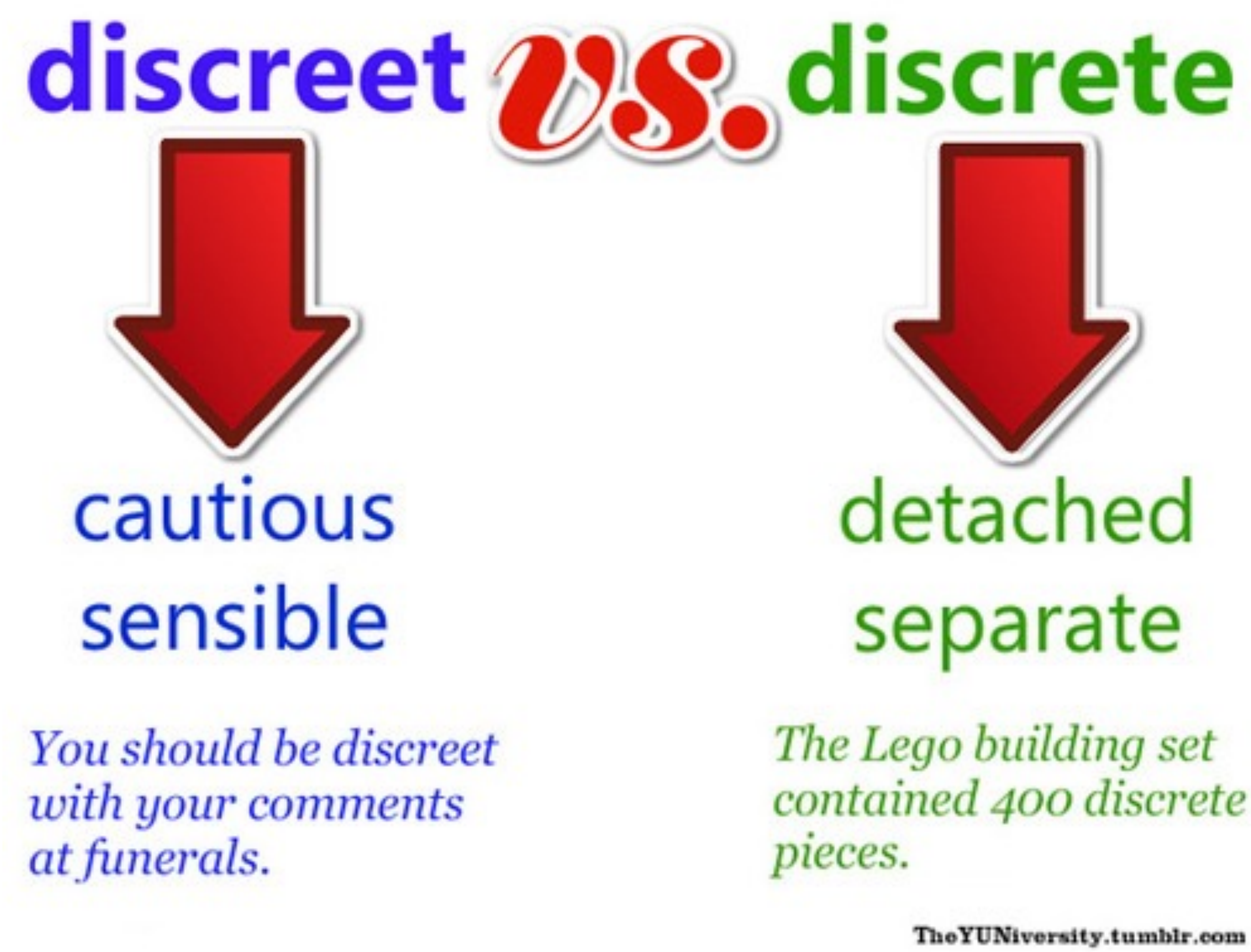
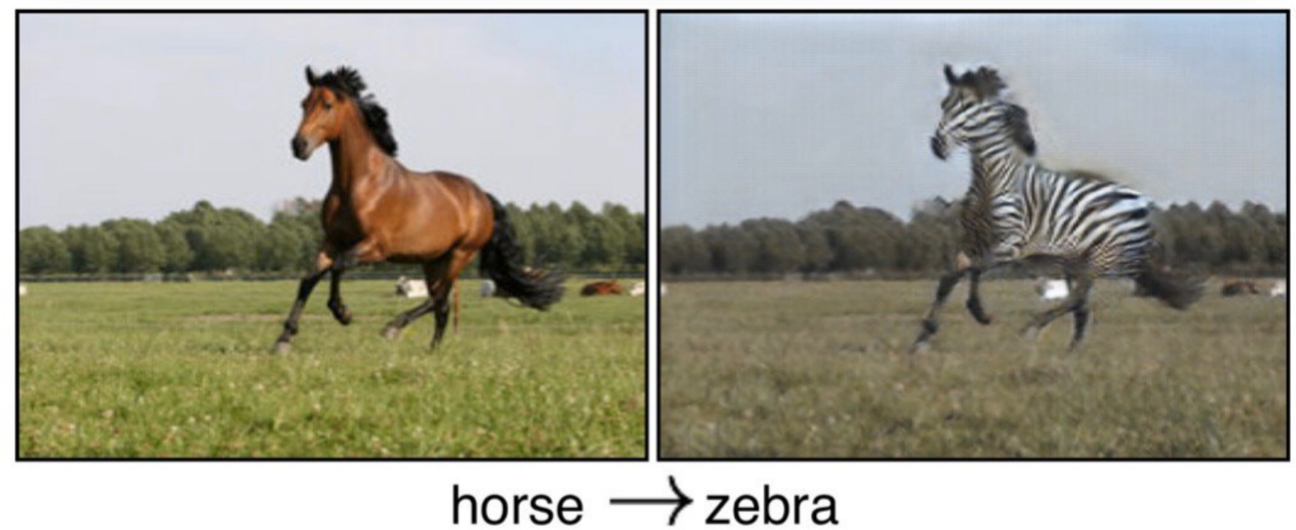
# STA 4273 / CSC 2547
# Spring 2018

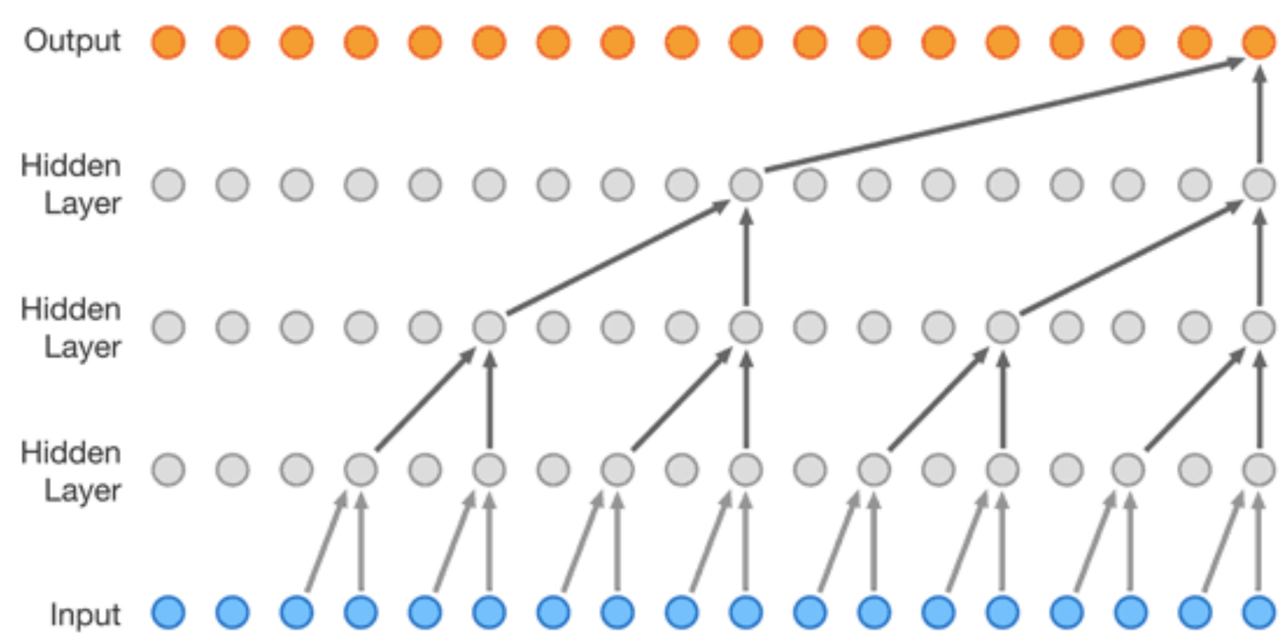
# Learning Discrete Latent Structure



# 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)



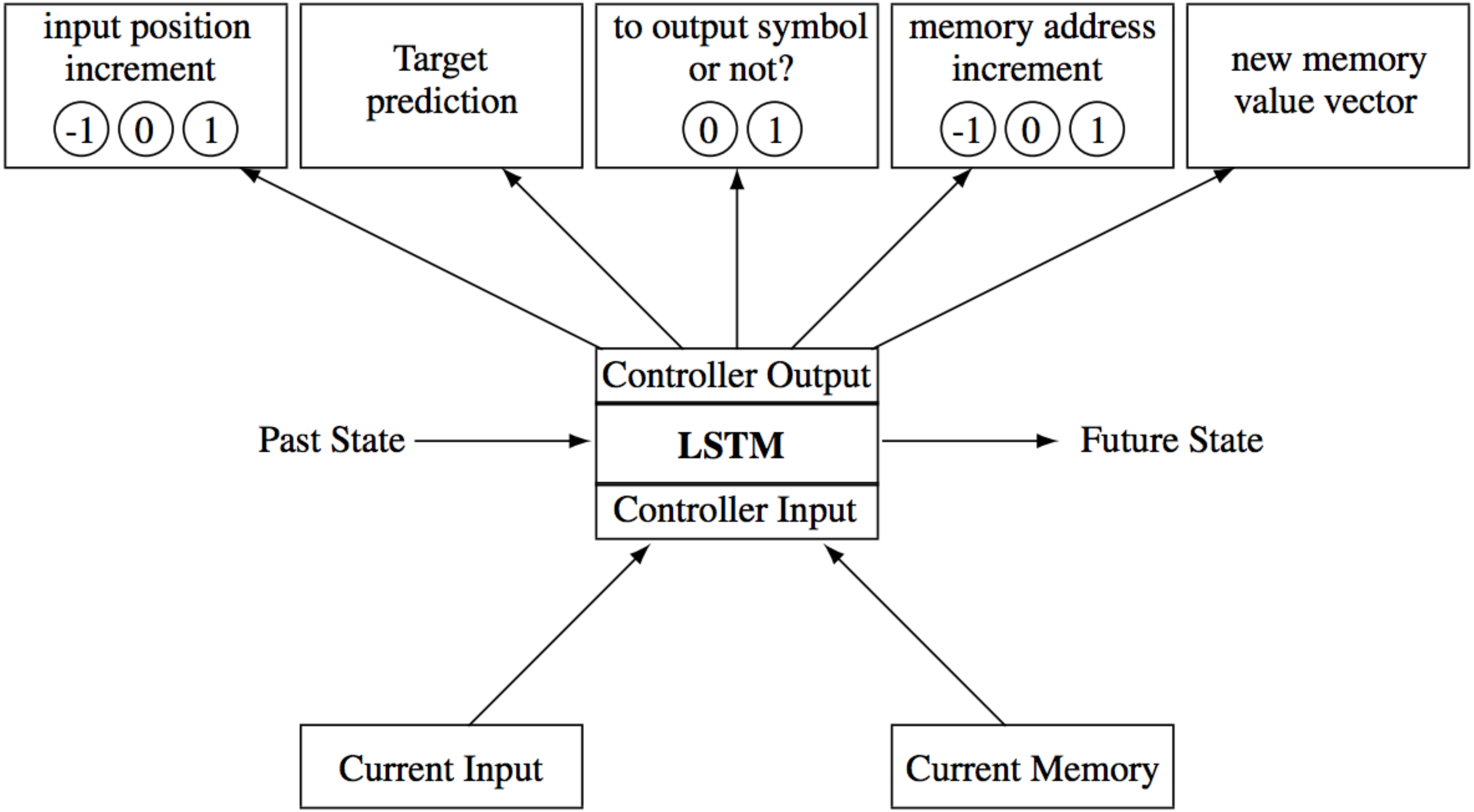
# What is still hard?

