Generative Models

David Duvenaud
Deep Learning Summer School 2018

VECTOR INSTITUTE

ML as a bag of tricks

Fast special cases:

- K-means
- Kernel Density Estimation
- SVMs
- Boosting
- Random Forests

Extensible family:

- Mixture of Gaussians
- Latent variable models
- Gaussian processes
- Deep neural nets
- Bayesian neural nets

Regularization as bag of tricks

Fast special cases:

- Early stopping
- Ensembling
- L2 Regularization
- Gradient noise
- Dropout
- Expectation-Maximization

Extensible family:

- Stochastic variational inference

A language of models

- Hidden Markov Models, Mixture of Gaussians, Logistic Regression, VAEs, Normalizing flows
- These are simply examples from a composable language of probabilistic models.

AI as a bag of tricks

Russel and Norvig's parts of AI:

- Machine learning
- Natural language processing
- Knowledge representation
- Automated reasoning
- Computer vision
- Robotics

Extensible family:

- Deep probabilistic latent-variable models + decision theory

Losses are log-likelihoods

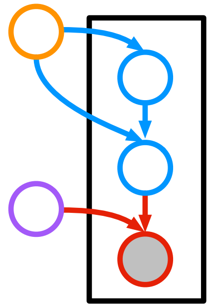
- Squared loss is just unnormalized Normal log-pdf
- “Cross-entropy” now means Categorical log-pmf ?!
  - Actual definition: $H(p, q) = -\sum_x p(x)\, \log q(x)$
- “Teacher forcing” is just evaluating the likelihood of a sequential model $p(x) = \prod_i p_\theta(x_i \mid x_{<i})$

What are Generative Models?

- Discriminative: Trained to answer a single query, p(class | image)
- Generative: Trained to model data distribution too: p(class, image) or simply p(image)
- Any distribution can be conditioned and sampled from (with some work).
- Can do ancestral sampling if $p(x, z) = p(z)p(x|z)$

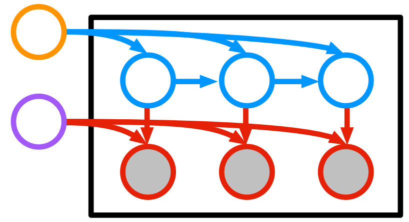
Why should you care?

- Modeling the joint distribution lets us answer any query about the domain: p(class | image), p(image | class), p(bottom of image | top of image)
    - Conditional probability is an extension of logic that tells us how to combine evidence automatically
- Generative models are composable. Useful for modeling and semi-supervised learning.
- Samples let us check the models
- Latent variables sometimes interpretable



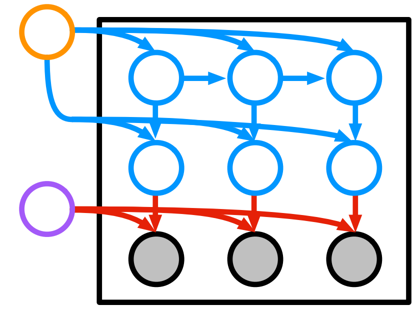
[1]

Gaussian mixture model



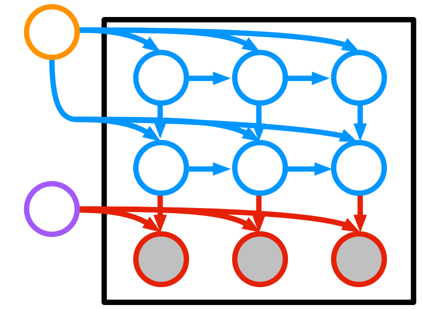
[2]

Linear dynamical system



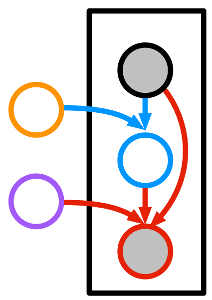
[3]

Hidden Markov model



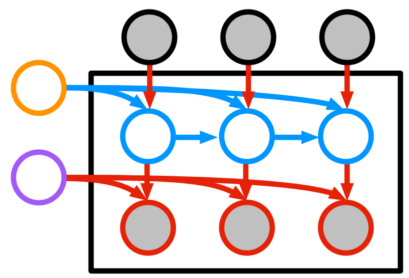
[4]

Switching LDS



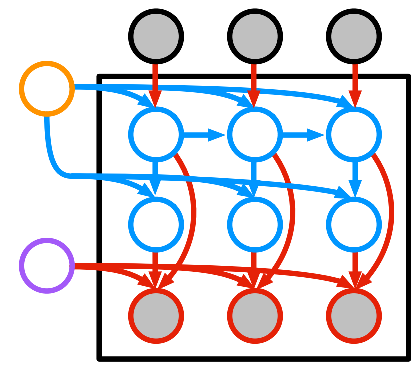
[5]

Mixture of Experts



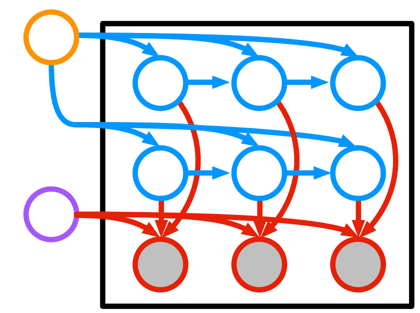
[2]

Driven LDS



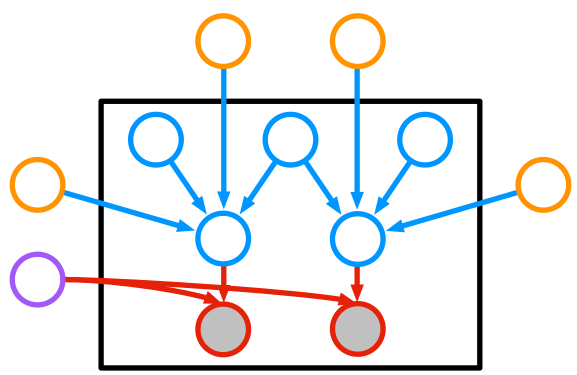
[6]

IO-HMM



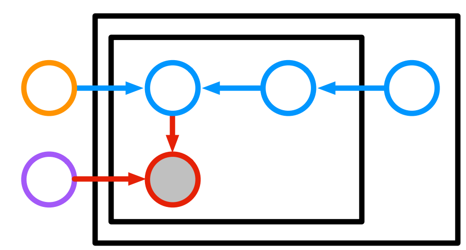
[7]

Factorial HMM



[8,9]

Canonical correlations analysis



[10]

admixture / LDA / NMF

[1] Palmer, Wipf, Kreutz-Delgado, and Rao. Variational EM algorithms for non-Gaussian latent variable models. NIPS 2005.
[2] Ghahramani and Beal. Propagation algorithms for variational Bayesian learning. NIPS 2001.
[3] Beal. Variational algorithms for approximate Bayesian inference, Ch. 3. U of London Ph.D. Thesis 2003.
[4] Ghahramani and Hinton. Variational learning for switching state-space models. Neural Computation 2000.
[5] Jordan and Jacobs. Hierarchical Mixtures of Experts and the EM algorithm. Neural Computation 1994.
[6] Bengio and Frasconi. An Input Output HMM Architecture. NIPS 1995.
[7] Ghahramani and Jordan. Factorial Hidden Markov Models. Machine Learning 1997.
[8] Bach and Jordan. A probabilistic interpretation of Canonical Correlation Analysis. Tech. Report 2005.
[9] Archambeau and Bach. Sparse probabilistic projections. NIPS 2008.
[10] Hoffman, Bach, Blei. Online learning for Latent Dirichlet Allocation. NIPS 2010.

Courtesy of Matthew Johnson

Differentiable latent-variable models

- Model distributions implicitly by a variable pushed through a deep net:

$$y = f_\theta(x)$$

- Approximate intractable distribution by a tractable distribution parameterized by a deep net:

$$p(y|x) = \mathcal{N}(y|\mu = f_\theta(x), \Sigma = g_\theta(x))$$

- Optimize all parameters using stochastic gradient descent

4 Main Approaches

- Sequential Models

$$p(x) = \prod_i p_\theta(x_i \mid x_{<i})$$

- Variational Autoencoders

$$x = f_\theta(z) + \epsilon$$

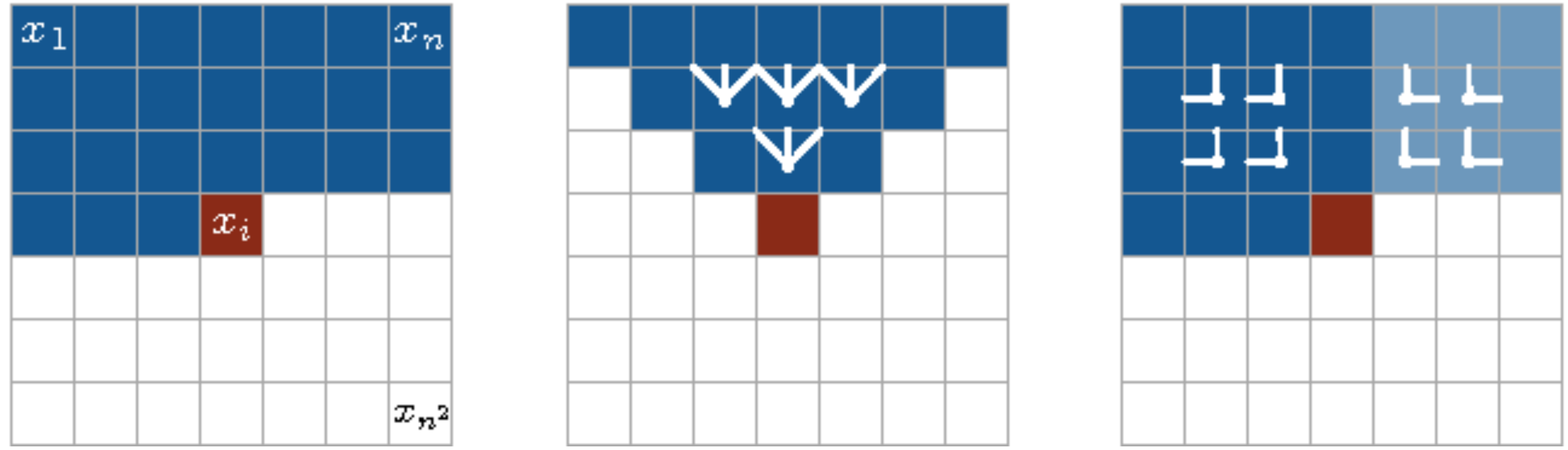
- Normalized models

$$x = f_\theta(z), \quad p(x) = p(z) \left| \det\left(\nabla f\right) \right|^{-1}$$

- Implicit models (GANs)

$$x = f_\theta(z)$$

$$p(x) = \prod_i p_\theta(x_i \mid x_{<i})$$



Pixel Recurrent Neural Networks

Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu

Variational Inference

- Need to compute $p_\theta(z|x) = \dfrac{p_\theta(x|z)p(z)}{\int p_\theta(x|z')p(z')dz'}$
- Optimize a distribution $q_\phi(z|x)$ to match $p_\theta(z|x)$
- What if there is a latent variable z per-datapoint, and global parameters?
- Optimize each $q_\phi(z_i|x_i)$ to match each $p_\theta(z_i|x_i)$, then update theta. Slow!


ADDING PRIORS ISN'T COOL
INTEGRATING OVER AN ENTIRE
HYPOTHESIS SPACE IS COOL

Variational Autoencoder

- Train a recognition network to output approximately optimal variational distributions $q_\phi(z_i | x_i)$ given x_i
- Total freedom in designing recognition procedure
- Can be evaluated by how well it matches $p_\theta(z_i | x_i)$

Consequences of using a recognition network

- Don't need to re-optimize q(z|x) each time theta changes. Much faster!
- Recognition net won't necessary give optimal phi_i
- Can have fast test-time inference (vision)
- Can train recognition net jointly with generator

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

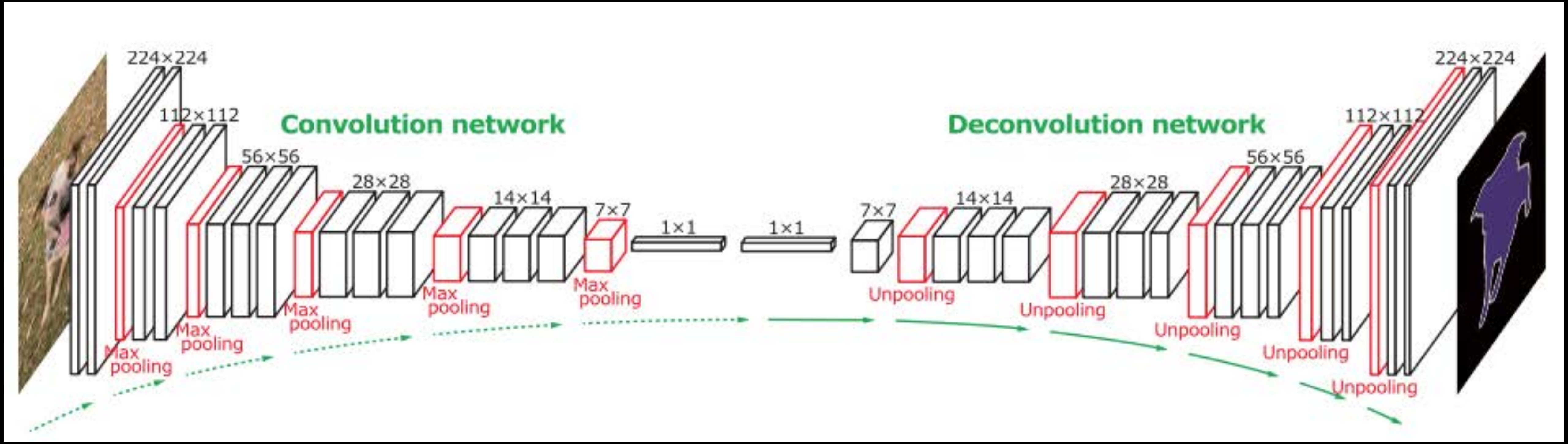
Discovering the structure inherent in a set of patterns is a fundamental aim of statistical inference or learning. One fruitful approach is to build a parameterized stochastic generative model, independent draws from which are likely to produce the patterns. For all but the simplest generative models,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

Variations: Decoder

- Often, $p(x \mid z) = \mathcal{N}(x \mid f_\theta(z), diag(g_\theta(z)))$
- Final step has independence assumption, causes noisy samples, blurry means
- p(x|z) can be anything: RNN, pixelRNN, real NVP, deconv net

Variations

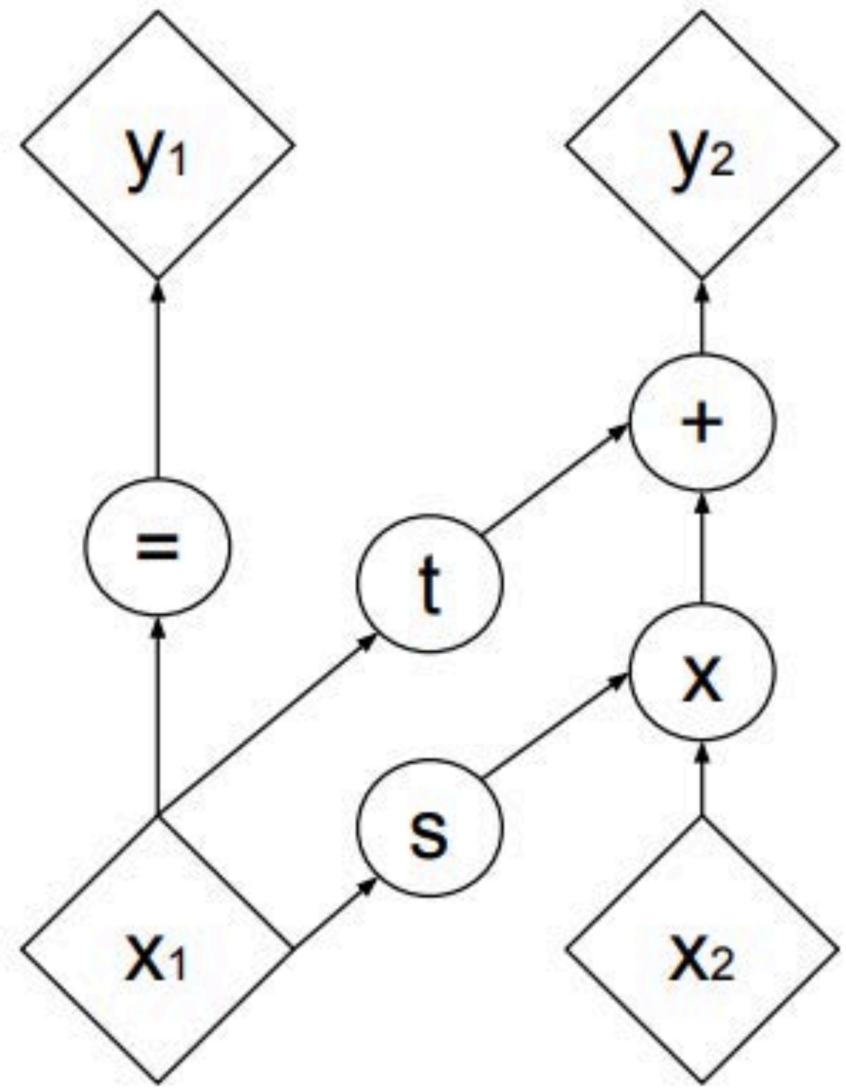
- Decoder often looks like inverse of encoder
- Encoders can come from supervised learning



Learning Deconvolution Network for Semantic Segmentation
http://arxiv.org/abs/1505.04366.

Real-Valued Non-Volume-Preserving Transformations

- aka Real NVP
- divides up variables into two parts, updates only one half with a scale and shift



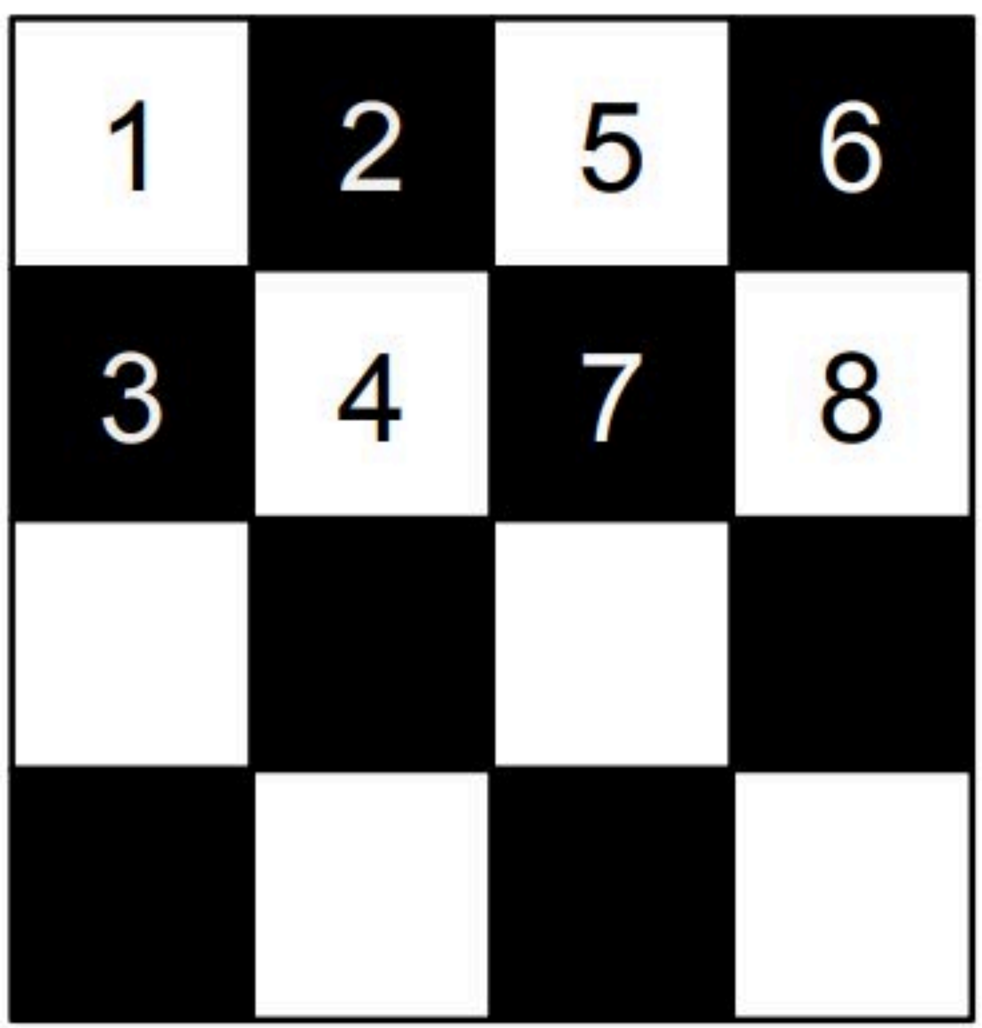
Real-Valued Non-Volume-Preserving Transformations

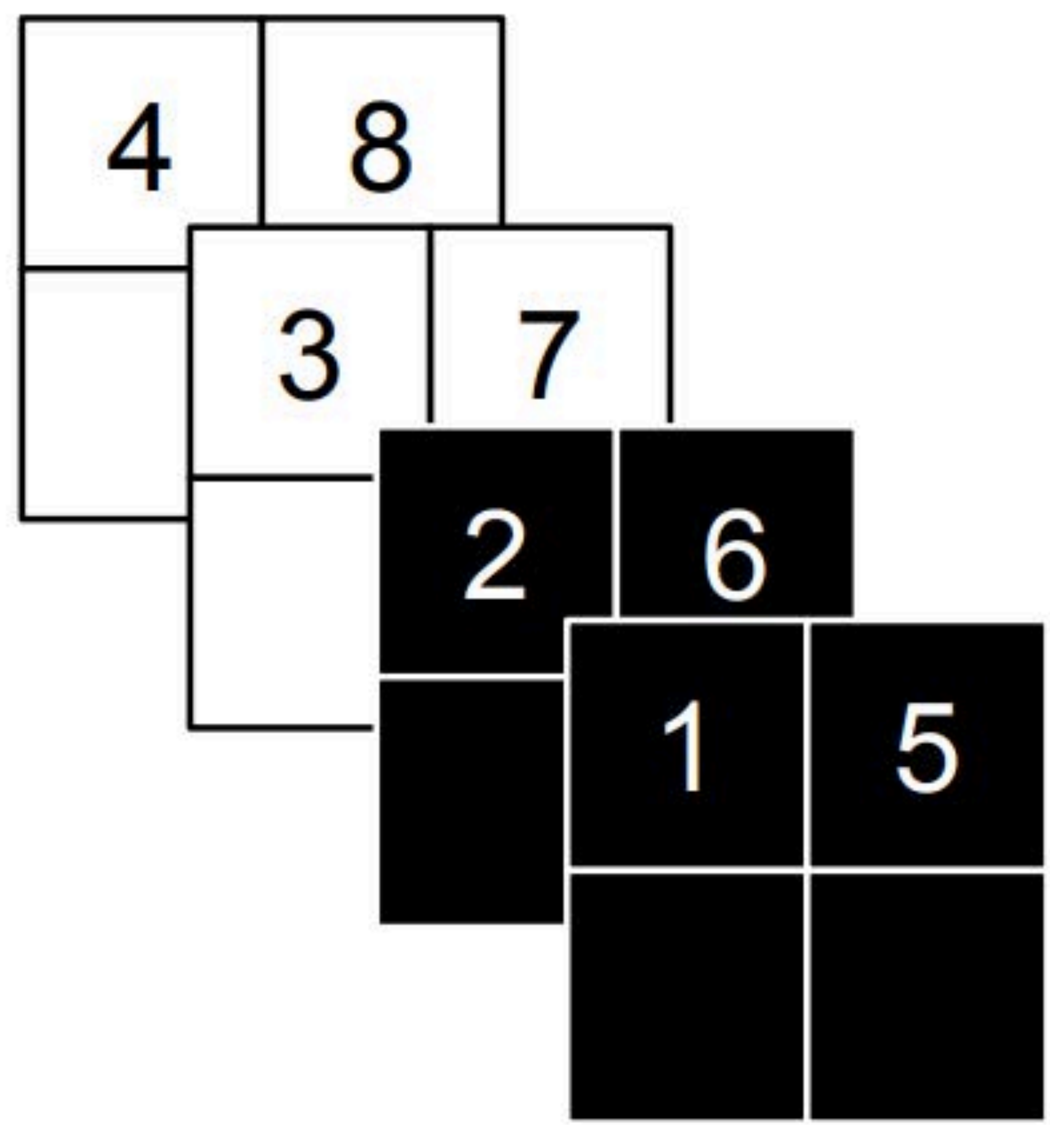
- change of variables formula is tractable due to lower-diagonal Jacobian

$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \mathrm{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix}$$

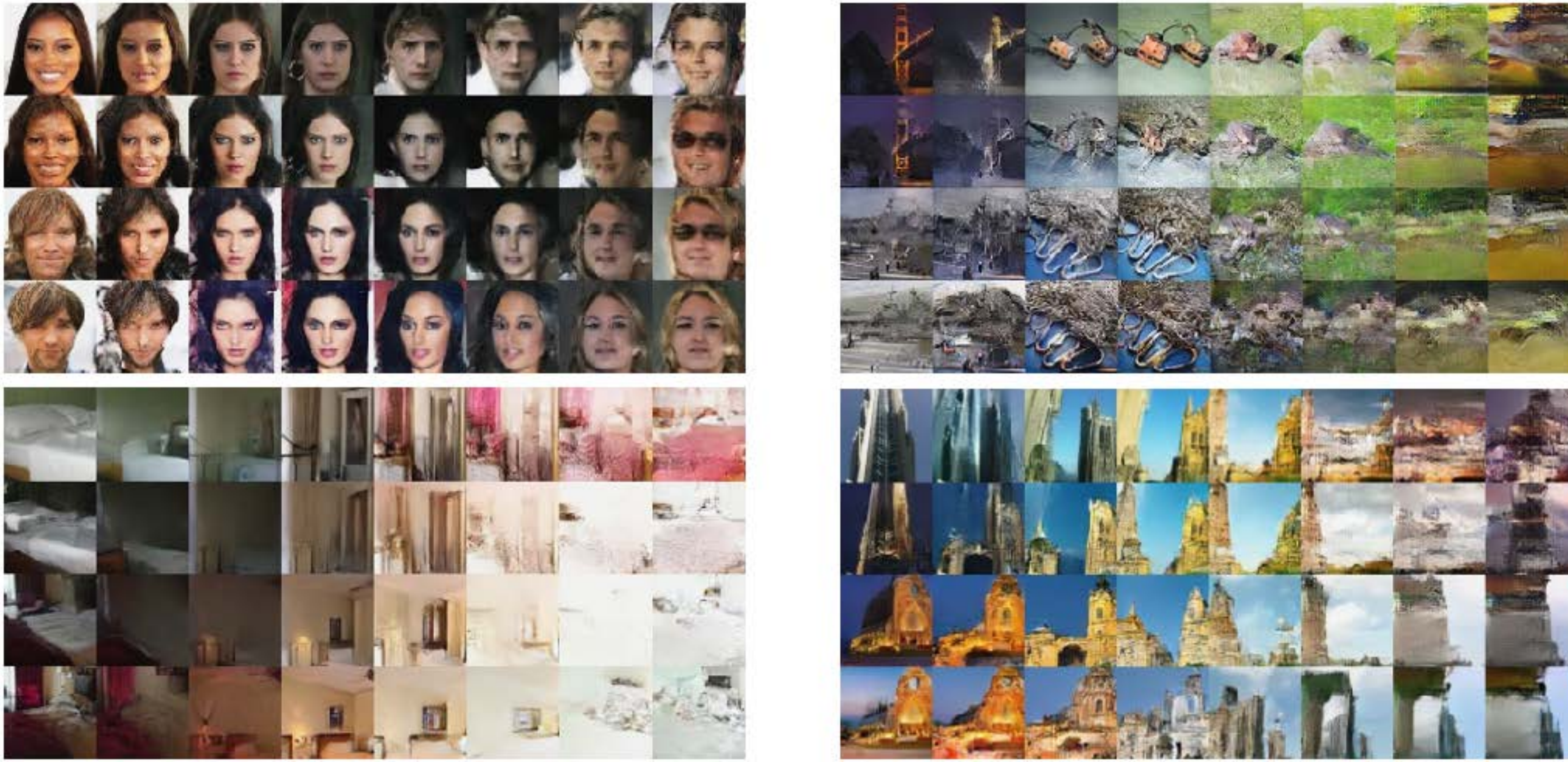
Real-Valued Non-Volume-Preserving Transformations

