Recognition net won't necessary give optimal phi_i

- Can have fast test-time inference (vision)

VAE ELBO

```python
def elbo(theta, phi, x):
    z_mu, z_log_sigma = nn_predict_gaussian(phi, x)   # Encode
    z = sample_diag_gaussian(z_mu, z_log_sigma)        # Sample latents
    mu_x = neural_net_predict(theta, z)                # Decode

    logq_z = diag_gaussian_log_density(z, z_mu, z_log_sigma)
    logp_z = diag_gaussian_log_density(z, 0,    np.log(1.0))
    logp_x_given_z = bernoulli_log_density(x, mu_x)
    return np.mean(logp_x_given_z + logp_z - logq_z)
```

Show VAE demo

Simple but not obvious

- It took a long time get here!
    - Independently developed as denoising autoencoders (Bengio et al.) and amortized inference (many others)
    - Helmholtz machine - same idea in 1995 but used discrete latent variables

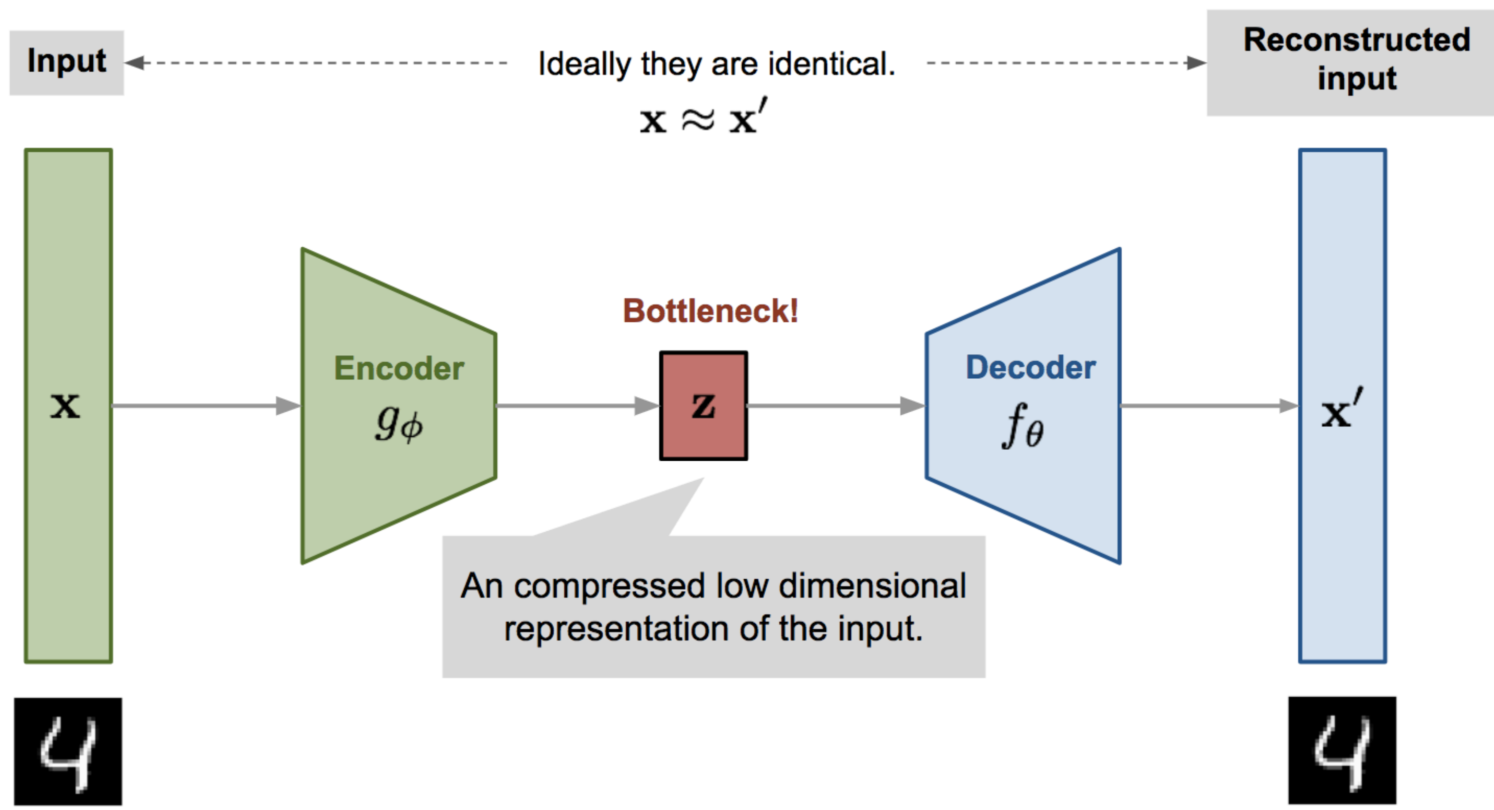
The Helmholtz Machine

Peter Dayan
Geoffrey E. Hinton
Radford M. Neal
*Department of Computer Science, University of Toronto,
6 King's College Road, Toronto, Ontario M5S 1A4, Canada*

Richard S. Zemel
CNL, The Salk Institute, PO Box 85800, San Diego, CA 92186-5800 USA

Discovering the structure inherent in a set of patterns is a fundamental aim of statistical inference or learning. One fruitful approach is to build a parameterized stochastic generative model, independent draws from which are likely to produce the patterns. For all but the simplest generative models, each pattern can be generated in exponentially many ways. It is thus intractable to adjust the parameters to maximize

Autoencoder Motivation



https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

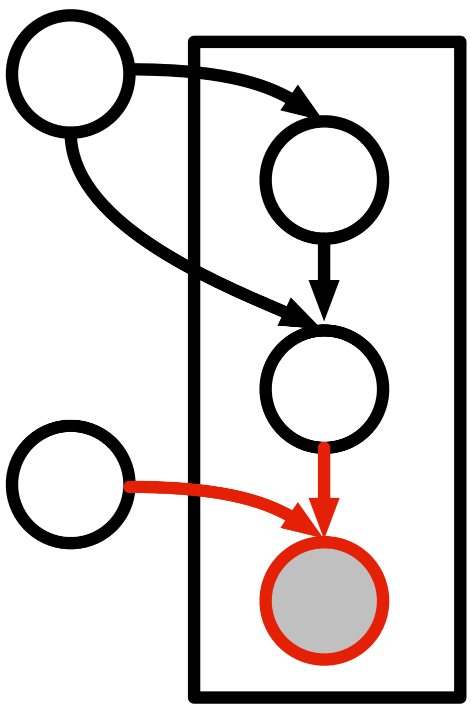
- Want compact representation of data.
  Need to prevent enc = dec = identity.

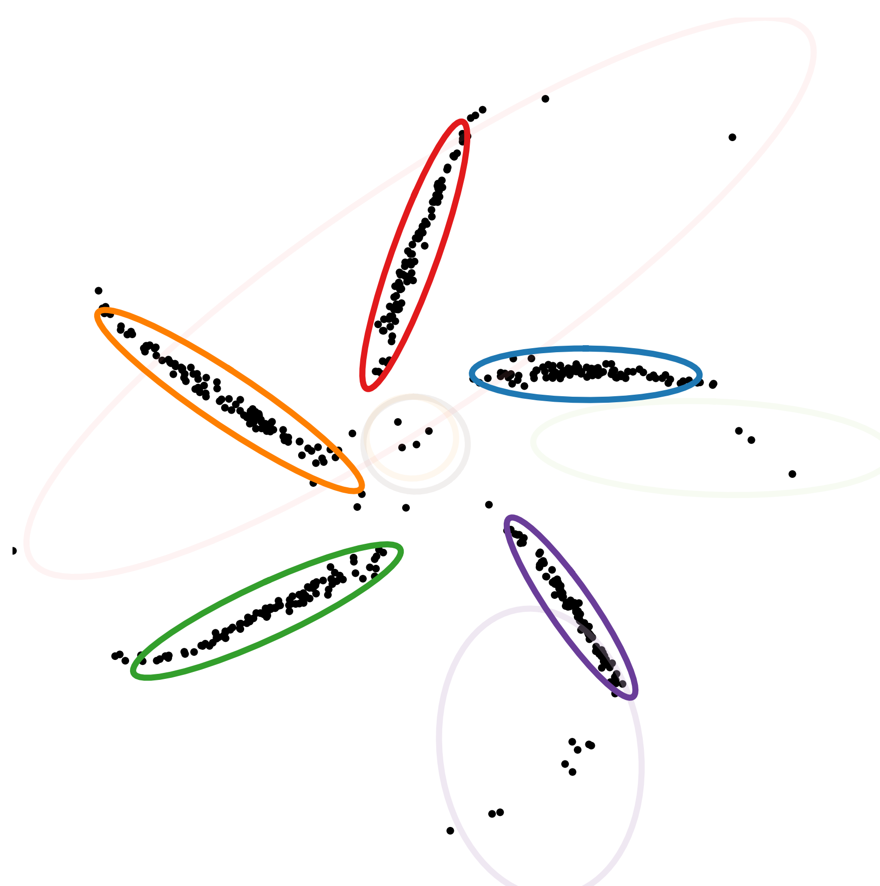
- Hack 1: Low-dim z. But what dimension?

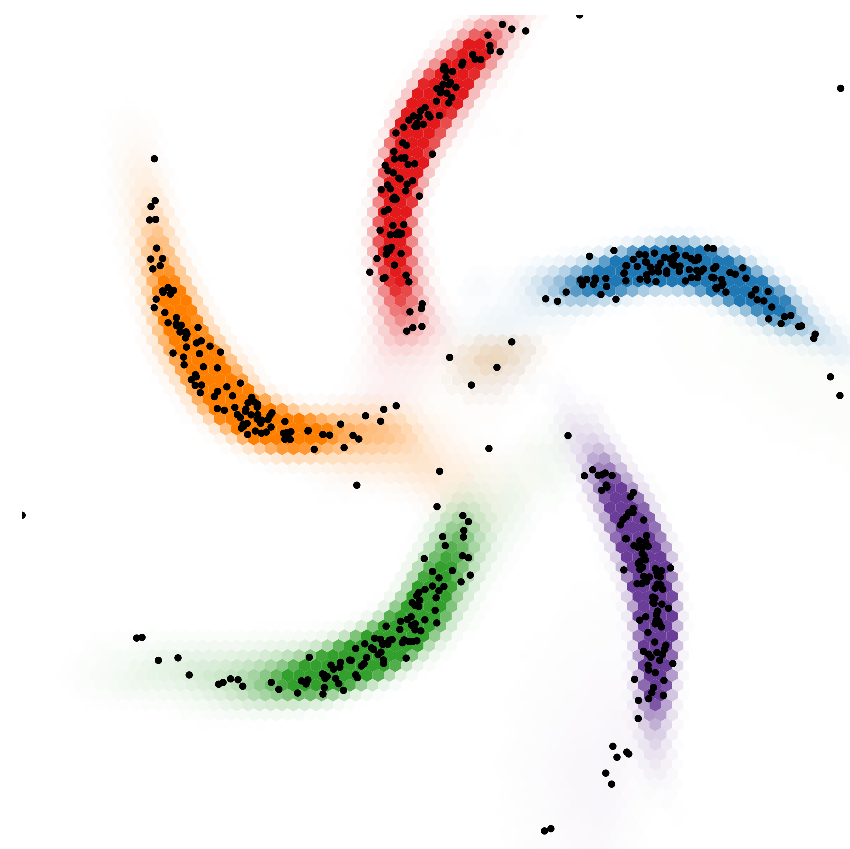
- Hack 2: Add noise to data before encoding, reconstruct original data. But how much noise?

- Hack 3: Add noise to latents after encoding, reconstruct original data. But how much noise?