- 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

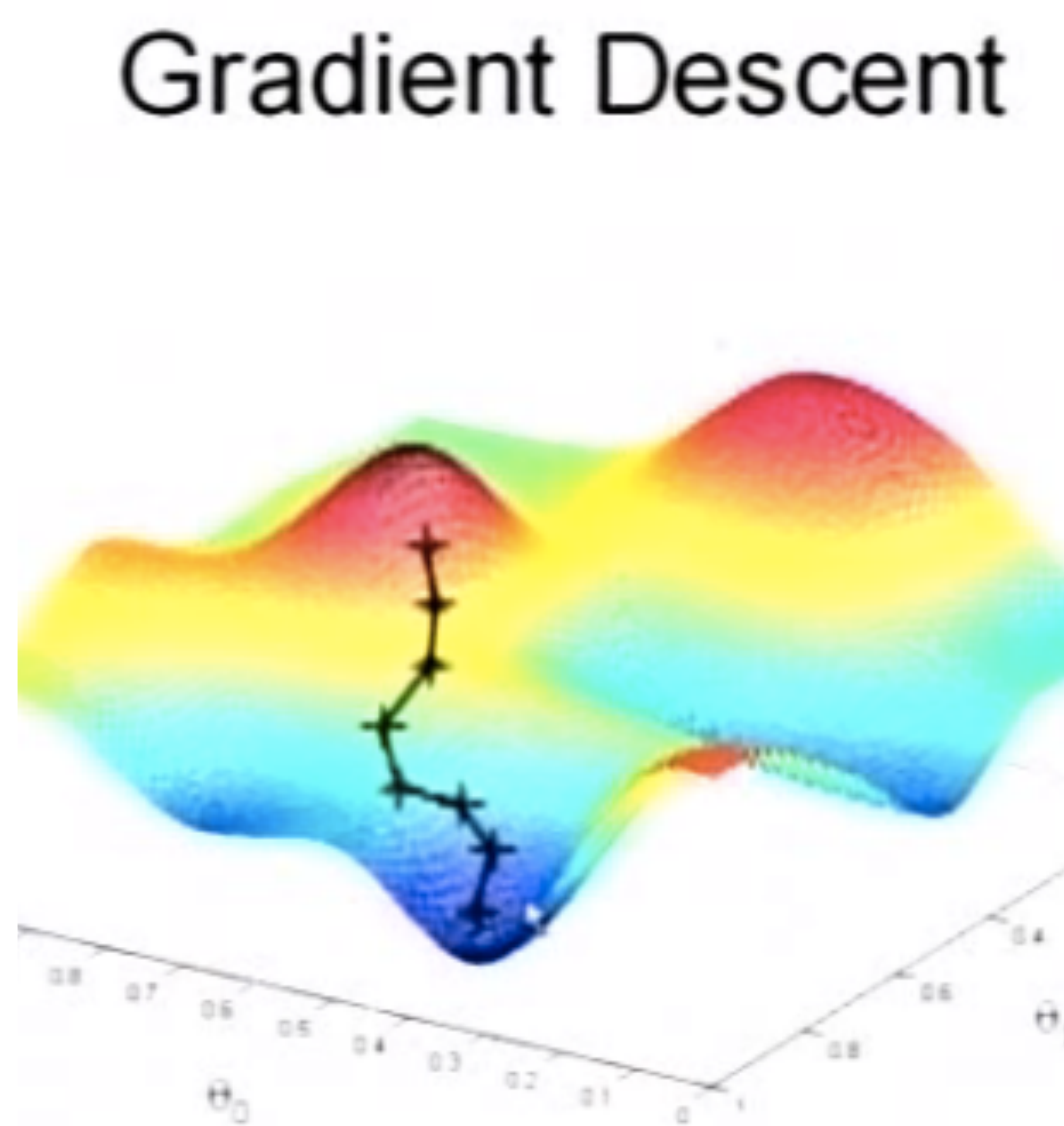
Adversarial Generation of Natural Language.
Sai Rajeswar, Sandeep Subramanian, Francis Dutil,
Christopher Pal, Aaron Courville, 2017




"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

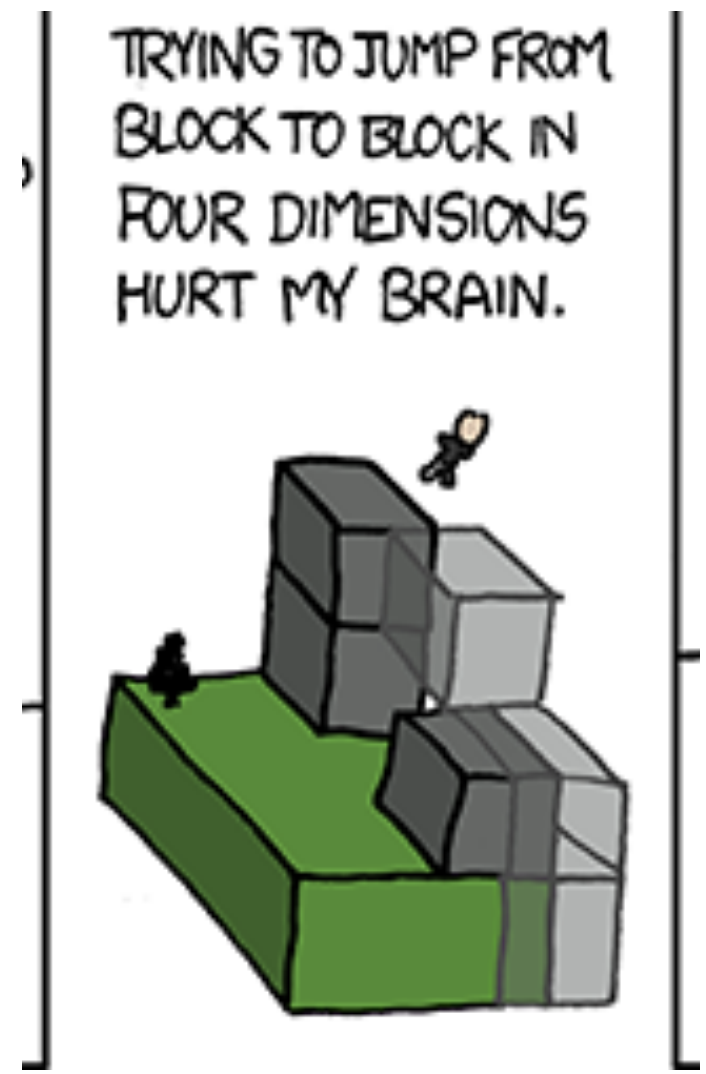
# Why are the easy things easy?

- 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 backdrop 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



# This course:
# How can we optimize anyways?

- This course is about how to optimize or integrate out parameters even when we don't have backprop

- And, what could we do if we knew how? Discover models, learn algorithms, choose architectures

- Not necessarily the same as discrete optimization - we often want to optimize continuous parameters that might be used to make discrete choices.

- Focus will be on gradient estimators that use some structure of the function being optimized, but lots doesn't fit in this framework. Also, want automatic methods (no GAs)

# Things we can do with learned discrete structures

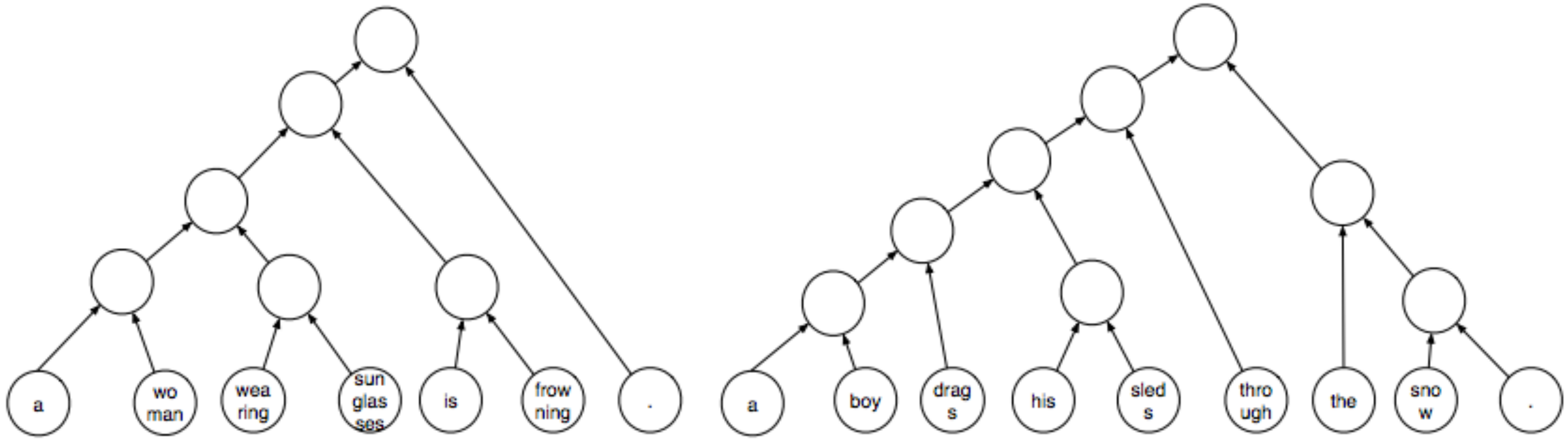



Figure 2: Examples of tree structures learned by our model which show that the model discovers simple concepts such as noun phrases and verb phrases.

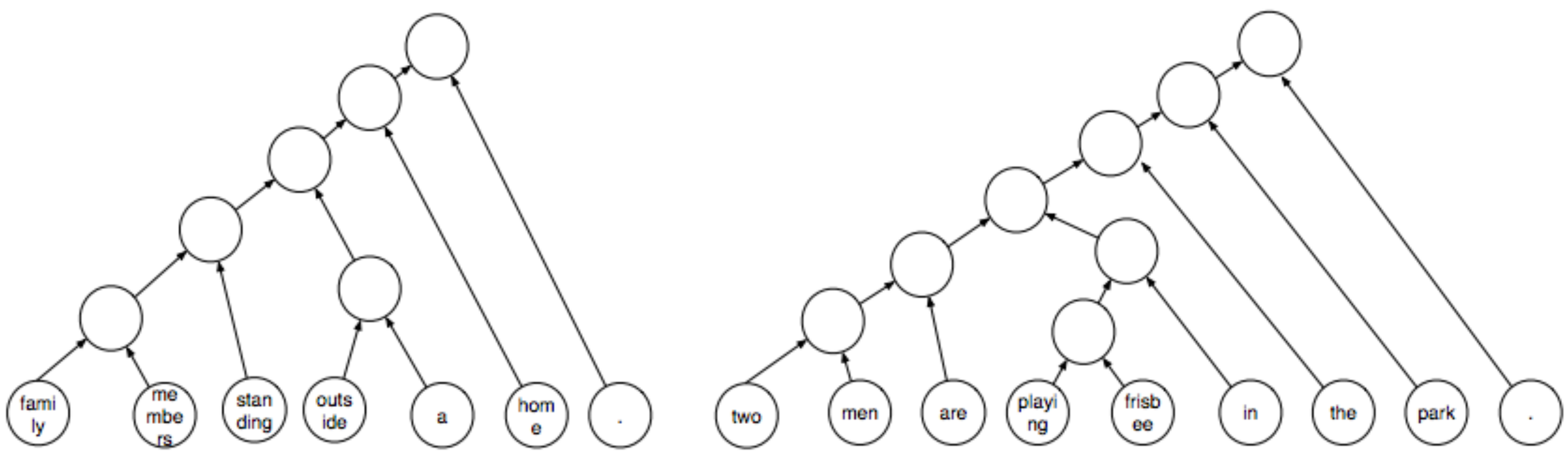



Figure 3: Examples of unconventional tree structures.