- Need to interleave many layers with different partitions





Density estimation using Real NVP. Ding et al, 2016



Density estimation using Real NVP. Ding et al, 2016

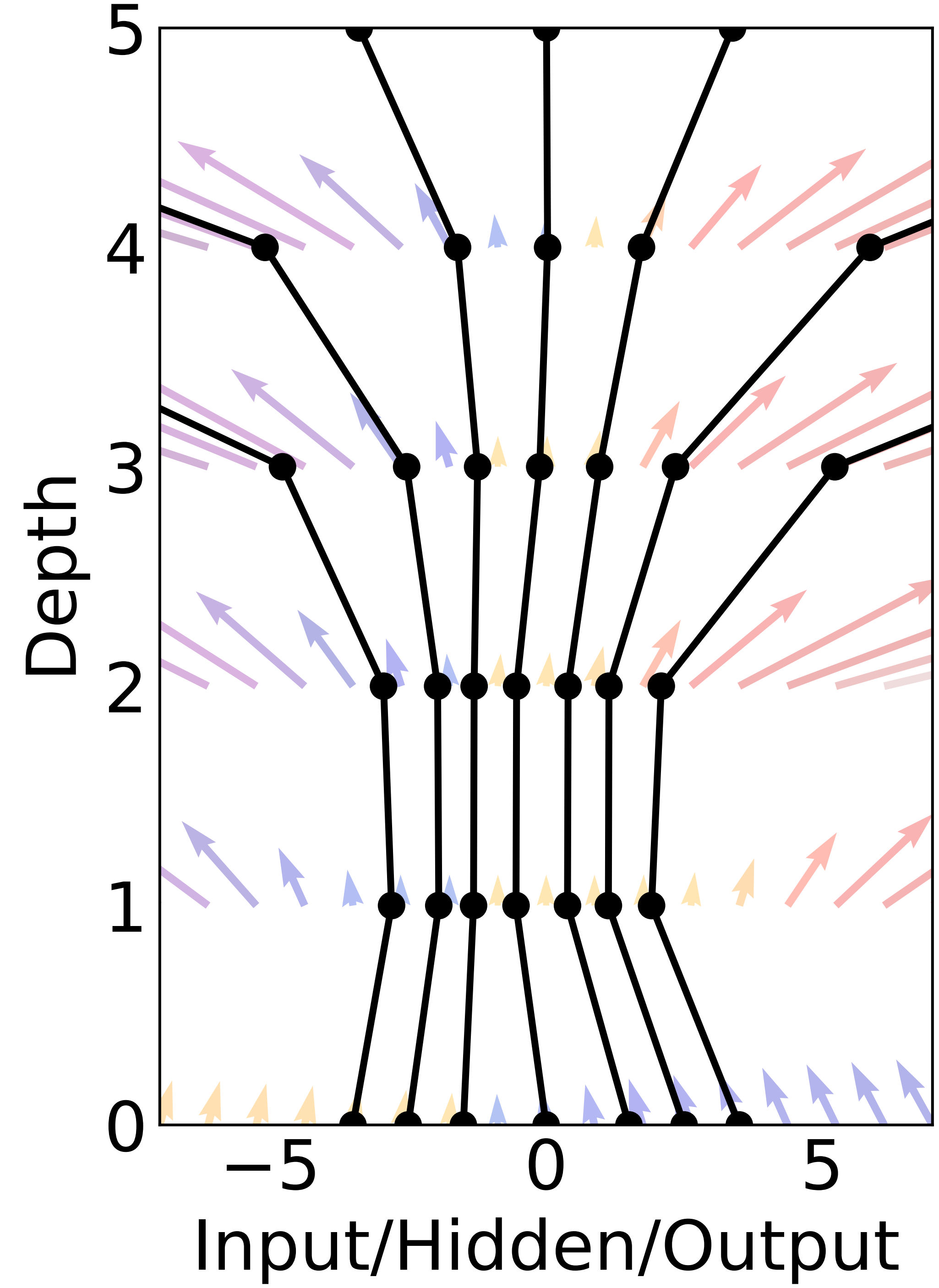
Flows as Euler integrators

- Middle layers look like:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

- Limit of smaller steps:

$$\frac{d\mathbf{h}(\mathbf{t})}{dt} = f(\mathbf{h}(t), \theta(t))$$



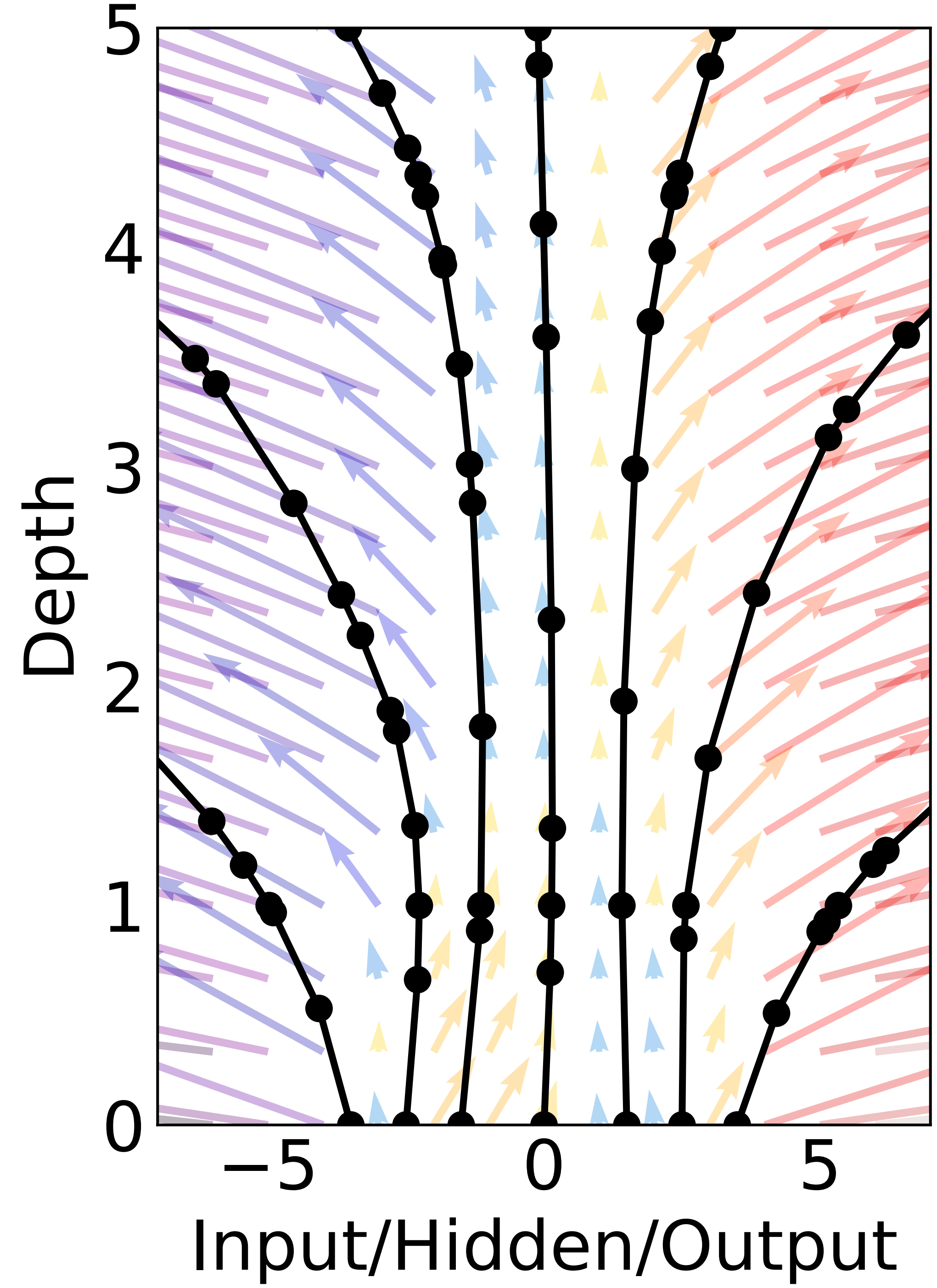
Flows as Euler integrators

- Middle layers look like:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

- Limit of smaller steps:

$$\frac{d\mathbf{h}(\mathbf{t})}{dt} = f(\mathbf{h}(t), \theta(t))$$



Normalizing Flows

$$x_1 = f(x_0) \implies p(x_1) = p(x_0) \left| \det \frac{\partial f}{\partial x_0} \right|^{-1}$$

- Determinant of Jacobian has cost O(D^3).
- Matrix determinant lemma gives O(DH^3) cost.
- Normalizing flows use 1 hidden unit. Deep & skinny

$$x(t+1) = x(t) + uh(w^T x(t) + b)$$

$$\log p(x(t+1)) = \log p(x(t)) - \log \left| 1 + u^T \frac{\partial h}{\partial x} \right|$$

Continuous Normalizing Flows

- What if we move to continuous transformations?

$$\frac{\partial \log p(x(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{dx}(t)\right)$$

- Time-derivative only depends on trace of Jacobian

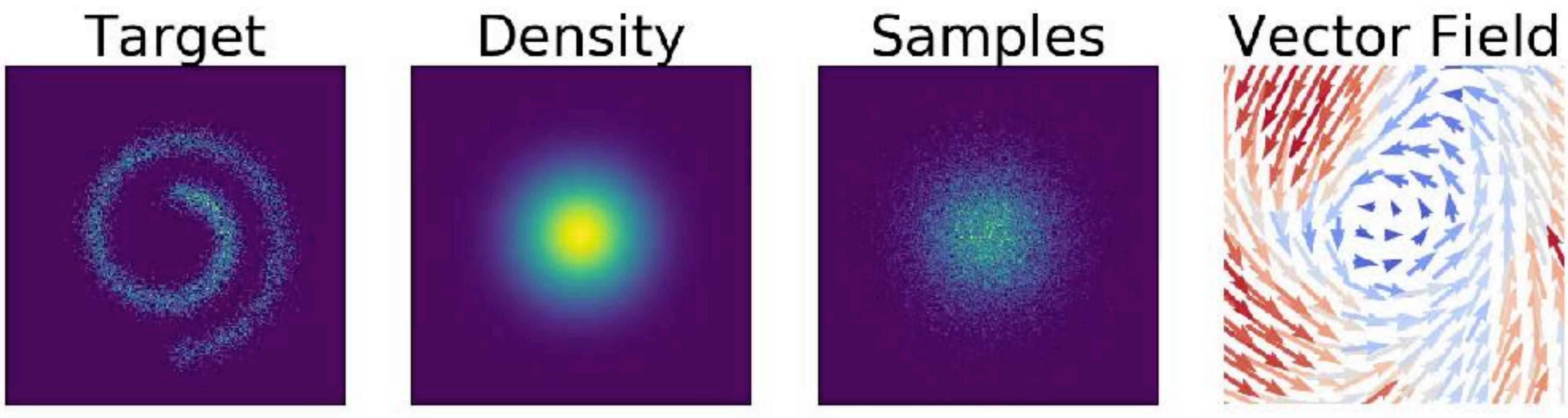
$$\frac{dx}{dt} = uh(w^T x + b), \quad \frac{\partial \log p(x)}{\partial t} = -u^T \frac{\partial h}{\partial x}$$

- Trace of sum is sum of traces - O(HD) cost!

$$\frac{dx}{dt} = \sum_n f_n(x), \quad \frac{d \log p(x(t))}{dt} = \sum_n \mathrm{tr}\left(\frac{\partial f}{\partial x}\right)$$

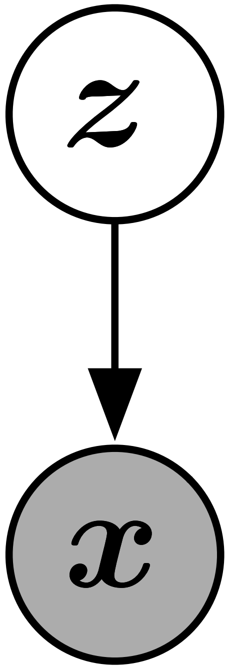
Training directly from data

- Best of all worlds:
  - Wide layers
  - No need to partition dimensions
  - Can evaluate density tractably?



Generator Network

$$\boldsymbol{x} = G(\boldsymbol{z}; \boldsymbol{\theta}^{(G)})$$



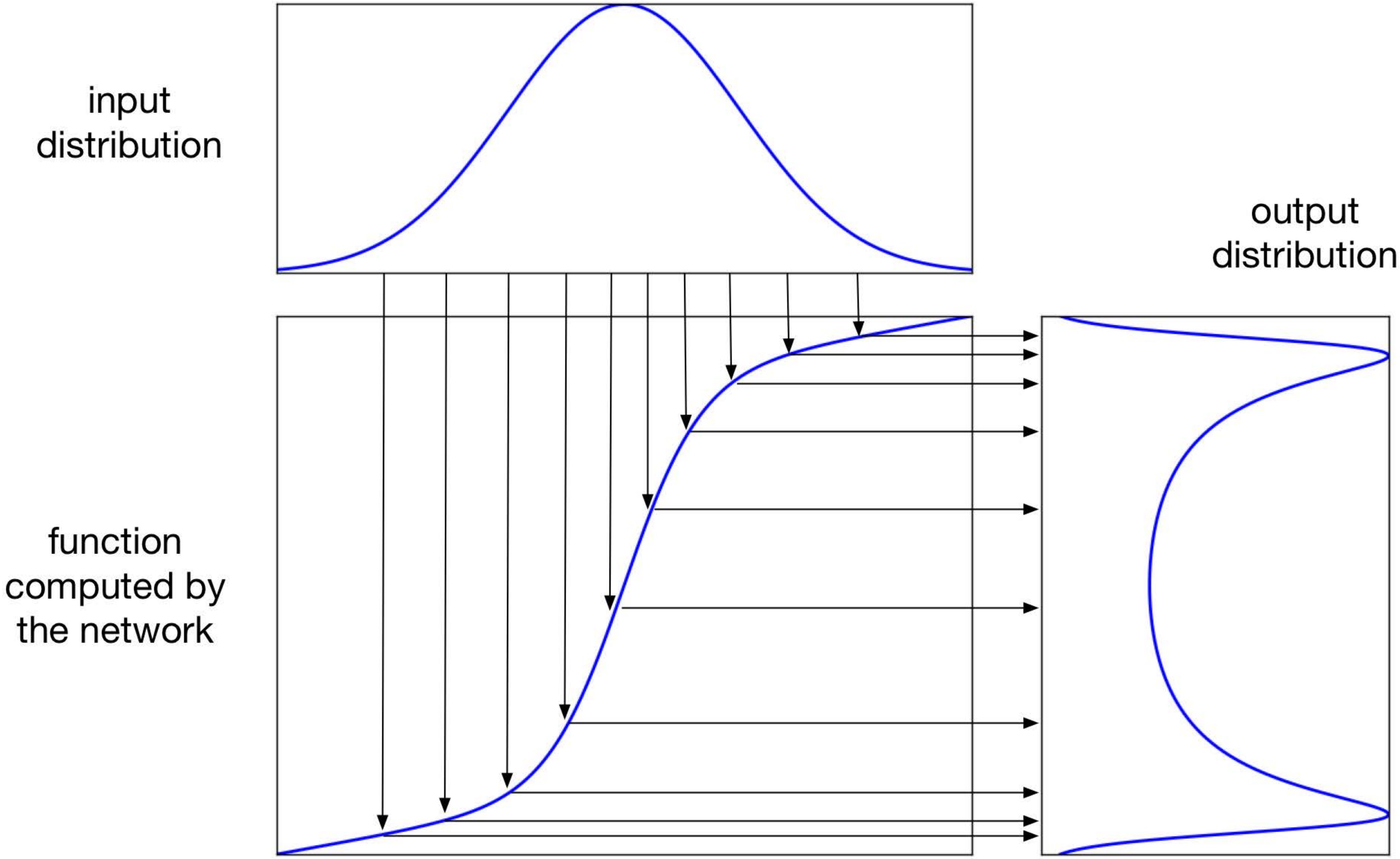
-Must be differentiable

- No invertibility requirement
- Trainable for any size of z
- Some guarantees require z to have higher dimension than x
- Can make x conditionally Gaussian given z but need not do so

(Goodfellow 2016)

Generative Adversarial Networks

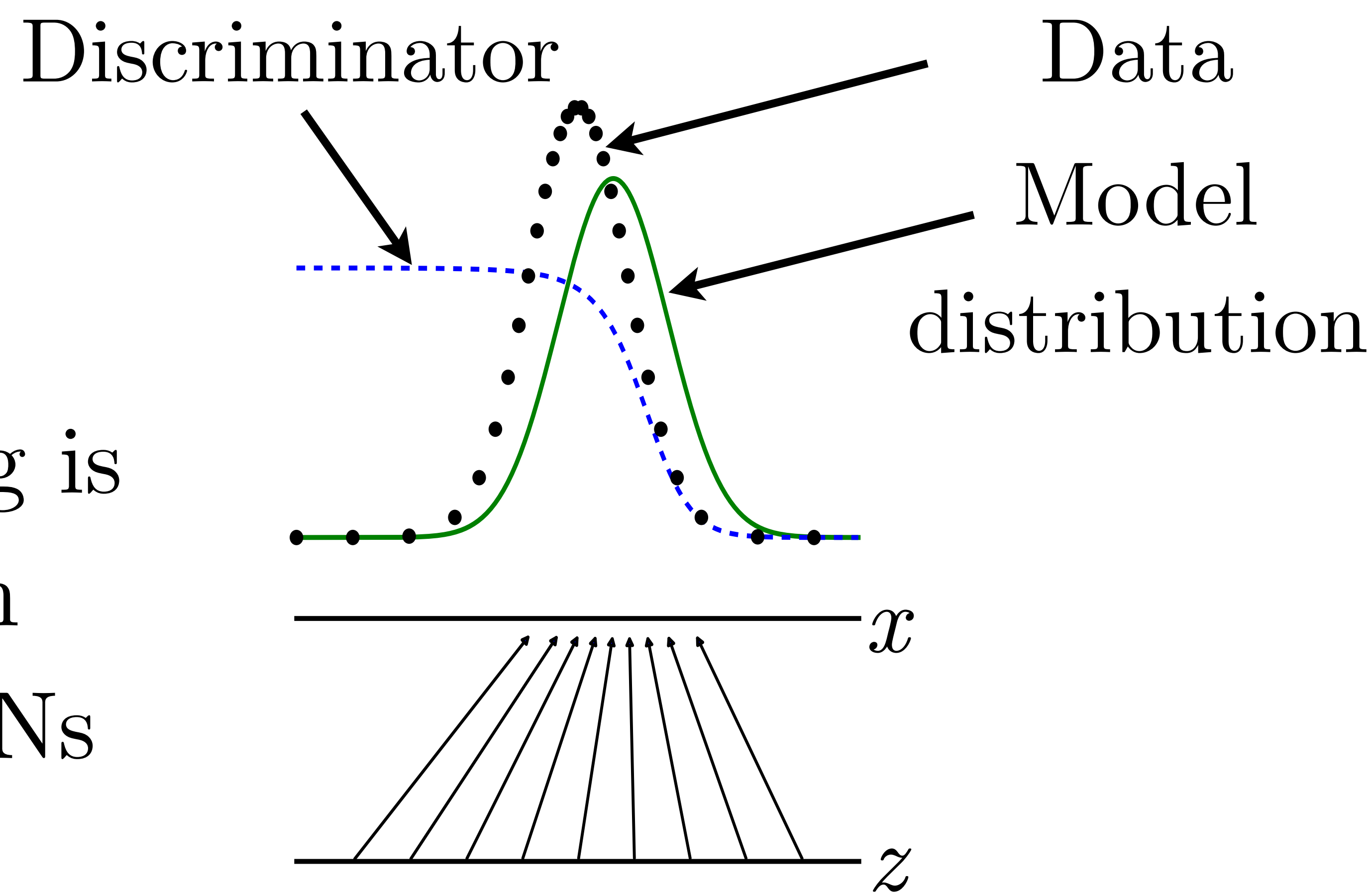
A 1-dimensional example:



Discriminator Strategy

Optimal $D(\boldsymbol{x})$ for any $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{model}}(\boldsymbol{x})$ is always

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$



Estimating this ratio using supervised learning is the key approximation mechanism used by GANs

Minimax Game

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(\boldsymbol{z})\right)\right)$$

$$J^{(G)} = -J^{(D)}$$

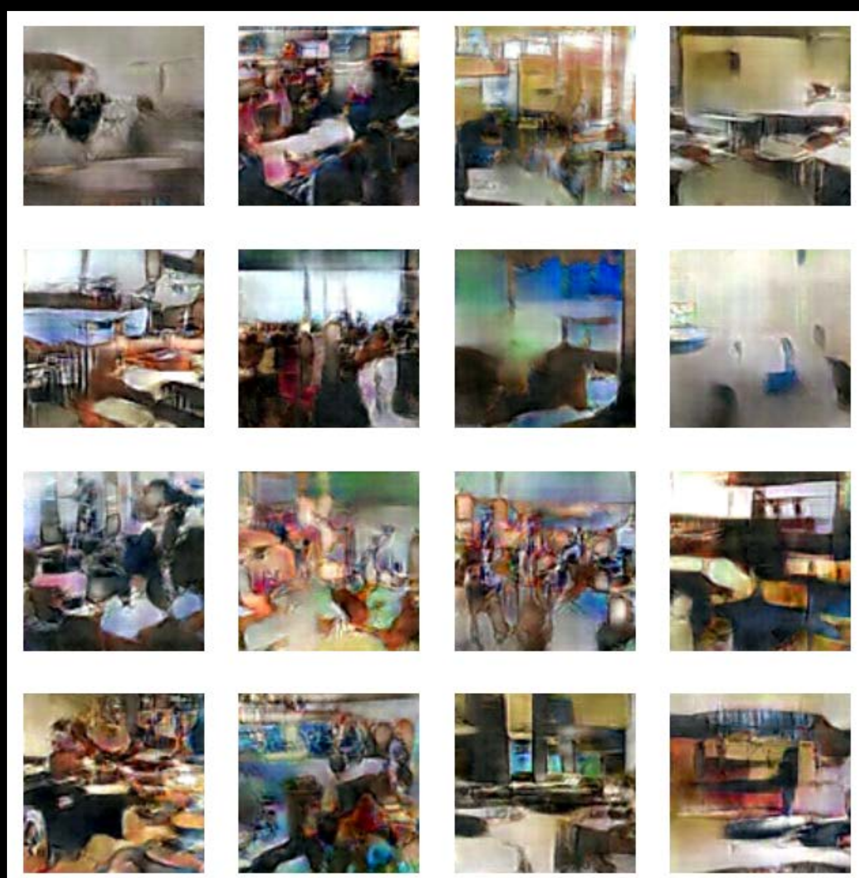
-Equilibrium is a saddle point of the discriminator loss

-Resembles Jensen-Shannon divergence

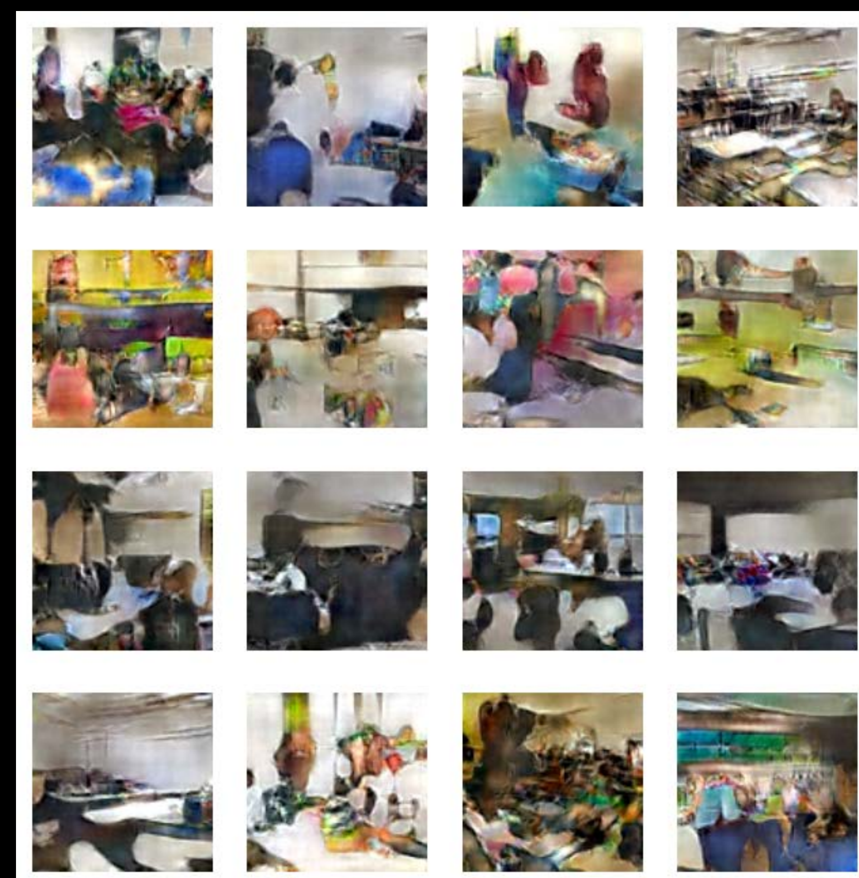
-Generator minimizes the log-probability of the discriminator being correct

(Goodfellow 2016)

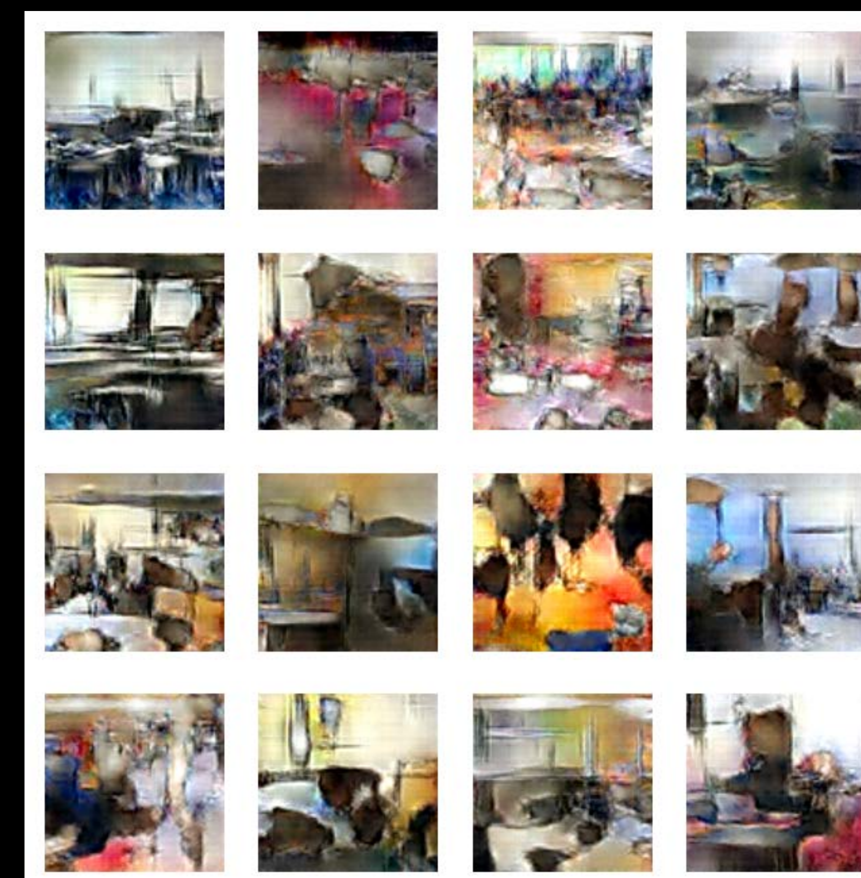
Can train GANs with any divergence



GAN (Jensen-Shannon)



Hellinger



Kullback-Leibler

Slide from Sebastian Nowozin

Relation to VAEs

- Same graphical model: z -> x
- VAEs have an explicit likelihood: p(x|z)
- GANs have no explicit likelihood
  - aka implicit models, likelihood-free models
- Can use same trick for implicit q(z|x). [Lars et al., 2017, Mohamed & Lakshminarayanan, 2016, Huszar, 2017, Tran, Ranganath, & Blei, 2017]

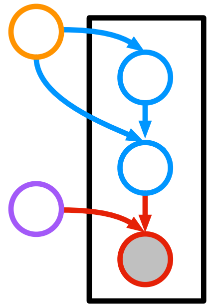
- Sequential Models: $$p(x) = \prod_i p_\theta(x_i \mid x_{<i})$$
  - Pros: Exact likelihoods, easy to train
  - Cons: O(N) layers to evaluate or sample, need to choose order
- Variational Autoencoders: $$x = f_\theta(z) + \epsilon$$
  - Pros: Cheap to evaluate and sample, low-D latents
  - Cons: Factorized likelihood gives noisy samples
- Explicitly normalized models: $$x = f_\theta(z), \quad p(x) = p(z) \left| \det \left(\nabla f \right) \right|^{-1}$$
  - Pros: Exact likelihoods, easy to train
  - Cons: Must cripple layers to maintain tractability, need huge models
- Implicit models: $$x = f_\theta(z)$$
  - Pros: Cheap to sample, no factorization
  - Cons: Hard to train, likelihood not available

Boltzmann Machines

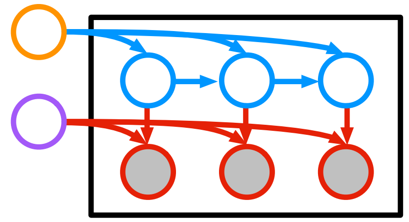
$$p(\boldsymbol{x}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{x}, \boldsymbol{z})\right)$$

$$Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{z}} \exp\left(-E(\boldsymbol{x}, \boldsymbol{z})\right)$$

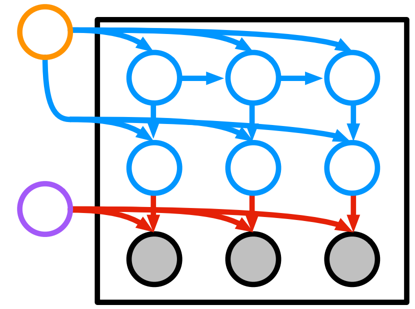
- Partition function is intractable
- May be estimated with Markov chain methods
- Generating samples requires Markov chains too


[1]

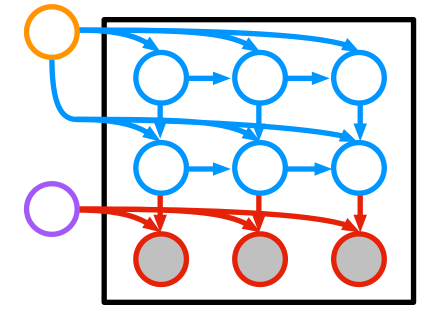
Gaussian mixture model


[2]

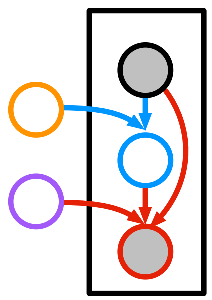
Linear dynamical system


[3]

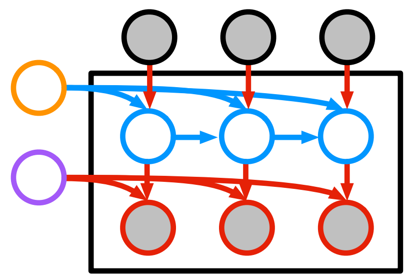
Hidden Markov model


[4]

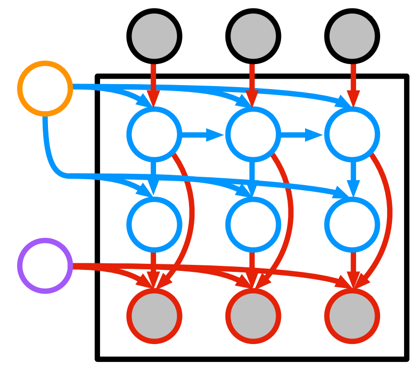
Switching LDS


[5]

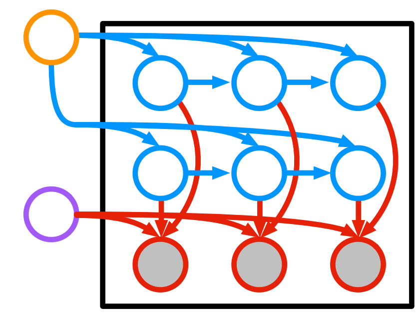
Mixture of Experts


[2]

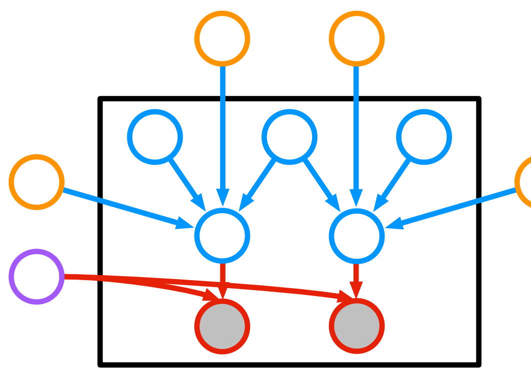
Driven LDS


[6]

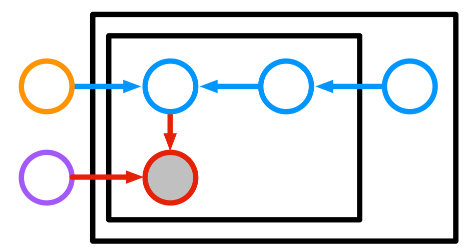
IO-HMM


[7]