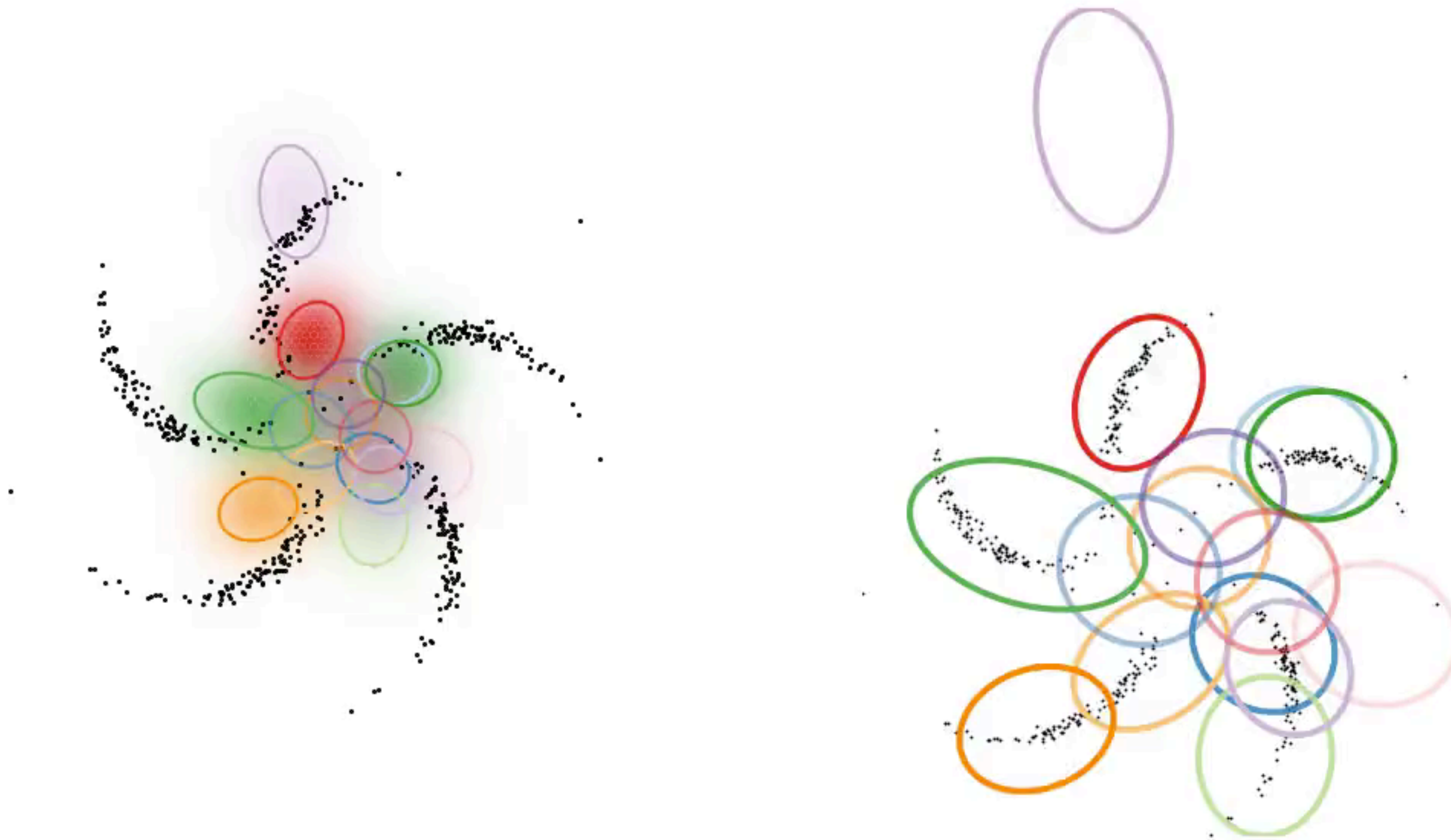
Modeling idea: graphical models on latent variables,
neural network models for observations







Composing graphical models with neural networks for structured representations and fast inference. Johnson, Duvenaud, Wiltschko, Datta, Adams, NIPS 2016



data space

latent space

Learning latent dimension

- Standard autoencoders require choosing latent dimension
- What happens if a VAE has more than it needs?

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{z}\sim q}[\log p(x|z)] - KL(q_\phi(z|x)||p(z))$$

- If q(z|x) is factorized, then KL term factorizes over dimensions, wants to make each q(z_i|x) look like p(z_i)
- If a dimension doesn’t help likelihood enough, it will ‘turn off’ and set q(z_i|x) = p(z_i), ignoring x. Then decoder can ignore too.

Reconstructions

- Start with input x
- Encode and sample: $z \sim q(z|x)$
- Decode and sample: $r \sim p(x|z)$
- Compare x and r
- If encoder is true posterior, $q(z|x) = p(z|x)$,
  then r is sampled from $p(x|x)$??
- Model can produce perfect $p(x)$ and also bad reconstructions

z doesn’t capture everything

(a) Multiple decoding of the same **z**

(b) Random samples from the our Auxiliary prior

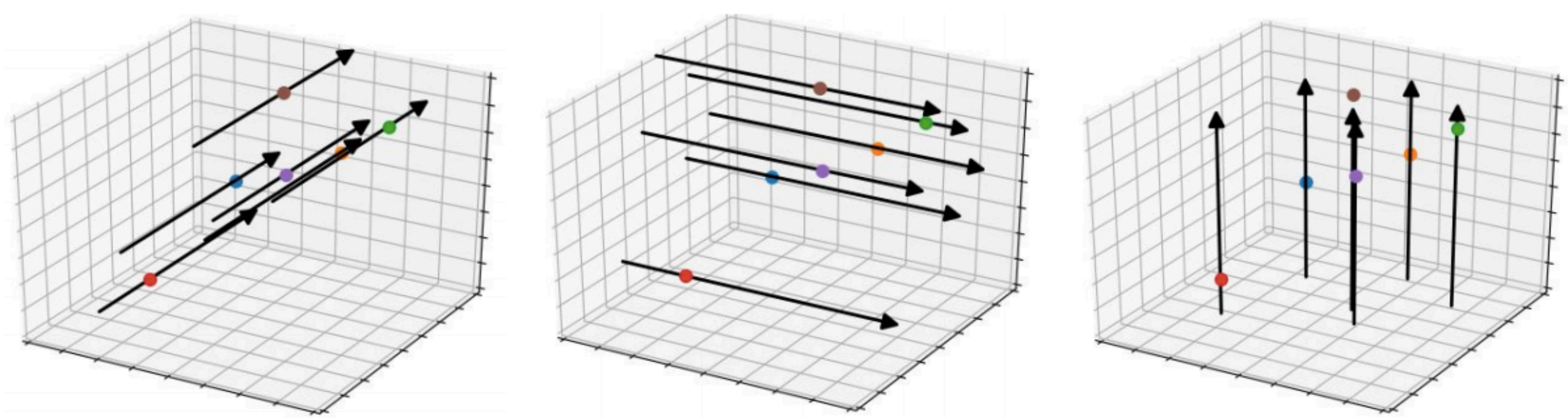
Preventing Posterior Collapse with delta-VAEs
Ali Razavi, Aäron van den Oord, Ben Poole, Oriol Vinyals, 2019

Benefits of compact latent code

- http://www.dpkingma.com/sgvb_mnist_demo/demo.html
- Nearby z's give similar x
- Recent work on 'disentangling' latent rep