Learning to Compose Words into Sentences with Reinforcement Learning
Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, Wang Ling, 2016

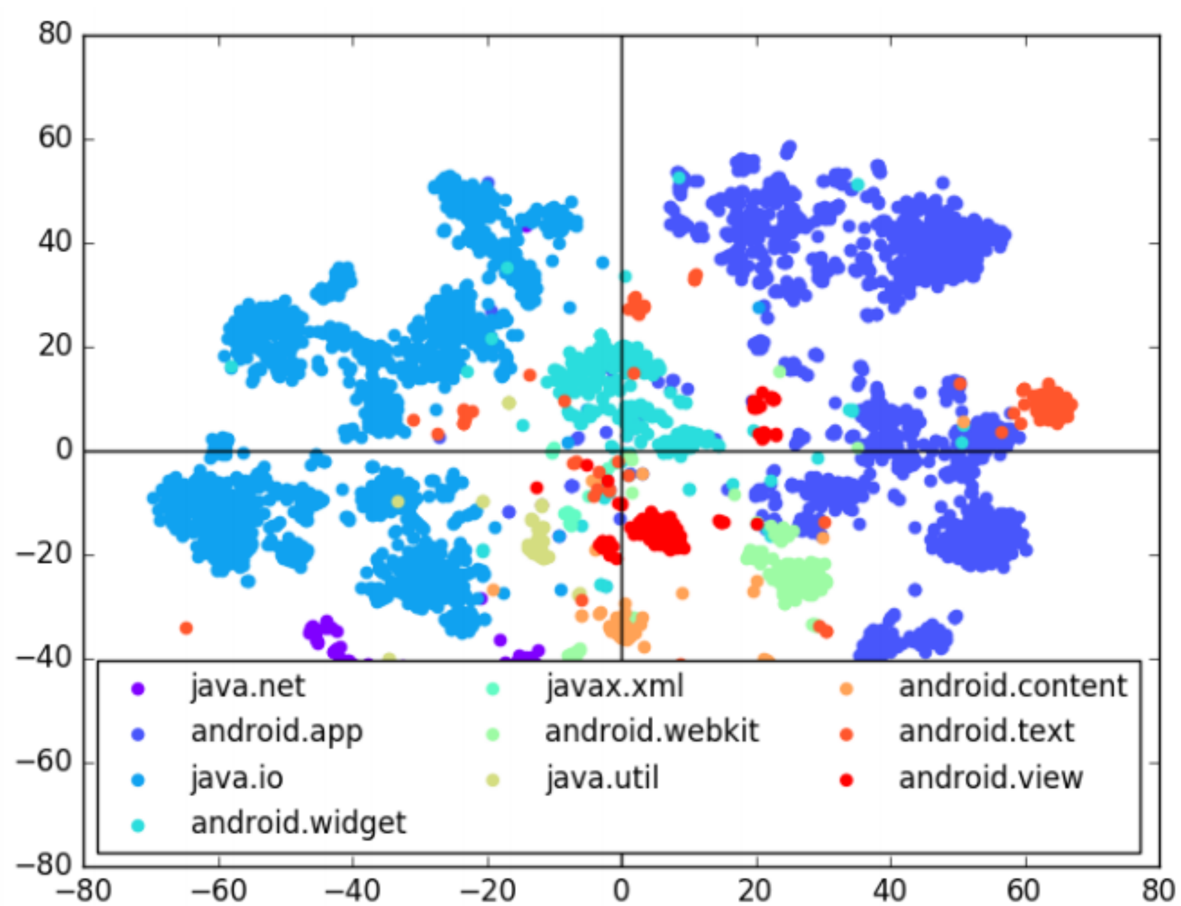
```
String s;
BufferedReader br;
FileReader fr;
try {
 fr = new FileReader($String);
 br = new BufferedReader(fr);
 while ((s = br.readLine()) != null) {}
 br.close();
} catch (FileNotFoundException _e) {
  _e.printStackTrace();
} catch (IOException _e) {
  _e.printStackTrace();
}
```

(a)

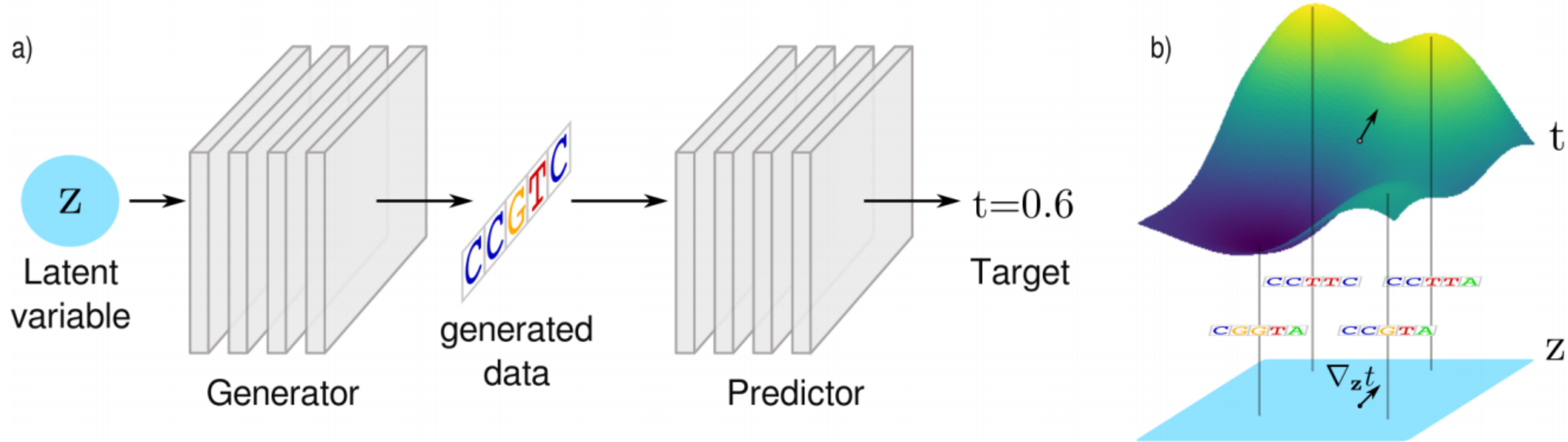
```
String s;
BufferedReader br;
FileReader fr;
try {
 fr = new FileReader($File);
 br = new BufferedReader(fr);
 while ((s = br.readLine()) != null){}
 br.close();
} catch (FileNotFoundException _e){
} catch (IOException _e){
}
```

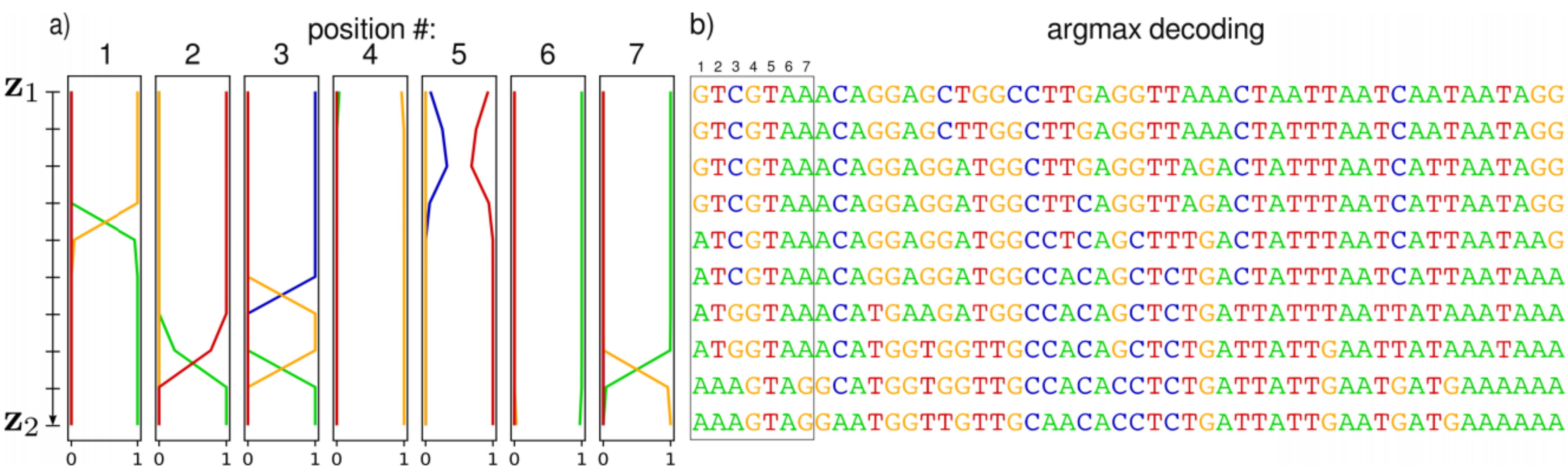
(b)