Factorial HMM


[8,9]

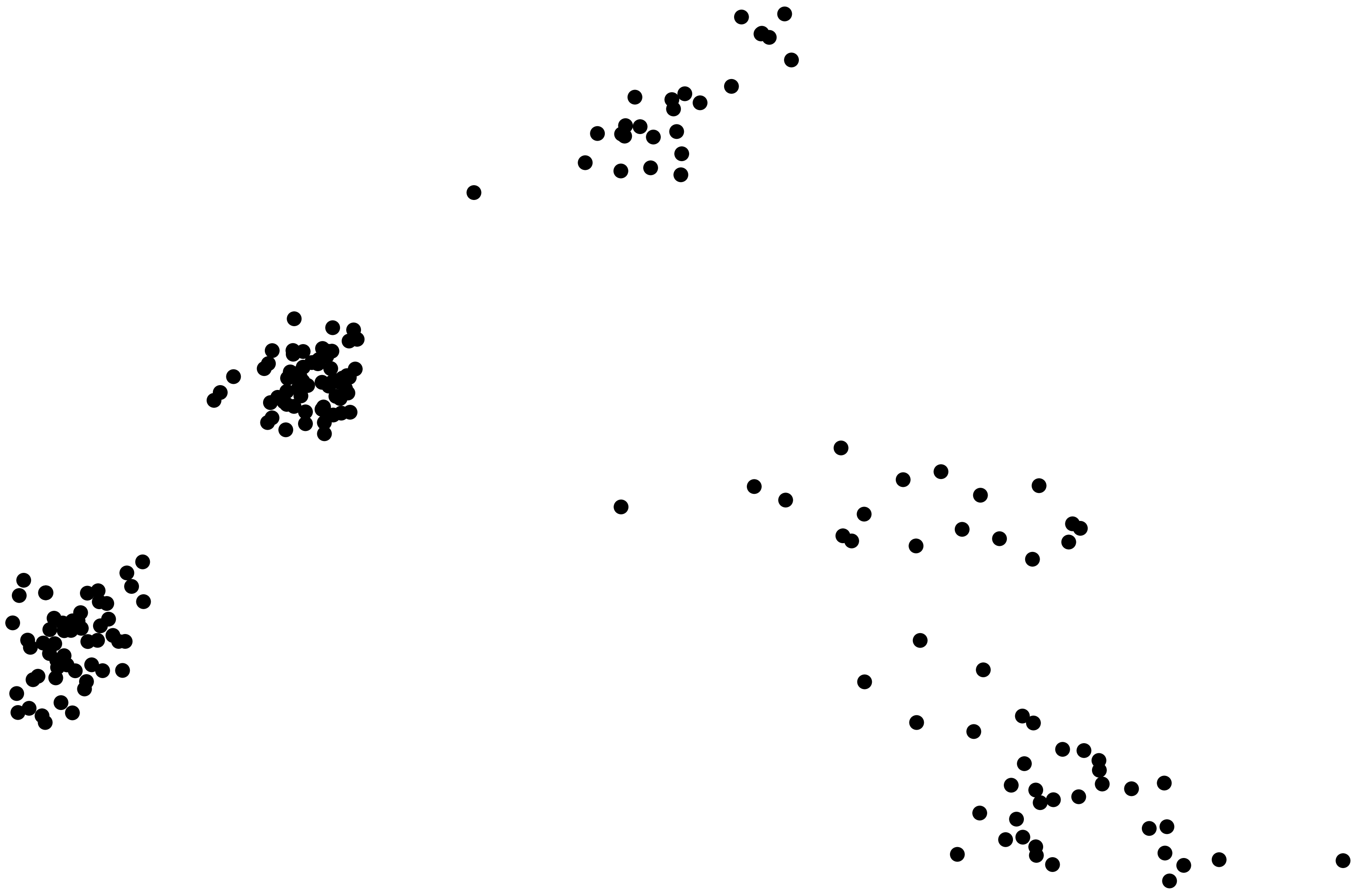
Canonical correlations analysis

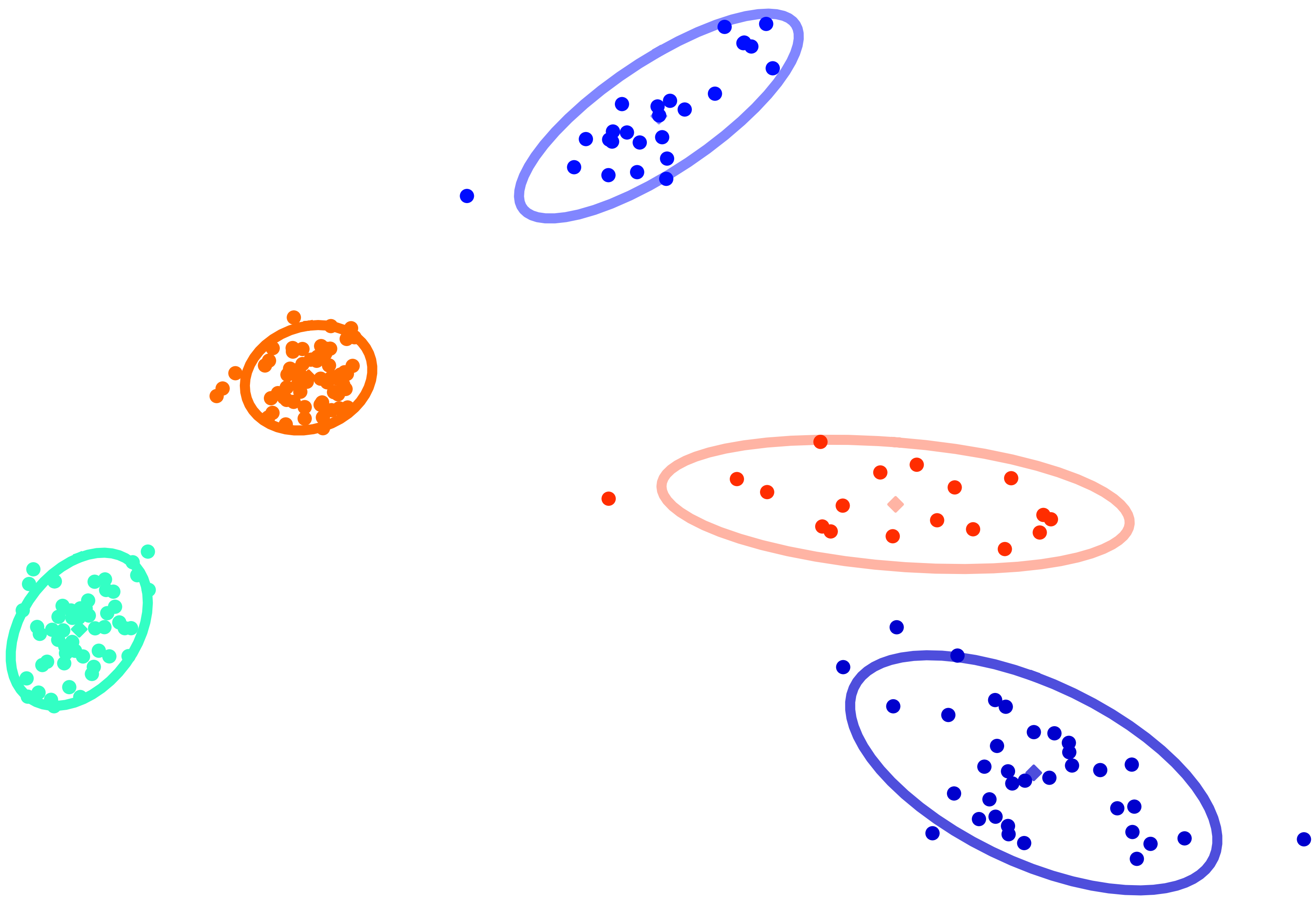

[10]

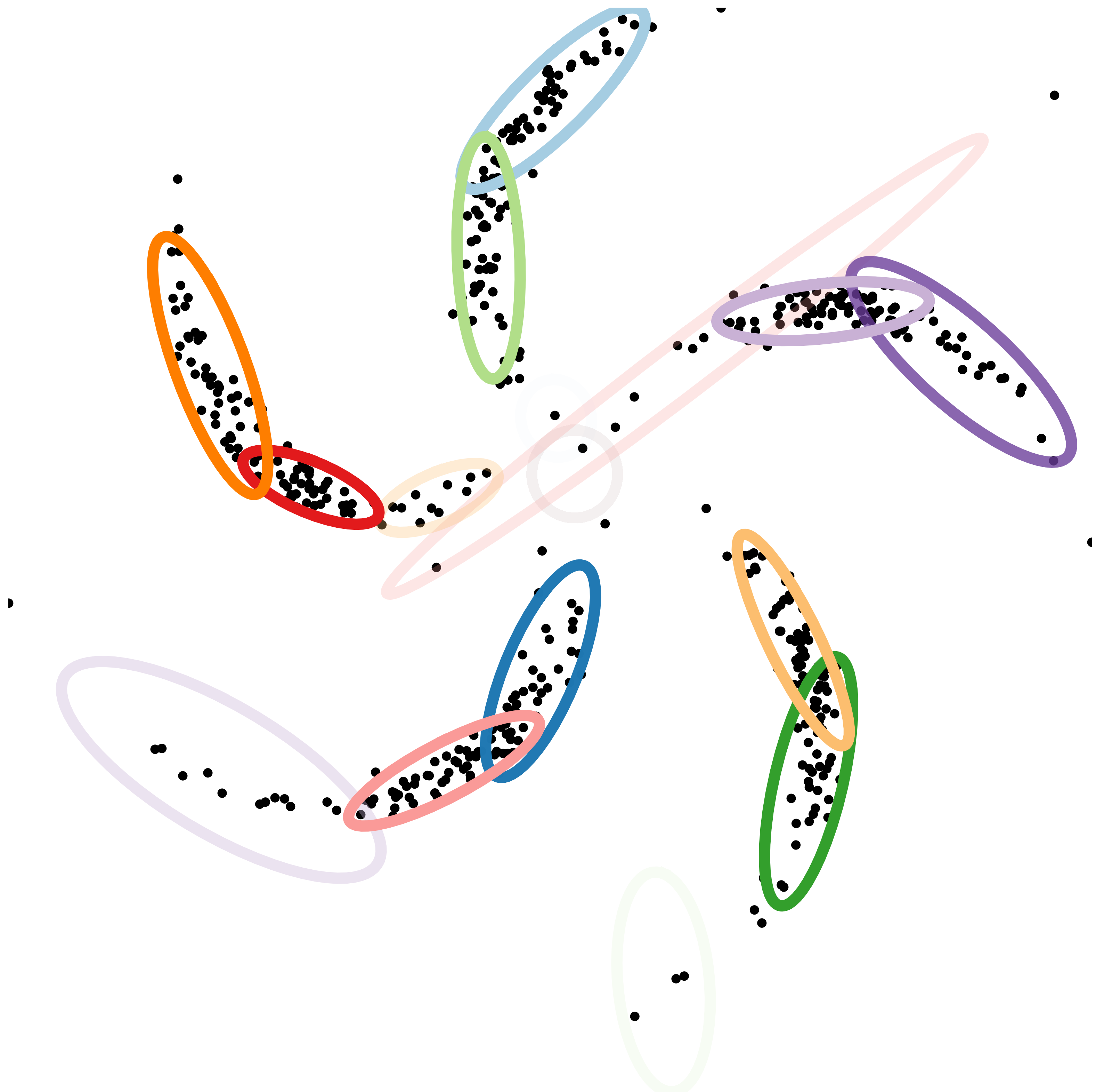
admixture / LDA / NMF

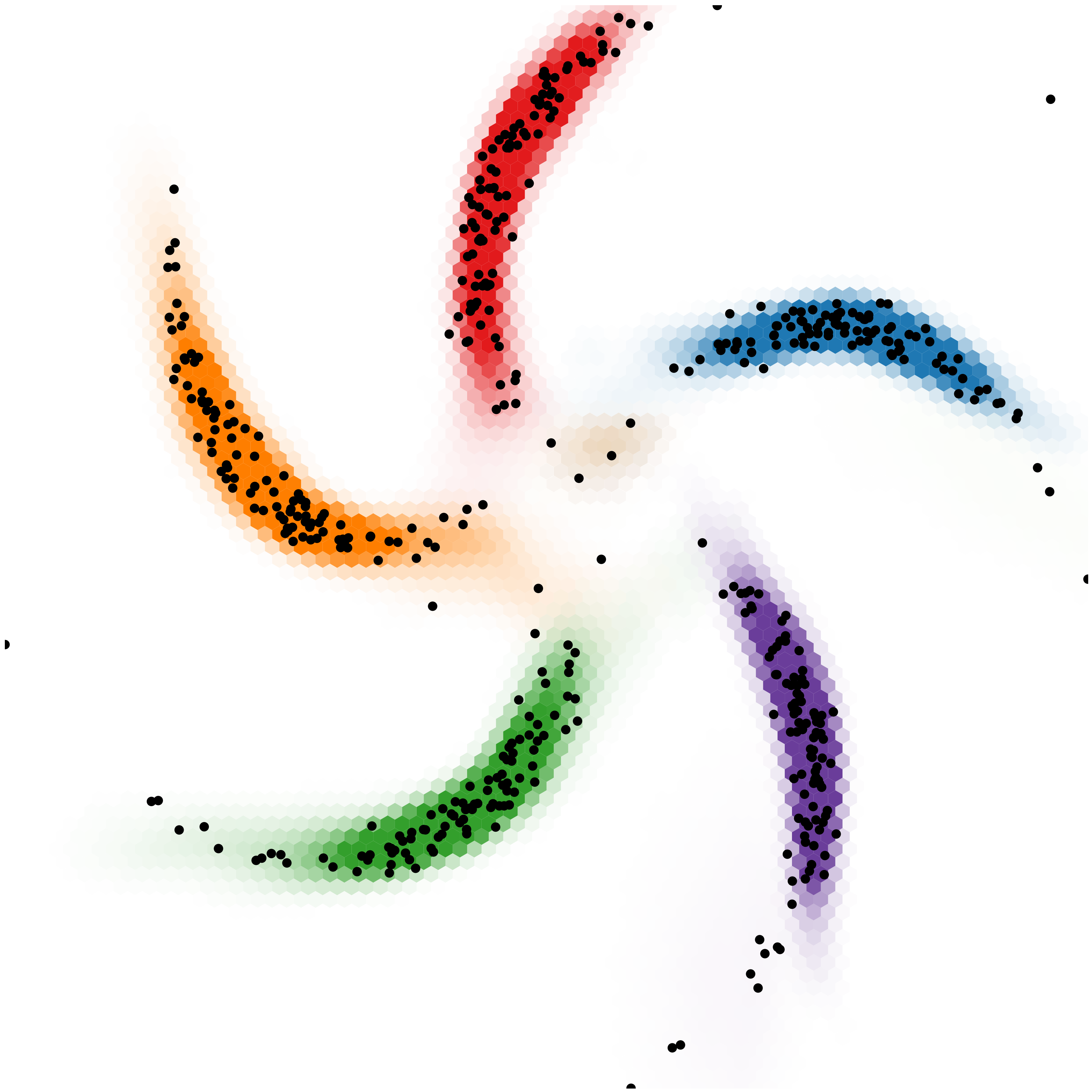
[1] Palmer, Wipf, Kreutz-Delgado, and Rao. Variational EM algorithms for non-Gaussian latent variable models. NIPS 2005.
[2] Ghahramani and Beal. Propagation algorithms for variational Bayesian learning. NIPS 2001.
[3] Beal. Variational algorithms for approximate Bayesian inference, Ch. 3. U of London Ph.D. Thesis 2003.
[4] Ghahramani and Hinton. Variational learning for switching state-space models. Neural Computation 2000.
[5] Jordan and Jacobs. Hierarchical Mixtures of Experts and the EM algorithm. Neural Computation 1994.
[6] Bengio and Frasconi. An Input Output HMM Architecture. NIPS 1995.
[7] Ghahramani and Jordan. Factorial Hidden Markov Models. Machine Learning 1997.
[8] Bach and Jordan. A probabilistic interpretation of Canonical Correlation Analysis. Tech. Report 2005.
[9] Archambeau and Bach. Sparse probabilistic projections. NIPS 2008.
[10] Hoffman, Bach, Blei. Online learning for Latent Dirichlet Allocation. NIPS 2010.

Courtesy of Matthew Johnson

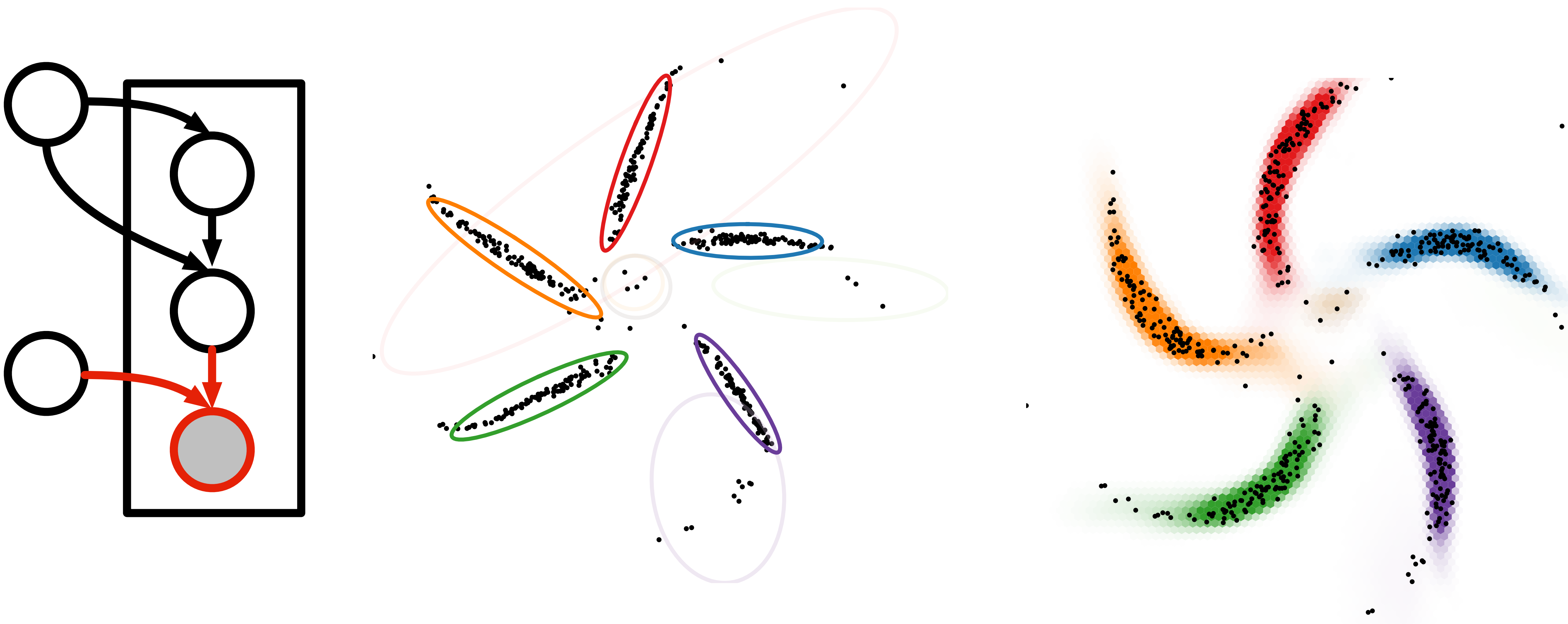




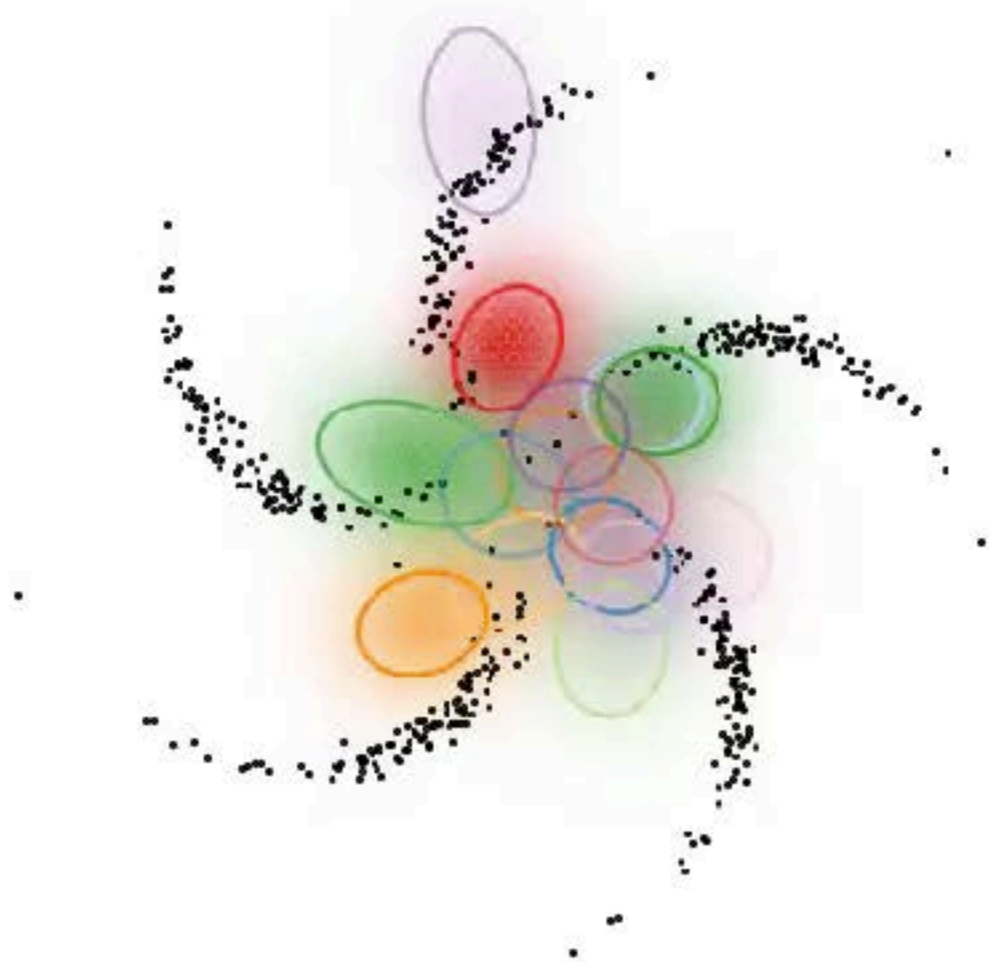




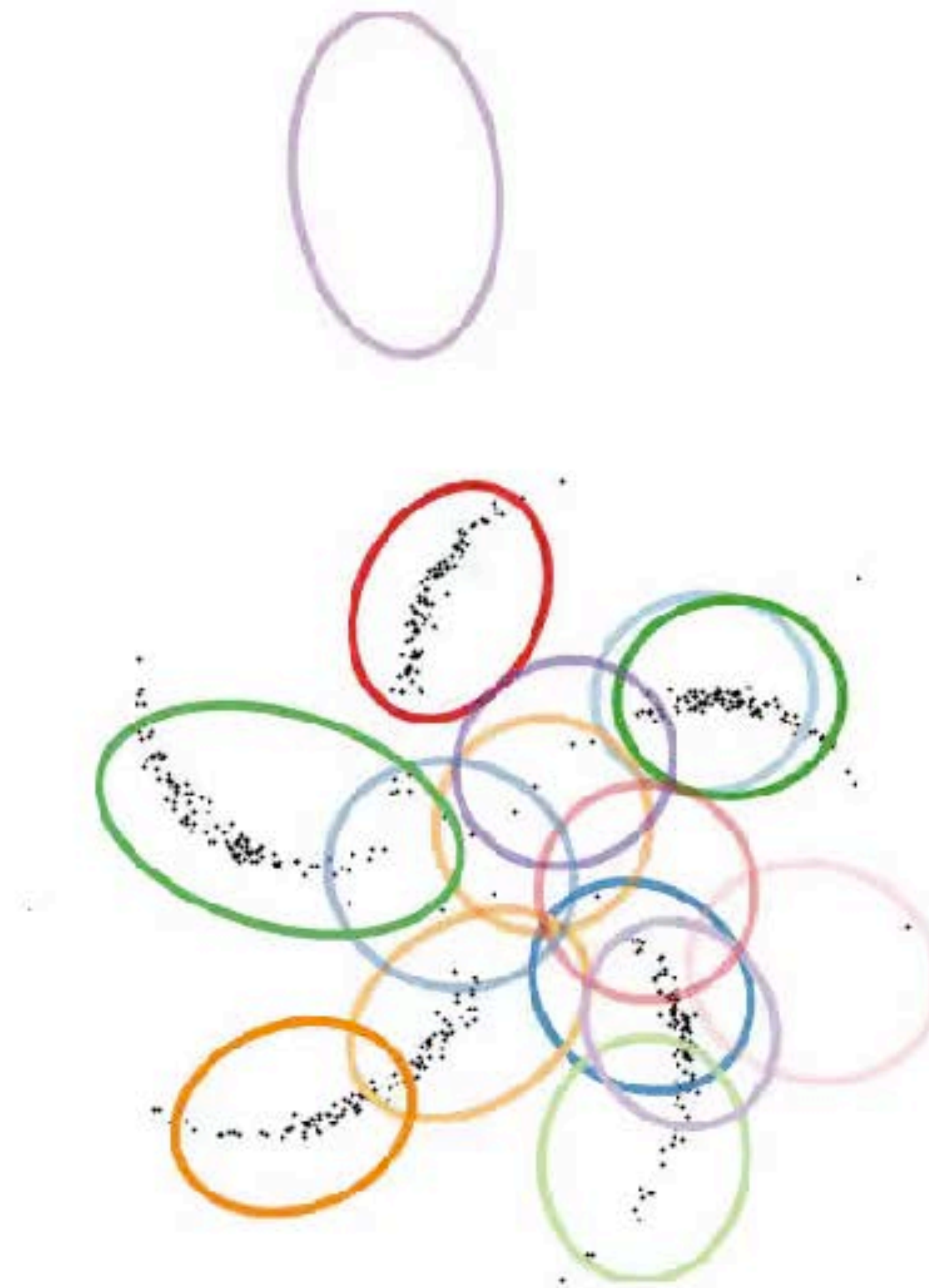
Modeling idea: graphical models on latent variables,
neural network models for observations



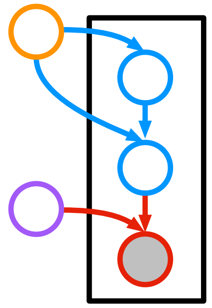
Composing graphical models with neural networks for structured representations and fast inference. Johnson, Duvenaud, Wiltschko, Datta, Adams, NIPS 2016



data space

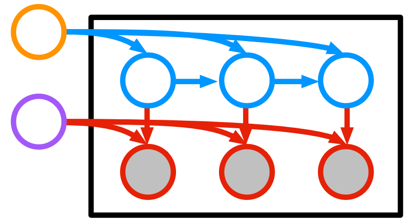


latent space



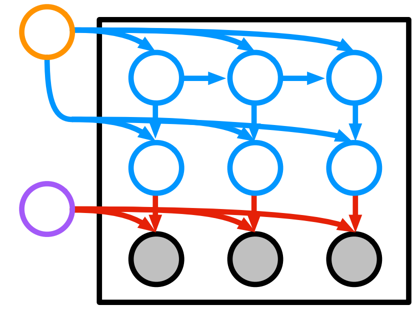
[1]

Gaussian mixture model



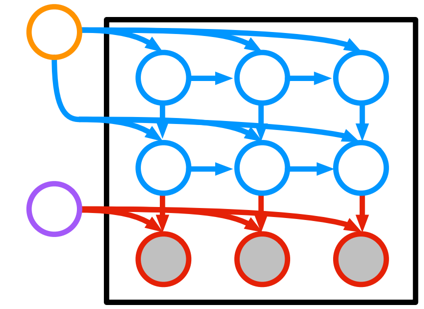
[2]

Linear dynamical system



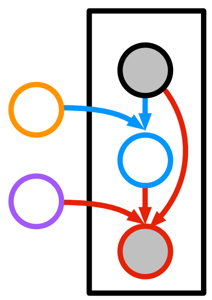
[3]

Hidden Markov model



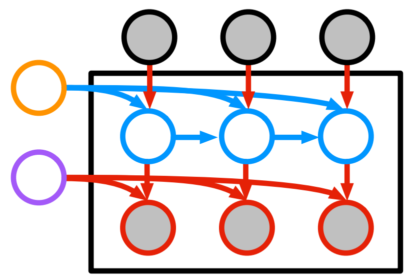
[4]

Switching LDS



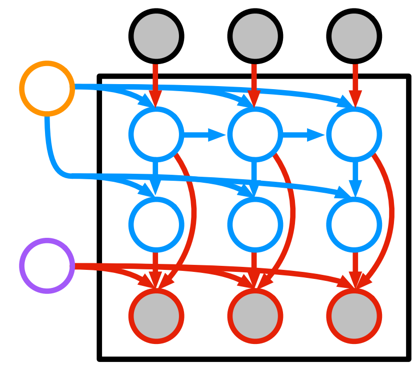
[5]

Mixture of Experts



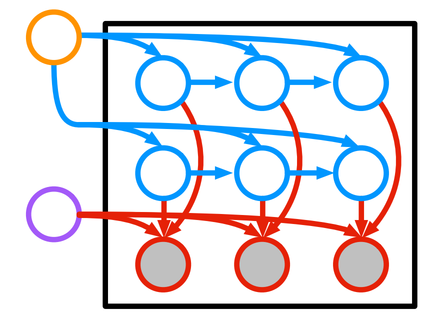
[2]

Driven LDS



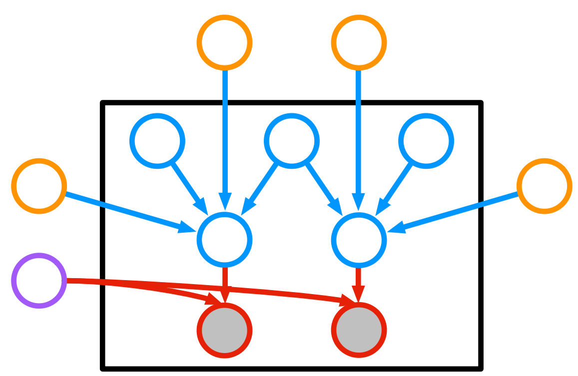
[6]

IO-HMM



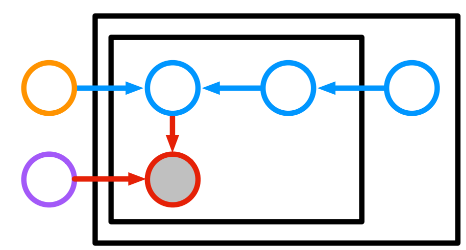
[7]

Factorial HMM



[8,9]

Canonical correlations analysis



[10]

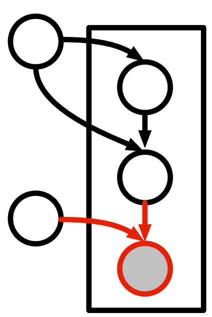
admixture / LDA / NMF

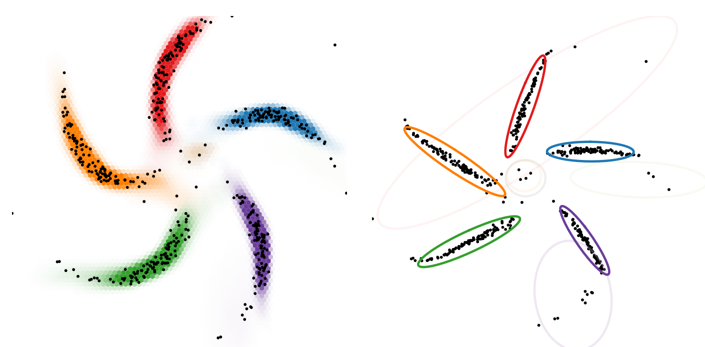
[1] Palmer, Wipf, Kreutz-Delgado, and Rao. Variational EM algorithms for non-Gaussian latent variable models. NIPS 2005.
[2] Ghahramani and Beal. Propagation algorithms for variational Bayesian learning. NIPS 2001.
[3] Beal. Variational algorithms for approximate Bayesian inference, Ch. 3. U of London Ph.D. Thesis 2003.
[4] Ghahramani and Hinton. Variational learning for switching state-space models. Neural Computation 2000.
[5] Jordan and Jacobs. Hierarchical Mixtures of Experts and the EM algorithm. Neural Computation 1994.
[6] Bengio and Frasconi. An Input Output HMM Architecture. NIPS 1995.
[7] Ghahramani and Jordan. Factorial Hidden Markov Models. Machine Learning 1997.
[8] Bach and Jordan. A probabilistic interpretation of Canonical Correlation Analysis. Tech. Report 2005.
[9] Archambeau and Bach. Sparse probabilistic projections. NIPS 2008.
[10] Hoffman, Bach, Blei. Online learning for Latent Dirichlet Allocation. NIPS 2010.