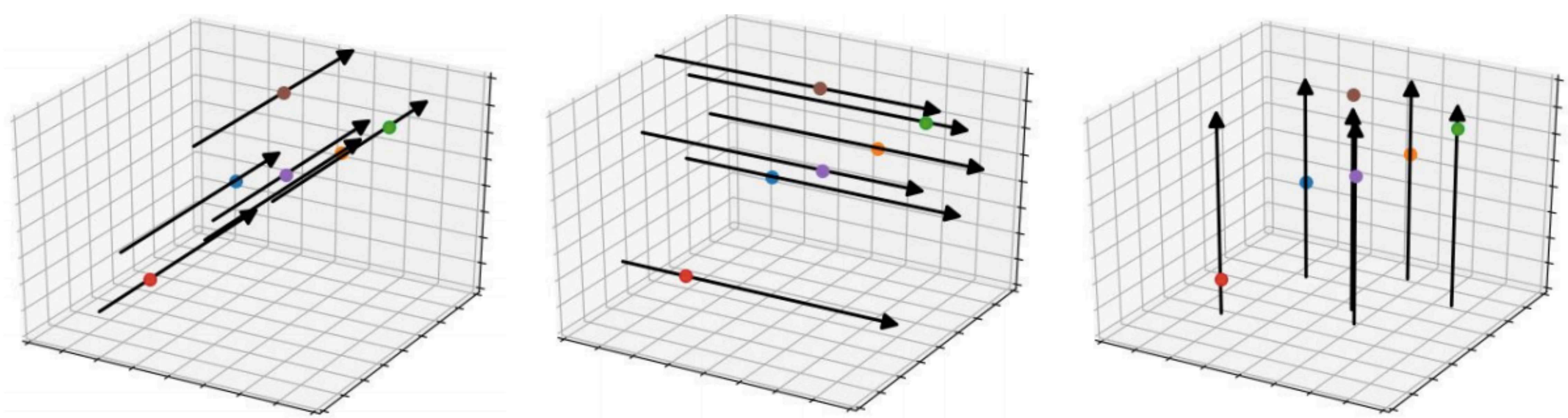
Disentanglement

Hue



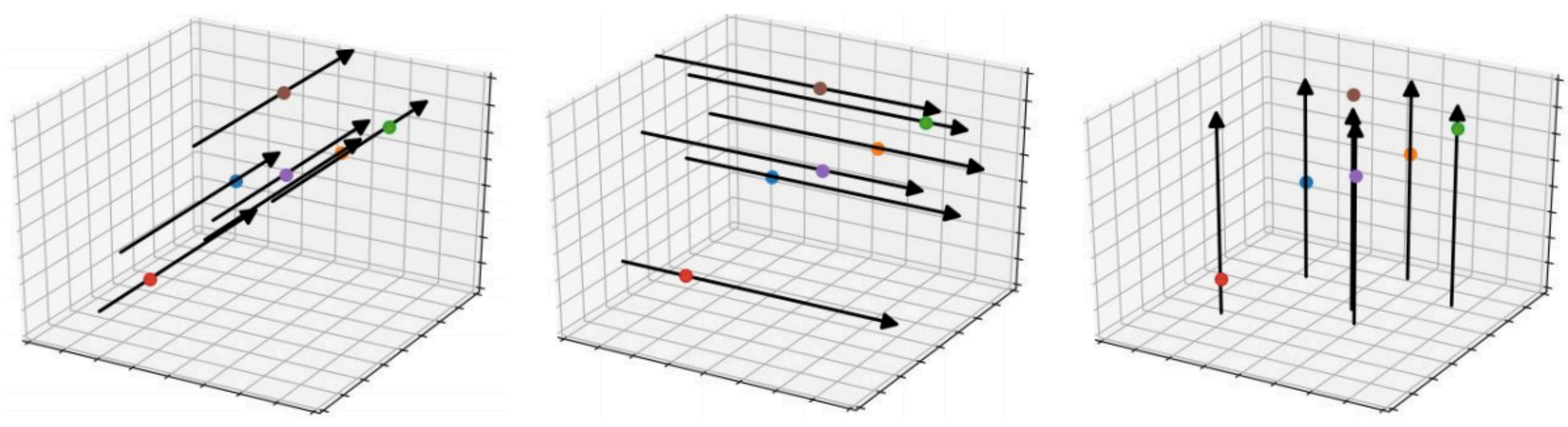
Disentanglement

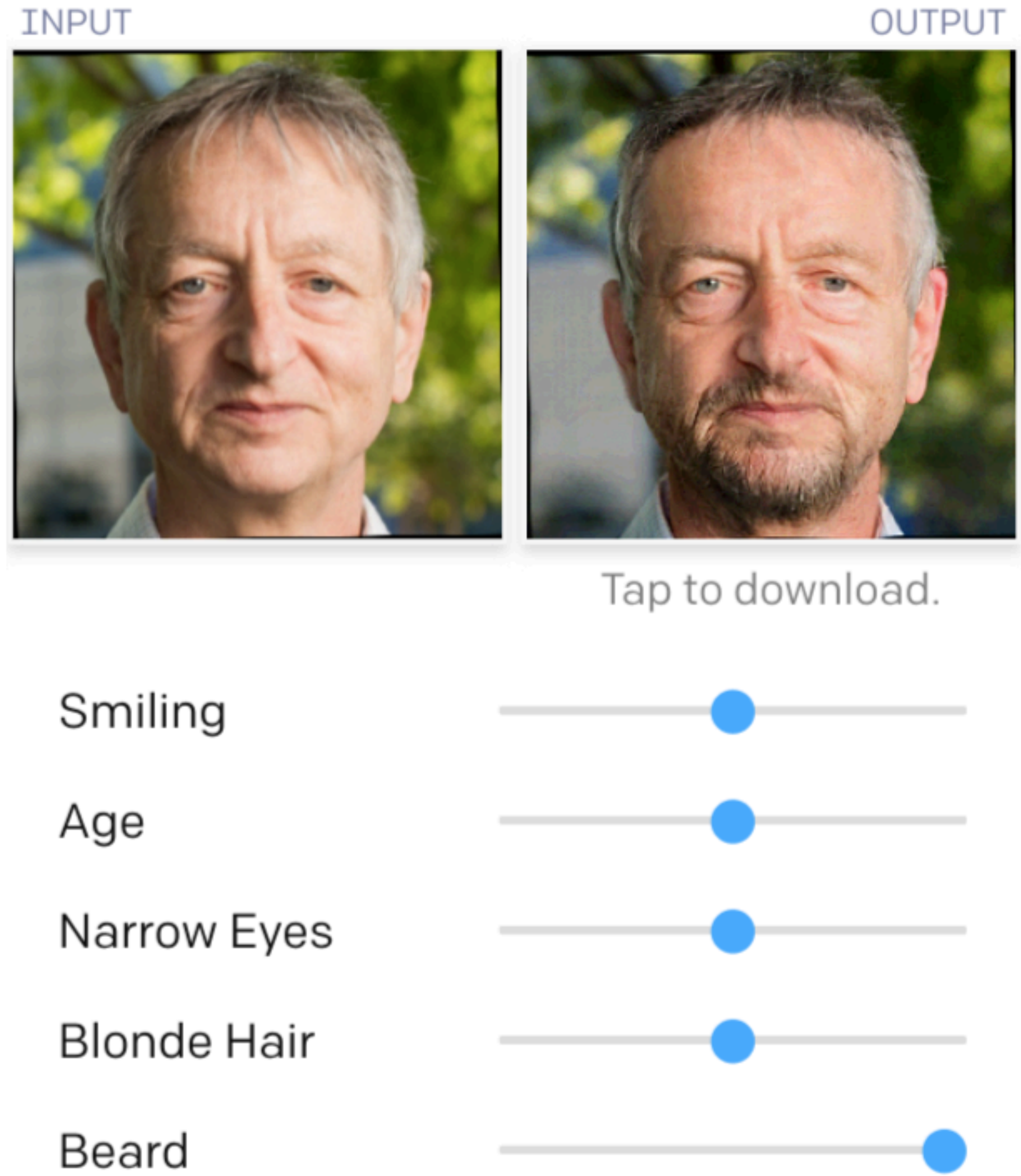
Mustache



Disentanglement

Smoldering Look





https://openai.com/blog/glow/

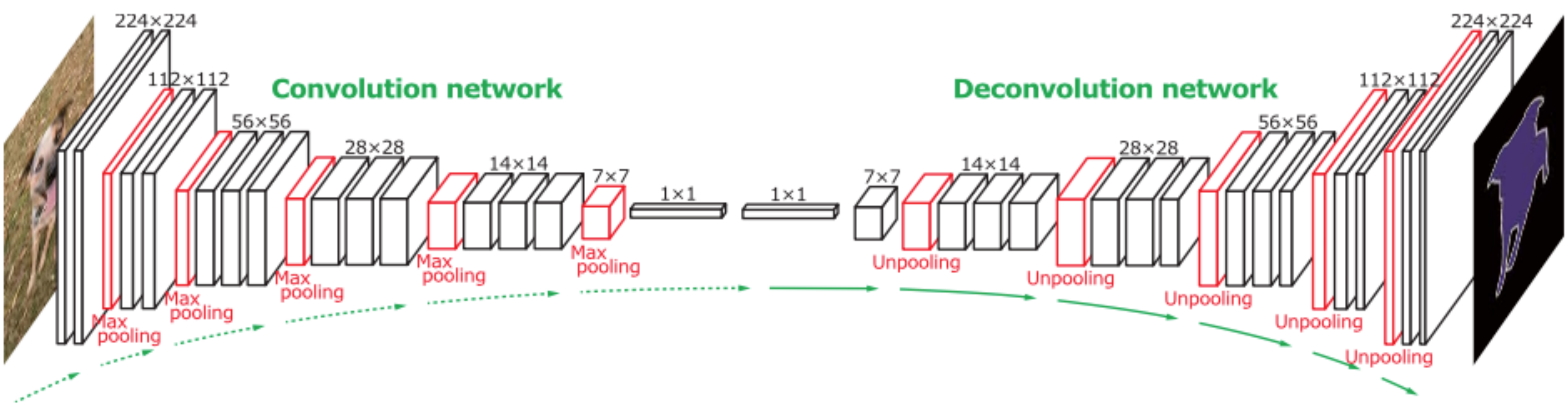
Designing the Decoder

- Only need p(z) and p(x|z) to be tractable
- Orginally, p(x|z) = N(x | dec(z, theta), sigma I). This is an instance of Naive Bayes.
- Final step has independence assumption, causes noisy samples, blurry means
- p(x|z) can be anything: rnn, pixelRNN, real NVP, de_convolutional net. More powerful but more expensive to compute.

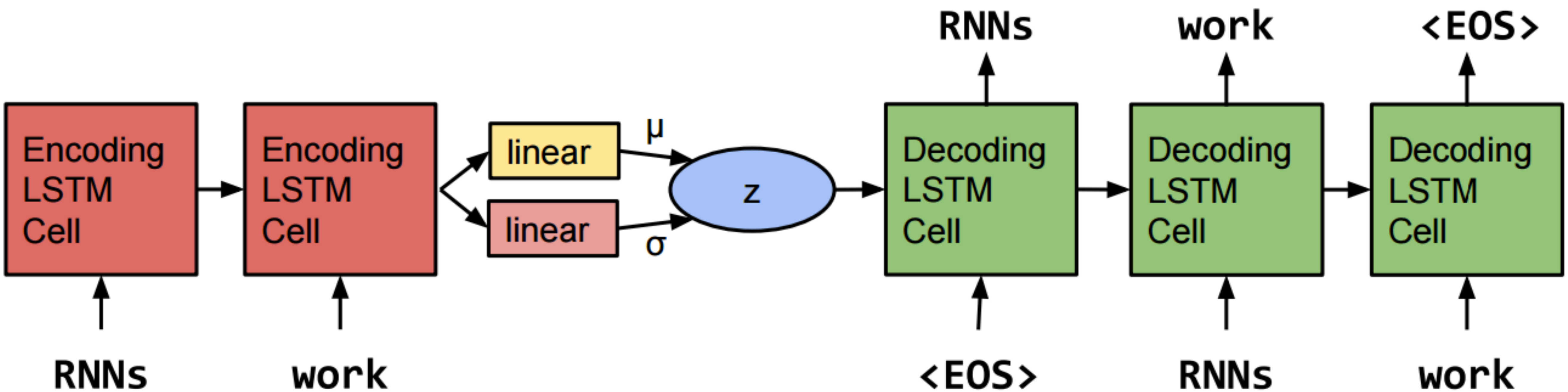
Figure 6: Samples from hierarchical PixelVAE on the 64x64 ImageNet dataset.

Designing the Decoder

- Decoder often looks like inverse of encoder
- Encoders can come from supervised learning



Text autoencoders



- *Generating Sentences from a Continuous Space.* Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, Samy Bengio

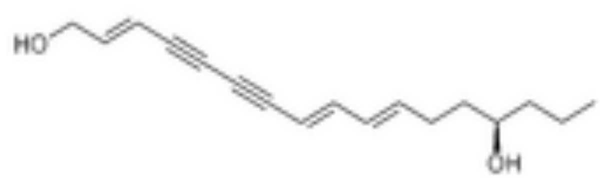
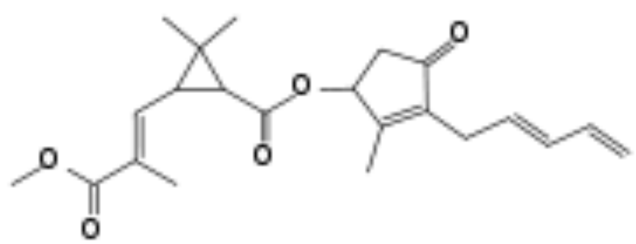
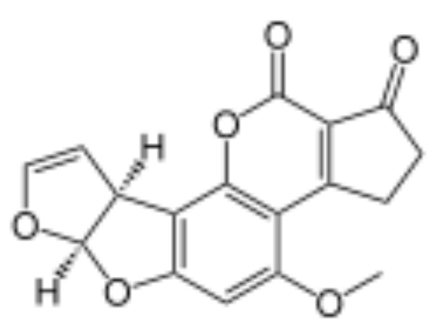
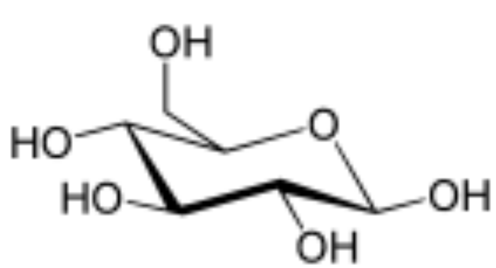
Text VAE - **Interpolation**

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .

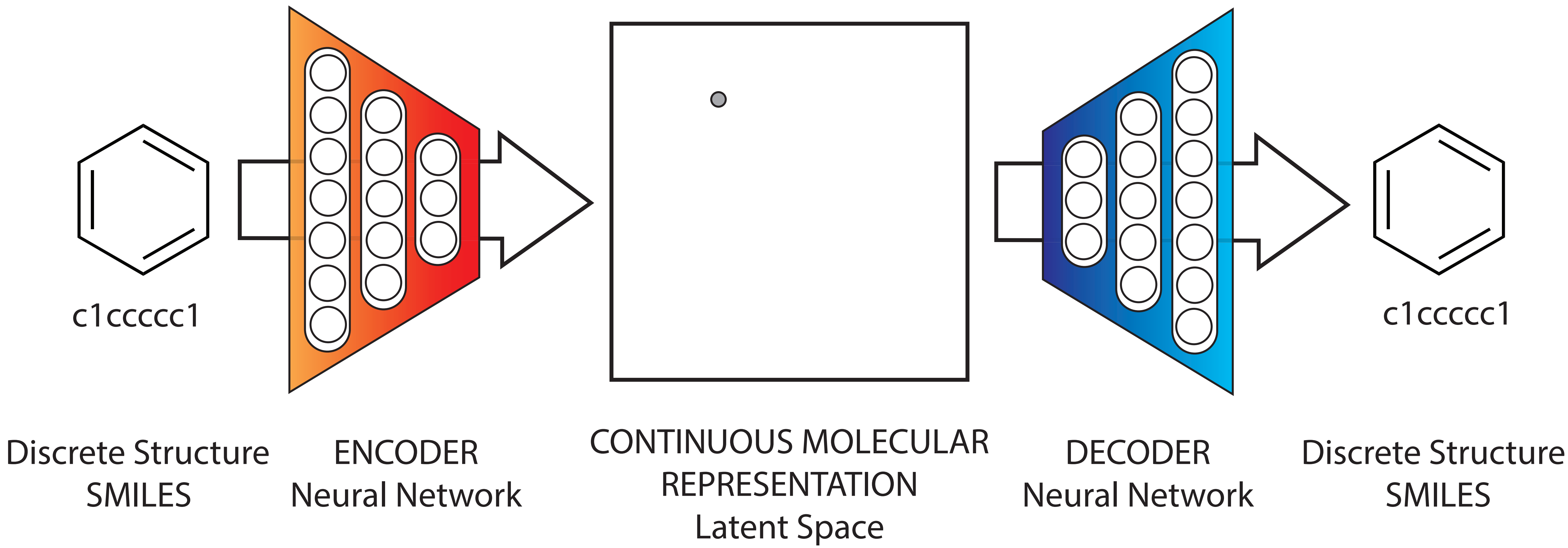
it made me want to cry .
no one had seen him since .
it made me feel uneasy .
no one had seen him .
the thought made me smile .
the pain was unbearable .
the crowd was silent .
the man called out .
the old man said .
the man asked .

he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

What is a molecule?