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

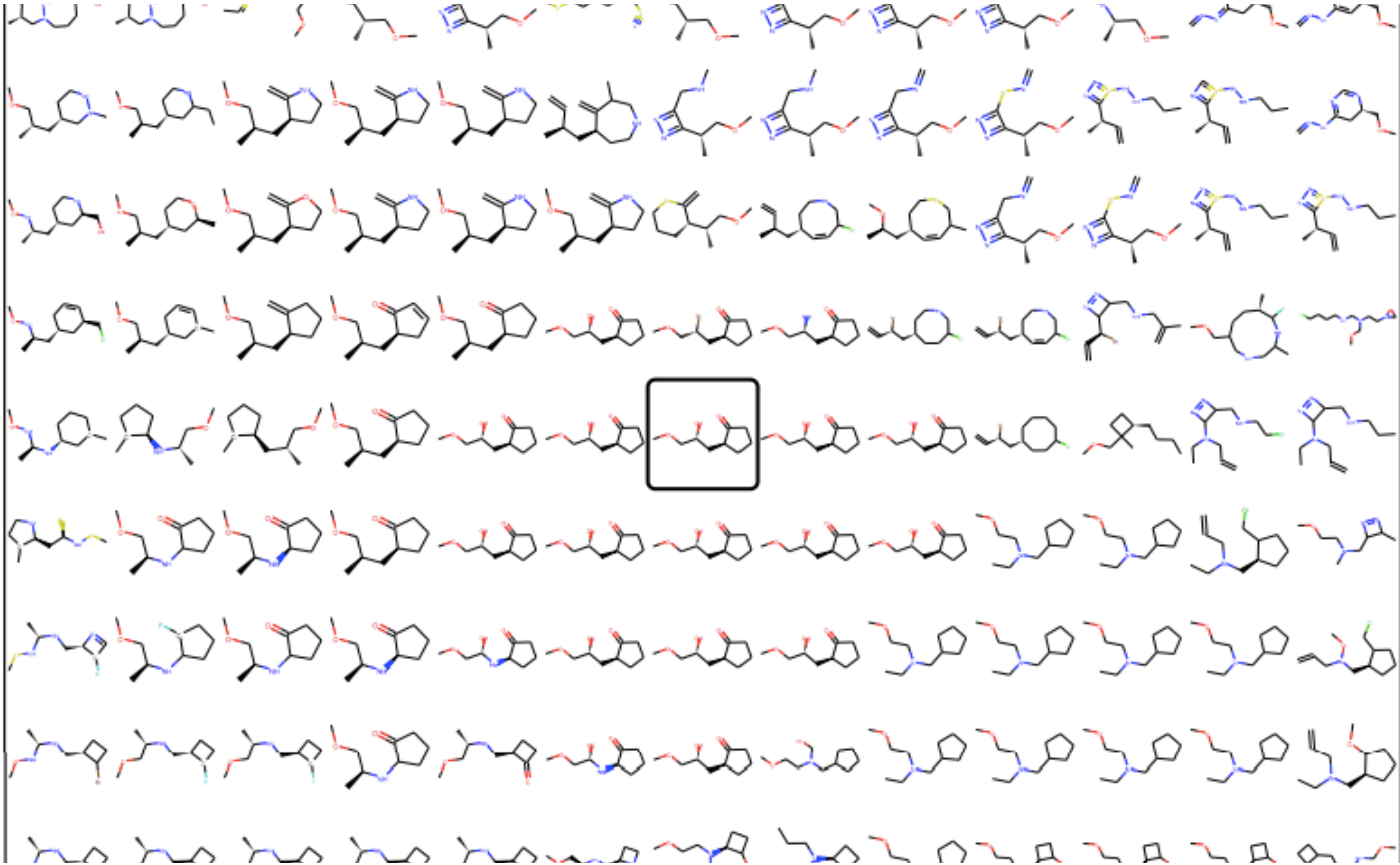



Neural Sketch Learning for Conditional Program Generation, ICLR 2018 submission



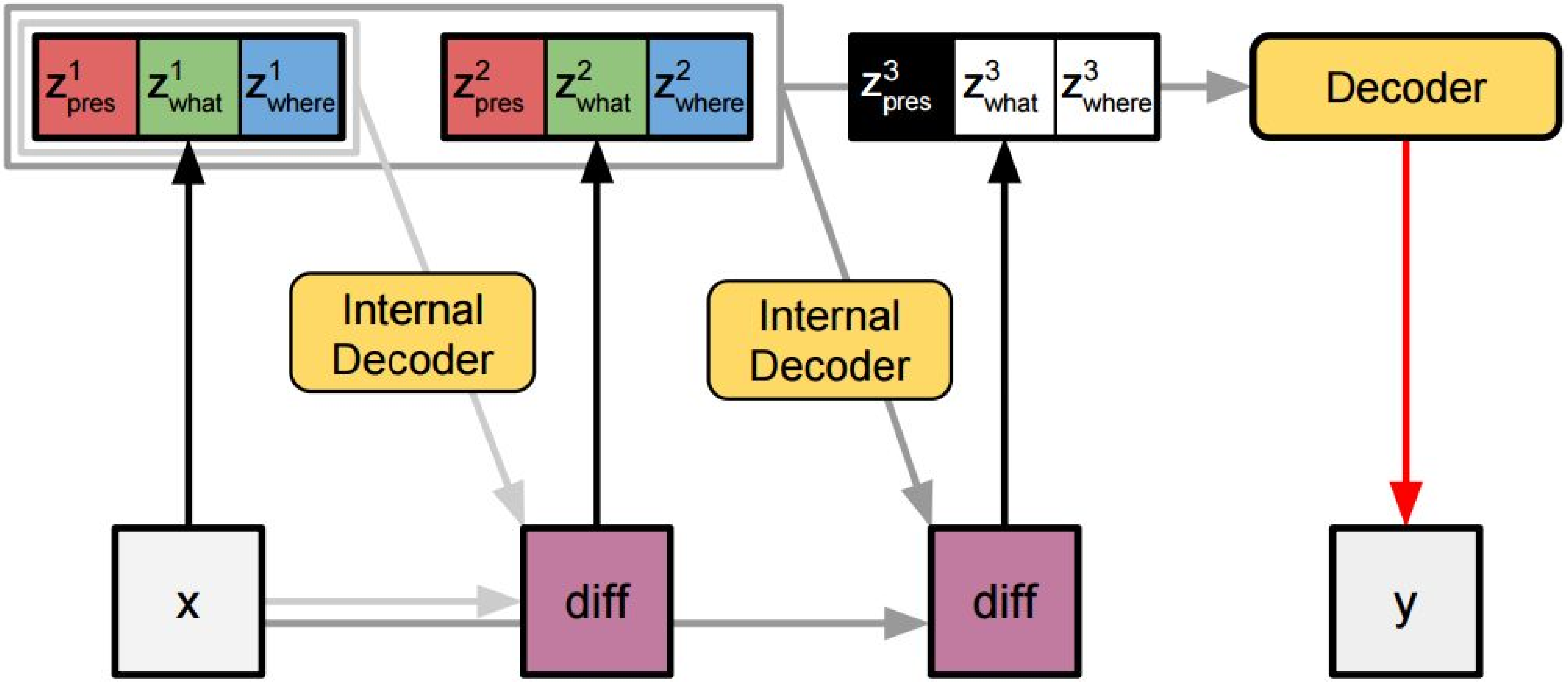




Generating and designing DNA with deep generative models. Killoran, Lee, Delong, Duvenaud, Frey, 2017



# Grammar VAE

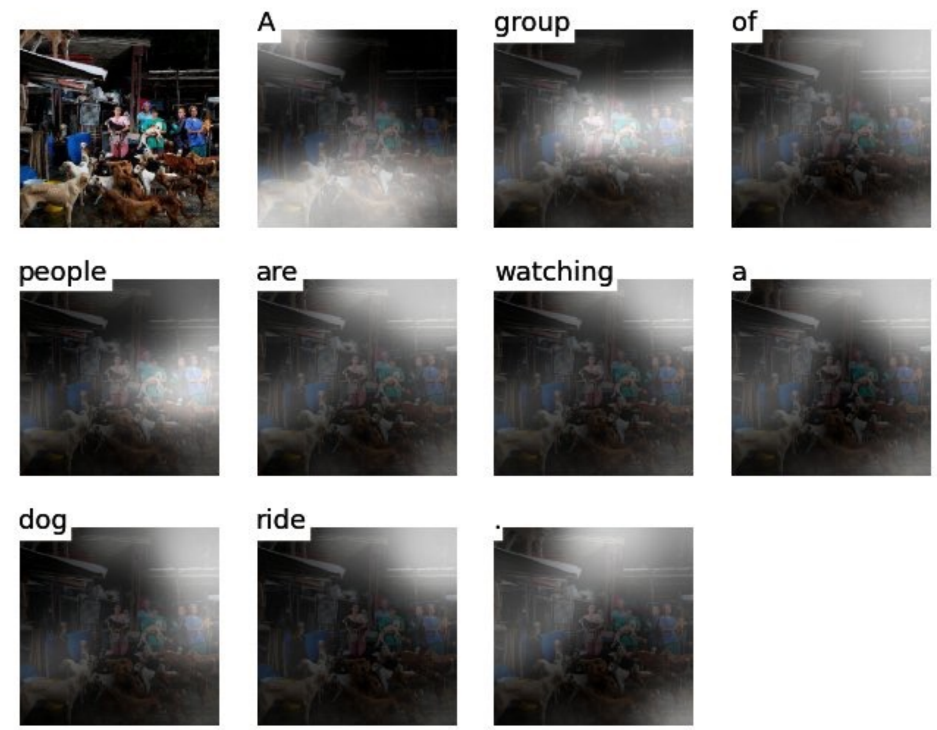
Matt Kusner, Brooks Paige, José Miguel Hernández-Lobato






# 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

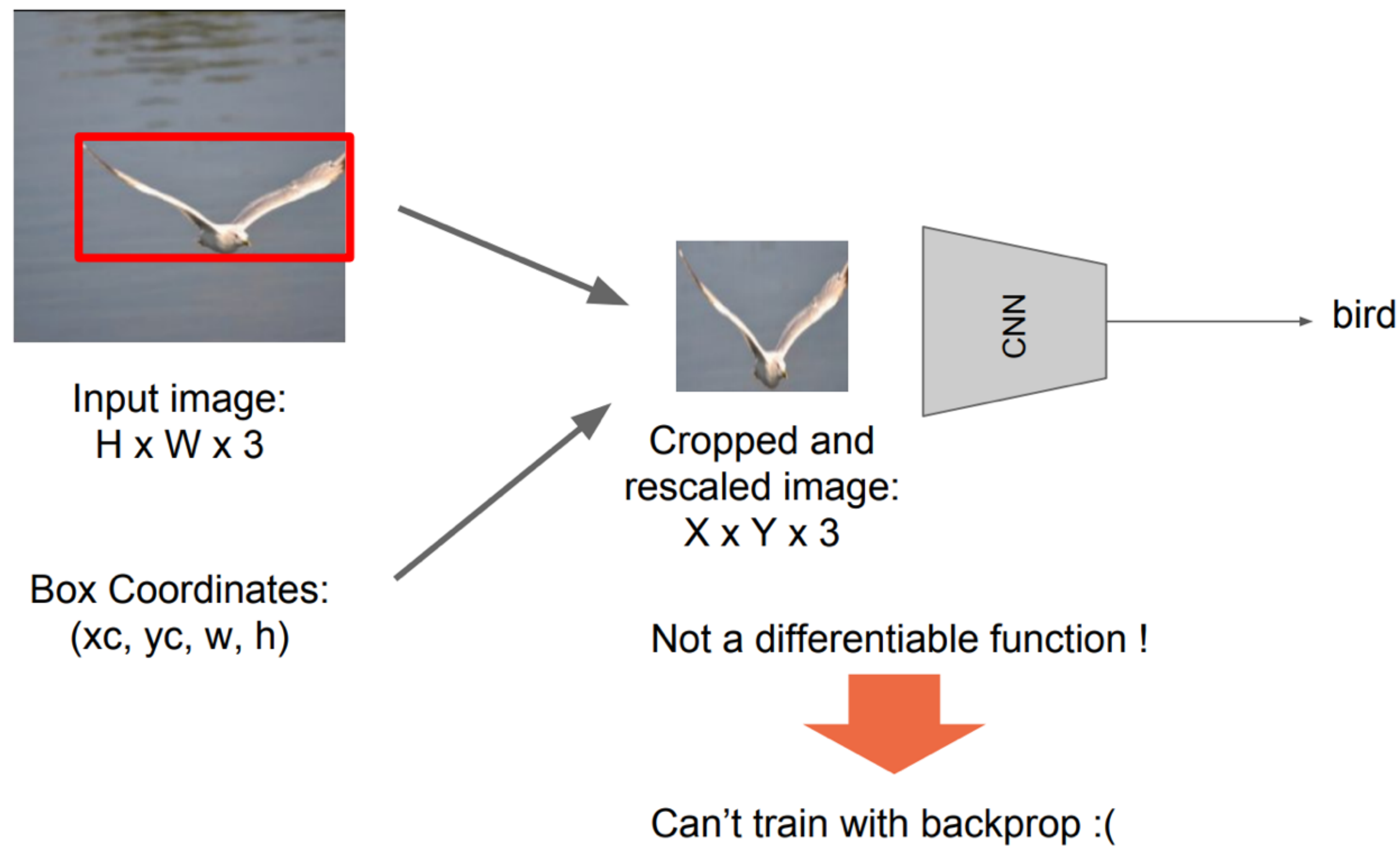



A group of people are watching a dog ride

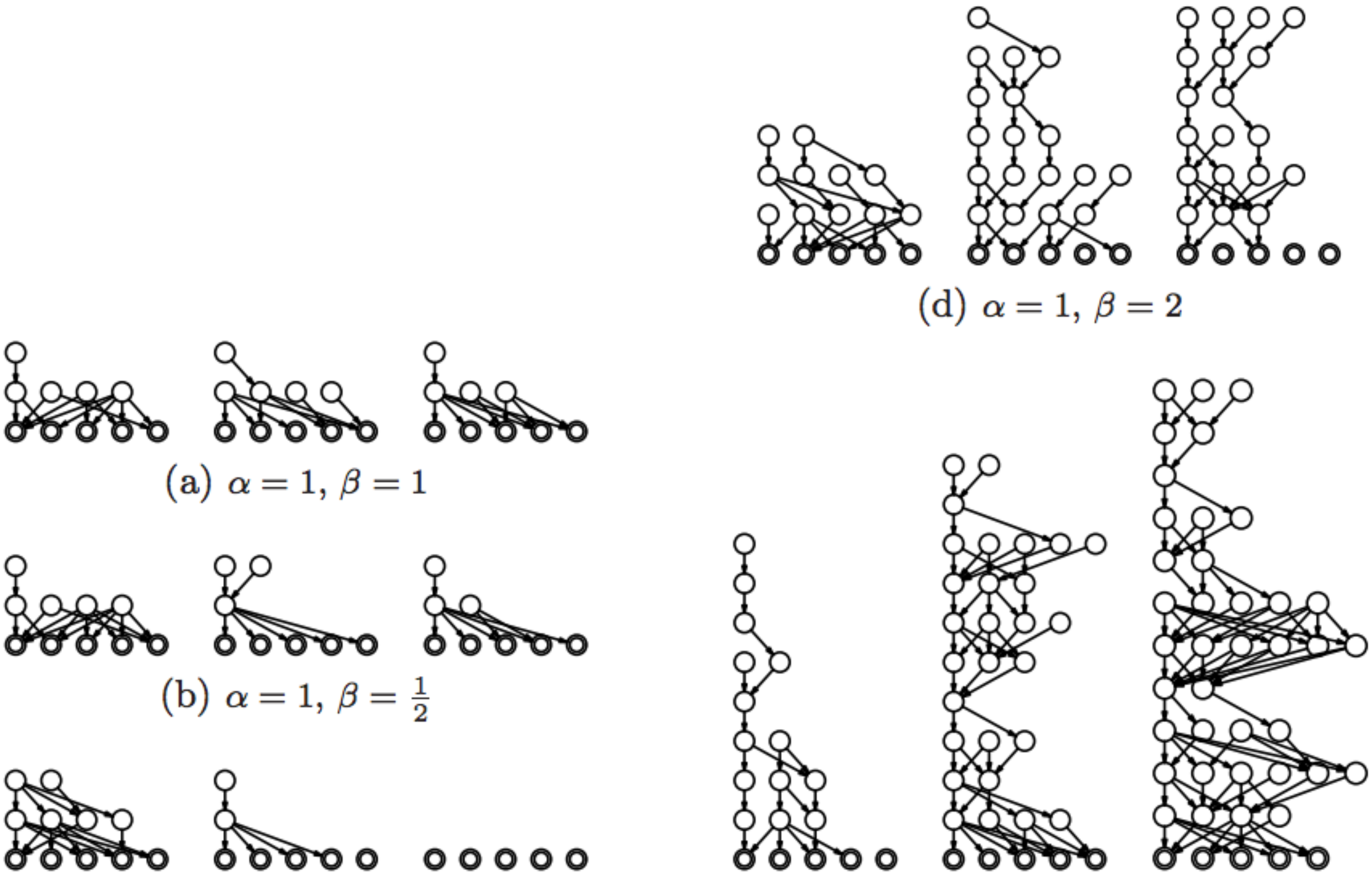
(Jamie Kyros)

# Hard attention models

- Want large or variable-sized memories or ‘scratch pads’
- Soft attention is a good computational substrate, scales linearly O(N) with size of model
- Want O(1) read/write
- This is “hard attention”




Source: http://imatge-upc.github.io/telecombcn-2016-dlcv/slides/D4L6-attention.pdf



(a) $\alpha = 1,\ \beta = 1$

(b) $\alpha = 1,\ \beta = \frac{1}{2}$

(c) $\alpha = \frac{1}{2},\ \beta = 1$

(d) $\alpha = 1,\ \beta = 2$

(e) $\alpha = \frac{3}{2},\ \beta = 1$

Fig 3: Samples from the CIBP-based prior on network structures, with five visible units.