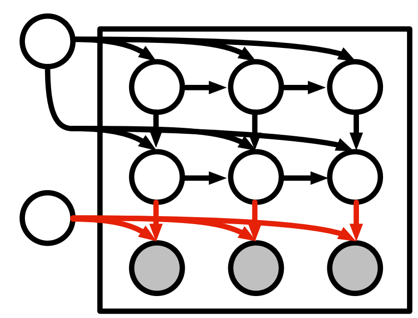
Courtesy of Matthew Johnson

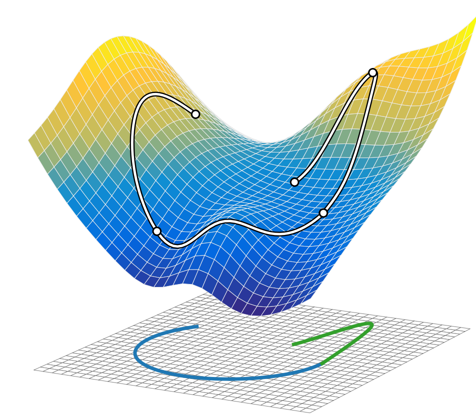
Probabilistic graphical models	Deep learning
+ structured representations	– neural net “goo”
+ priors and uncertainty	– difficult parameterization
+ data and computational efficiency	– can require lots of data
– rigid assumptions may not fit	+ flexible
– feature engineering	+ feature learning
– top-down inference	+ recognition networks

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

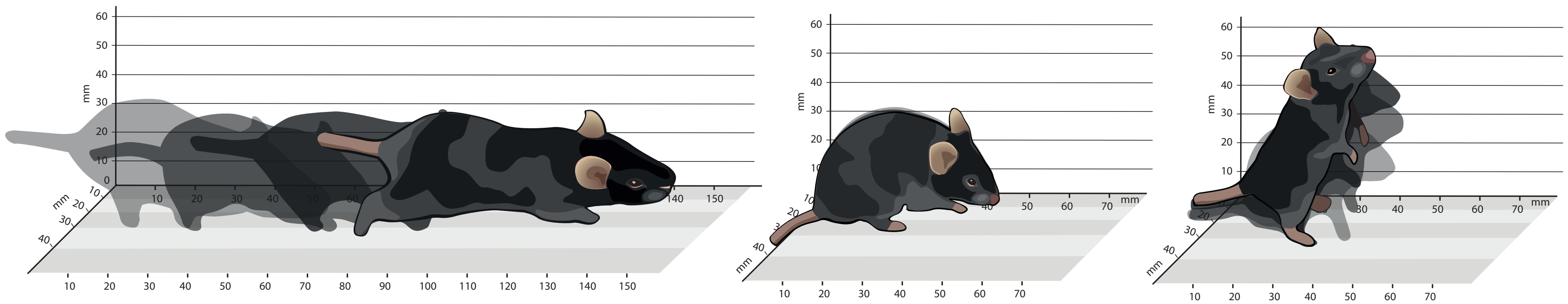






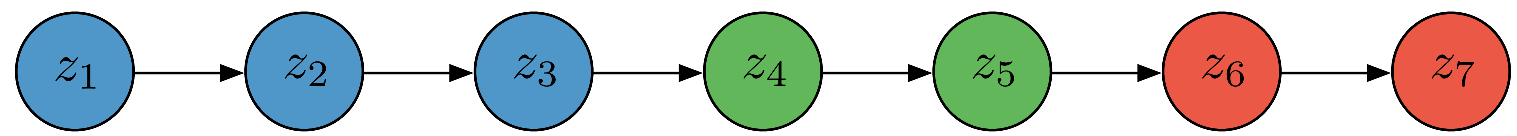


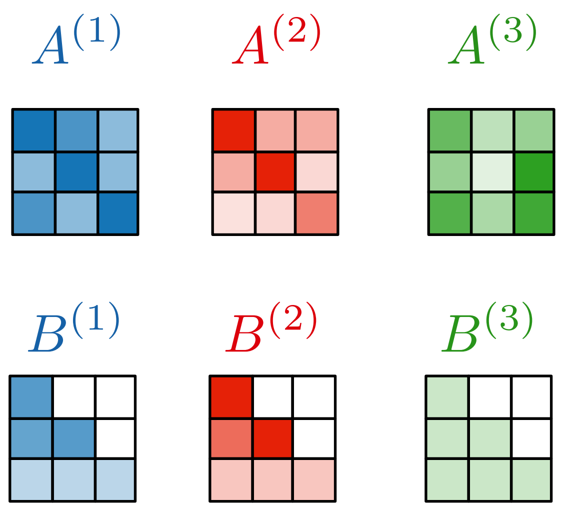
Application: learn syllable representation of behavior from video



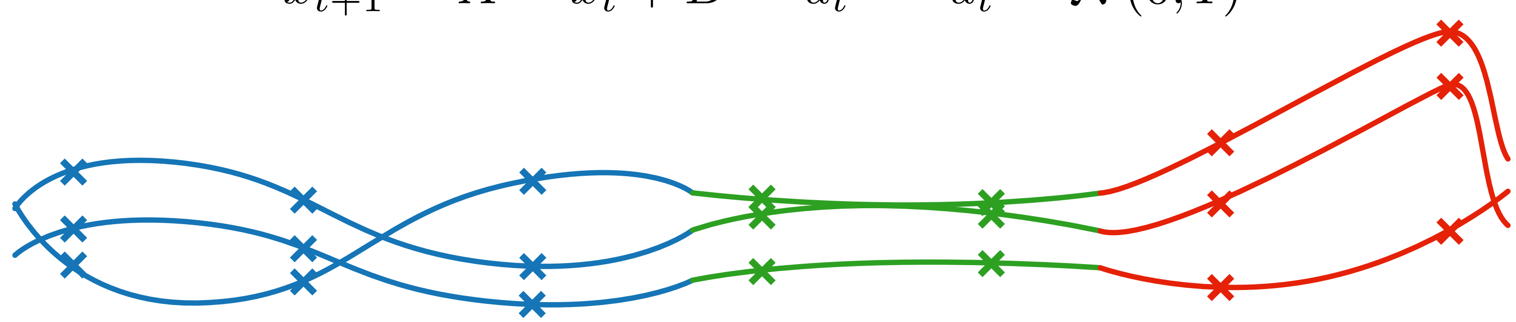
$$\pi = \begin{bmatrix} \text{—} & \pi^{(1)} & \text{—} \\ \text{—} & \pi^{(2)} & \text{—} \\ \text{—} & \pi^{(3)} & \text{—} \end{bmatrix}$$

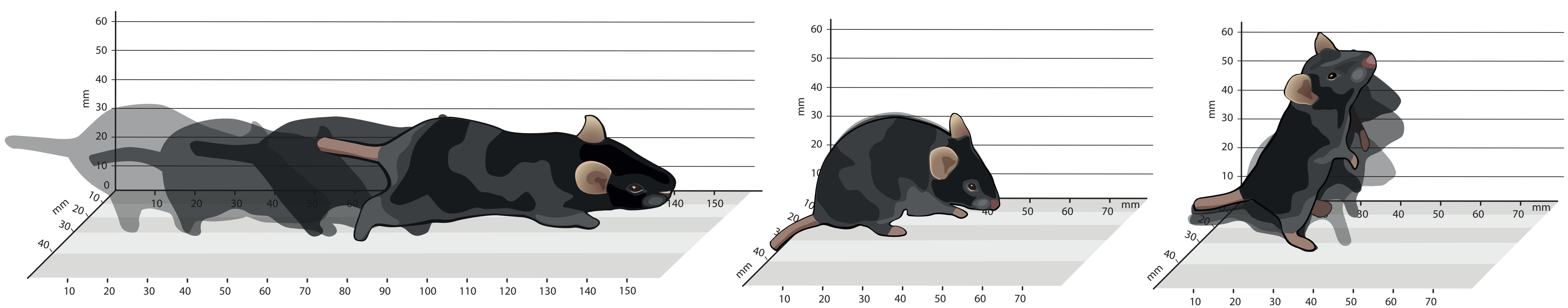
$$z_{t+1} \sim \pi^{(z_t)}$$

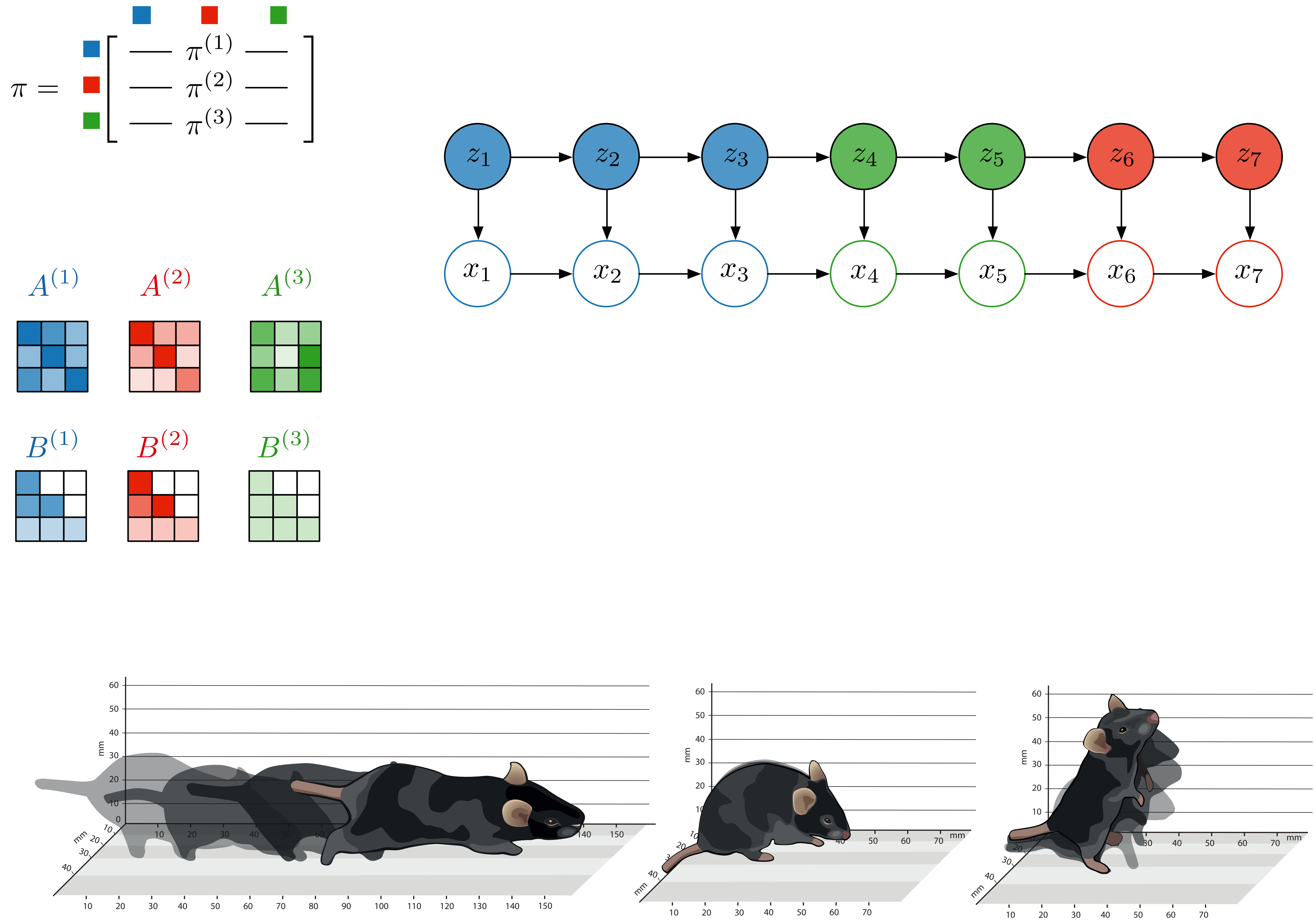




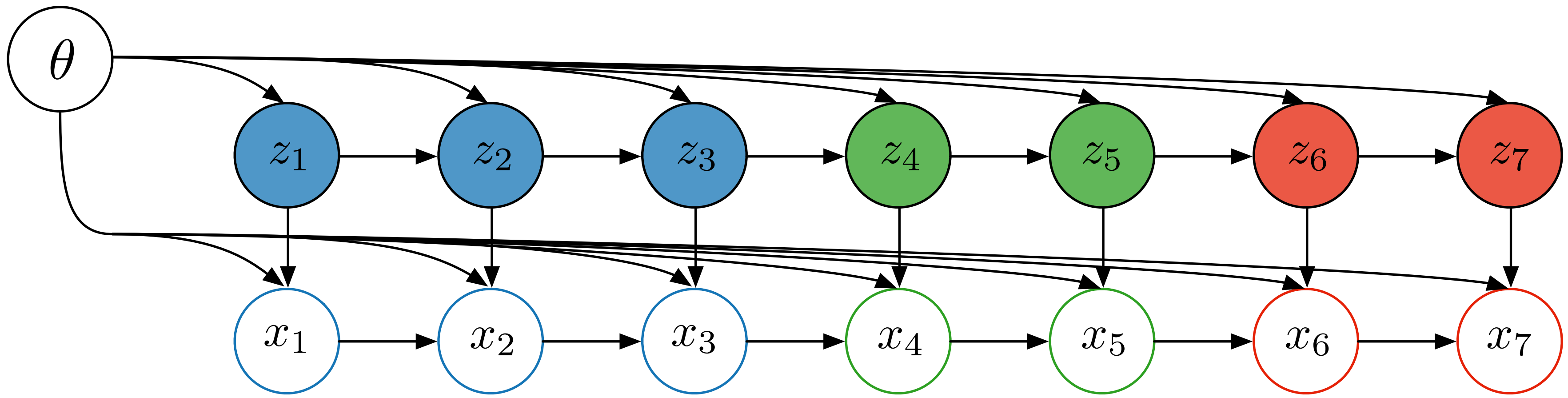
$$x_{t+1} = A^{(z_t)} x_t + B^{(z_t)} u_t \qquad u_t \overset{\text{iid}}{\sim} \mathcal{N}(0, I)$$



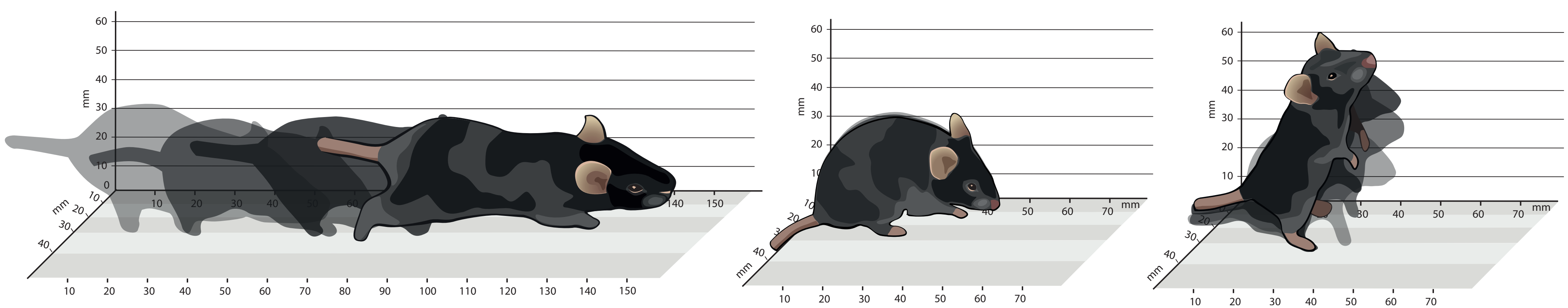




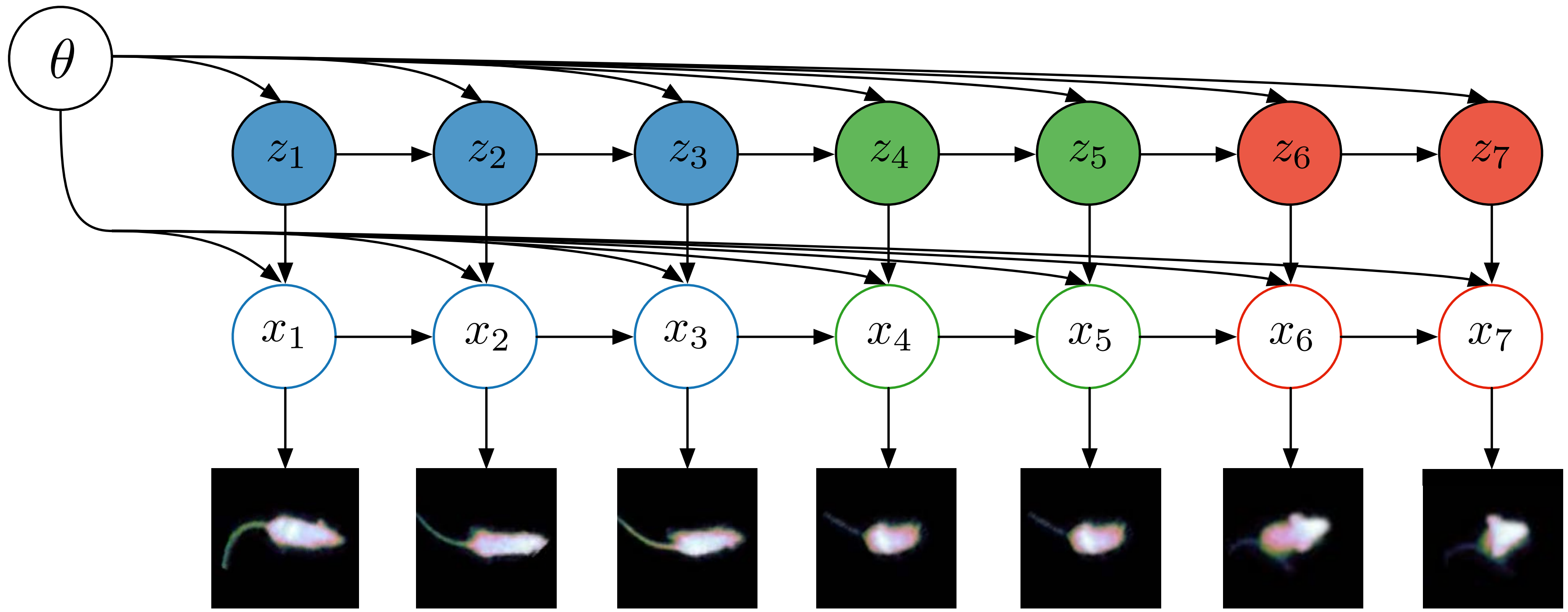
$\pi =$
$\pi^{(1)}$
$\pi^{(2)}$
$\pi^{(3)}$
$A^{(1)}$ $A^{(2)}$ $A^{(3)}$
$B^{(1)}$ $B^{(2)}$ $B^{(3)}$
z_1 z_2 z_3 z_4 z_5 z_6 z_7
x_1 x_2 x_3 x_4 x_5 x_6 x_7
mm

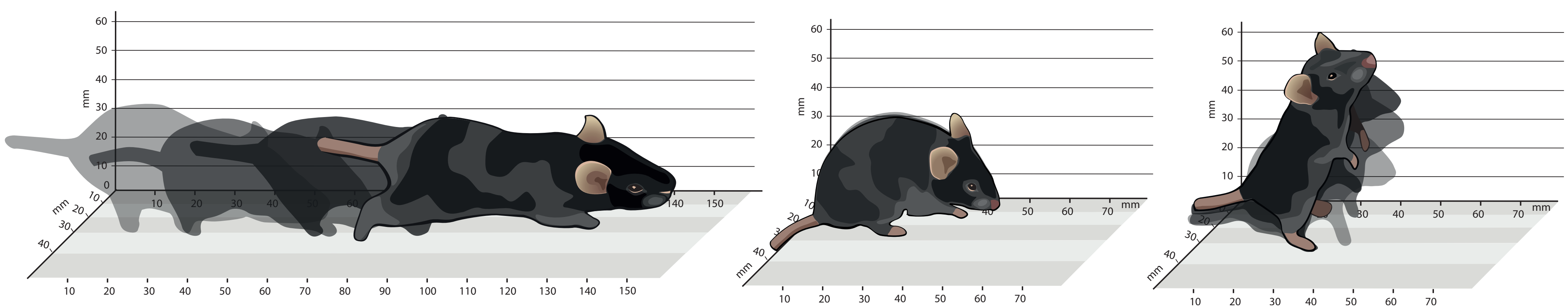


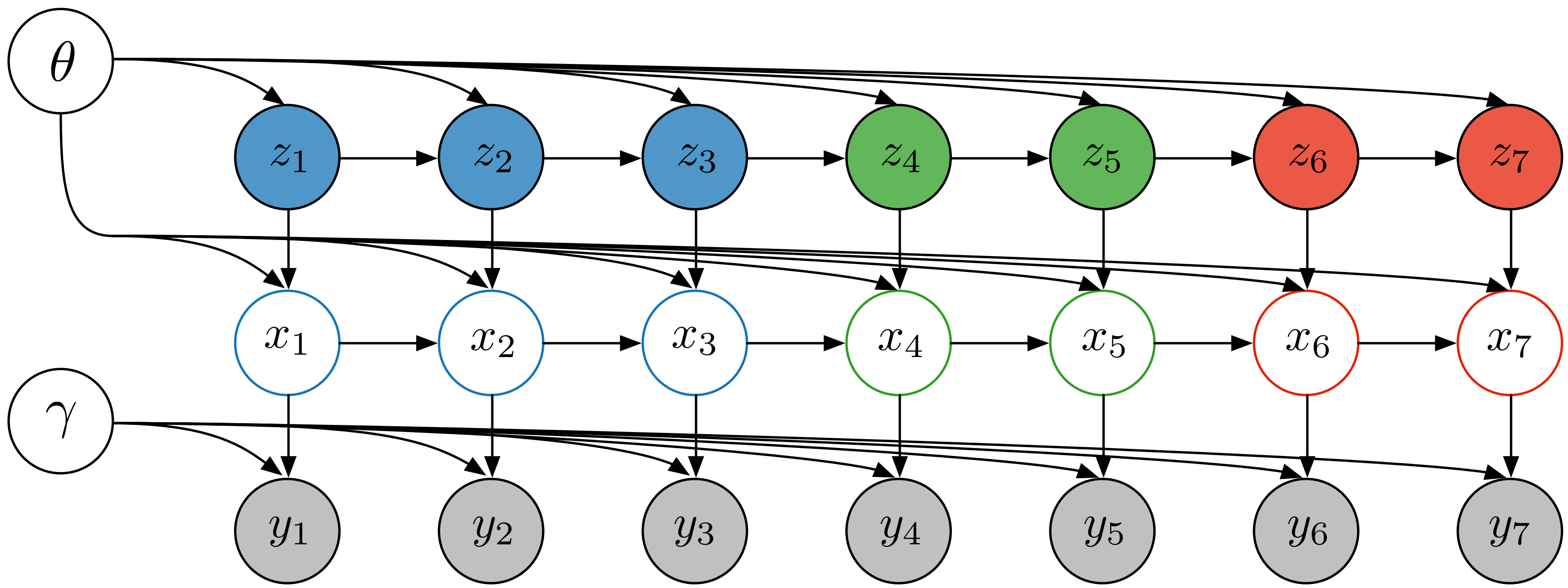
θ
z_1
z_2
z_3
z_4
z_5
z_6
z_7
x_1
x_2
x_3
x_4
x_5
x_6
x_7



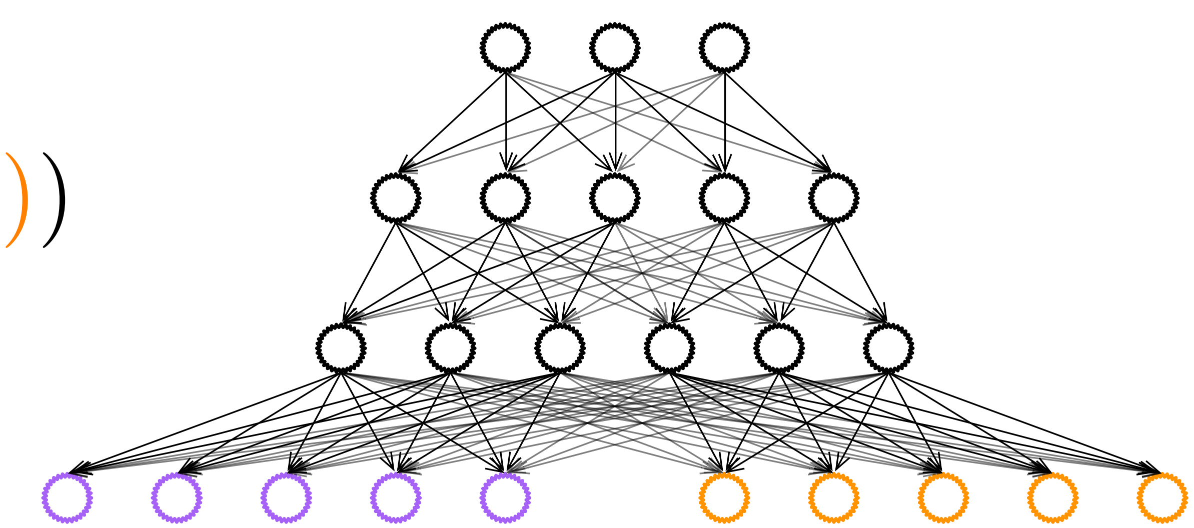
mm


θ
z_1 z_2 z_3 z_4 z_5 z_6 z_7
x_1 x_2 x_3 x_4 x_5 x_6 x_7

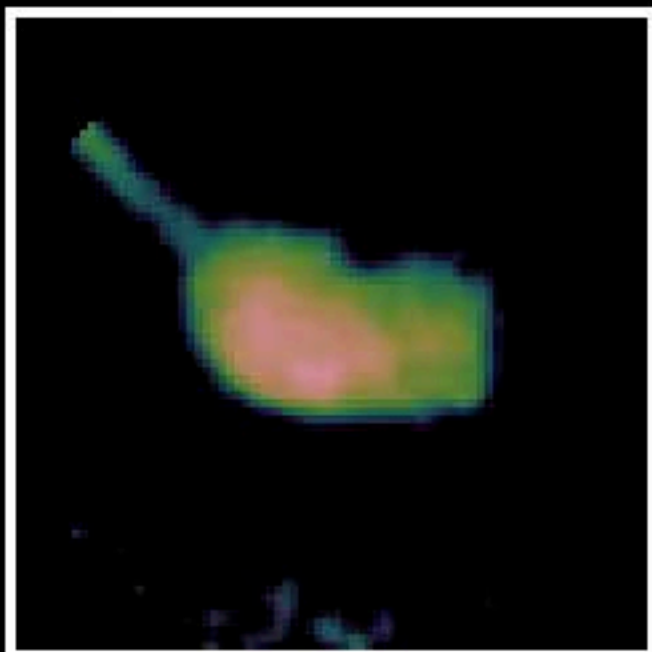

mm

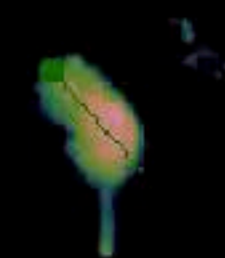


$$y_t \mid x_t, \gamma \;\sim\; \mathcal{N}(\mu(x_t;\gamma), \Sigma(x_t;\gamma))$$