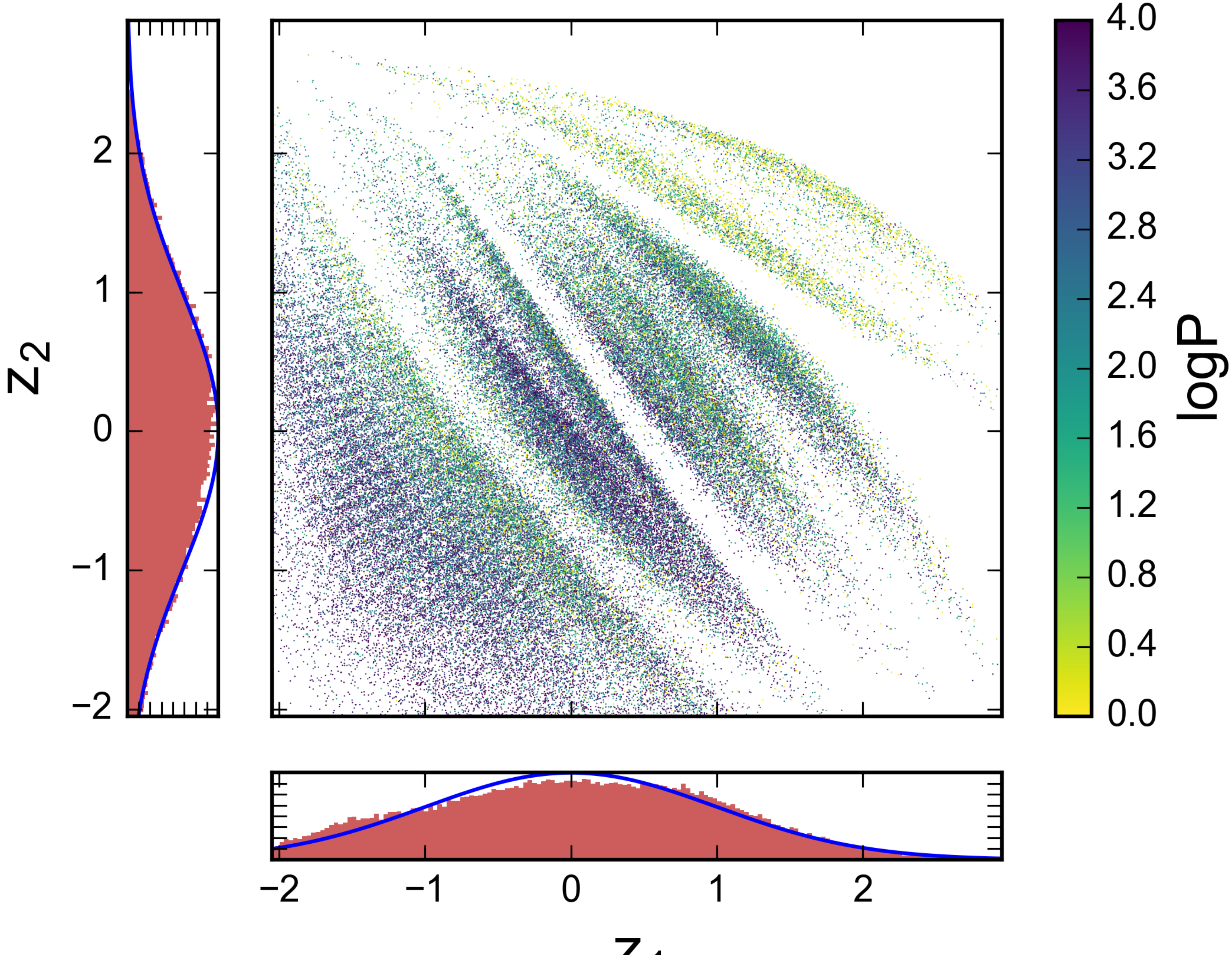
Graph	SMILES string
	CCC[C@@H](O)CC\C=C\C=C\C#CC#C\C=C\CO
	COC(=O)C(\C)=C\C1C(C)(C)[C@H]1C(=O)O[C@@H]2C(C)=C(C(=O)C2)CC=CC=C
	O1C=C[C@H]([C@H]1O2)c3c2cc(OC)c4c3OC(=O)C5=C4CCC(=O)5
	OC[C@@H](O1)[C@@H](O)[C@H](O)[C@@H](O)[C@@H](O)1