Learning the Structure of Deep Sparse Graphical Models
Ryan Prescott Adams, Hanna M. Wallach, Zoubin Ghahramani, 2010

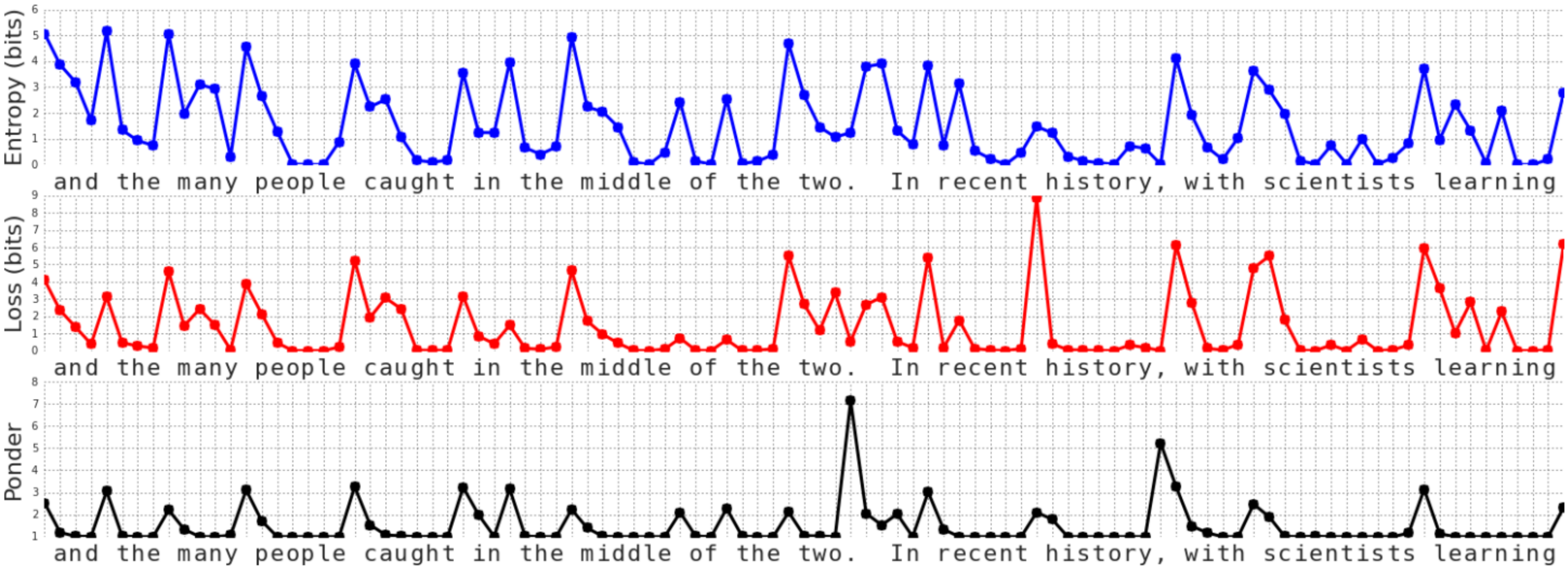
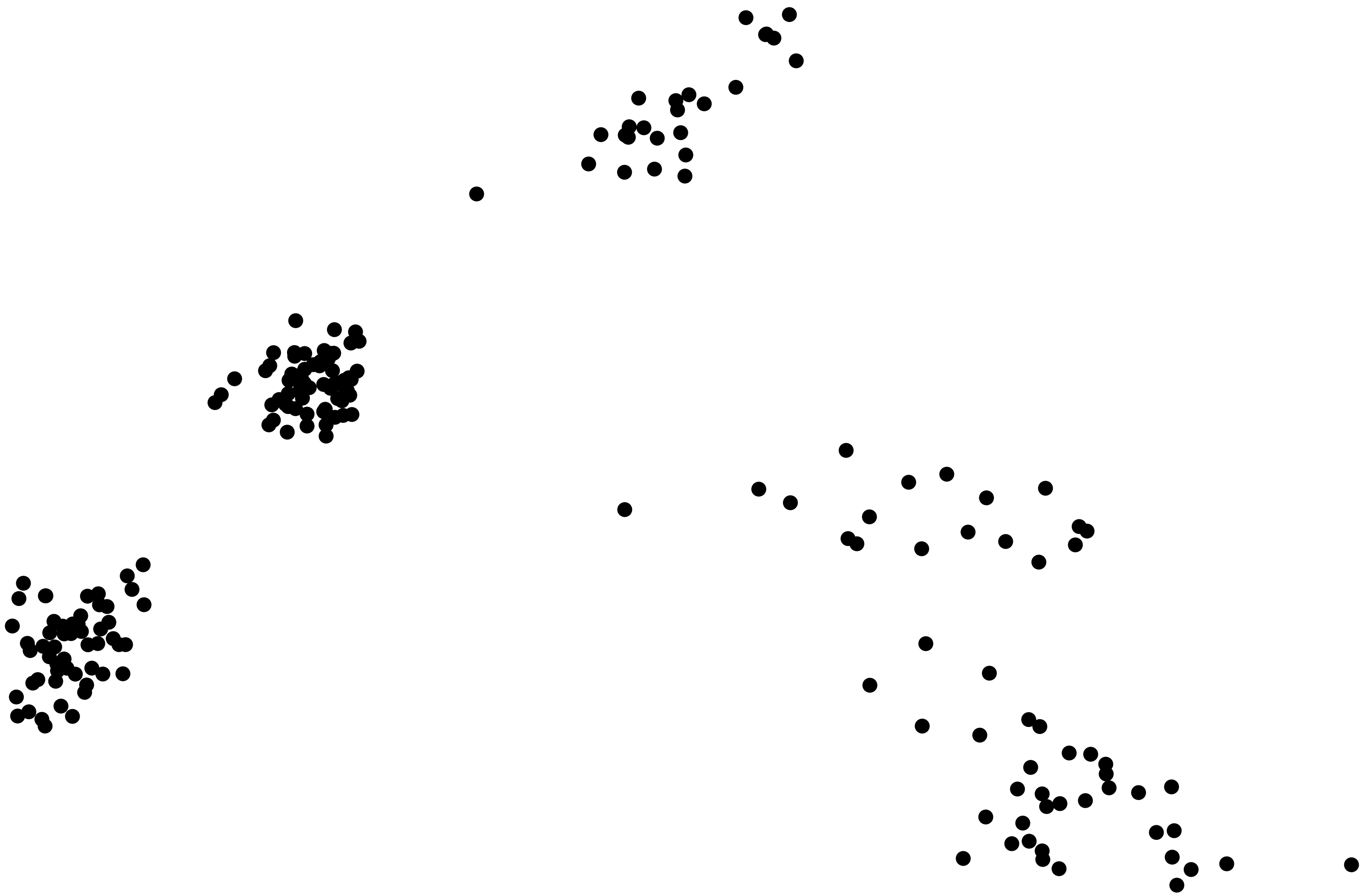
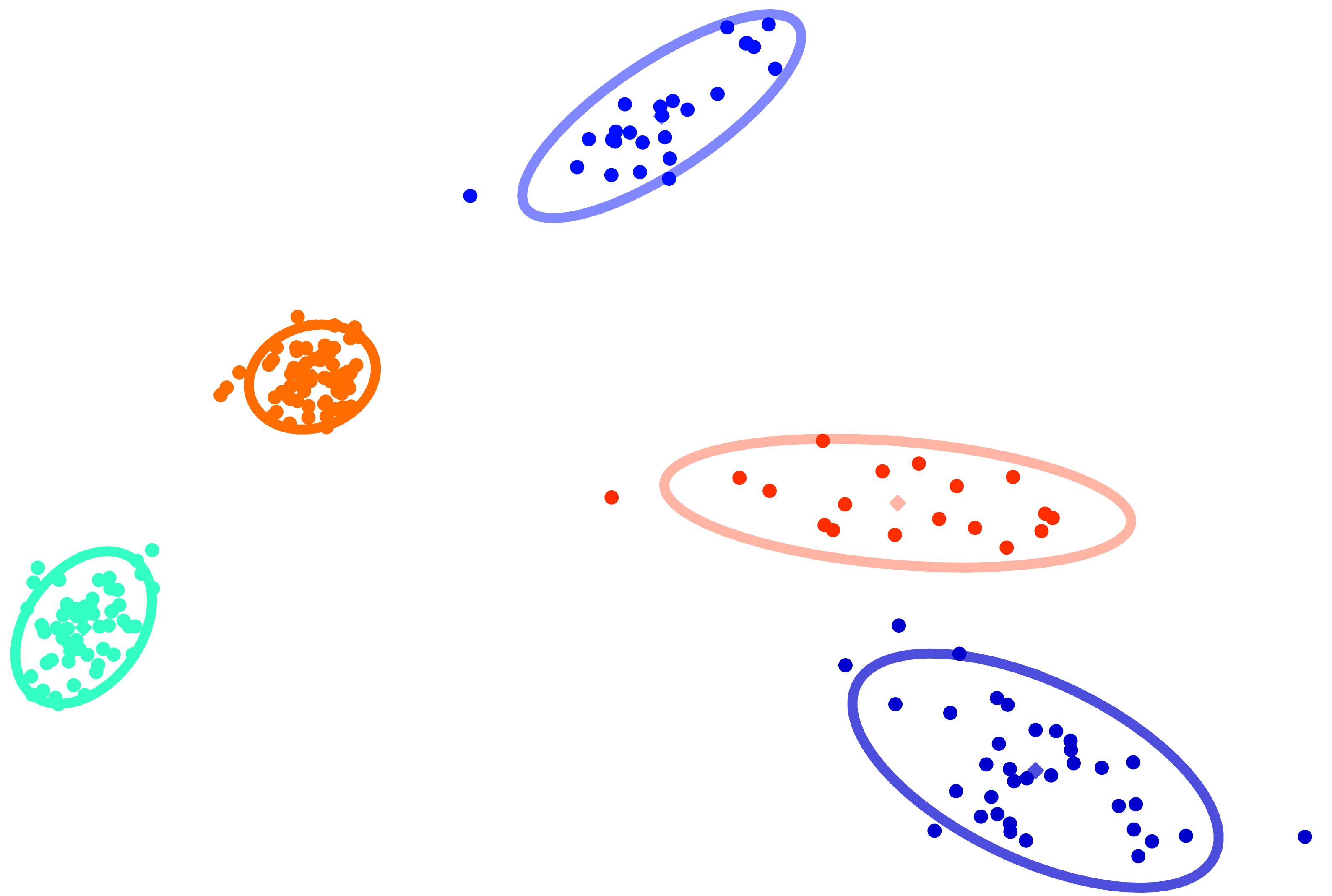
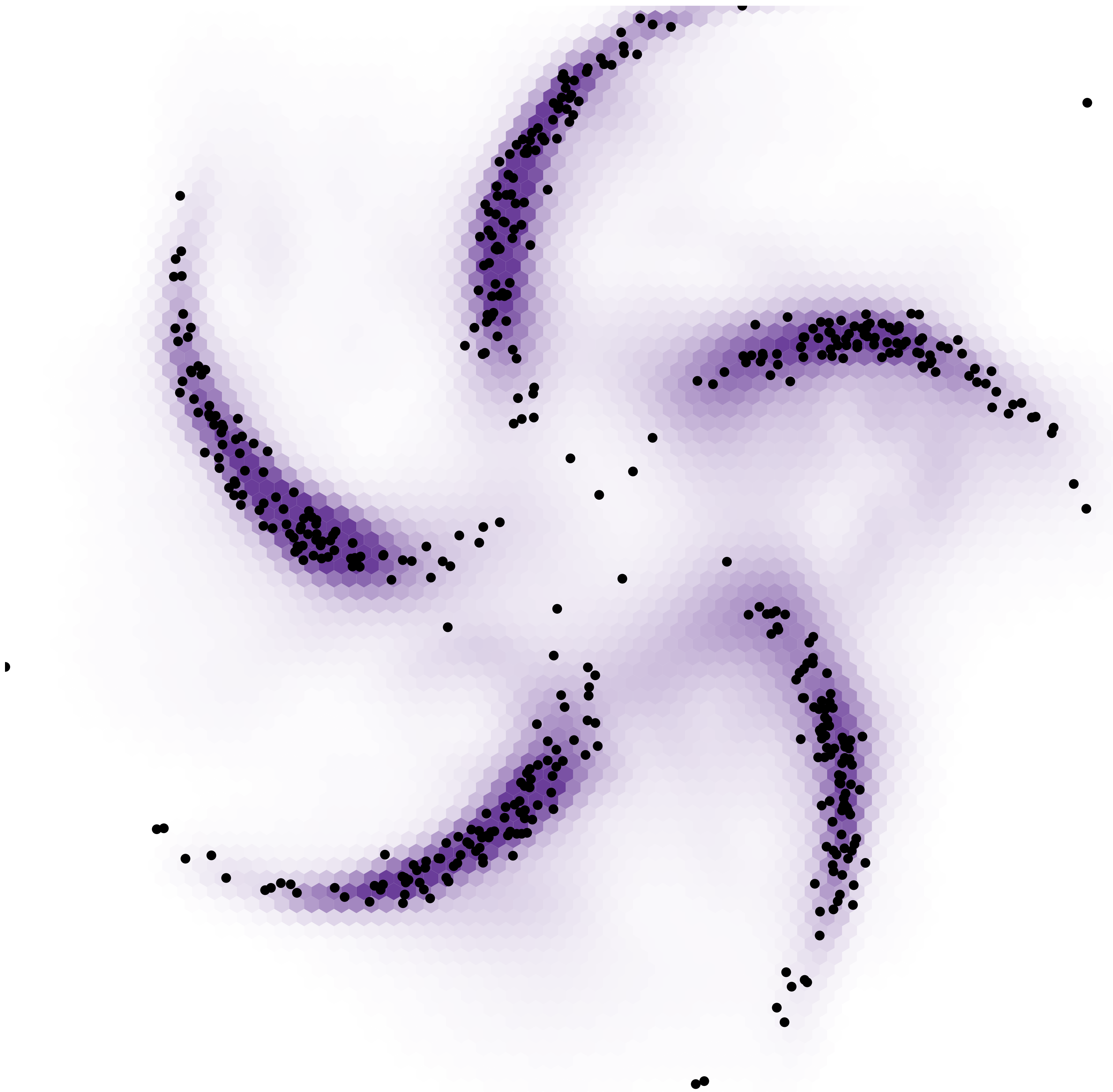