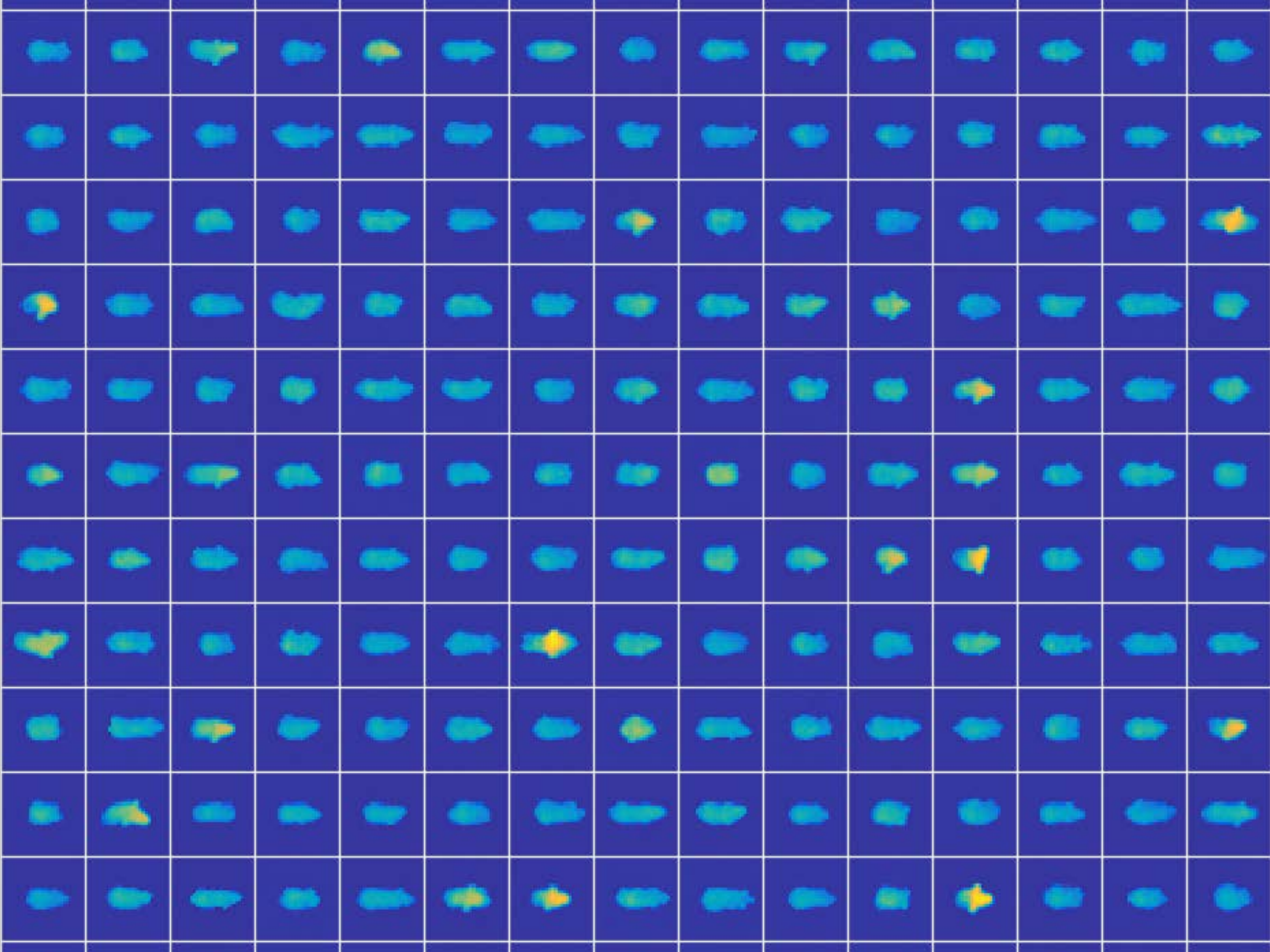


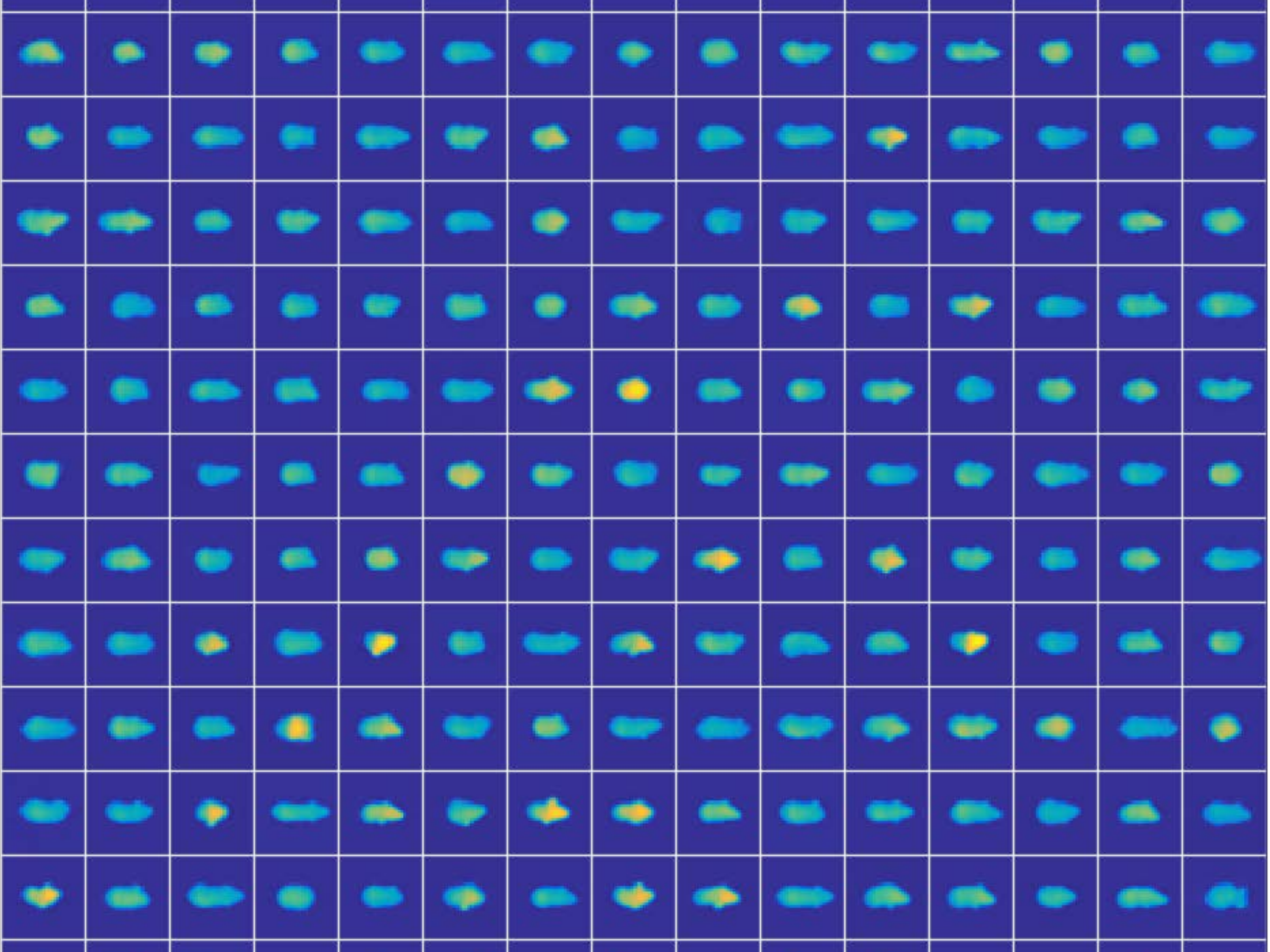
Frame 0

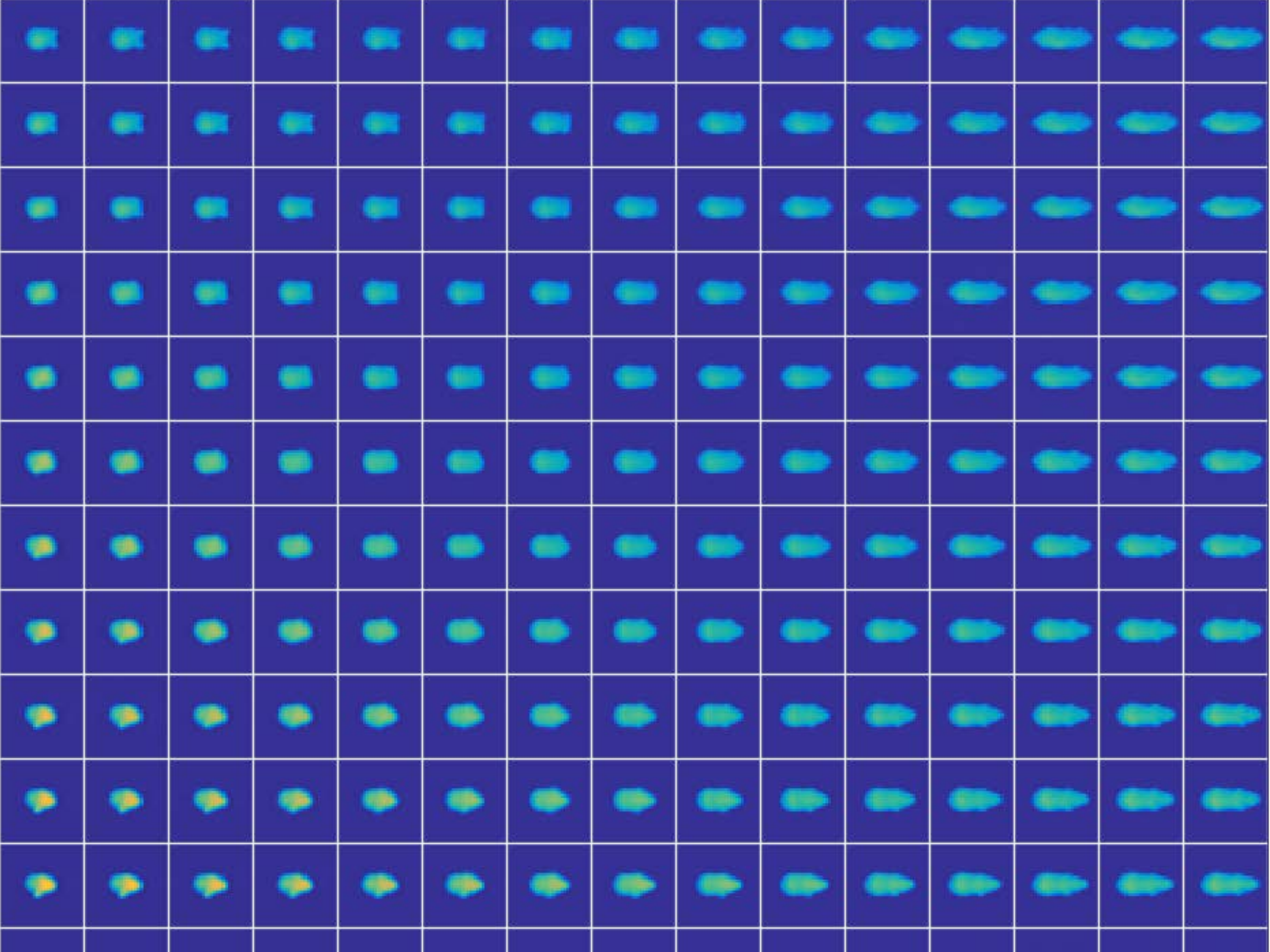


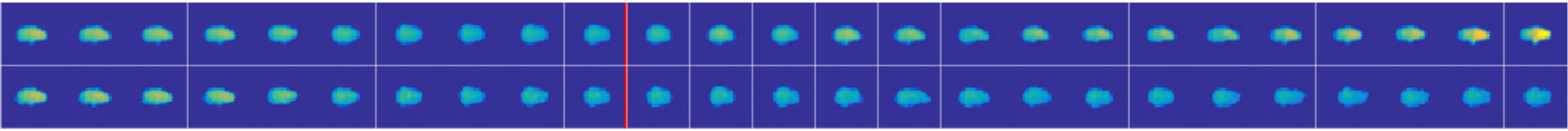


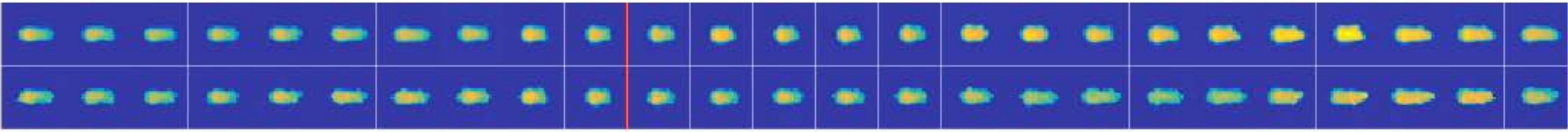
Alexander Wiltschko, **Matthew Johnson**, et al., Neuron 2015.

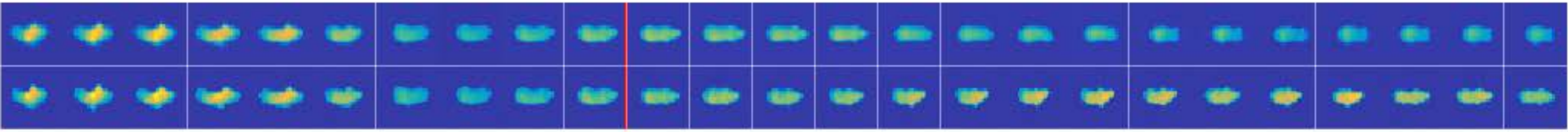


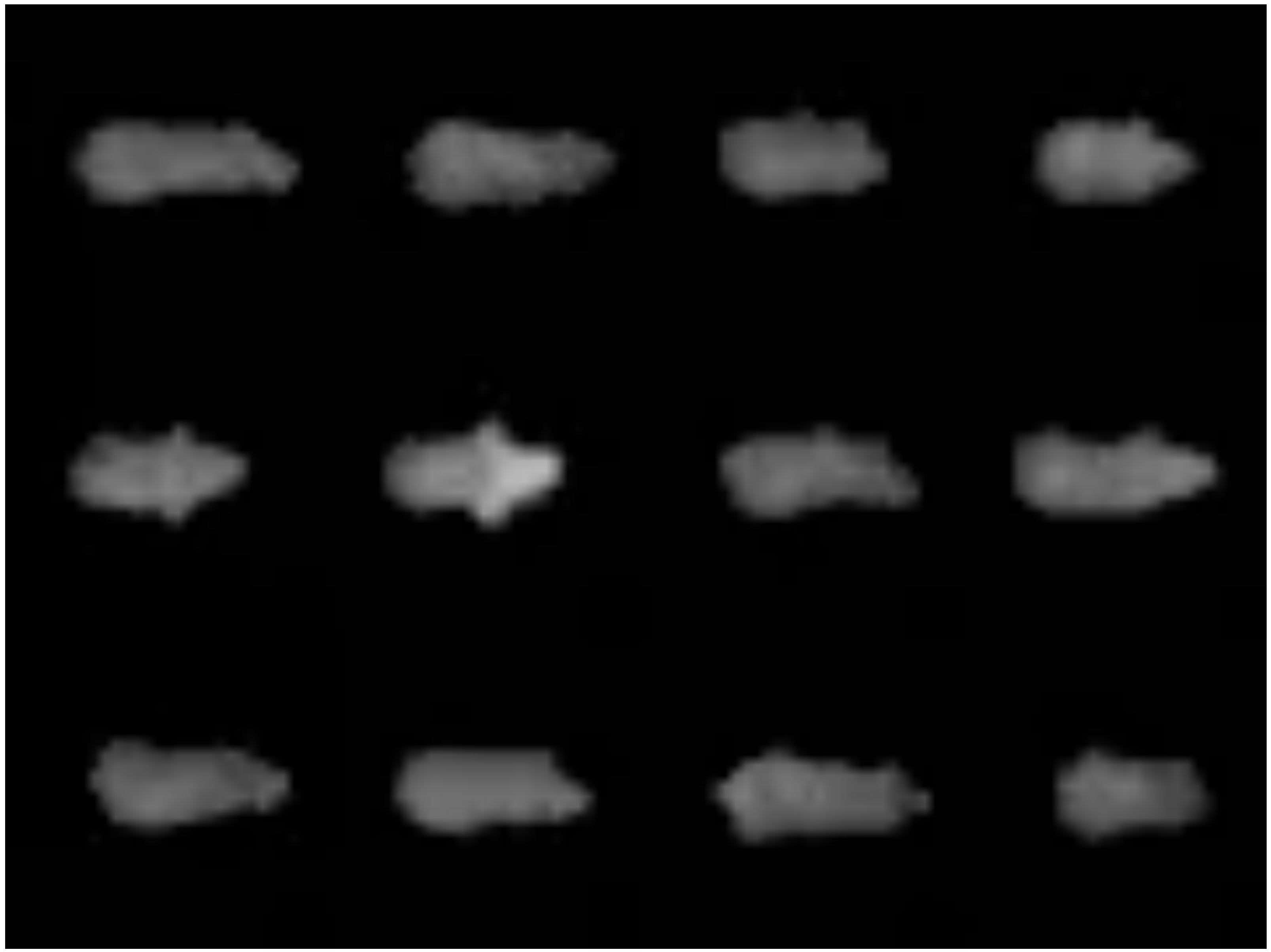




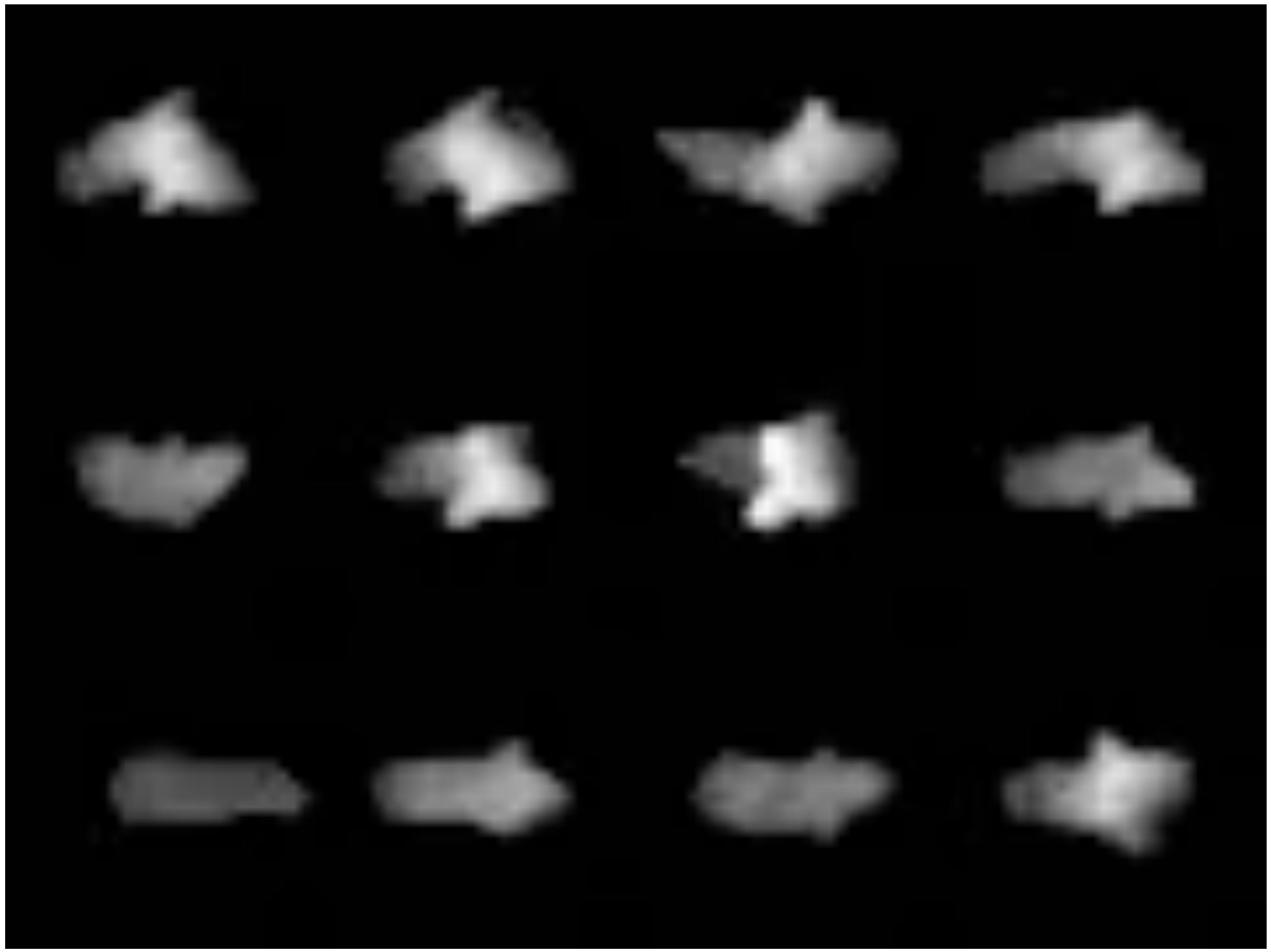




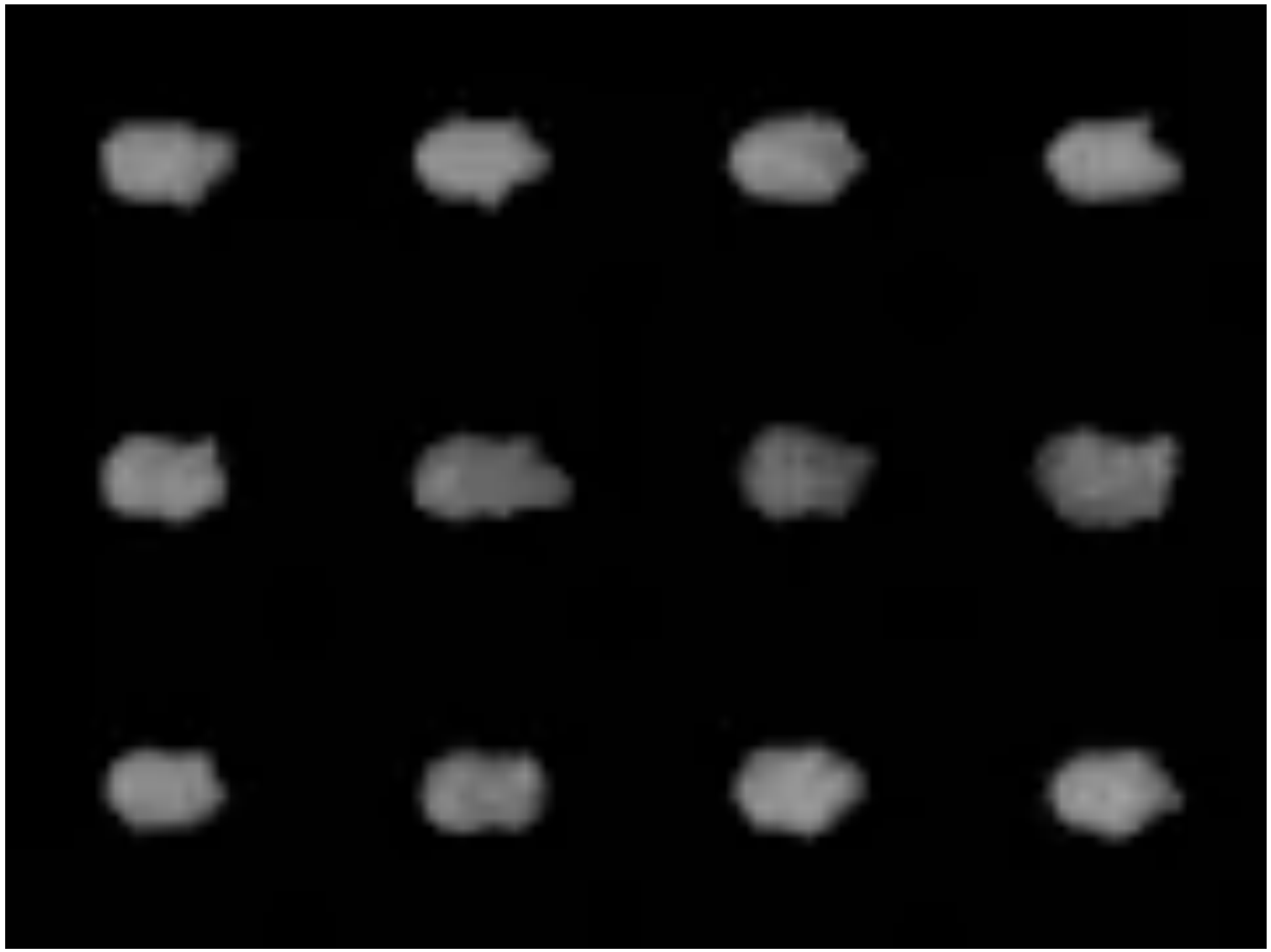




start rear



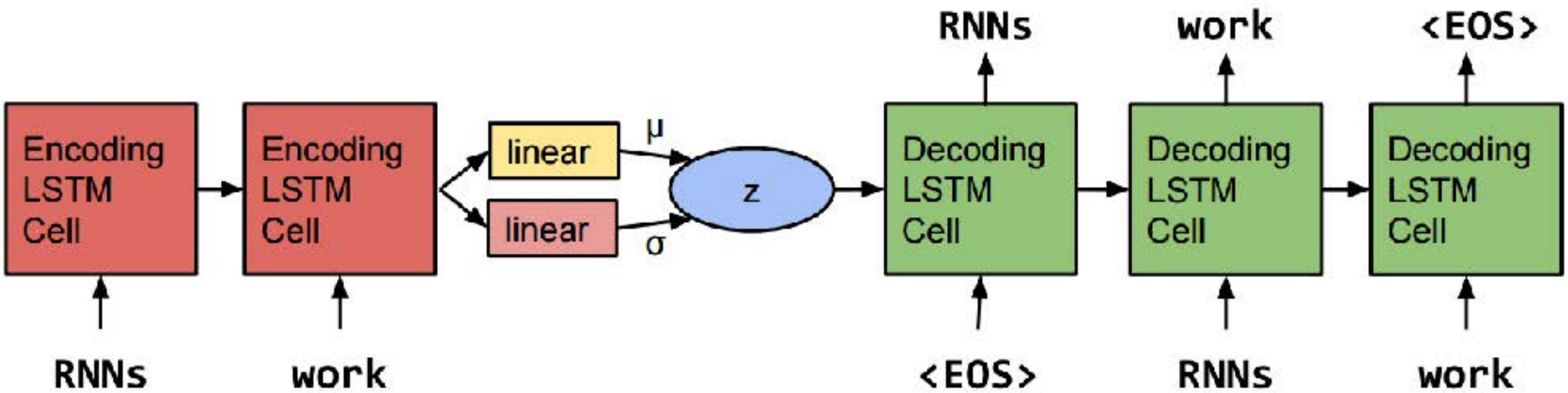
fall from rear



grooming

Application: Generative Design of Molecules

Text autoencoders



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

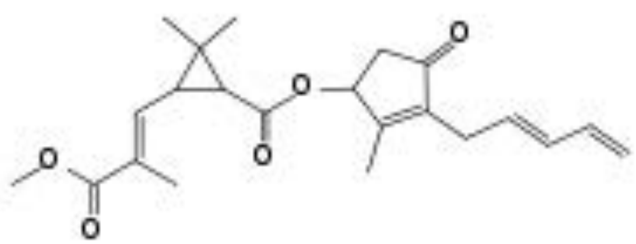
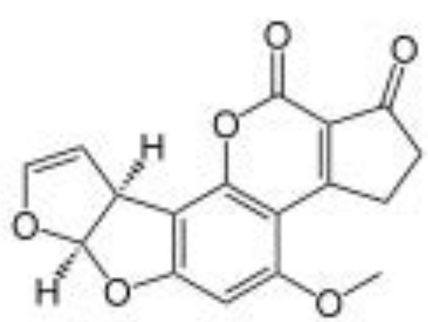
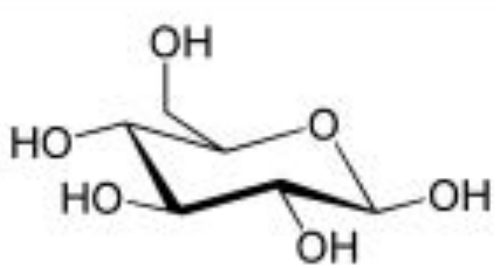
Text VAE - **Interpolation**

“ 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 .

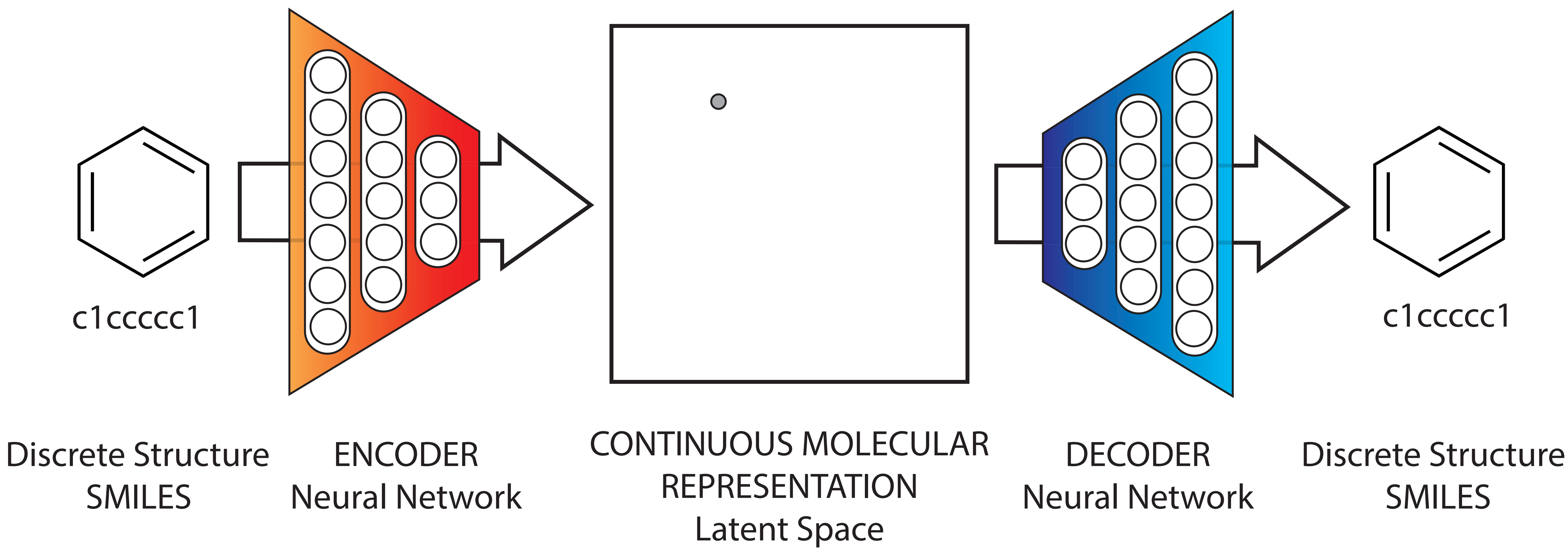
it made me want to cry .
no one had seen him since .
it made me feel uneasy .
no one had seen him .
the thought made me smile .
the pain was unbearable .
the crowd was silent .
the man called out .
the old man said .
the man asked .

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?