Repurposing text autoencoders

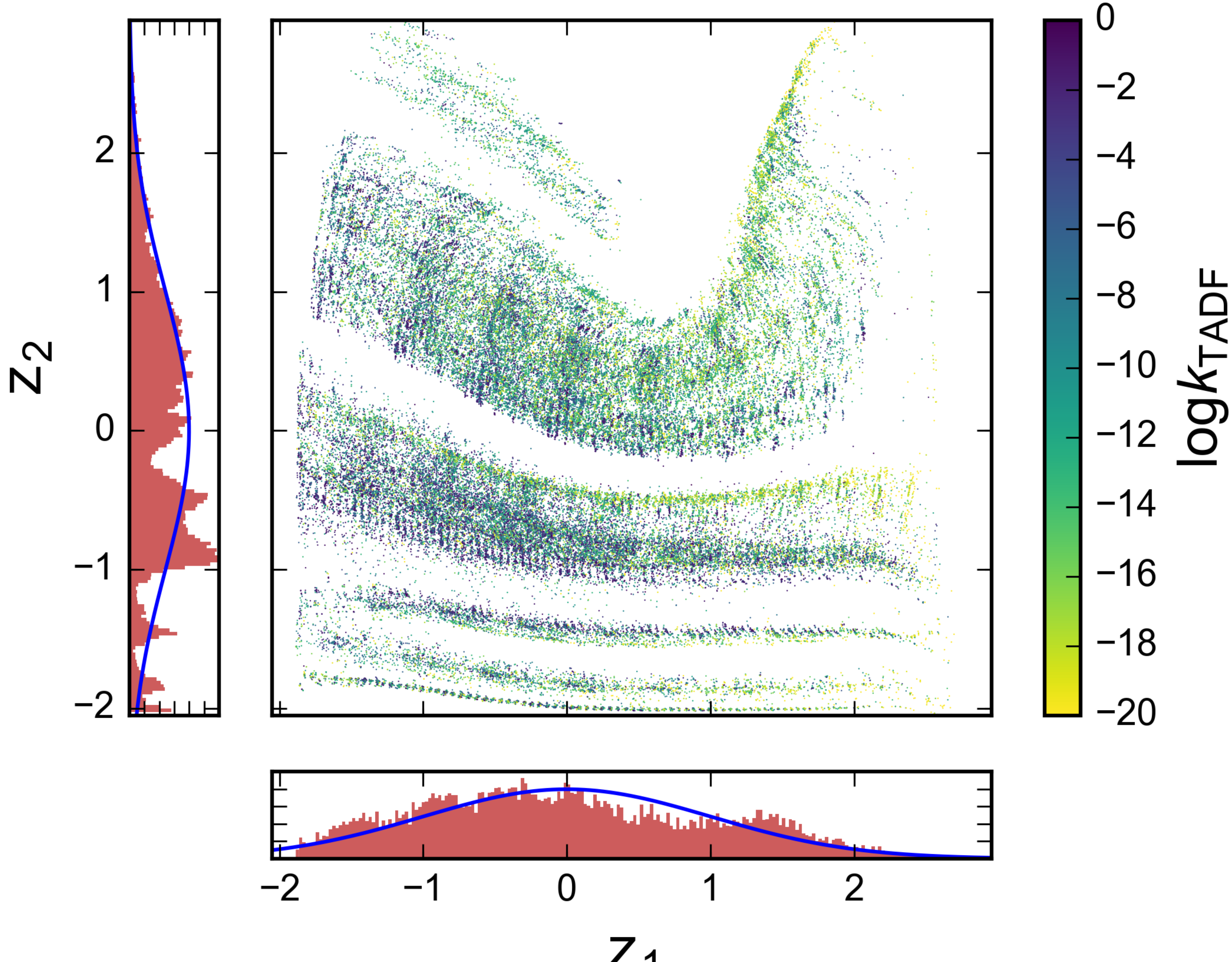


Can be trained on unlabeled data

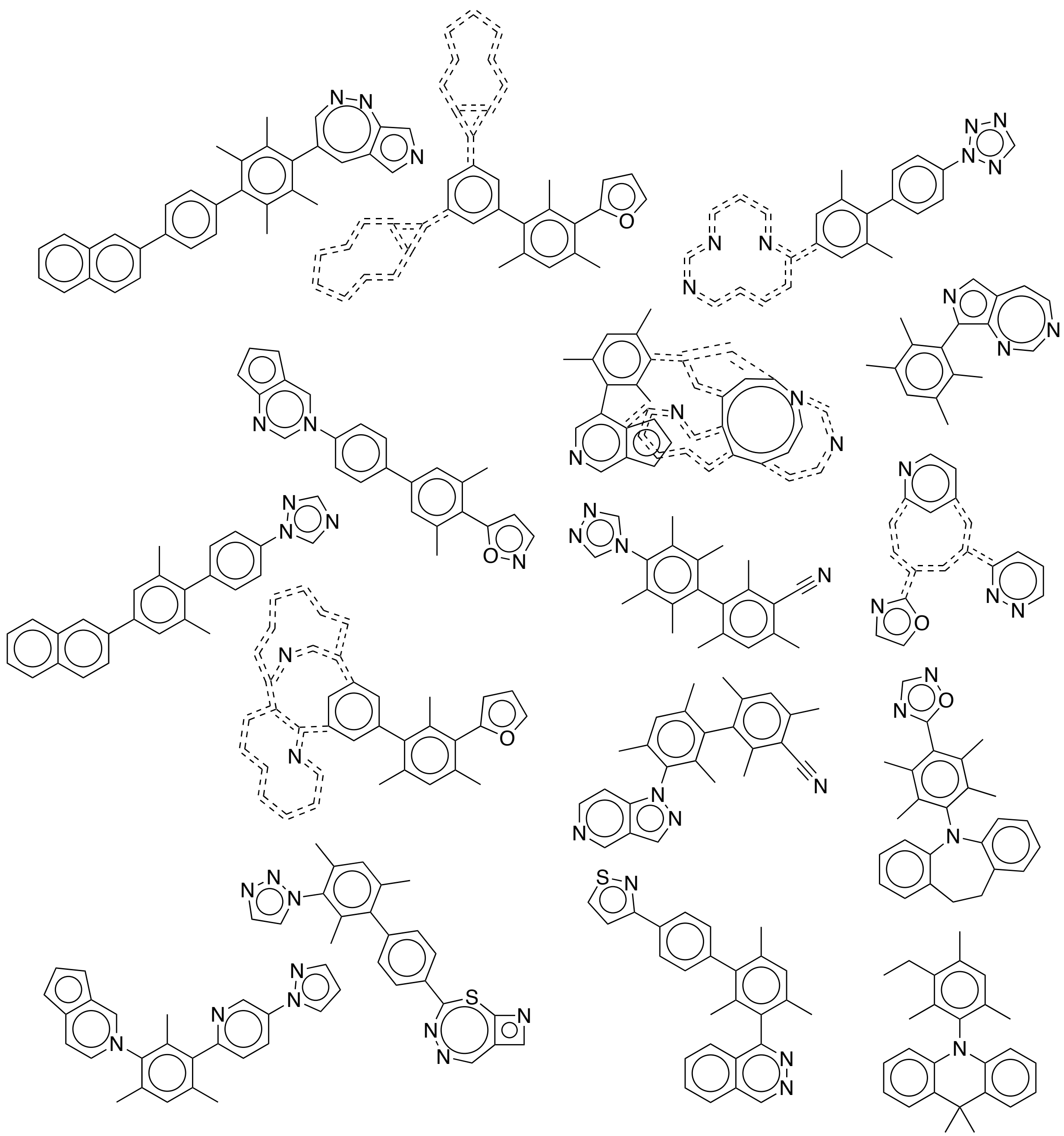
Map of 220,000 Drugs



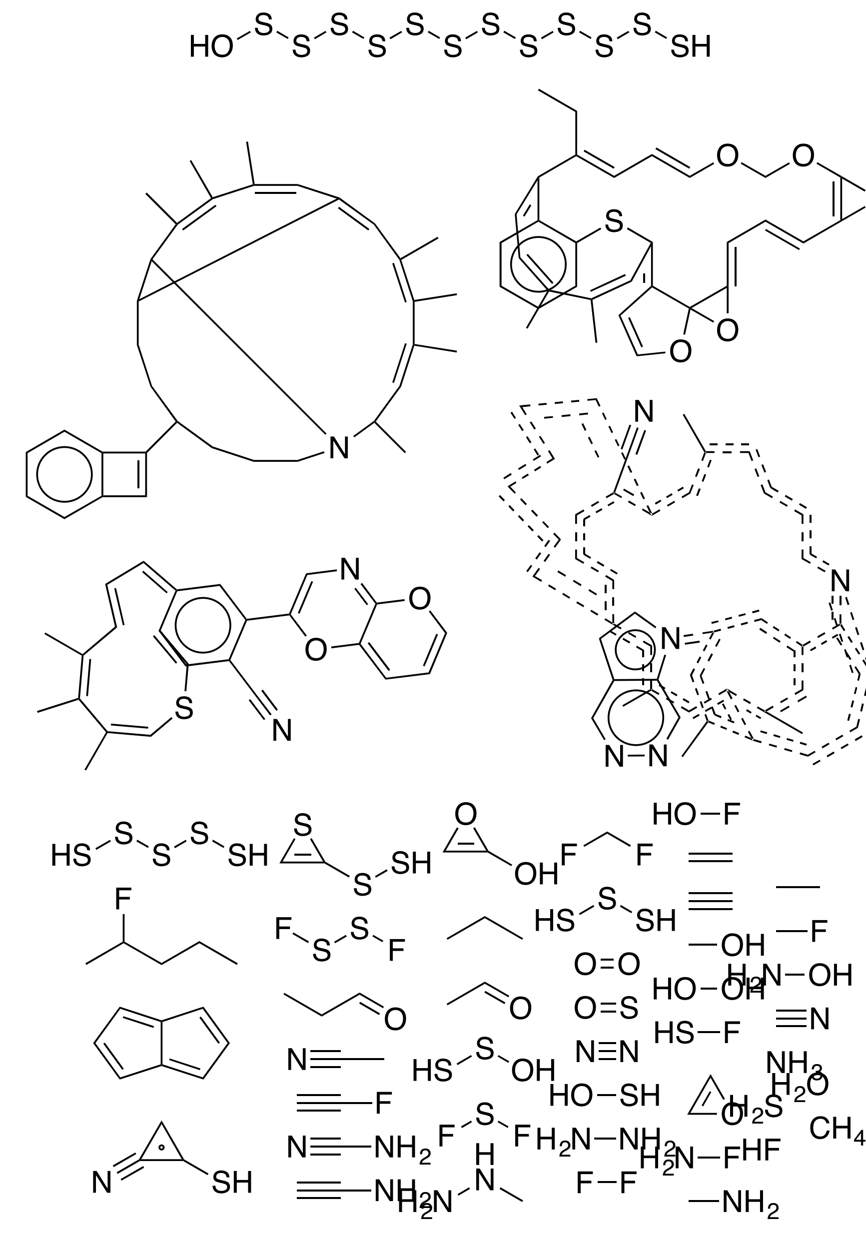
Map of 100,000 OLEDs



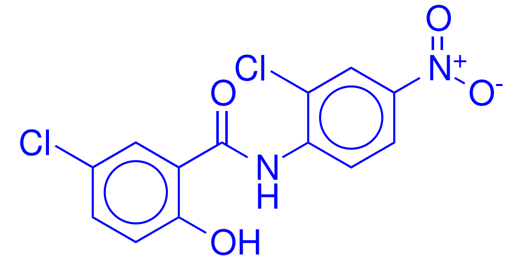
Random Organic LEDs

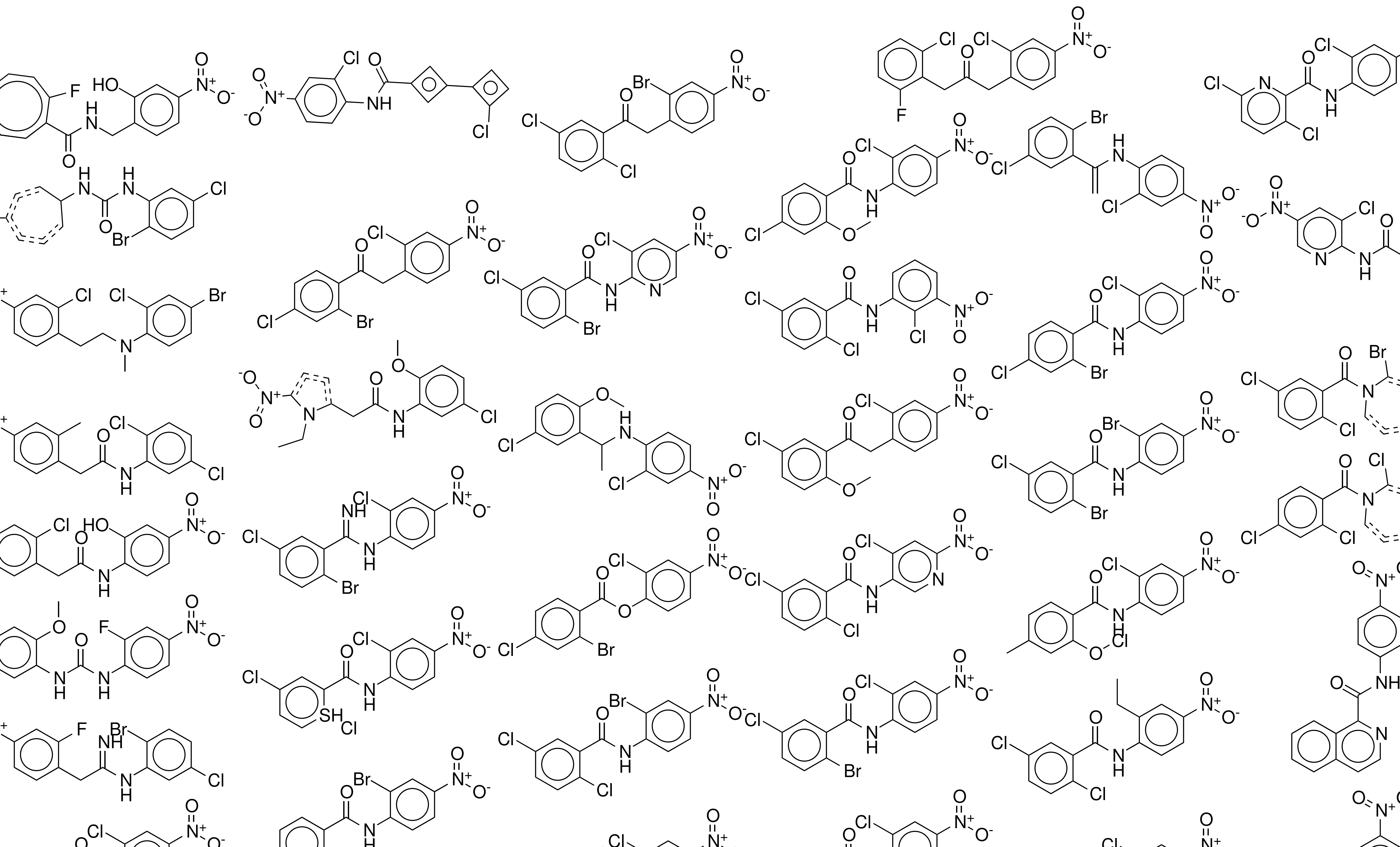


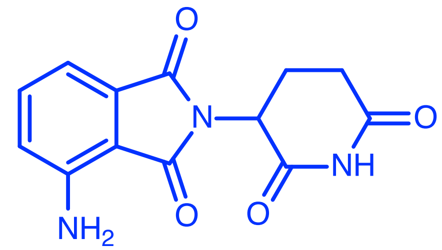
Variational autoencoder

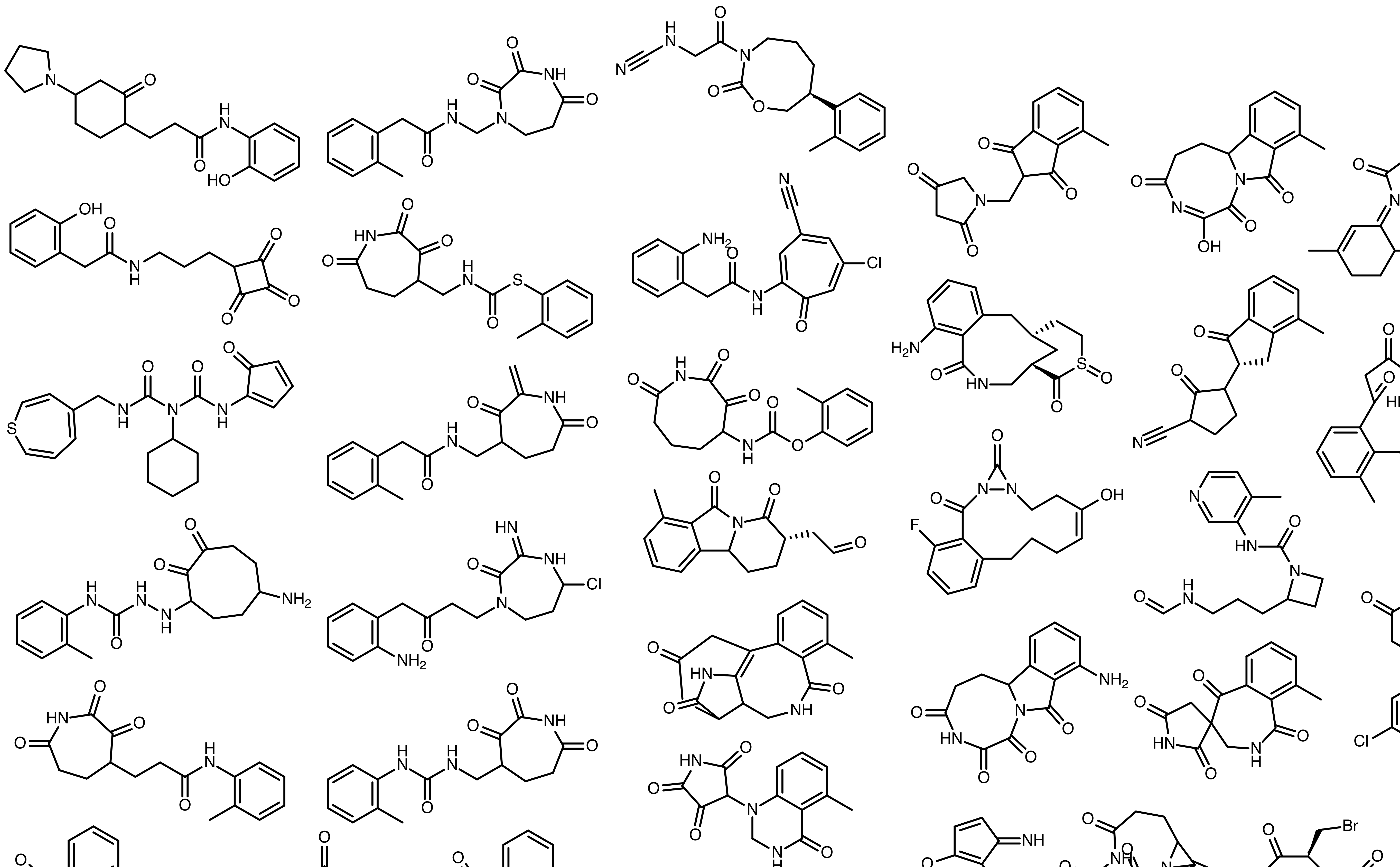


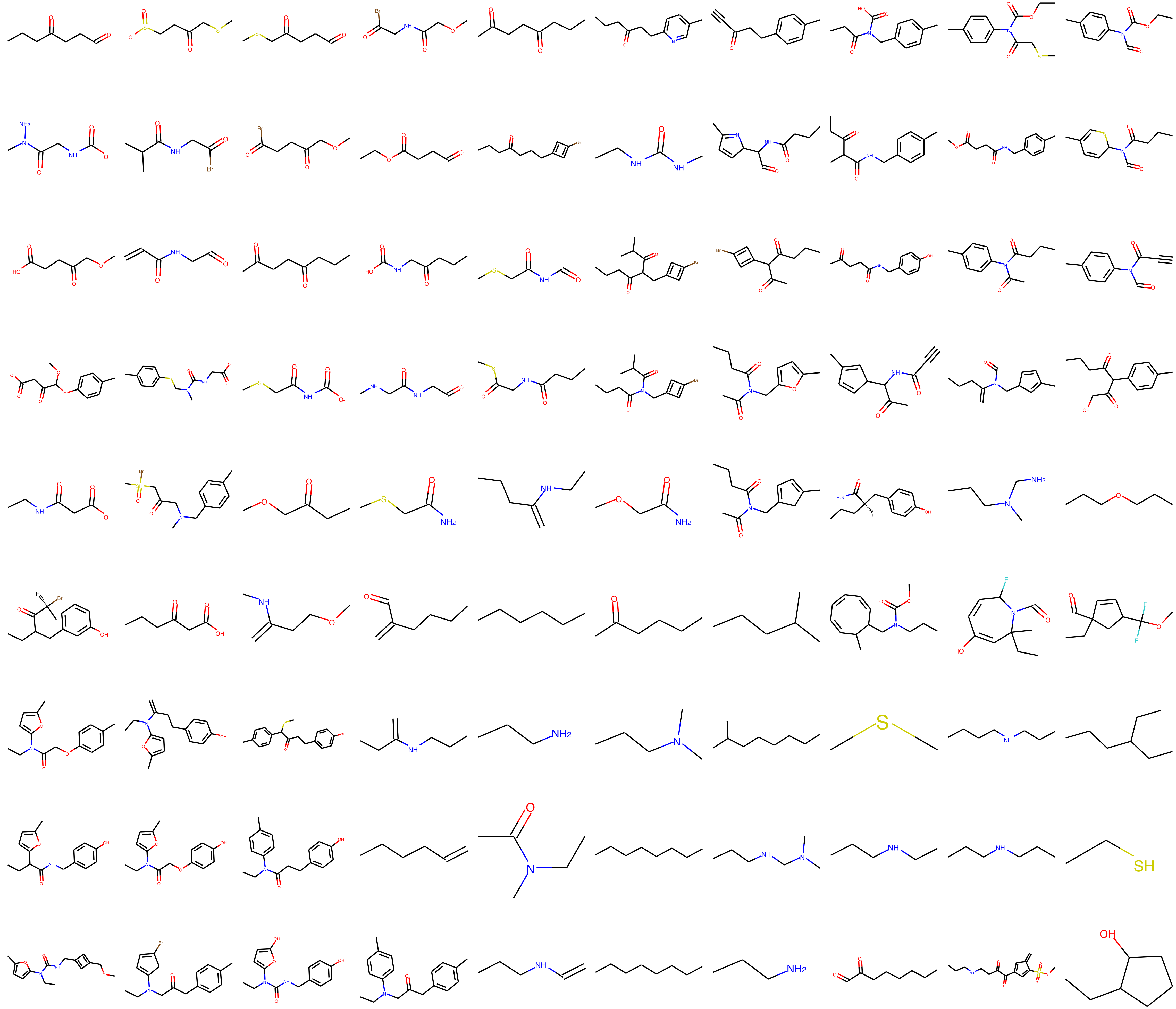
Standard autoencoder

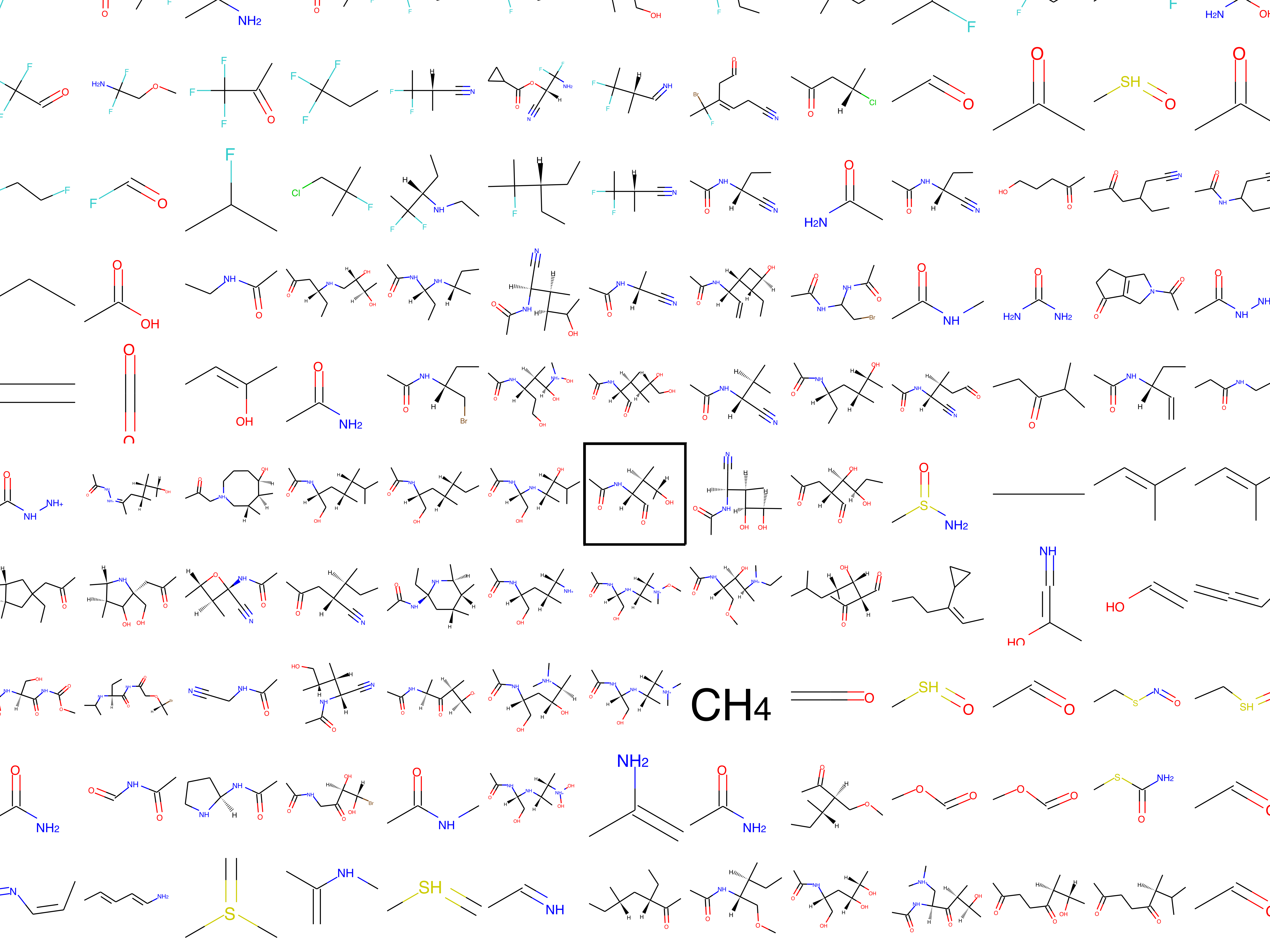