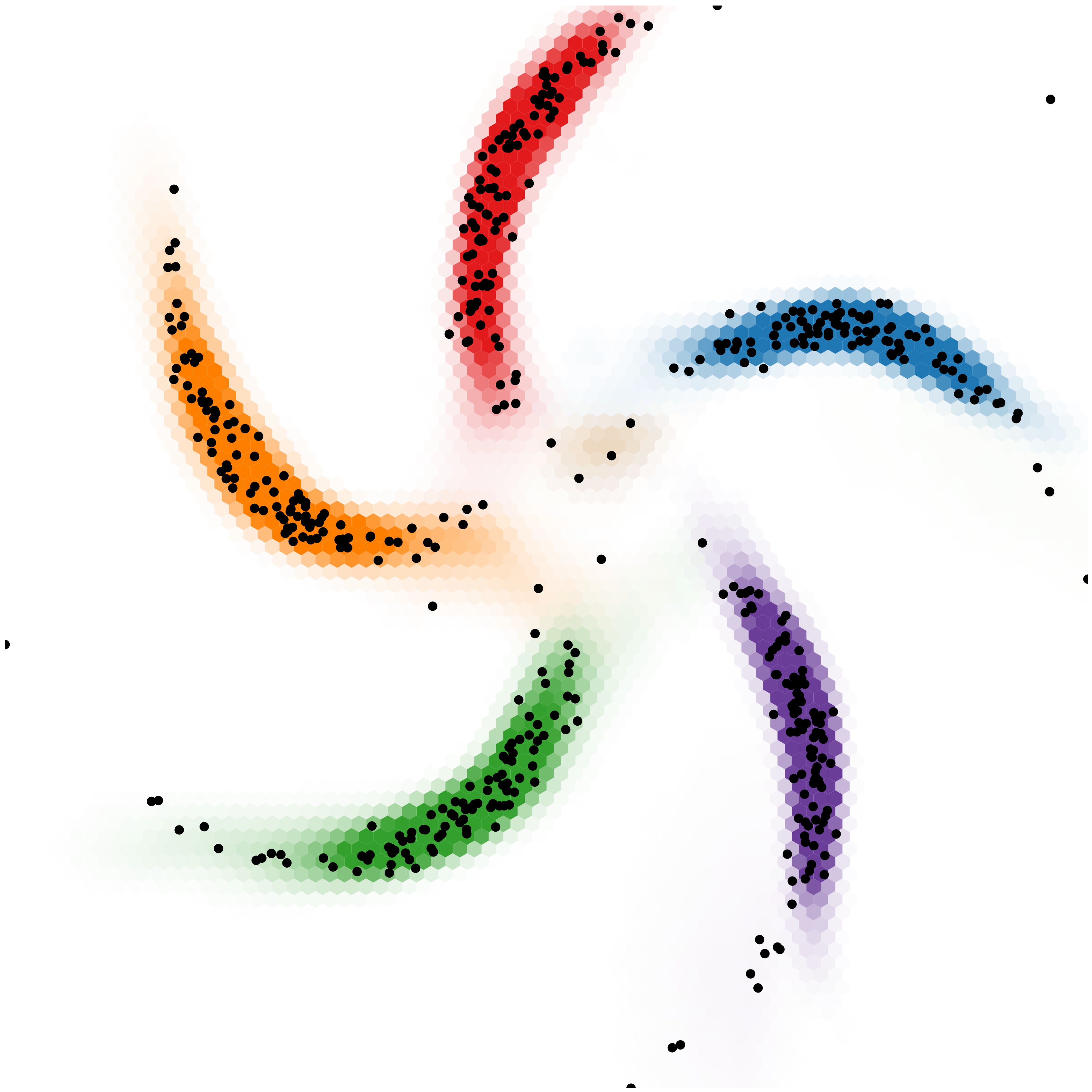

**Figure 23: Ponder Time, Prediction loss and Prediction Entropy During a Wikipedia Text Sequence.** Plot created using a network trained with $\tau = 6e^{-3}$