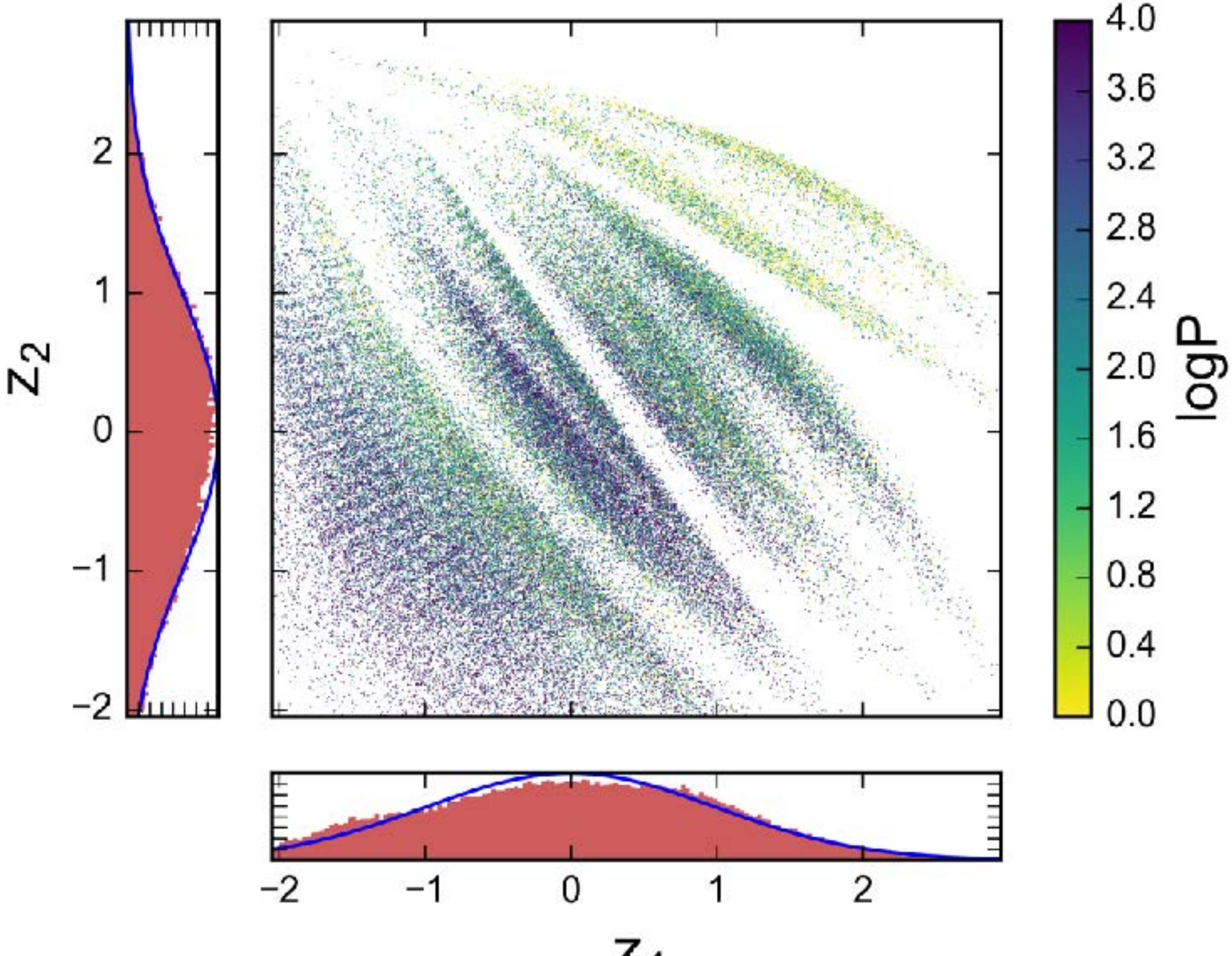
Graph	SMILES string
	CCC[C@@H](O)CC\C=C\C=C\C#CC#C\C=C\CO
	COC(=O)C(\C)=C\C1C(C)(C)[C@H]1C(=O)O[C@@H]2C(C)=C(C(=O)C2)CC=CC=C
	O1C=C[C@H]([C@H]1O2)c3c2cc(OC)c4c3OC(=O)C5=C4CCC(=O)5
	OC[C@@H](O1)[C@@H](O)[C@H](O)[C@@H](O)[C@@H](O)1