Molecules near 

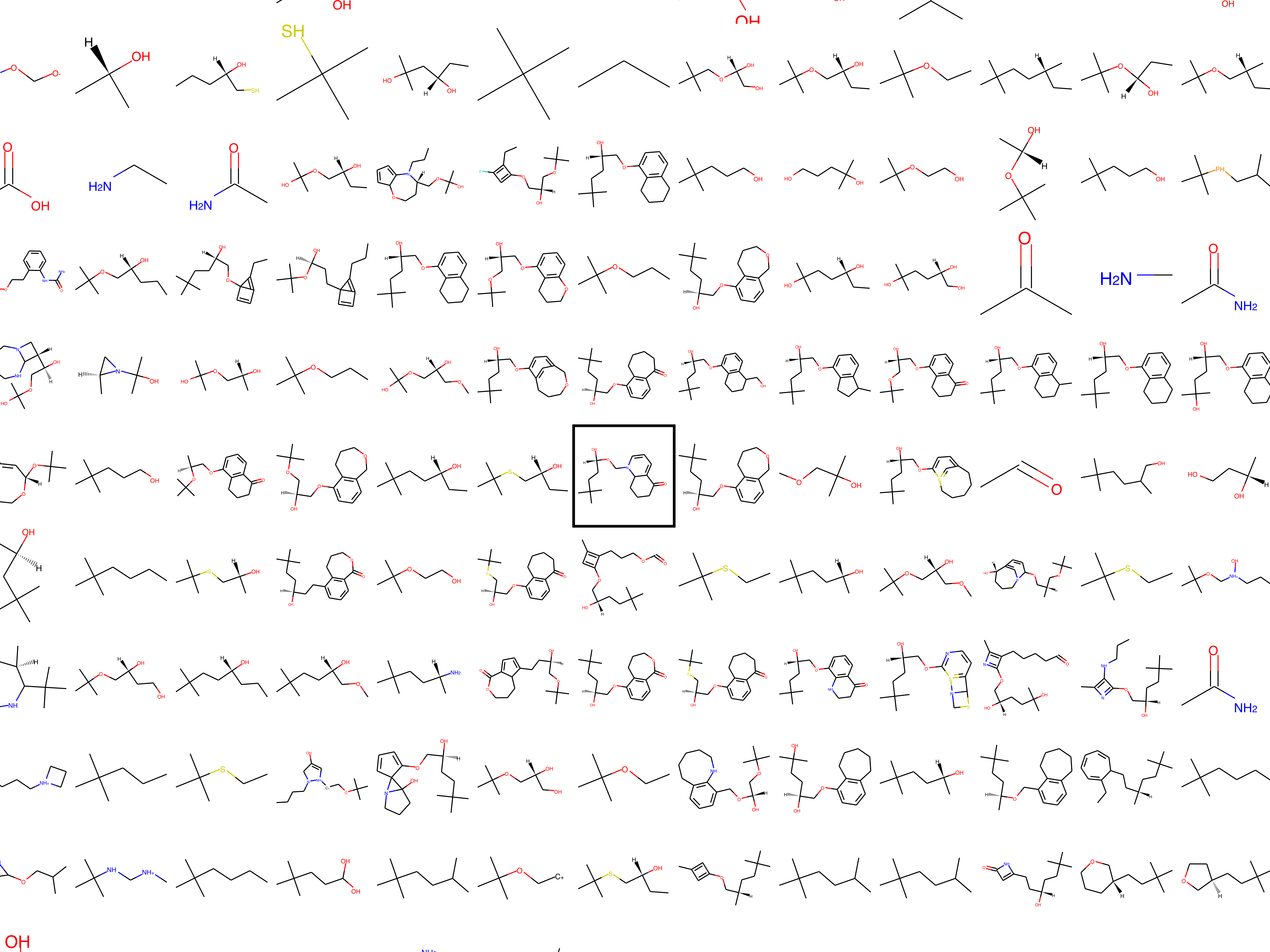


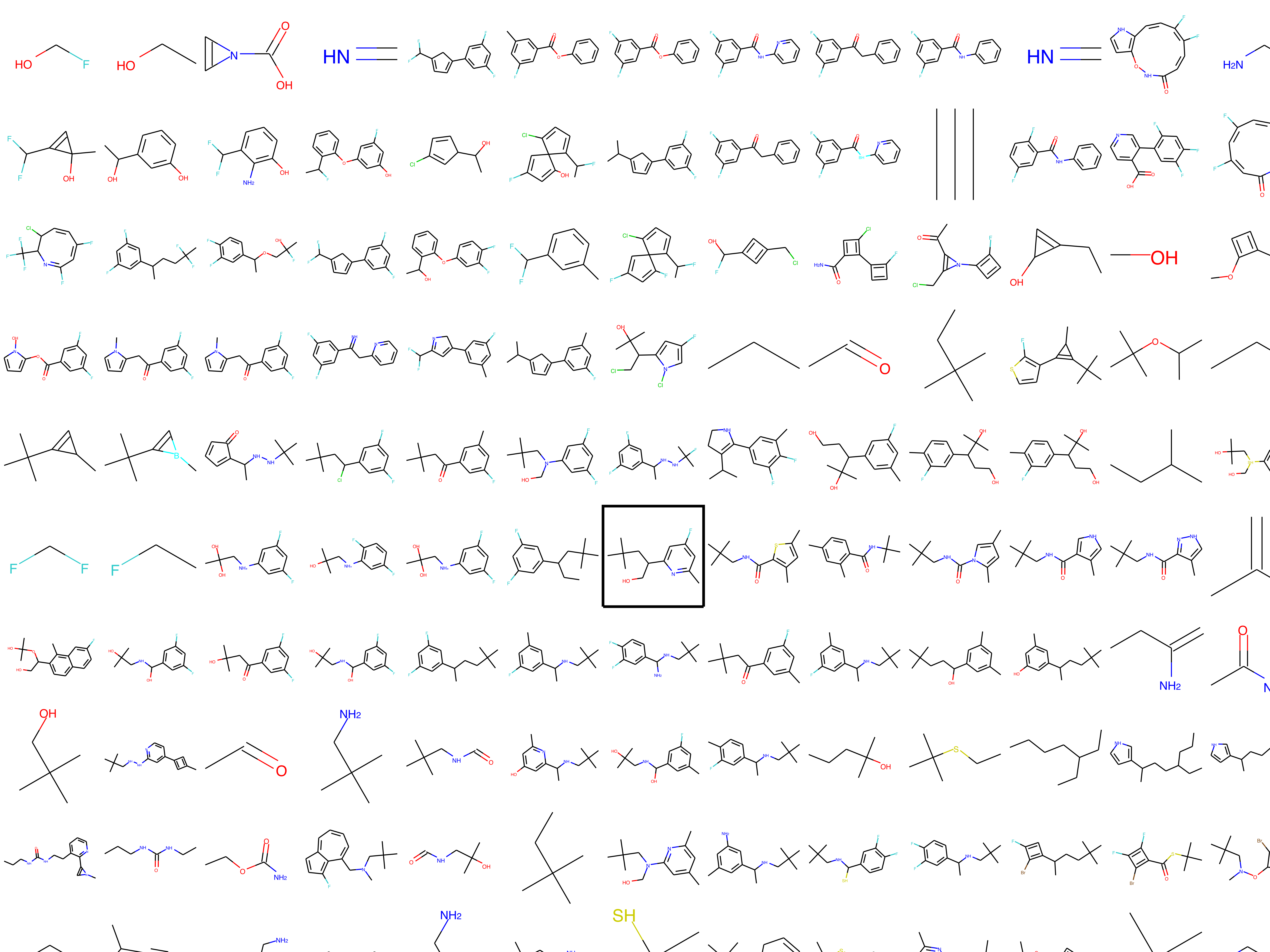
Molecules near 



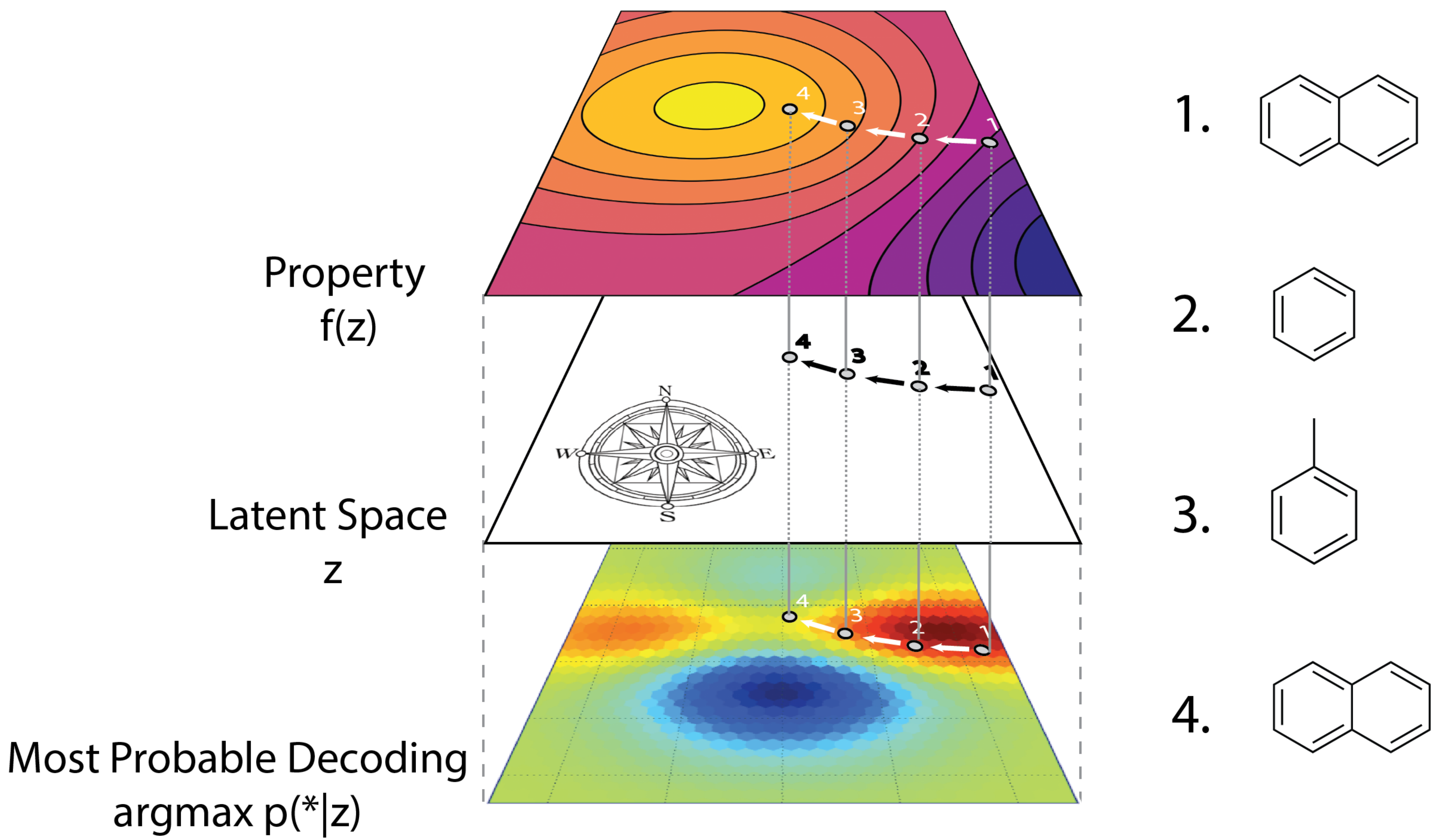




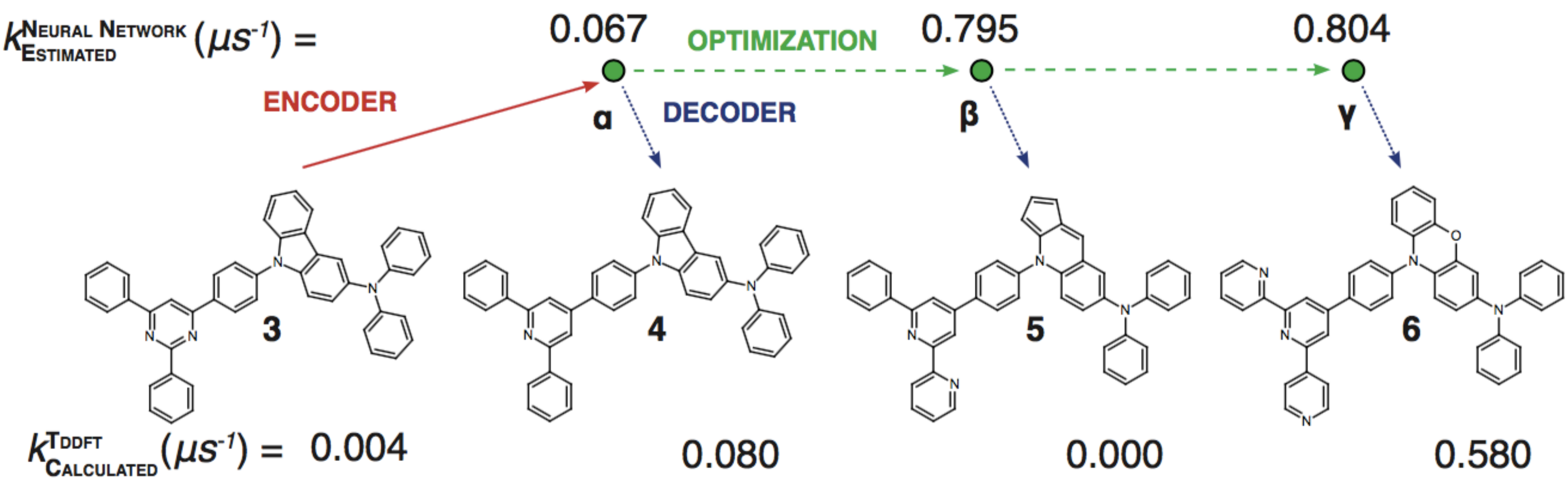




Gradient-based optimization



Gradient-based optimization



- Can’t necessarily start from given molecule, need to encode/decode
- Can’t go too far from start, wander into ‘holes’ or empty regions

Fixing up the encoder with a few steps of SVI

- Semi-Amortized Variational Autoencoders. Yoon Kim, Sam Wiseman, Andrew C. Miller, David Sontag, Alexander M. Rush. 2018