Repurposing text autoencoders

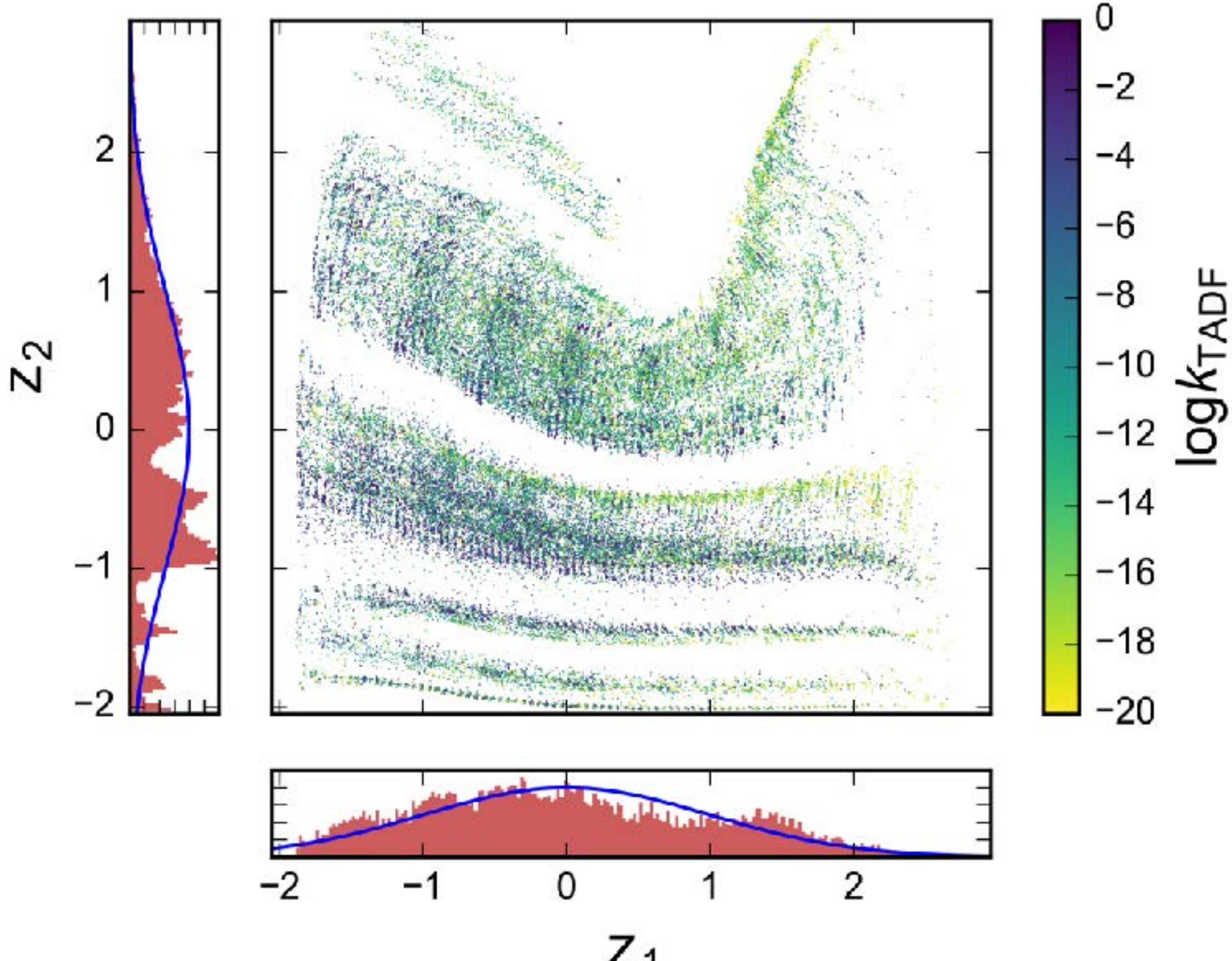


Can be trained on unlabeled data

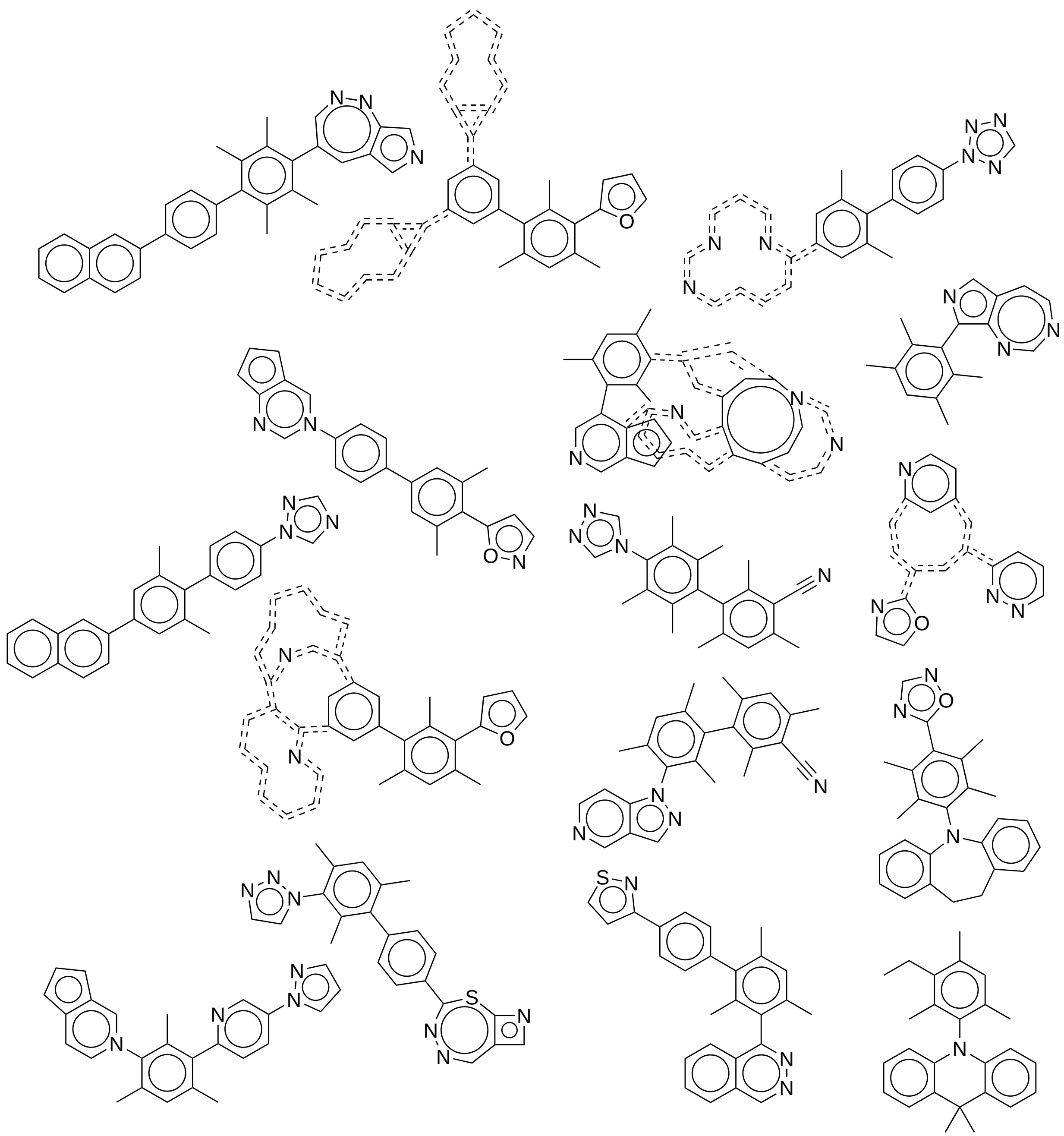
Map of 220,000 Drugs



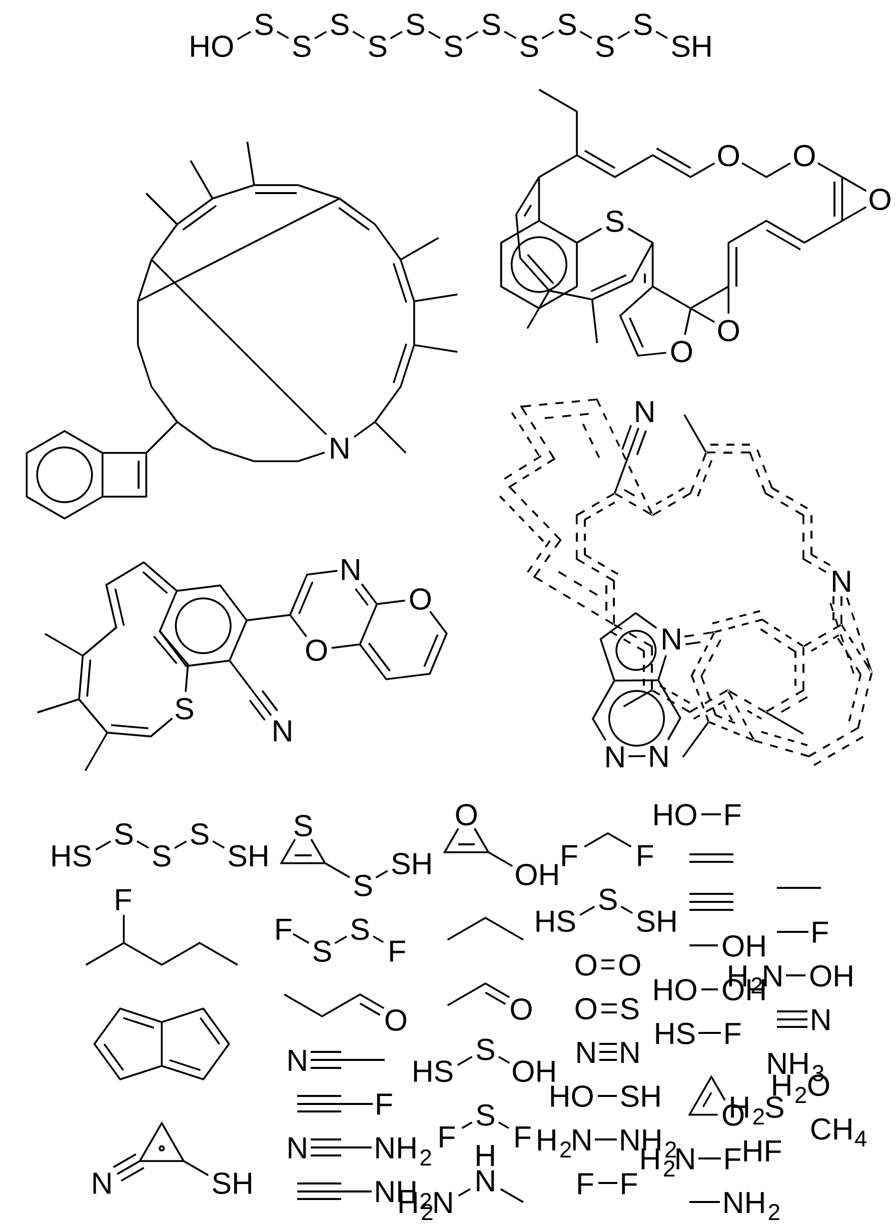
Map of 100,000 OLEDs



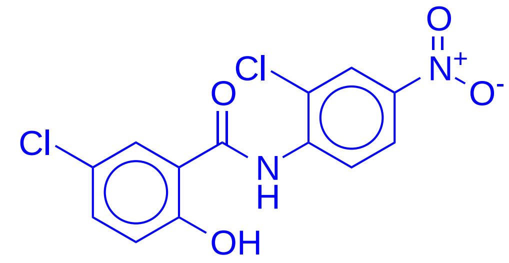
Random Organic LEDs

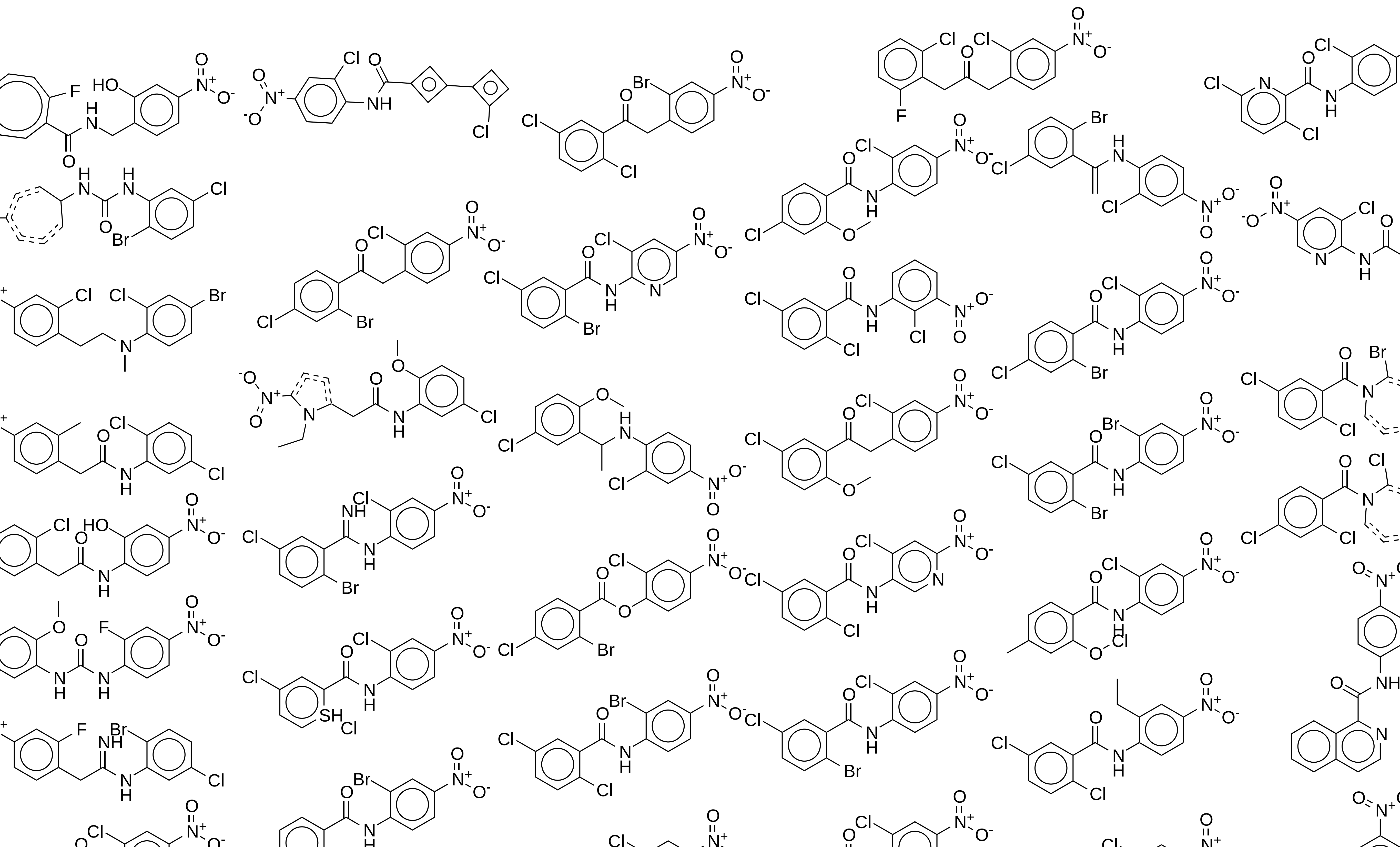


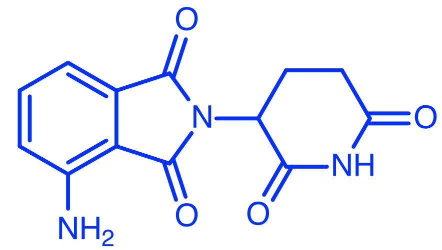
Variational autoencoder

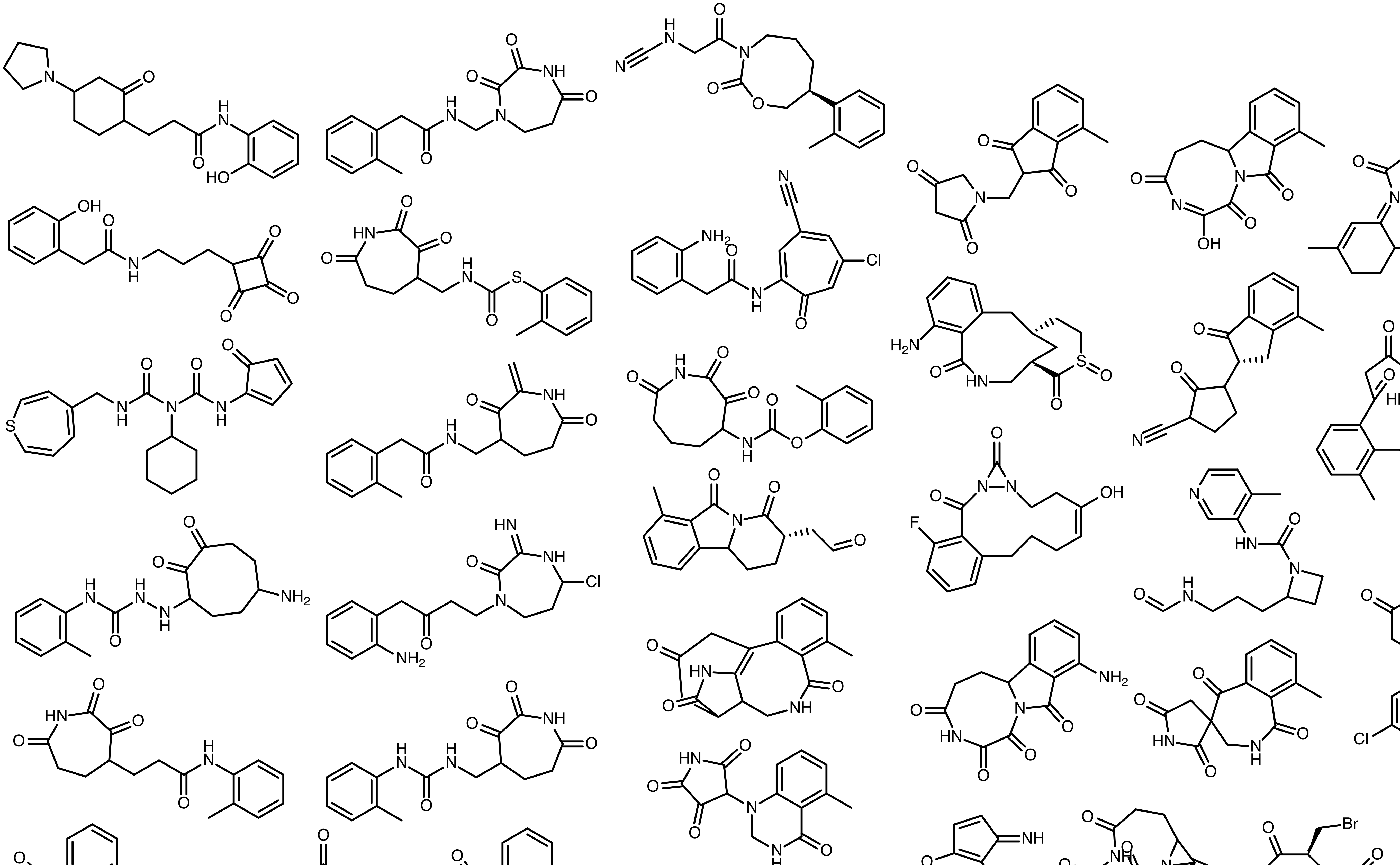


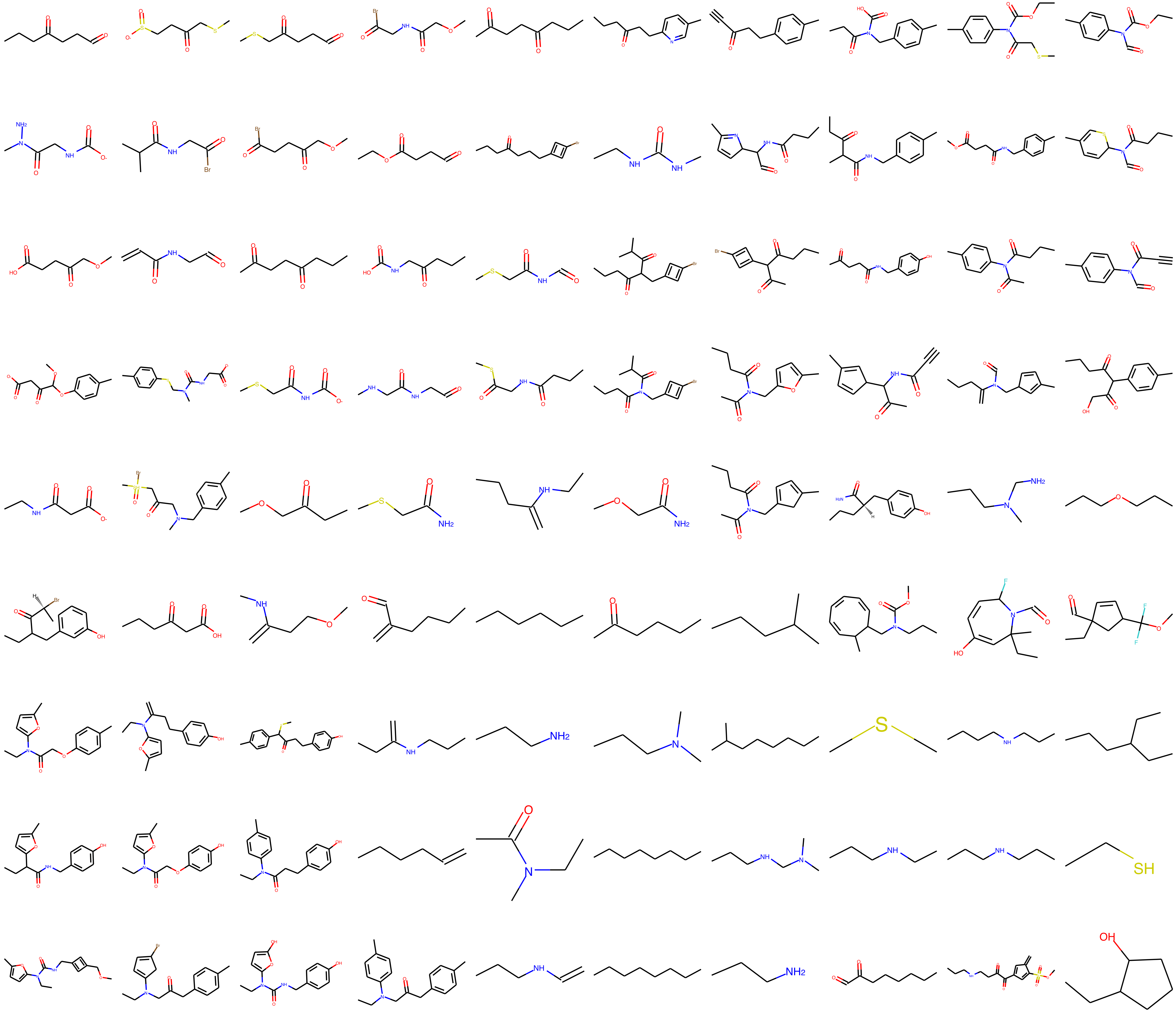
Standard autoencoder

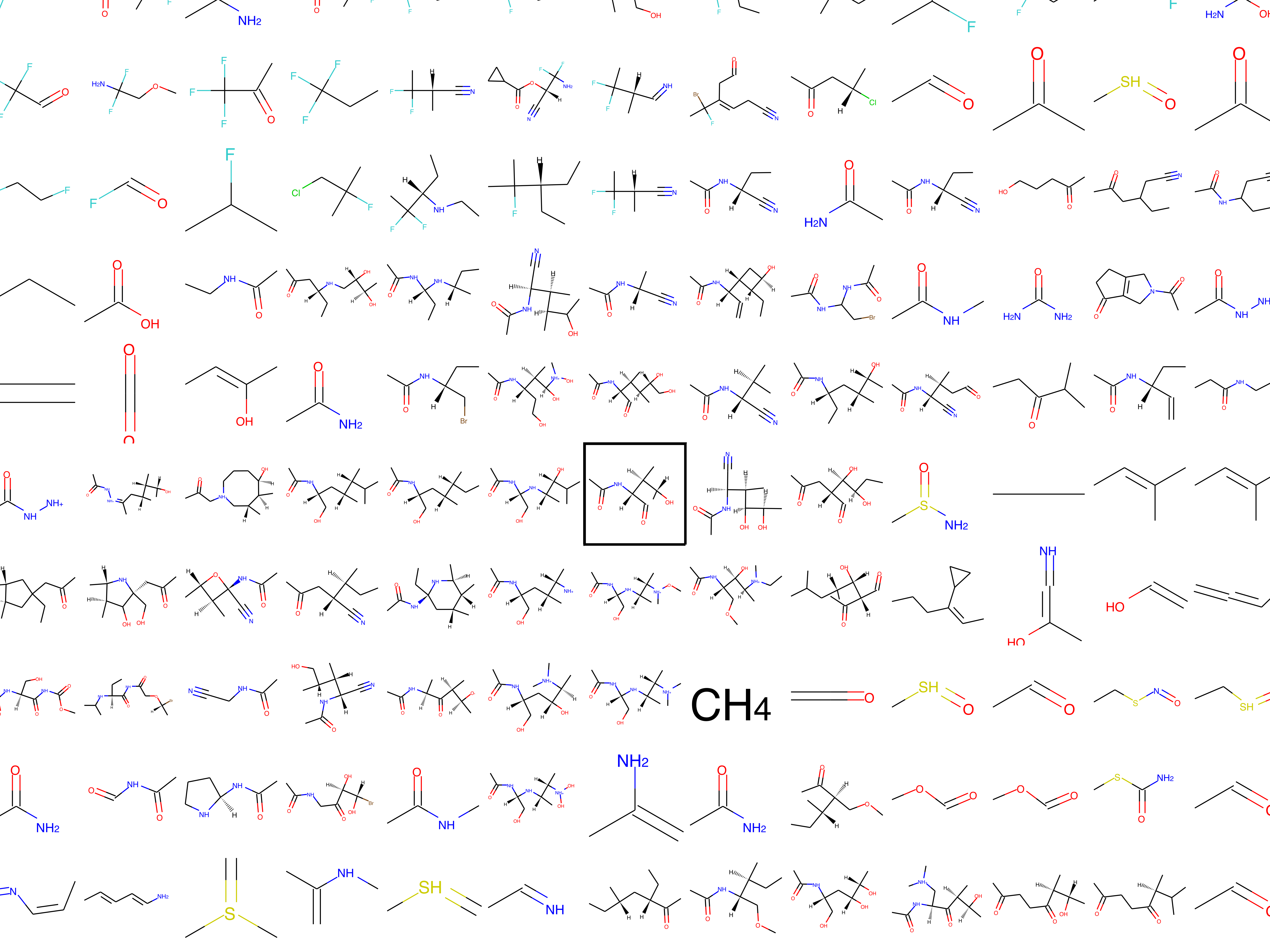
Molecules near 

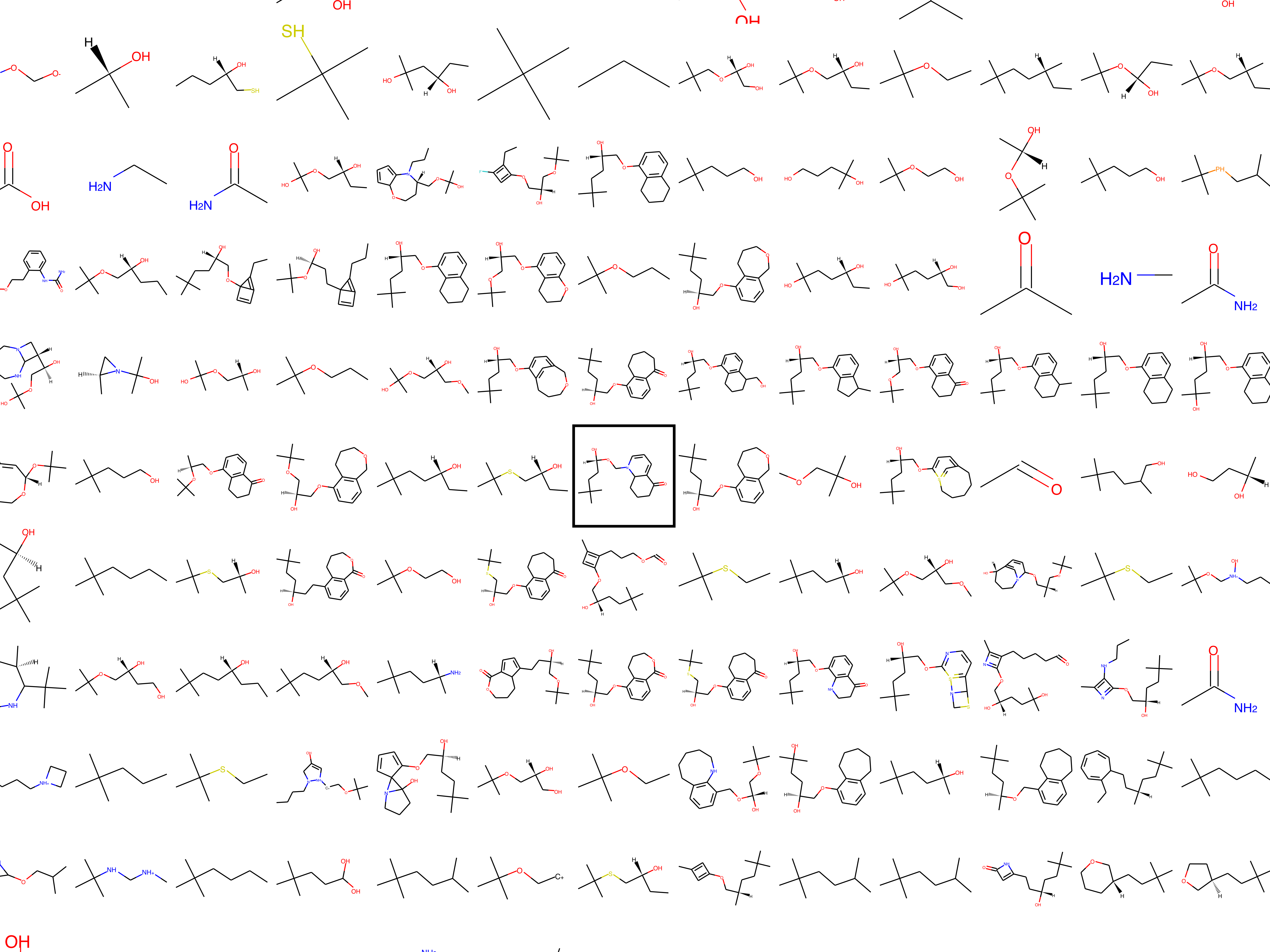


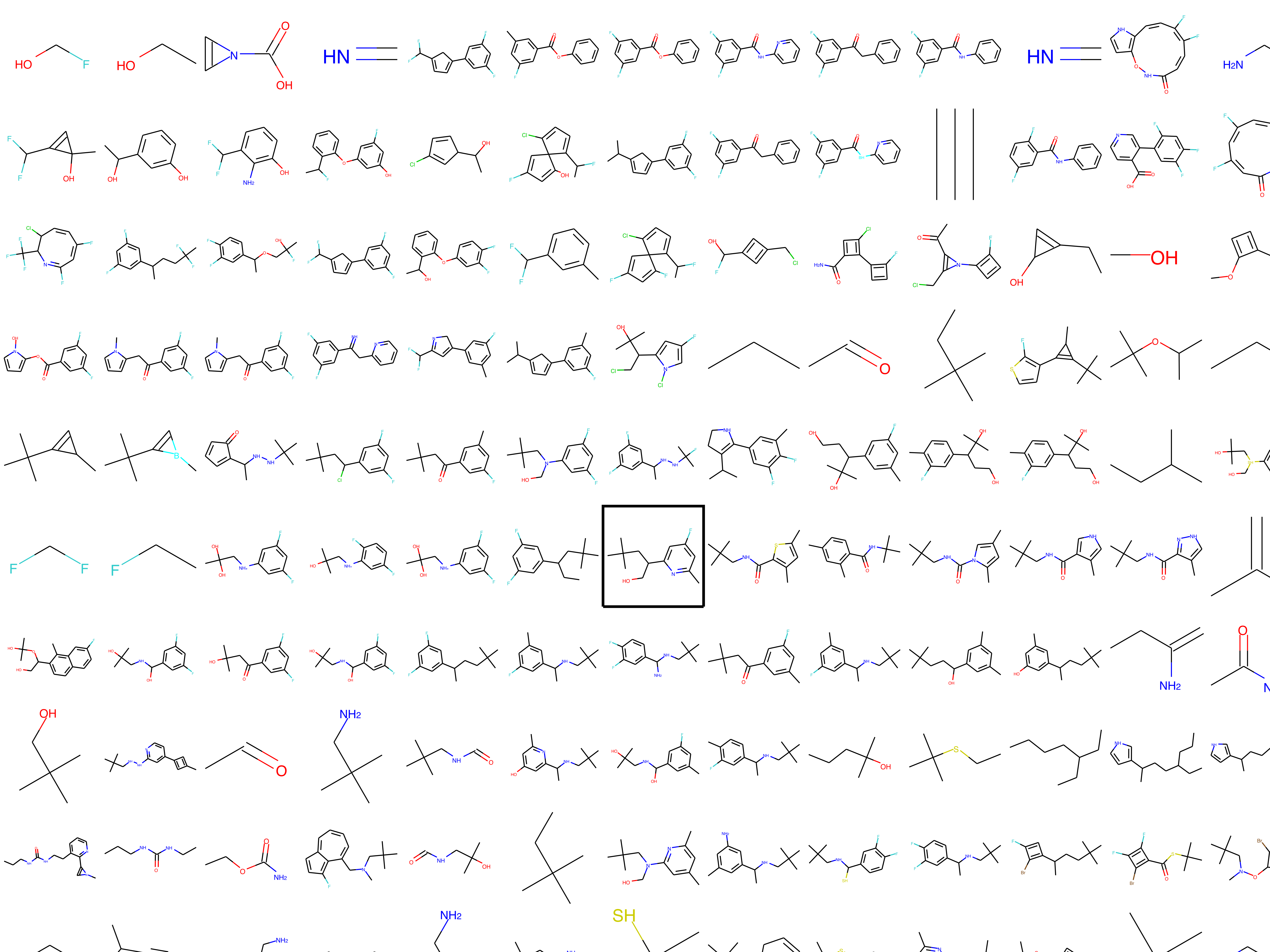
Molecules near 



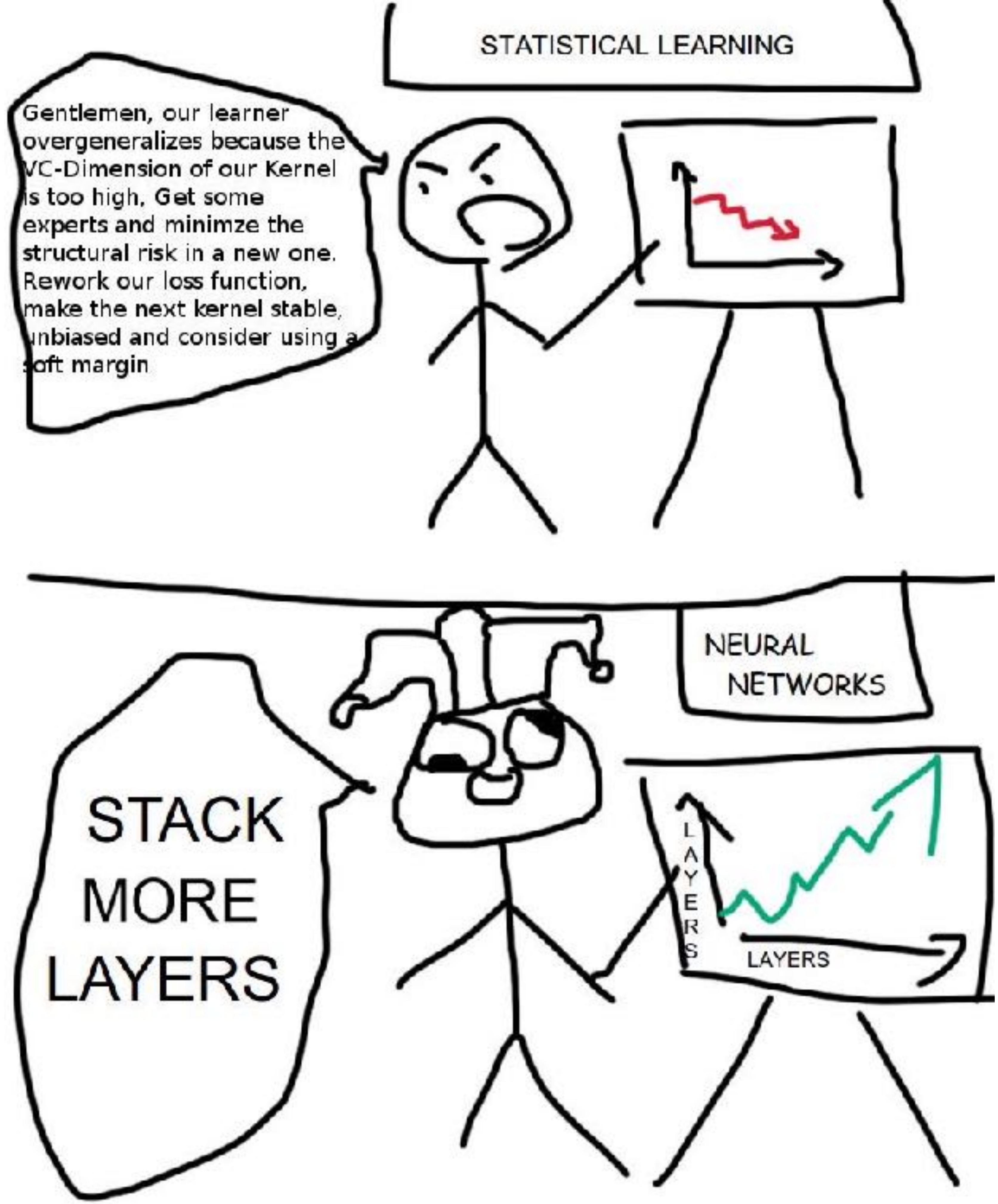


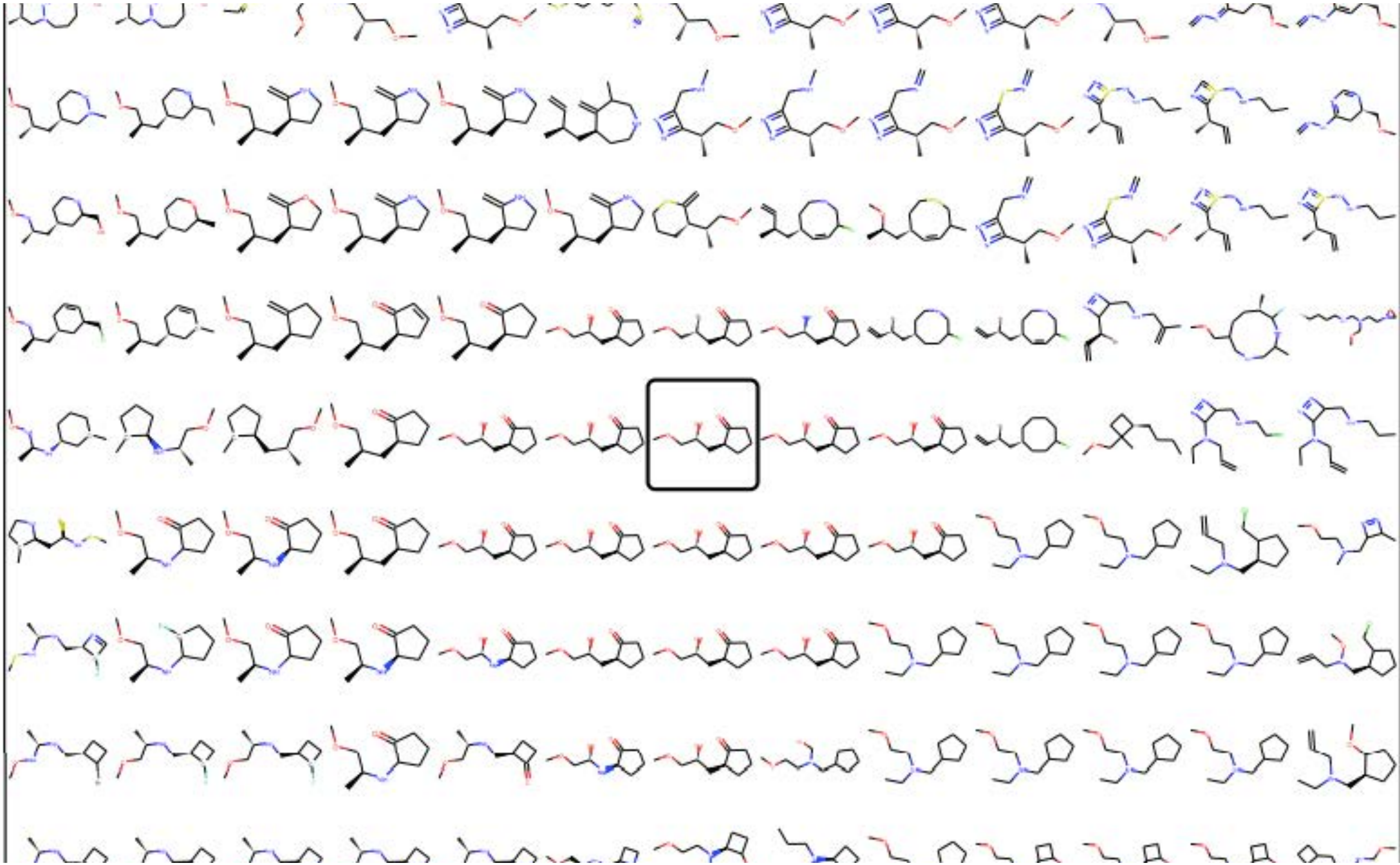






No chemistry-specific design!

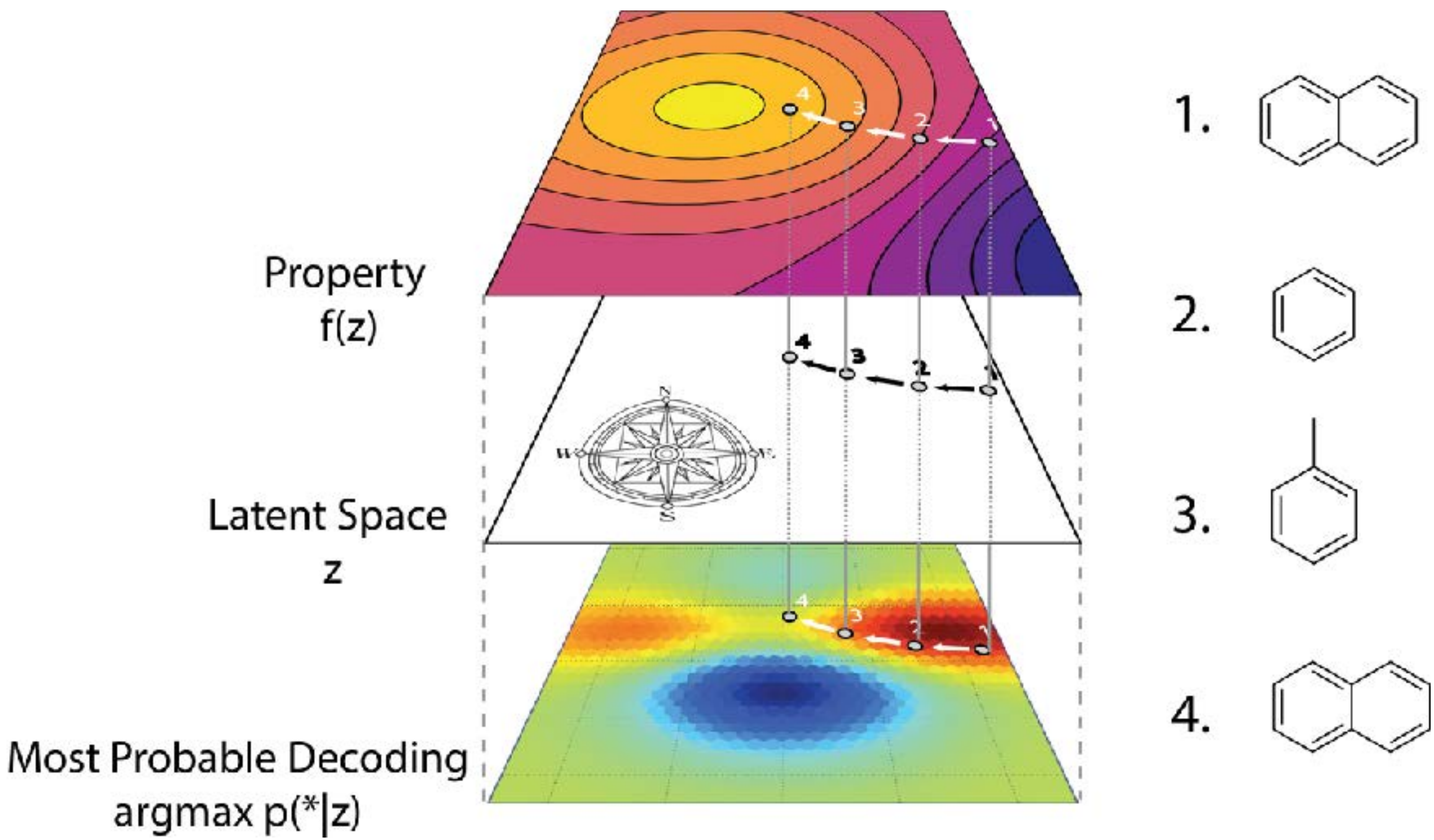




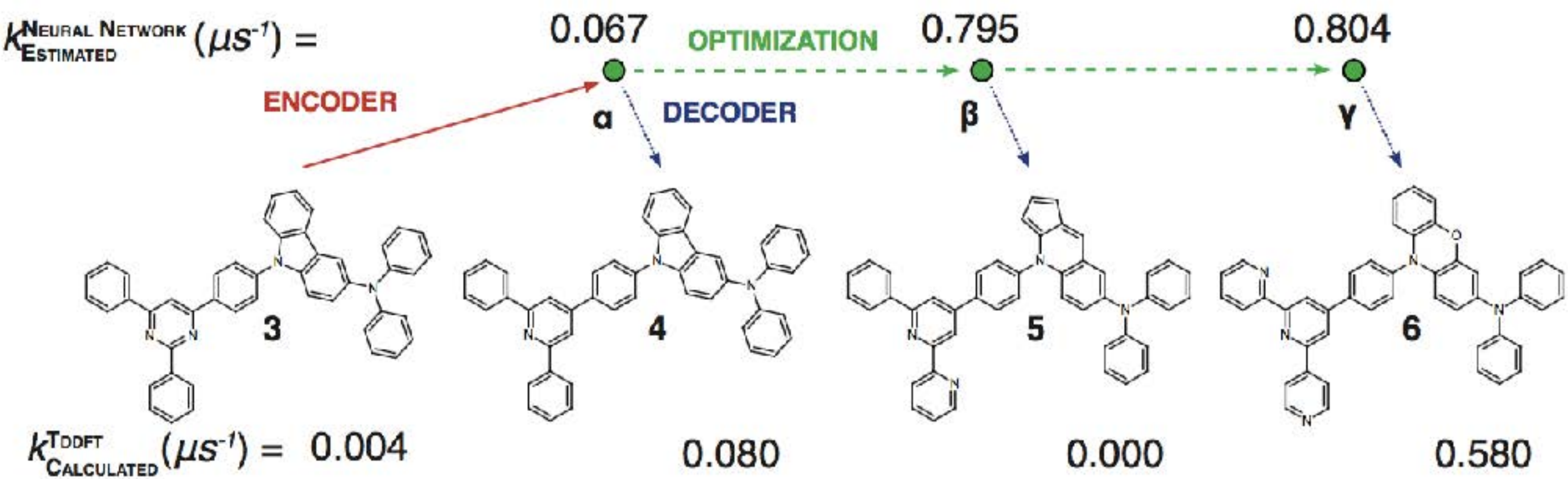
Grammar VAE

Matt Kusner, Brooks Paige, José Miguel Hernández-Lobato

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

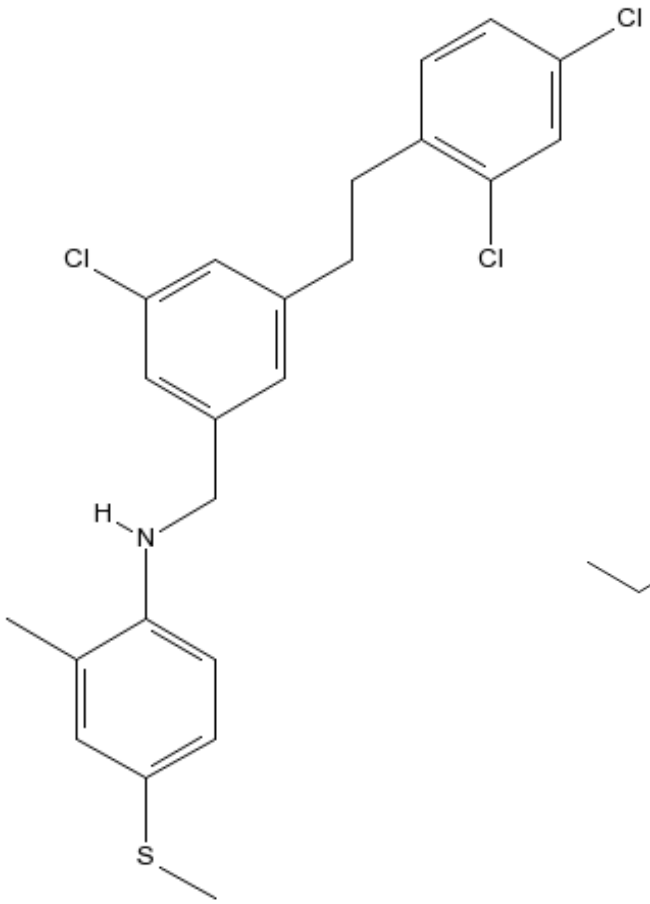
Be careful what you wish for

- Optimizing for solubility gave molecules with giant rings
- Needed to add hacky terms to objective
- Maybe not necessary, if there's downstream validation

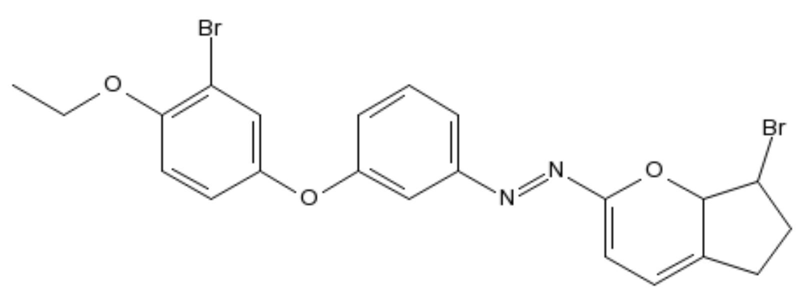
$$J^{\mathrm{logP}}(m) = \mathrm{logP}(m) - \mathrm{SA}(m) - \text{ring-penalty}(m)$$

Bayesian Optimization

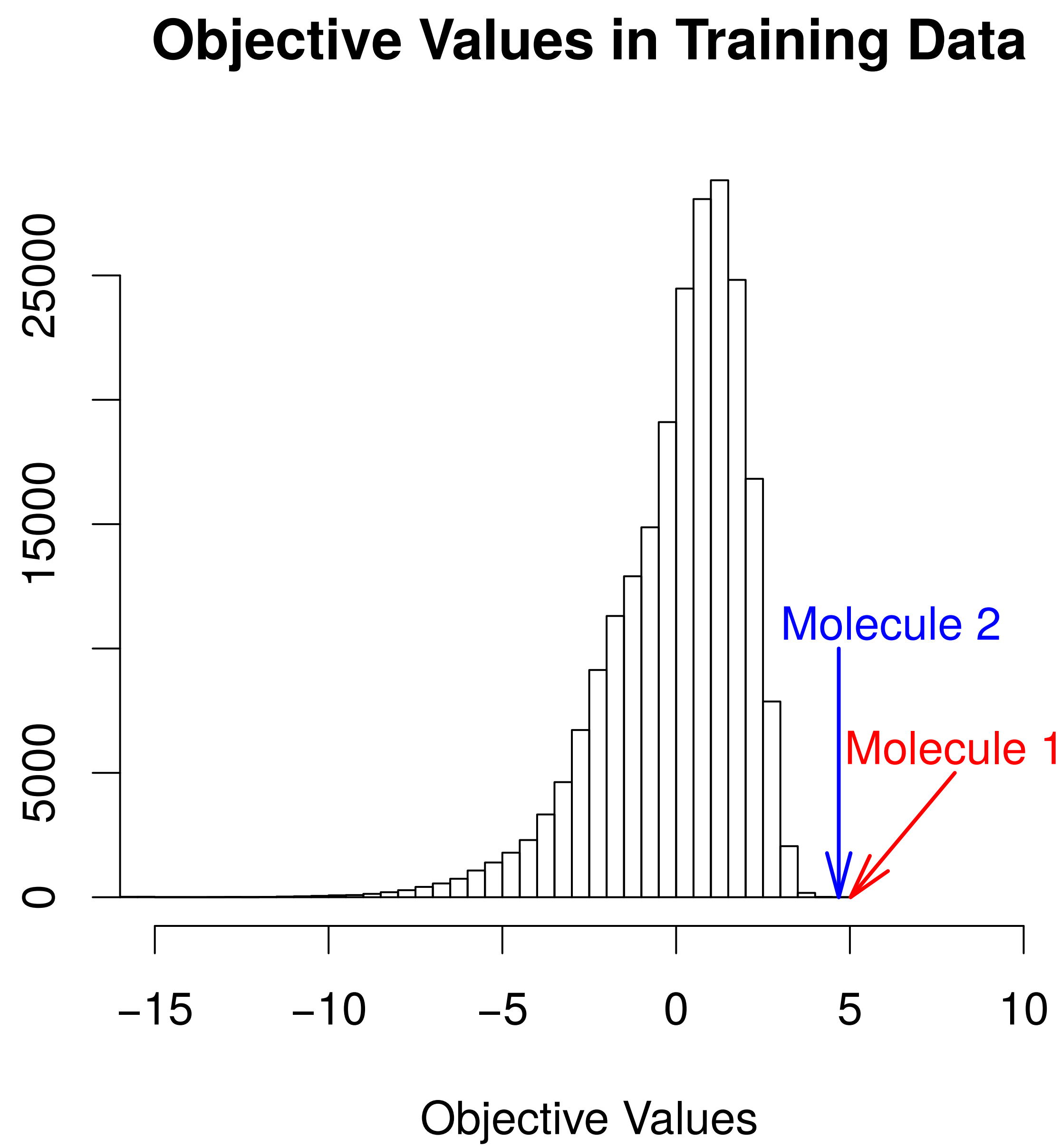
Sort of worked!



Molecule 1



Molecule 2



Science Translational Medicine

IN THE PIPELINE

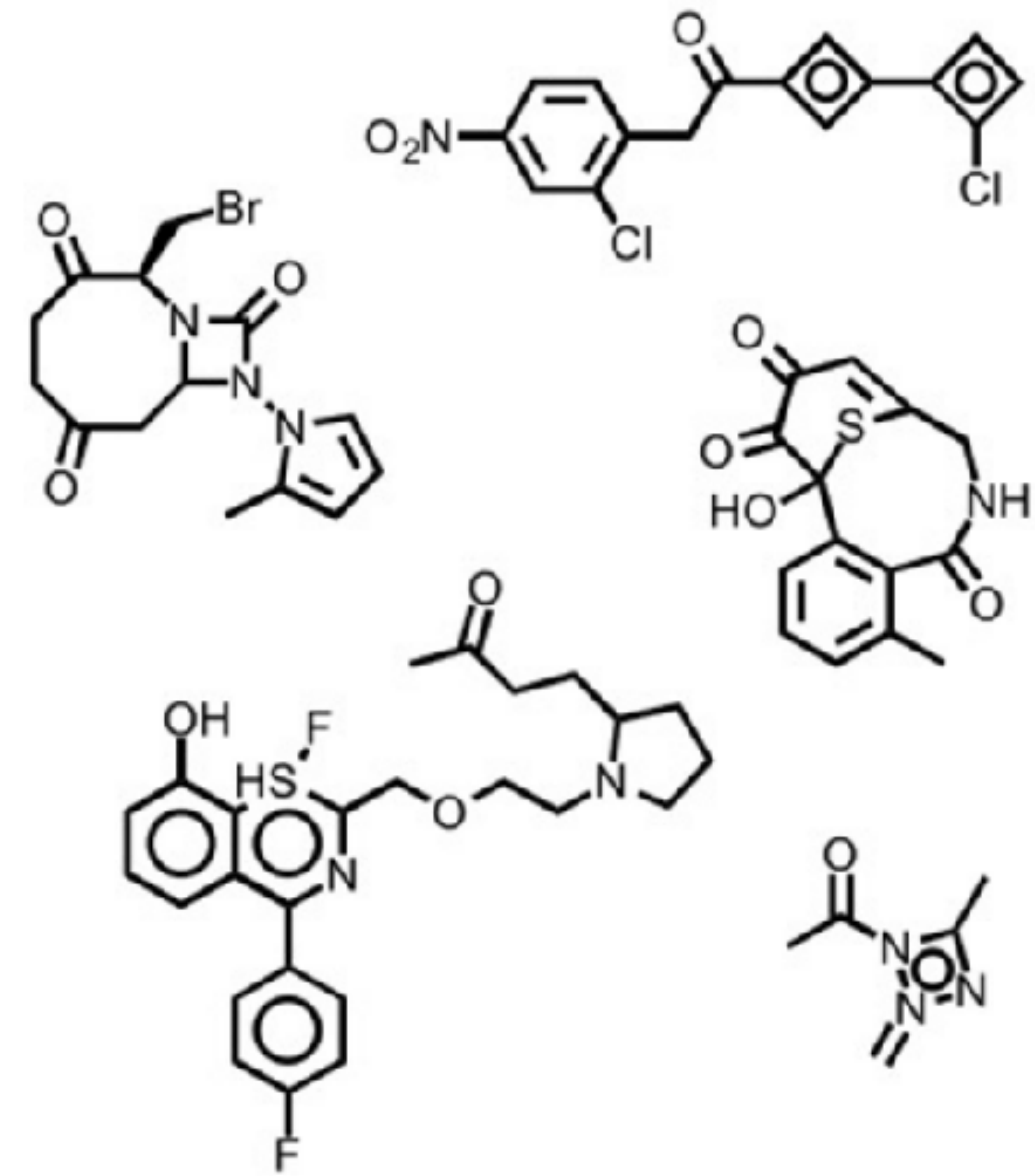
Derek Lowe's commentary on drug discovery and the pharma industry. An editorially independent blog from the publishers of *Science Translational Medicine*.