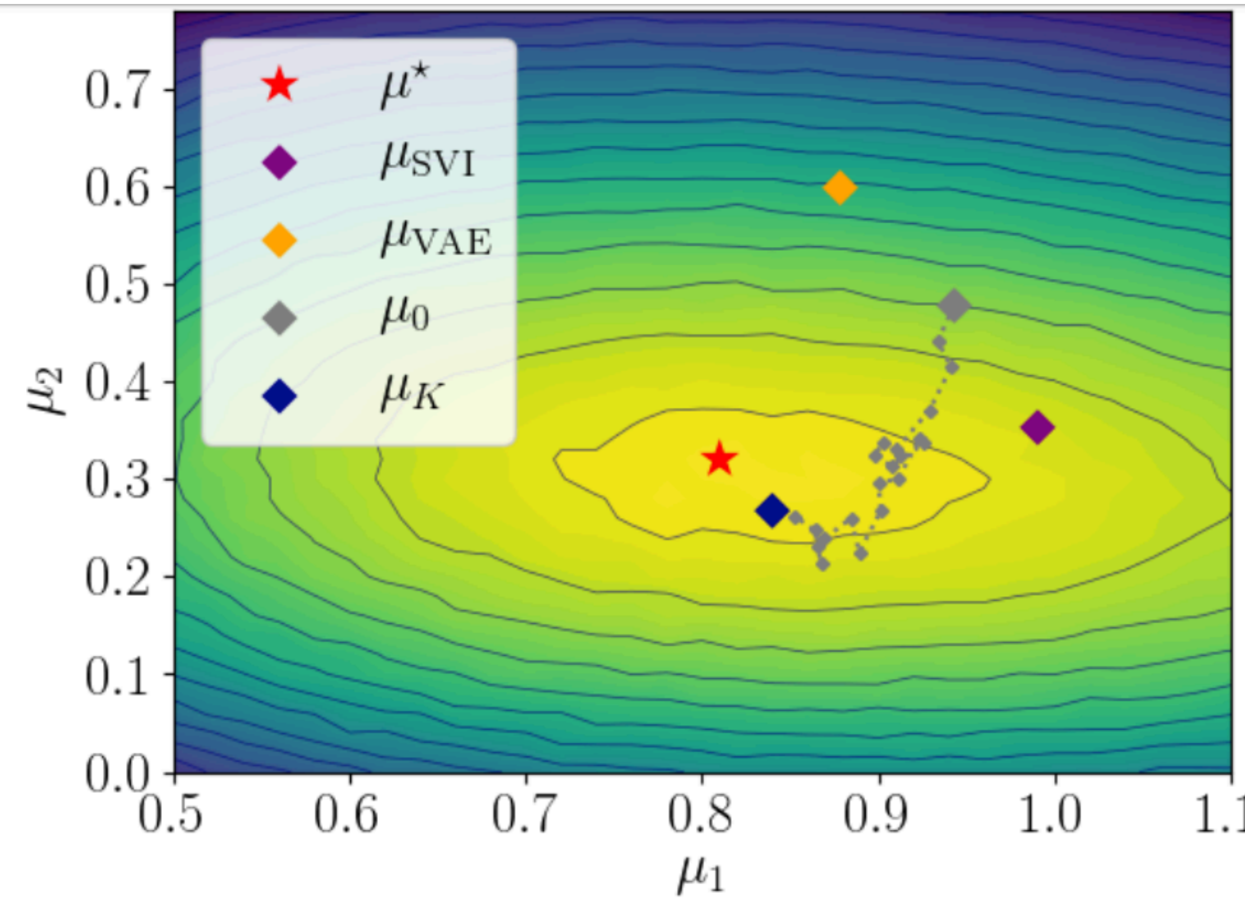
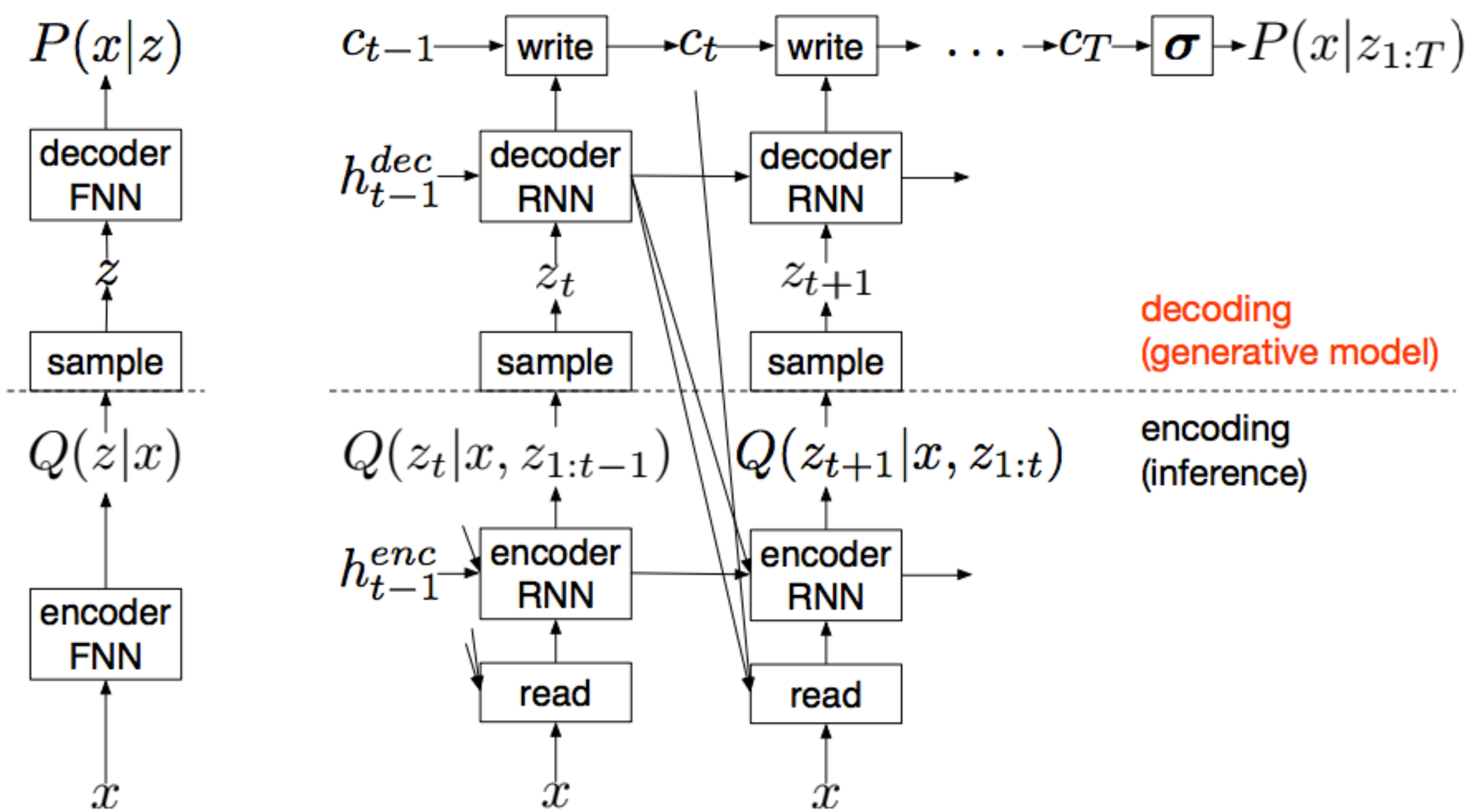


Figure 1. ELBO landscape with the oracle generative model as a function of the variational posterior means μ_1, μ_2 for a randomly chosen test point. Variational parameters obtained from VAE, SVI are shown as $\mu_{\text{VAE}}, \mu_{\text{SVI}}$ and the initial/final parameters from SA-VAE are shown as μ_0 and μ_K (along with the intermediate points). SVI/SA-VAE are run for 20 iterations. The optimal point, found from grid search, is shown as $\mu^\star$.

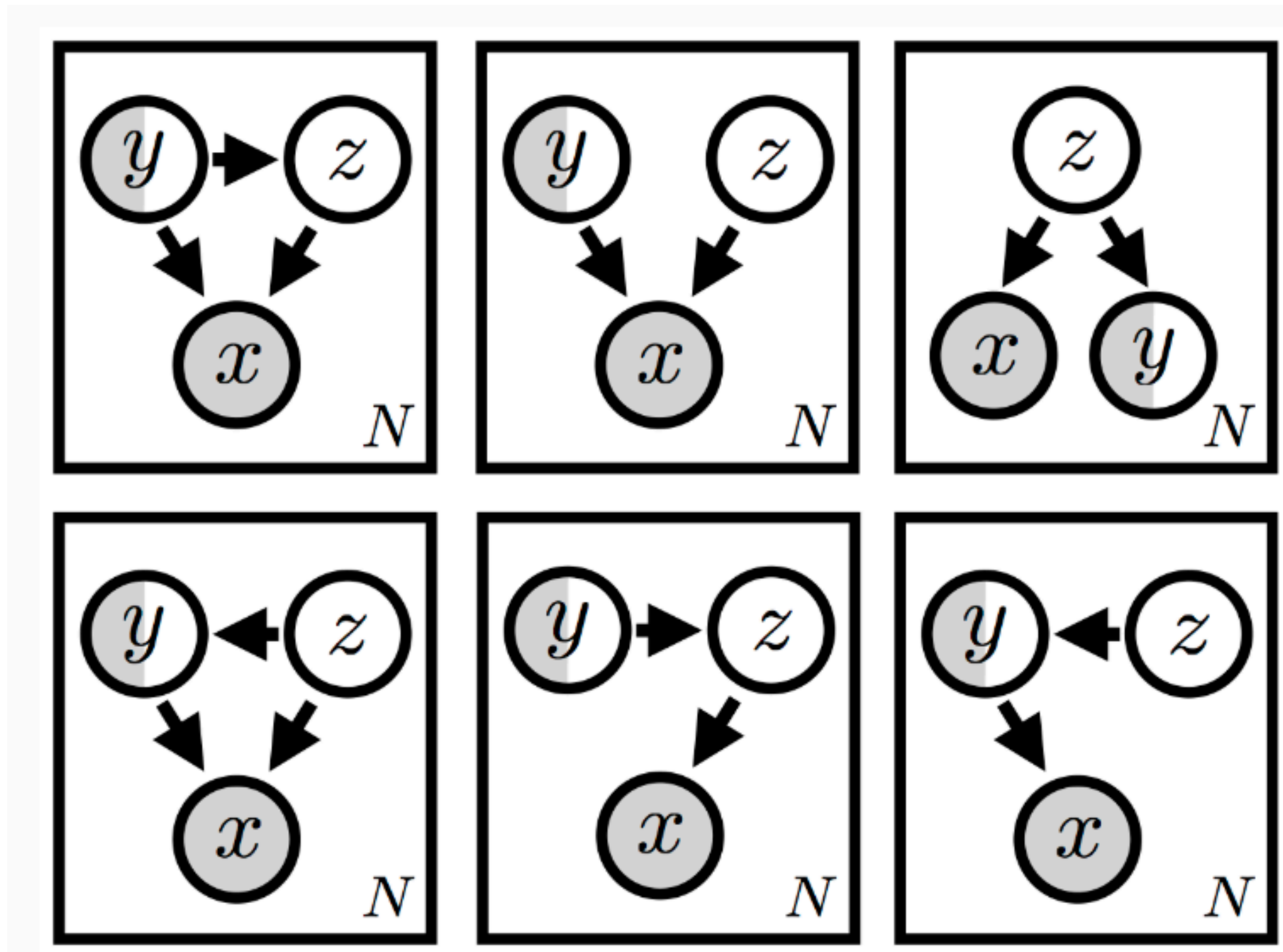
Encoder can look at decoder

- https://www.youtube.com/watch?v=Zt-7MI9eKEo



Semi-supervised learning

- Easy to add discrete labels
- Condition on them when observed, integrate out when unknown



https://pyro.ai/examples/ss-vae.html

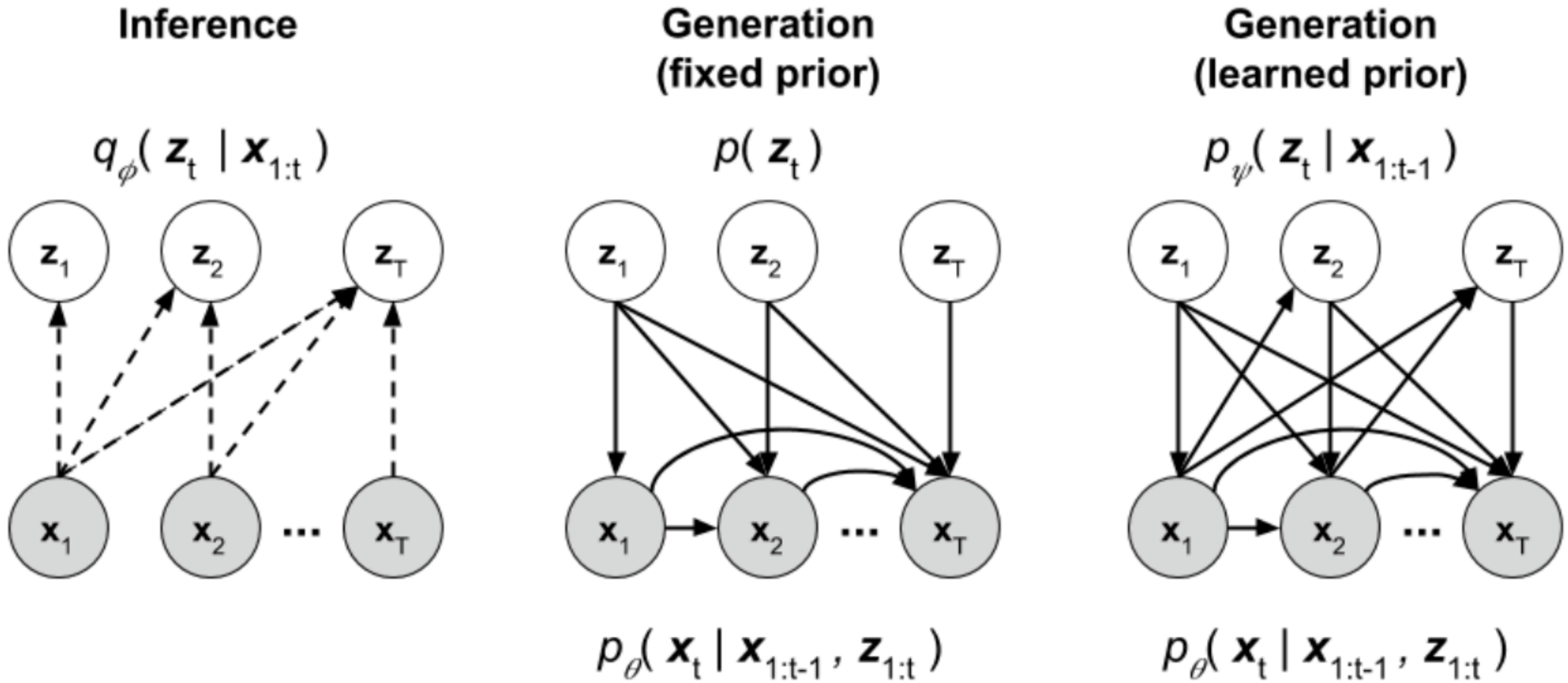
Generative Model vs. Approximate Inference Method

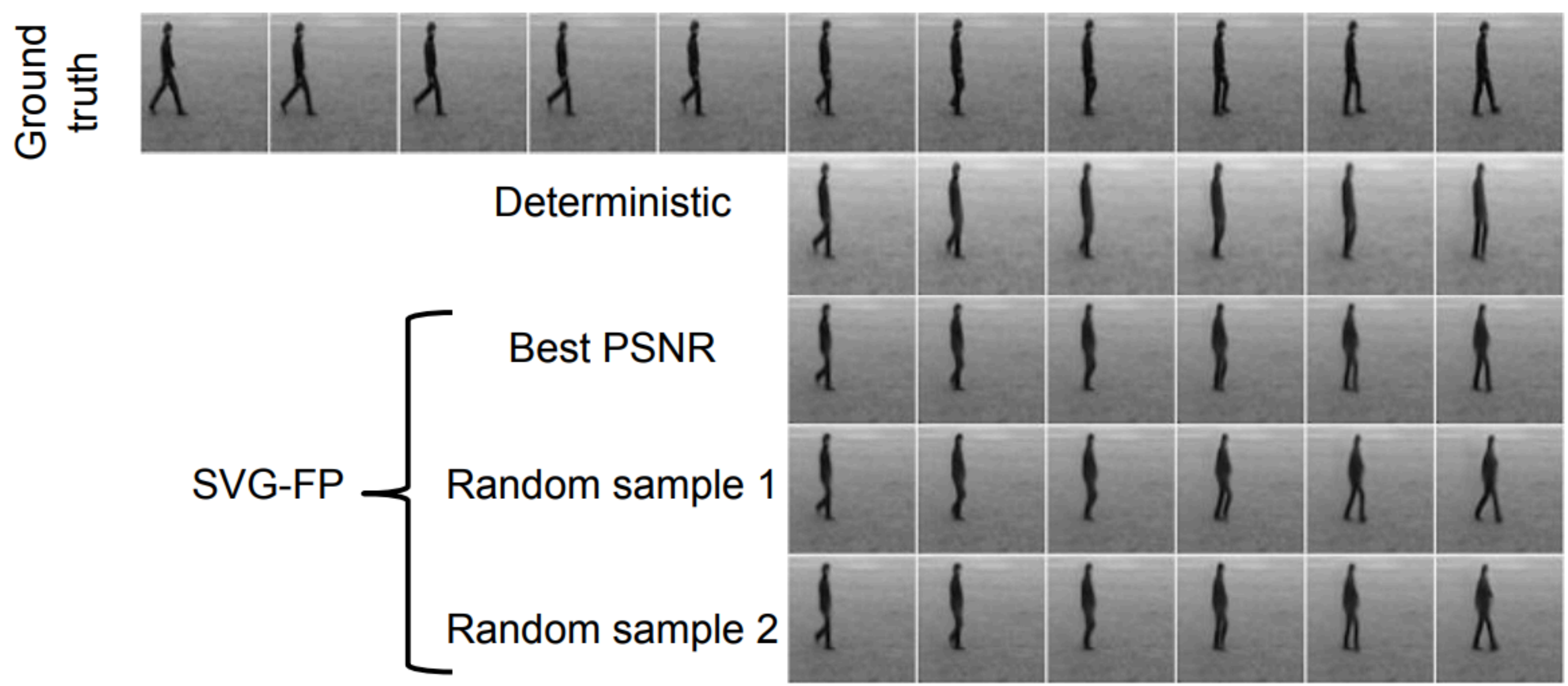
- Can consider generative model (decoder) on its own, and the encoder as just a tool to speed up inference.

Recognition networks

- Know p(symptoms | diseases),
  need p(diseases | symptoms)
- Bayes' rule: $p(d \mid s) = \dfrac{p(s \mid d)p(d)}{\sum_{d'} p(s \mid d')p(d')}$
- Too many possible combinations to compute exactly
- Train a net to approximate p(disease | symptoms): aka a "recognition net"

Denton & Fergus, 2018





Learning outcomes

- Amortized inference
- How to train a VAE
- Separation between model (decoder) and approximate inference strategy (encoder)

Generative Model Families

- Variational Autoencoders
$$x \sim p_\theta(x \mid z), \quad p(x) = \int p(x \mid z)p(z)dz$$

- Autoregressive Models: LSTMs, NICE, PixelRNN
$$x_i \sim p_\theta(x_i \mid x_{<i}), \quad p(x) = \prod_i p_\theta(x_i \mid x_{<i})$$

- Invertible models: Normalizing flows, Real NVP, FFJORD
$$x = f_\theta(z), \quad p(x) = p(z)\left|\det\left(\nabla_z f_\theta\right)\right|^{-1}$$

- Implicit models (GANs)
$$x = f_\theta(z), \quad p(x) \approxeq D_\phi(x)p_{data}(x)$$