By Derek Lowe

Calculating A Few Too Many New Compounds

By Derek Lowe | November 8, 2016

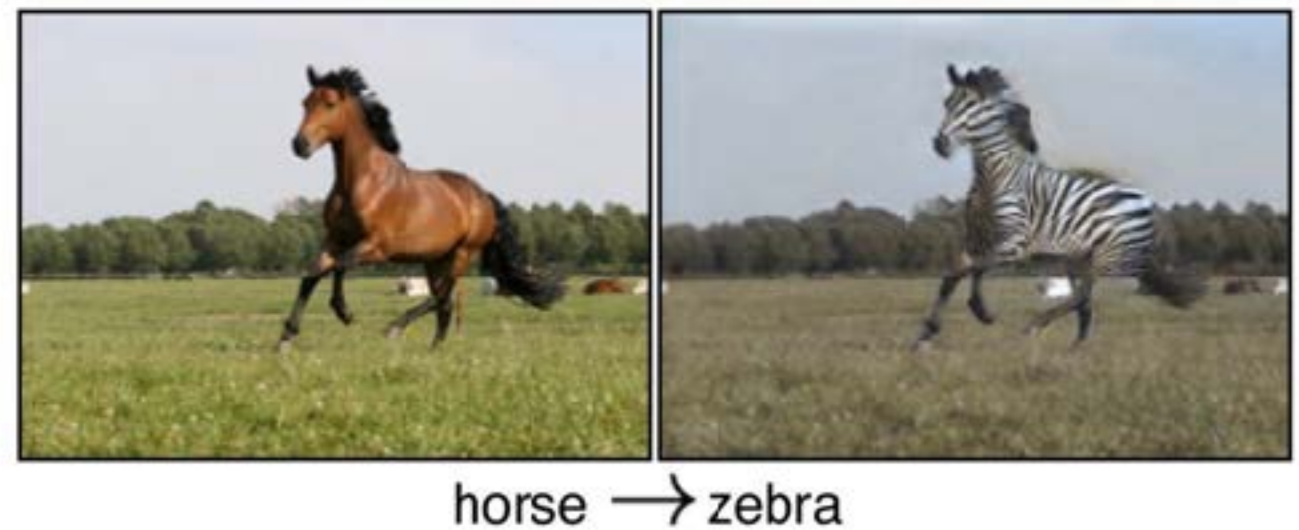
"No organic chemist could have looked at these without raising the alarm – this stuff is not, by many standards, publishable at all. When the authors do show this work to someone in the field, it will not go well. In fact, this blog post is an example of just such an encounter, and no, it isn't going well."

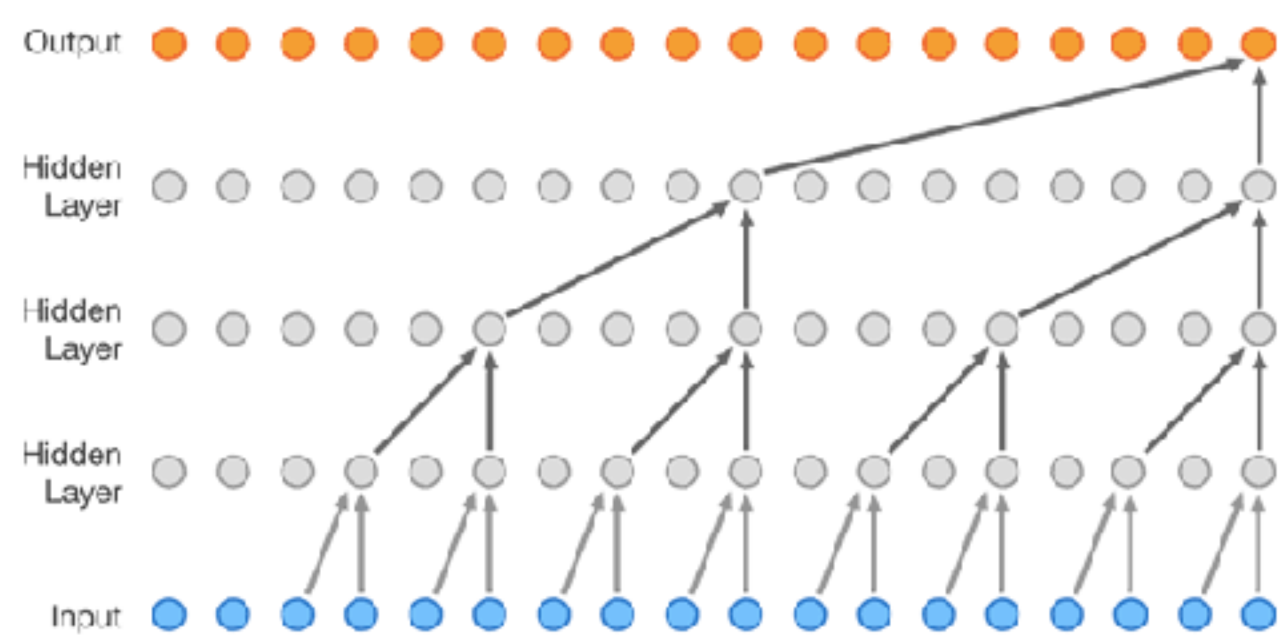


Frontiers

What recently became easy in machine learning?

- Training continuous latent-variable models (VAEs, GANs) to produce large images
- Training large supervised models with fixed architectures
- Building RNNs that can output grid-structured objects (images, waveforms)





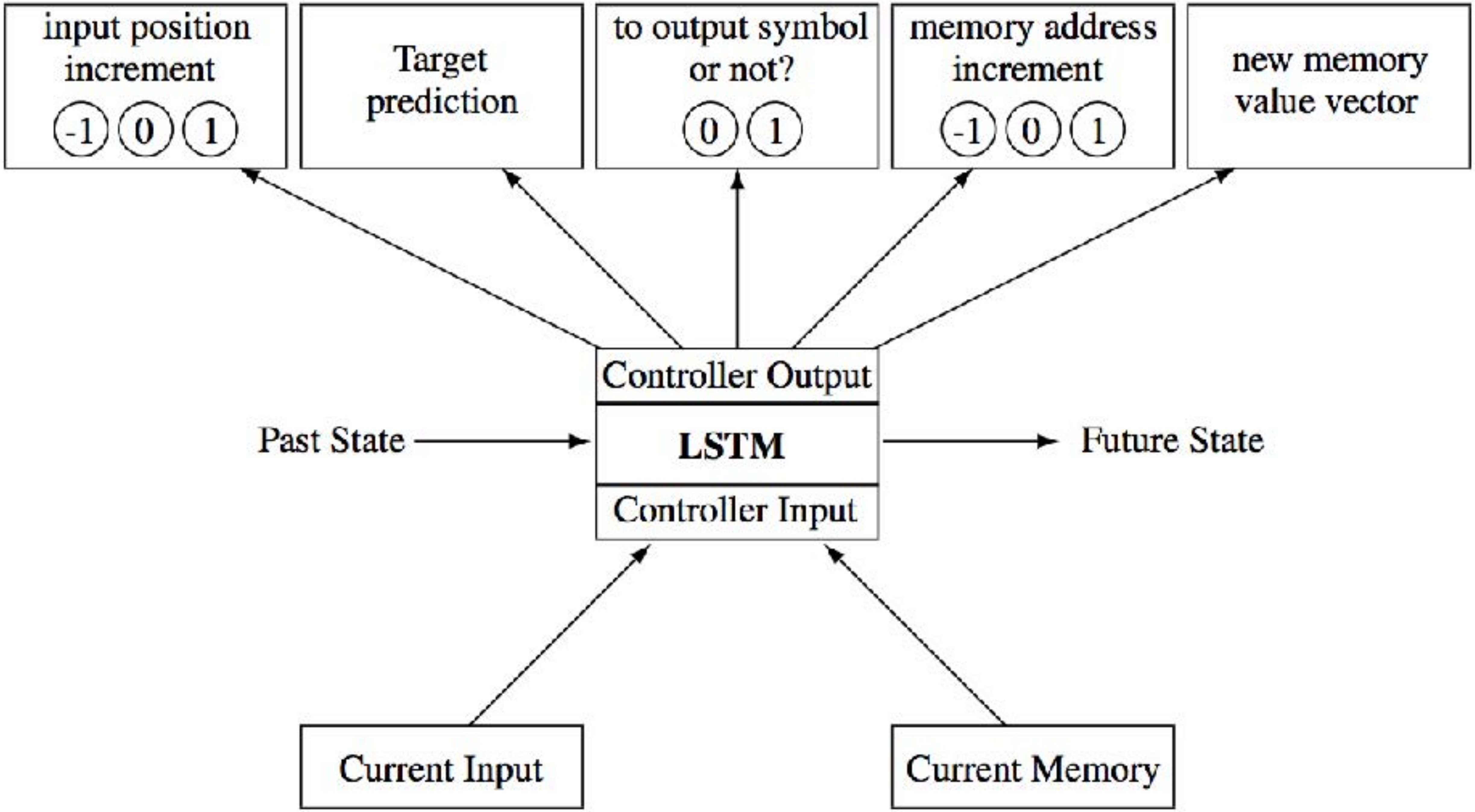
What is still hard?

- Training GANs to generate text
- Training VAEs with discrete latent variables
- Training agents to communicate with each other using words
- Training agent or programs to decide which discrete action to take.
- Training generative models of structured objects of arbitrary size, like programs, graphs, or large texts.

Level	Model	PTB	CMU-SE
Word	LSTM	what everything they take everything away from .	<s>will you have two moment ? </s>
		may tea bill is the best chocolate from emergency .	<s>i need to understand deposit length . </s>
		can you show show if any fish left inside .	<s>how is the another headache ? </s>
		room service , have my dinner please .	<s>how there , is the restaurant popular this cheese ? </s>
	CNN	meanwhile henderson said that it has to bounce for.	<s>i 'd like to fax a newspaper . </s>
		I'm at the missouri burning the indexing manufacturing and through .	<s>cruise pay the next in my replacement . </s>
			<s>what 's in the friday food ? ? </s>

Table 4: Word level generations on the Penn Treebank and CMU-SE datasets

Adversarial Generation of Natural Language.
Sai Rajeswar, Sandeep Subramanian, Francis Dutil,
Christopher Pal, Aaron Courville, 2017



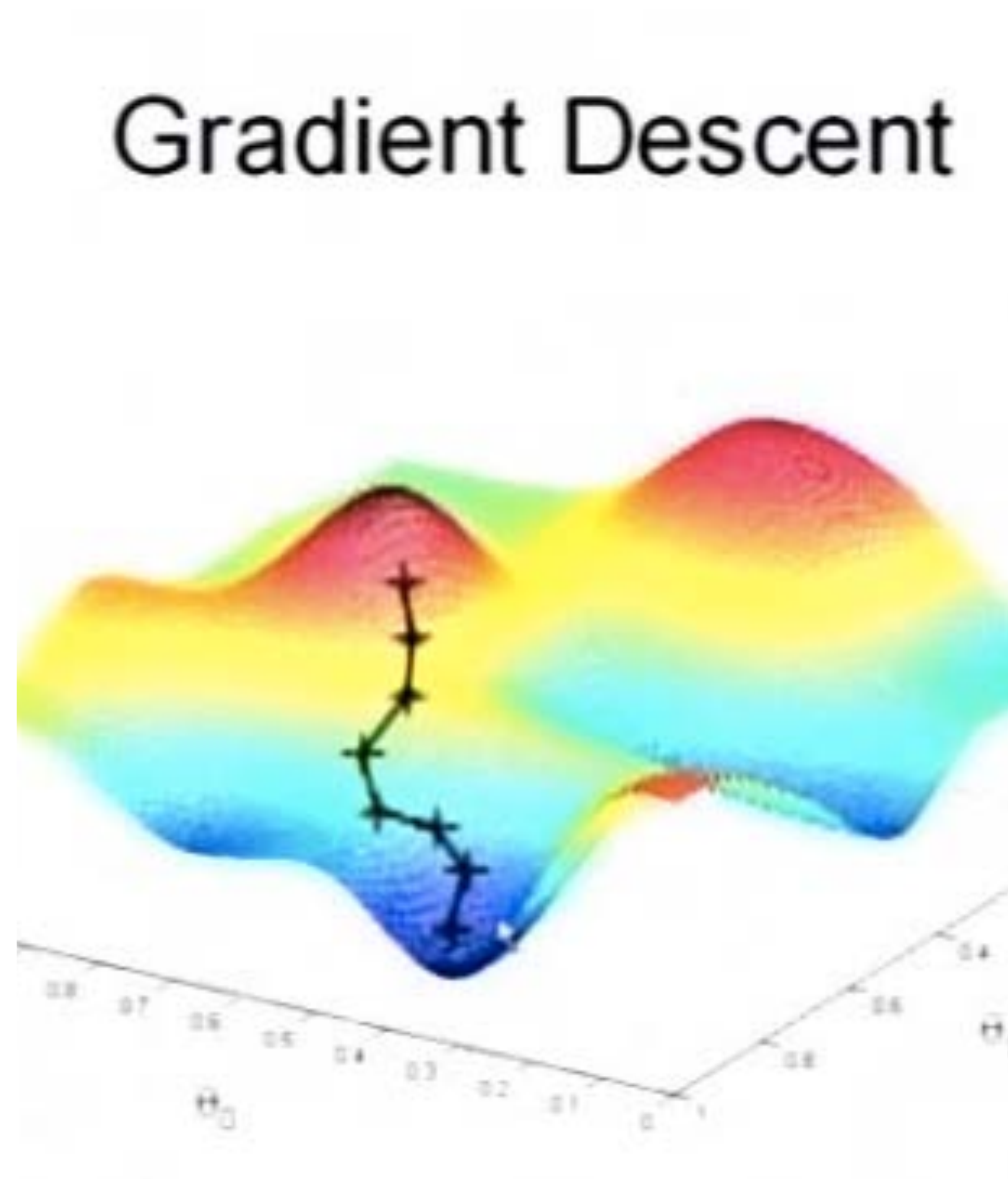
"We successfully trained the RL-NTM to solve a number of algorithmic tasks that are simpler than the ones solvable by the fully differentiable NTM."

Reinforcement Learning Neural Turing Machines

Wojciech Zaremba, Ilya Sutskever, 2015

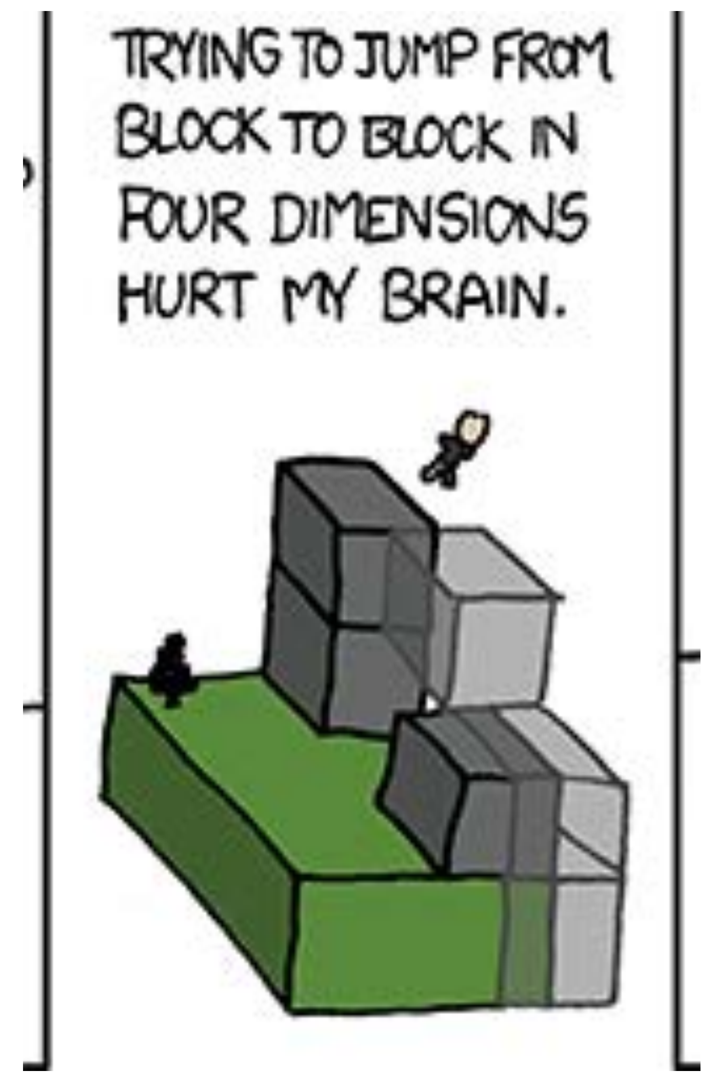
Why are the easy things easy?

- Gradients give more information the more parameters you have
- Backprop (reverse-mode AD) only takes about as long as the original function
- Local optima less of a problem than you think



Why are the hard things hard?

- Discrete structure means we can't use backprop to get gradients
- No cheap gradients means that we don't know which direction to move to improve
- Not using our knowledge of the structure of the function being optimized
- Becomes as hard as optimizing a black-box function



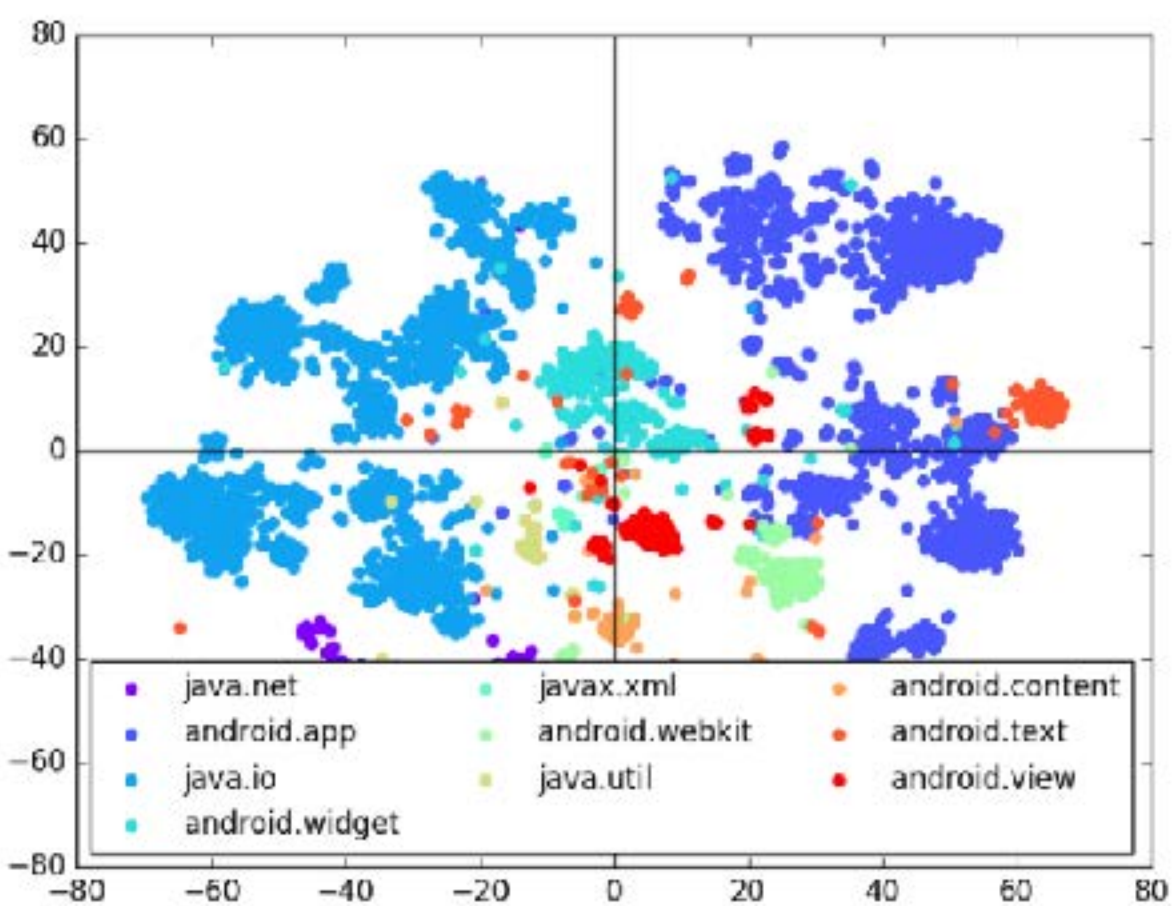
```
String s;
BufferedReader br;
FileReader fr;
try {
 fr = new FileReader($String);
 br = new BufferedReader(fr);
 while ((s = br.readLine()) != null) {}
 br.close();
} catch (FileNotFoundException _e) {
  _e.printStackTrace();
} catch (IOException _e) {
  _e.printStackTrace();
}
```

(a)

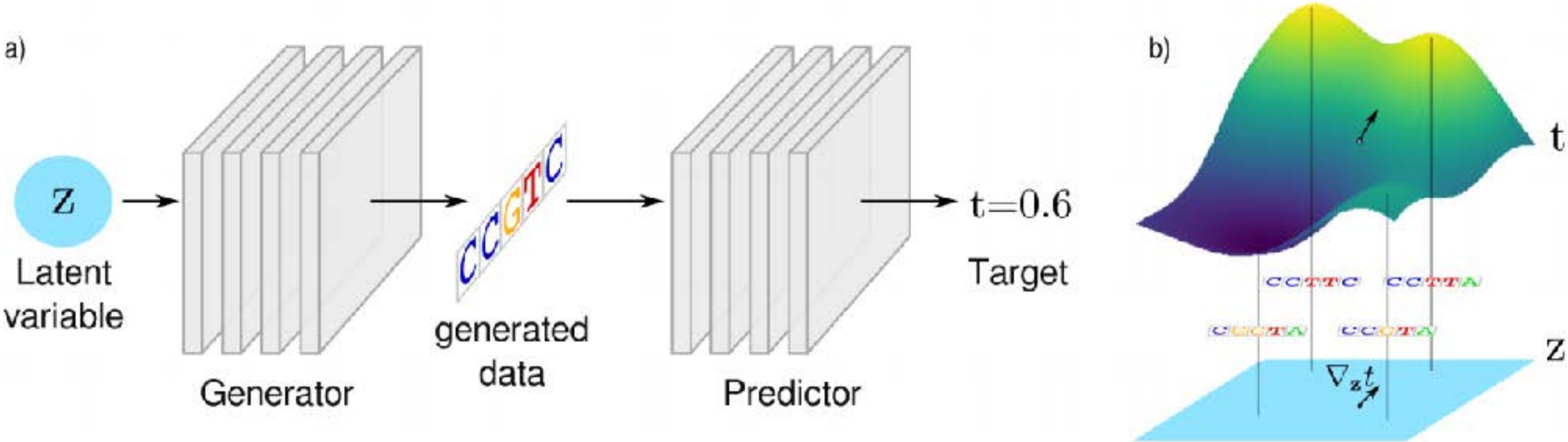
```
String s;
BufferedReader br;
FileReader fr;
try {
 fr = new FileReader($File);
 br = new BufferedReader(fr);
 while ((s = br.readLine()) != null){}
 br.close();
} catch (FileNotFoundException _e){
} catch (IOException _e){
}
```

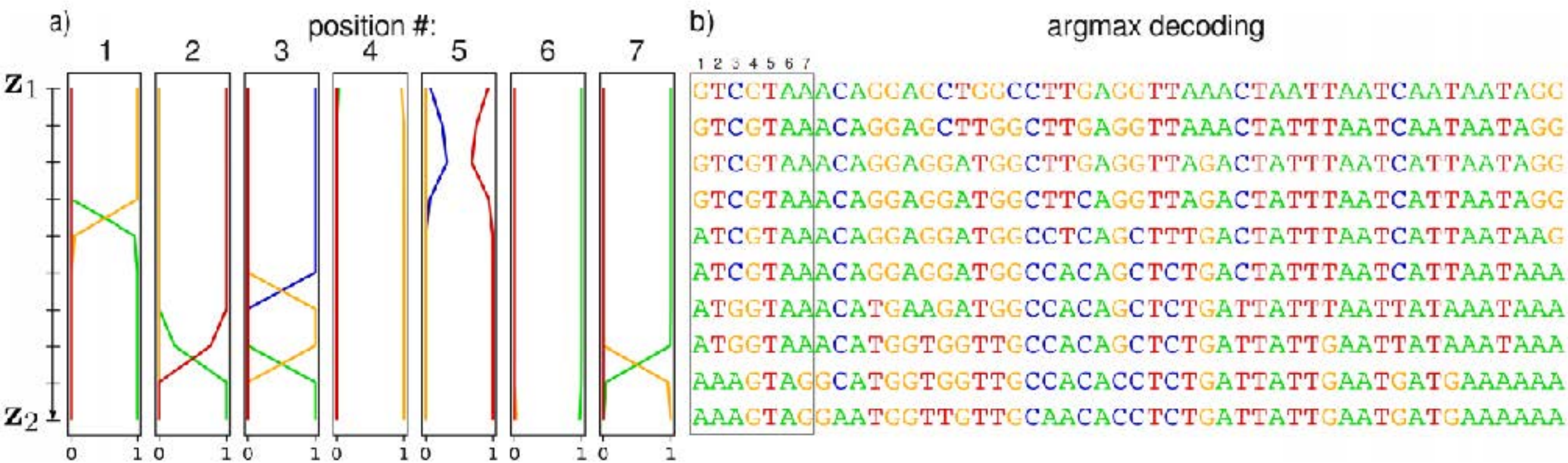
(b)

Figure 7: Programs generated in a typical run of BAYOU, given the API method name `readLine` and the type `FileReader`.

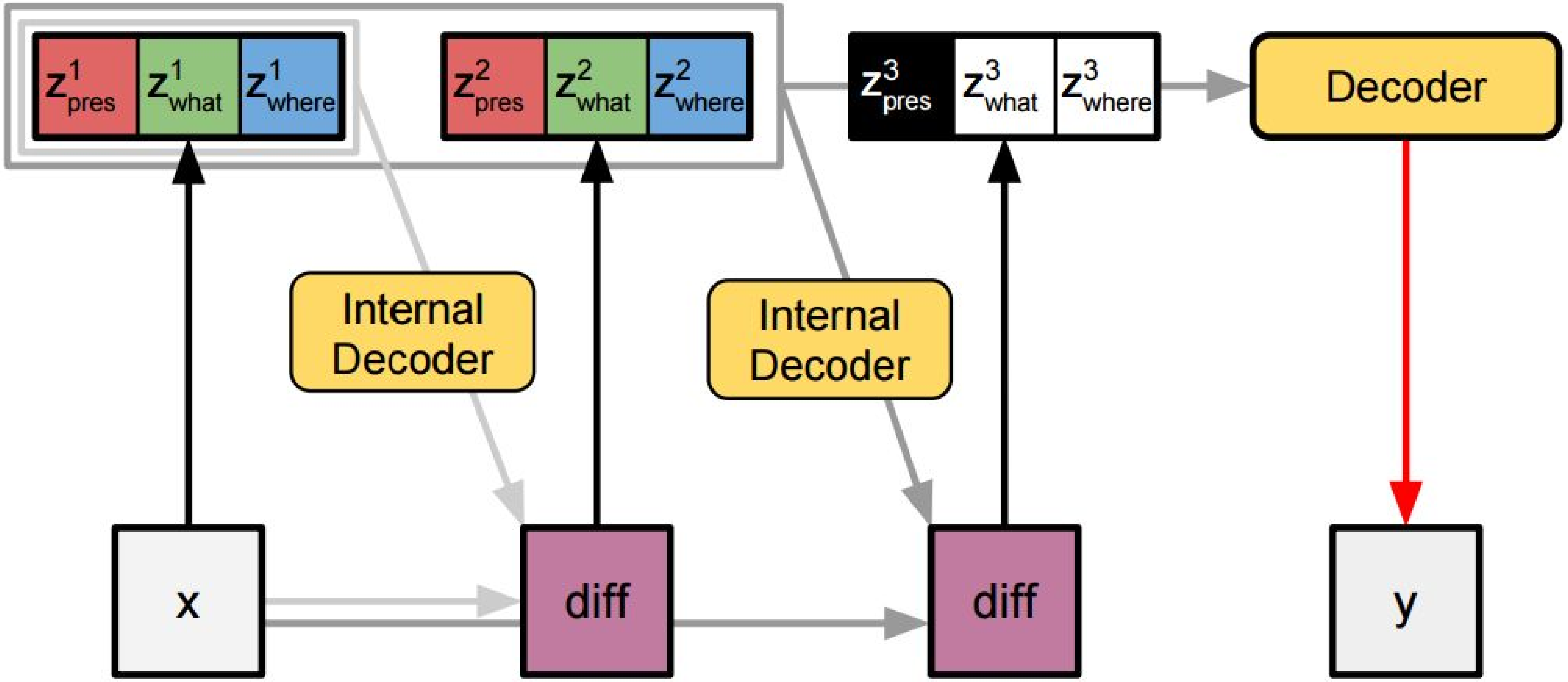


Neural Sketch Learning for Conditional Program Generation, ICLR 2018 submission





Generating and designing DNA with deep generative models. Killoran, Lee, Delong, Duvenaud, Frey, 2017



Attend, Infer, Repeat: Fast Scene Understanding with Generative Models

S.M. Eslami,N. Heess, T. Weber, Y. Tassa, D. Szepesvari, K.Kavukcuoglu, G. E. Hinton

History of Generative Models

- **1940s - 1960s** Motivating probability and Bayesian inference
- **1980s - 2000s** Bayesian machine learning with MCMC
- **1990s - 2000s** Graphical models with exact inference
- **1990s** - **2015** Bayesian Nonparametrics with MCMC (Indian Buffet process, Chinese restaurant process)
- **1990s - 2000s** Bayesian ML with mean-field variational inference
- **1995 -1996** Helmholtz machine, wake-sleep (*almost* invented variational autoencoders)
- **2000s - 2013** Deep undirected graphical models (RBMs, pretraining)
- **2000s** - **2013** Autoencoders, denoising autoencoders

Modern Generative Models

- **2000s -** Probabilistic Programming
- **2000s -** Invertible density estimation
- **2010 -** Stan - Bayesian Data Analysis with HMC
- **2013 -** Variational autoencoders, reparamaterization trick becomes widely known
- **2014 -** Generative adversarial nets
- **2015 -** Deep reinforcement learning
- **2016 -** New gradient estimators (muprop, Q-prop, concrete + Gumbel-softmax, REBAR, RELAX)

Other Frontiers

- Generating long action-conditional video
- Modeling uncertainty in the generative process
- Coherent multi-scale models

- Ultimate application: data-efficient model-based RL
- Expected utility framework separates modeling from decision-making

Takeaways

- Different approaches to generative modeling have different tradeoffs.
  - GANs pay high cost at training time, flexible and cheap sampling at test time
- Simple components form a composable language of models.
- Watch out for reinventing Bayes' rule. Approximating the optimal provides a lot of guidance.

Thanks!