Adaptive Computation Time for Recurrent Neural Networks
Alex Graves, 2016



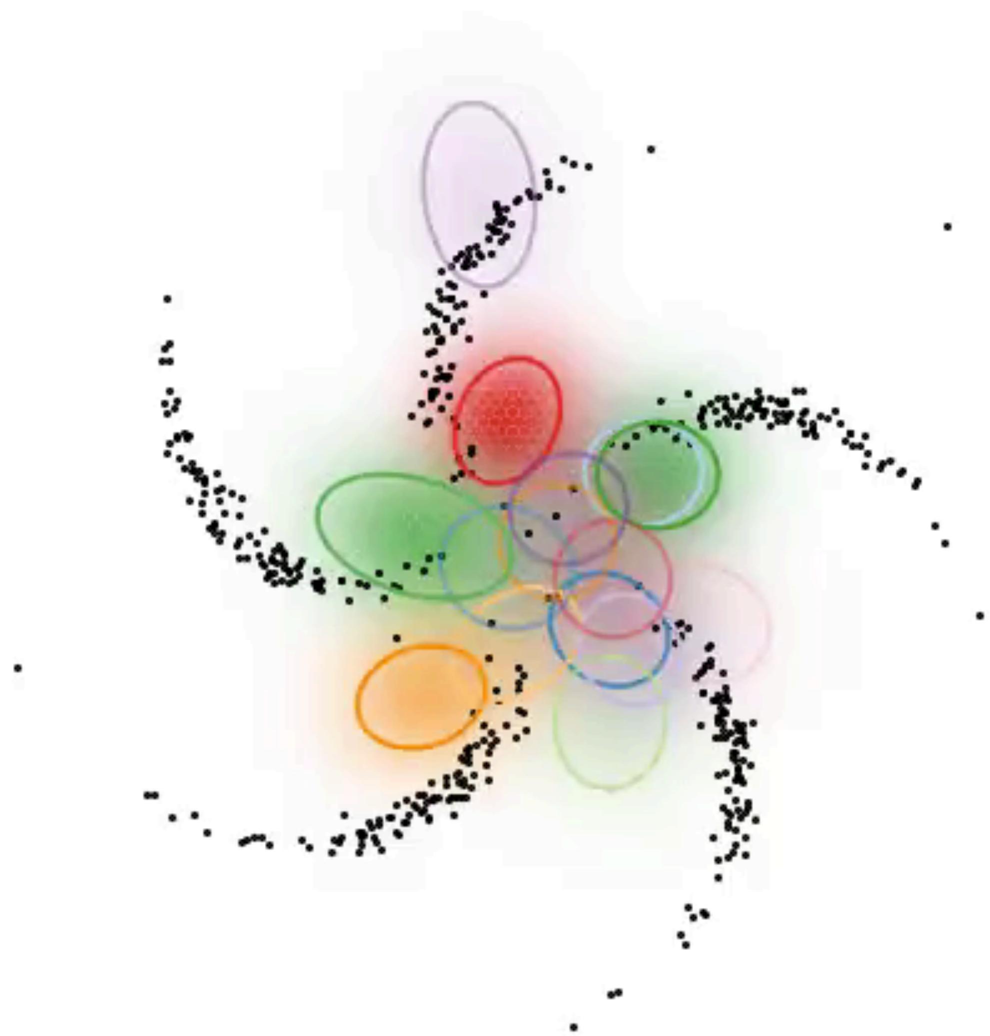




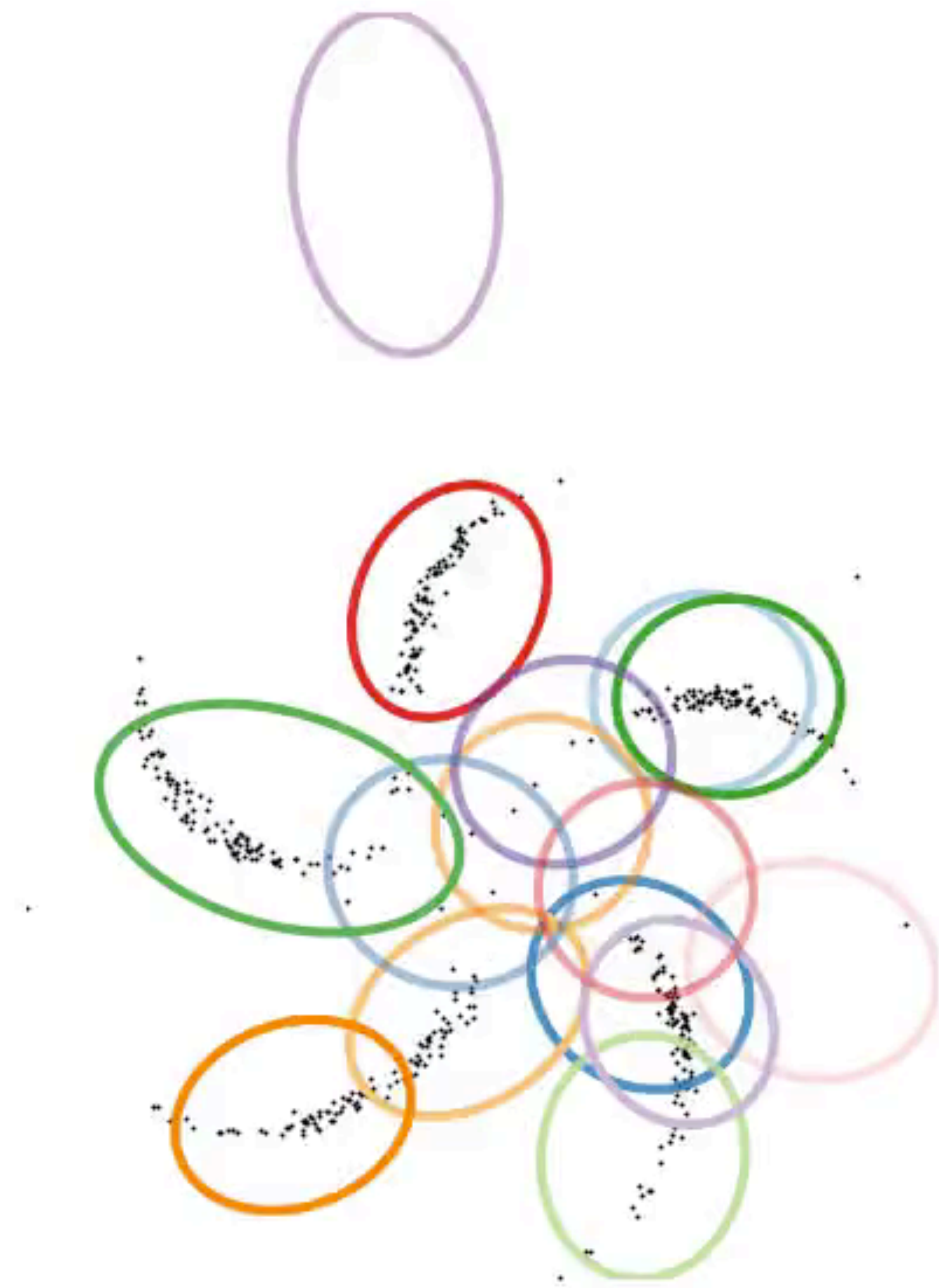


**Modeling idea:** graphical models on latent variables, neural network models for observations

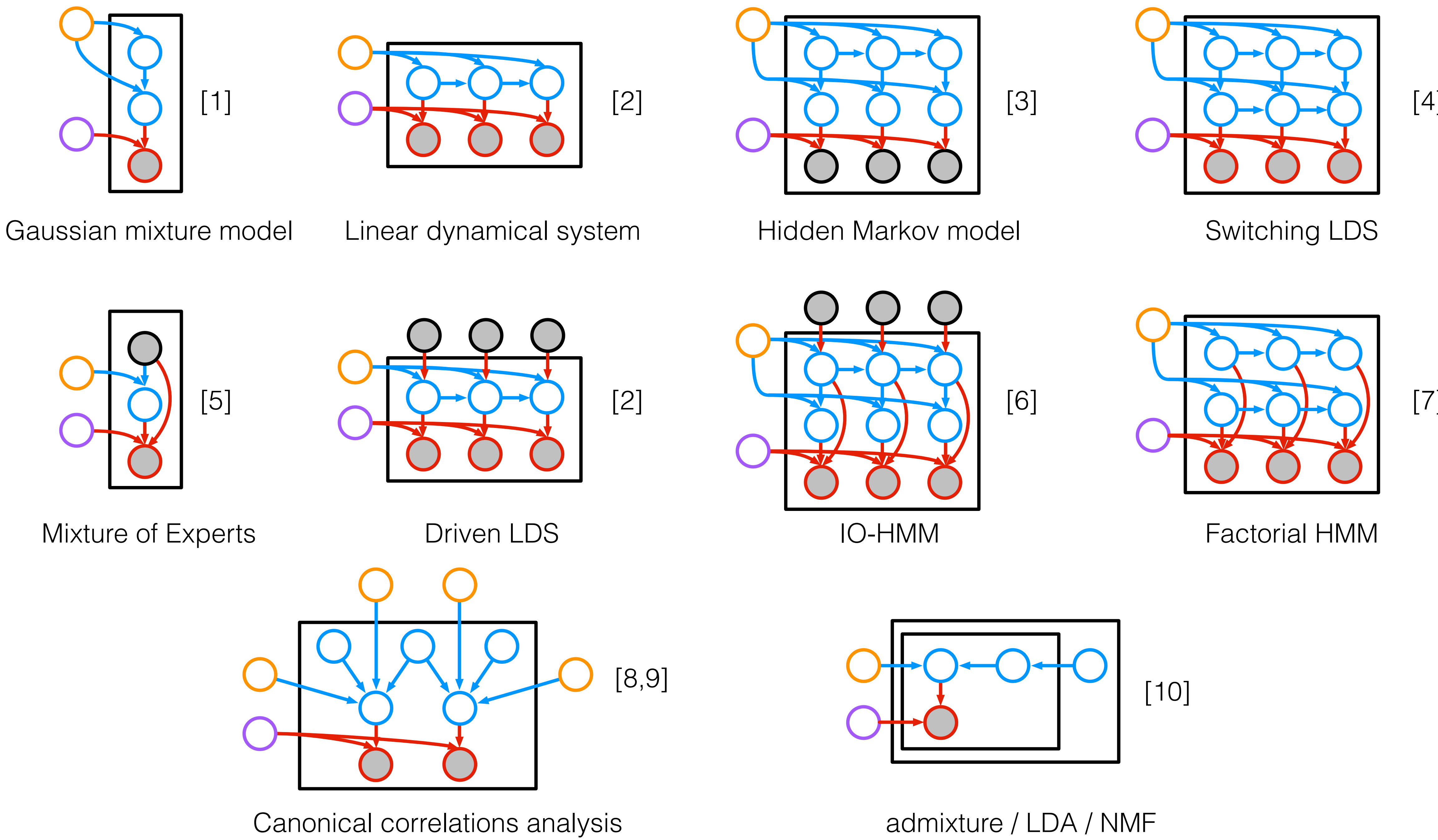
Composing graphical models with neural networks for structured representations and fast inference. Johnson, Duvenaud, Wiltschko, Datta, Adams, NIPS 2016



data space



latent space




[1] 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

+ structured representations
+ priors and uncertainty
+ data and computational efficiency
– rigid assumptions may not fit
– feature engineering
– top-down inference

Deep learning

– neural net “goo”
– difficult parameterization
– can require lots of data
+ flexible
+ feature learning
+ recognition networks

# Today: Overview and intro

- Motivation and overview
- Structure of course
- Project ideas
- Ungraded background quiz
- Actual content: History, state of the field, REINFORCE and reparameterization trick

# Structure of course

- I give first two lectures
- Next 7 lectures mainly student presentations
  - each covers 5-10 papers on a given topic
  - will finalize and choose topics next week
- Last 2 lectures will be project presentations

# Student lectures

- 7 weeks, 84 people(!) about 10 people each week.
- Each day will have one theme, 5-10 papers
- Divided into 4-5 presentations of about 20 mins each
- Explain main idea, scope, relate to previous work and future directions
- Meet me on Friday or Monday before to organize

# Grading structure

- 15% One assignment on gradient estimators
- 15% Class presentations
- 15% Project proposal
- 15% Project presentation
- 40% Project report and code

# Assignment

- Q1: Show REINFORCE is unbiased.  Add different control variates/baselines and see what happens.
- Q2: Derive variance of REINFORCE, reparam trick, etc, and how it grows with the dimension of the problem.
- Q3: Show that stochastic policies are suboptimal in some cases, optimal in others.
- Q4: Pros and cons of different ways to represent discrete distributions.
- Bonus 1: Derive optimal surrogates for REBAR, LAX, RELAX
- Bonus 2: Derive optimal reparameterization for a Gaussian
- Hints galore

# Tentative Course Dates

- Assignment due Feb. 1
- Project proposal due Feb. 15
    - ~2 pages, typeset, include preliminary lit search
- Project Presentations: March 16th and 23rd
- Projects due: mid-April

# Learning outcomes

- How to optimize and integrate in settings where we can't just use backprop
- Familiarity with the recent generative models and RL literature
- Practice giving presentations, reading and writing papers, doing research
- Ideally: Original research, and most of a NIPS submission!

# Project Ideas - Easy

- Compare different gradient estimators in an RL setting.
- Compare different gradient estimators in a variational optimization setting.
- Write a distill article with interactive demos.
- Write a lit review, putting different methods in the same framework.

# Project ideas - medium

- Train GANs to produce text or graphs.
- Train huge HMM with O(KT) cost per iteration [like van den Oord et al.,2017]
- Train a model with hard attention, or different amounts of compute depending on input. [e.g. Graves 2016]
- A theory paper analyzing the scalability of different estimators in different settings.
- Meta-learning with discrete choices at both levels
- Train a VAE with continuous latents but with a non-differentiable decoder (e.g. a renderer), or surrogate loss for text

# Project ideas - hard

- Build a VAE with discrete latent variables of different size depending on input. E.g. latent lists, trees, graphs.
- Build a GAN that outputs discrete variables of variable size. E.g. lists, trees, graphs, programs
- Fit a hierarchical latent variable model to a single dataset (a la Tenenbaum, or Grosse)
- Propose and examine new gradient estimator / optimizer / MCMC alg.
- Theory paper: Unify existing algorithms, or characterize their behavior

# Ungraded Quiz

# Next week:
# Advanced gradient estimators

- Most mathy lecture of the course
- Should prep you and give context for for A1
- Only calculus and probability
- Not as scary as it looks!

# Lecture 0:
# State of the field and basic gradient estimators

# 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)

# Differentiable models

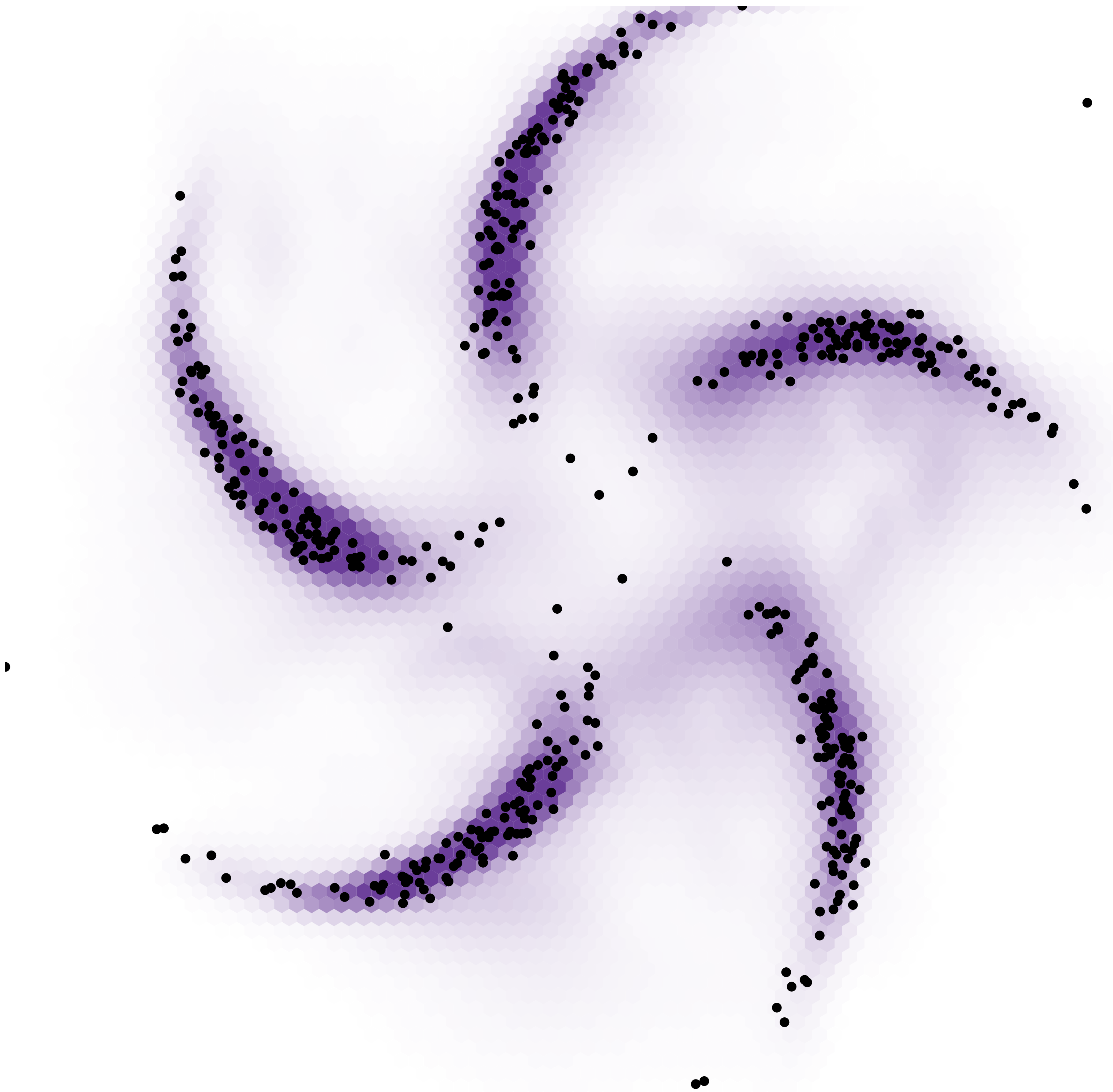
- Model distributions implicitly by a variable pushed through a deep net:

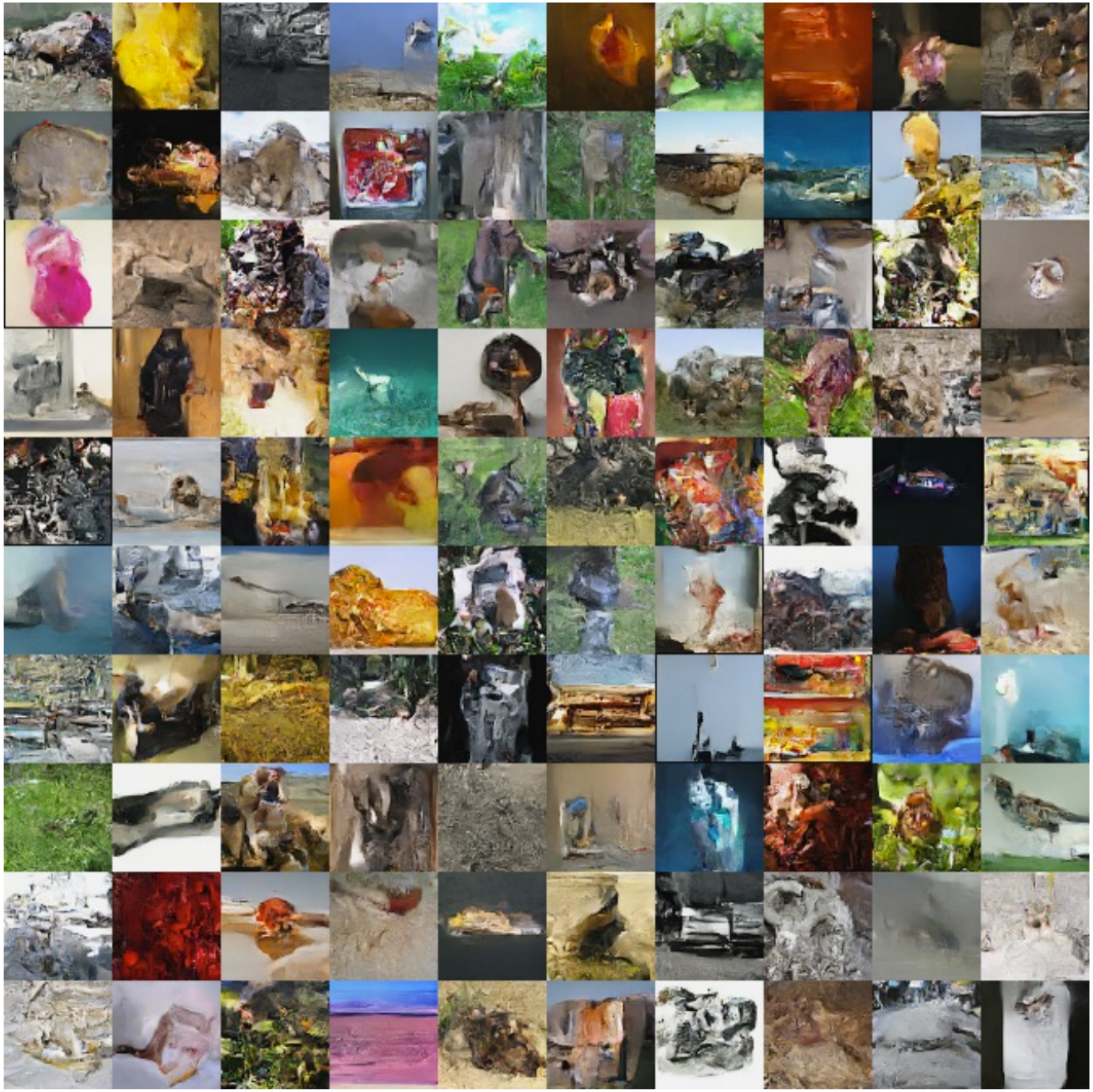
$$y = f_\theta(x)$$

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

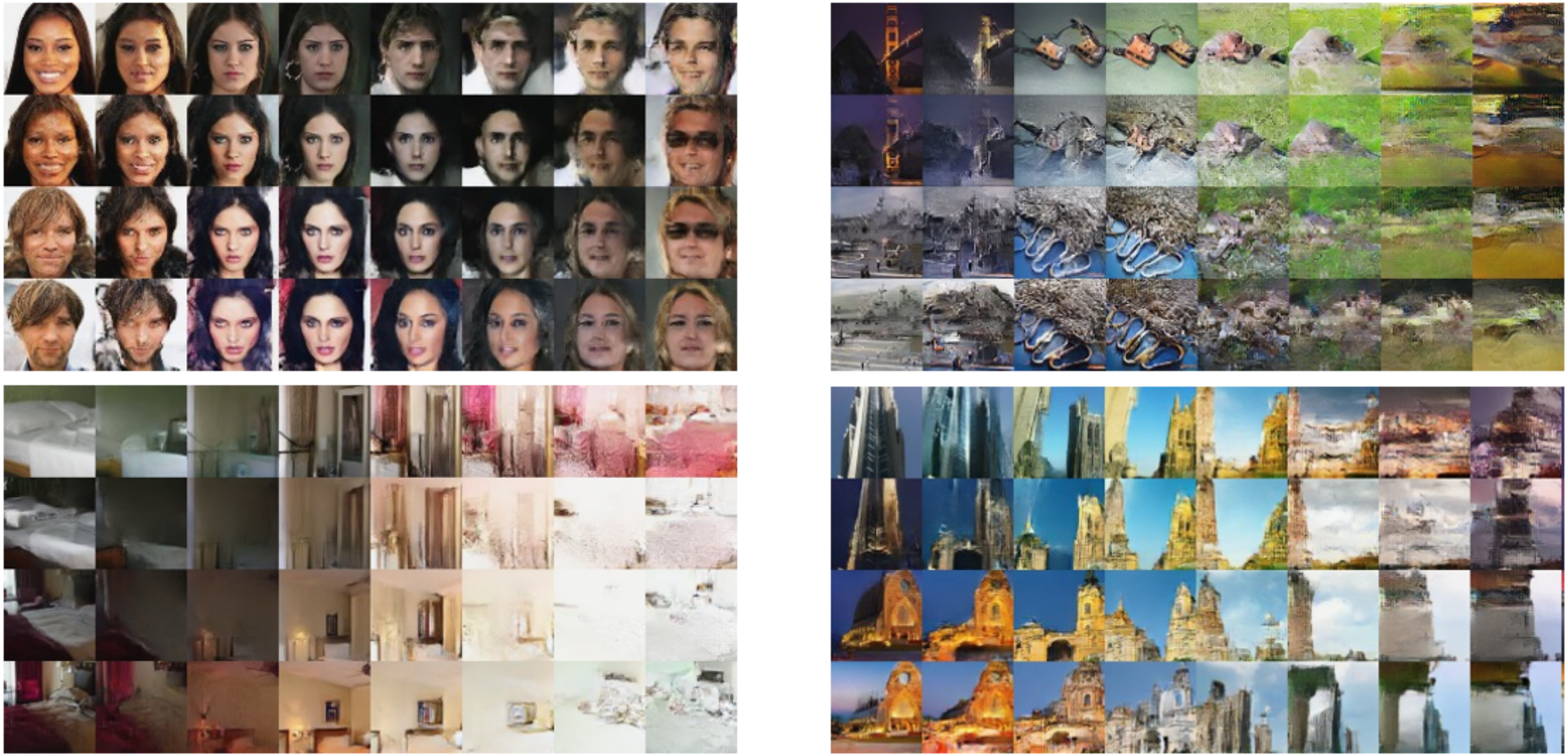
$$p(y|x) = \mathcal{N}(y|\mu = f_\theta(x), \Sigma = g_\theta(x))$$

- Optimize all parameters using stochastic gradient descent

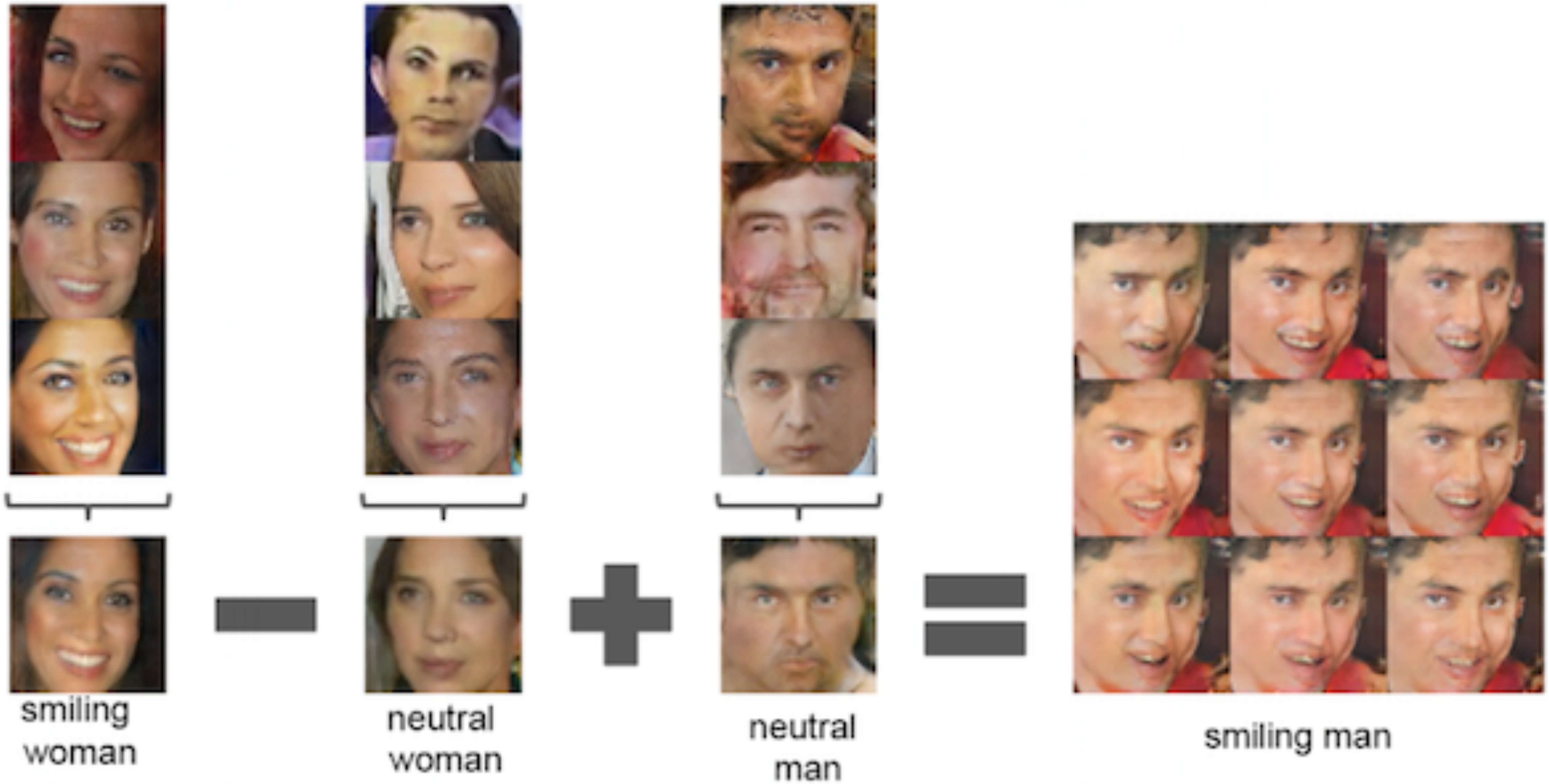




Density estimation using Real NVP. Ding et al, 2016



Density estimation using Real NVP. Ding et al, 2016




Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Alec Radford, Luke Metz, Soumith Chintala, 2015

# Advantages of latent variable models

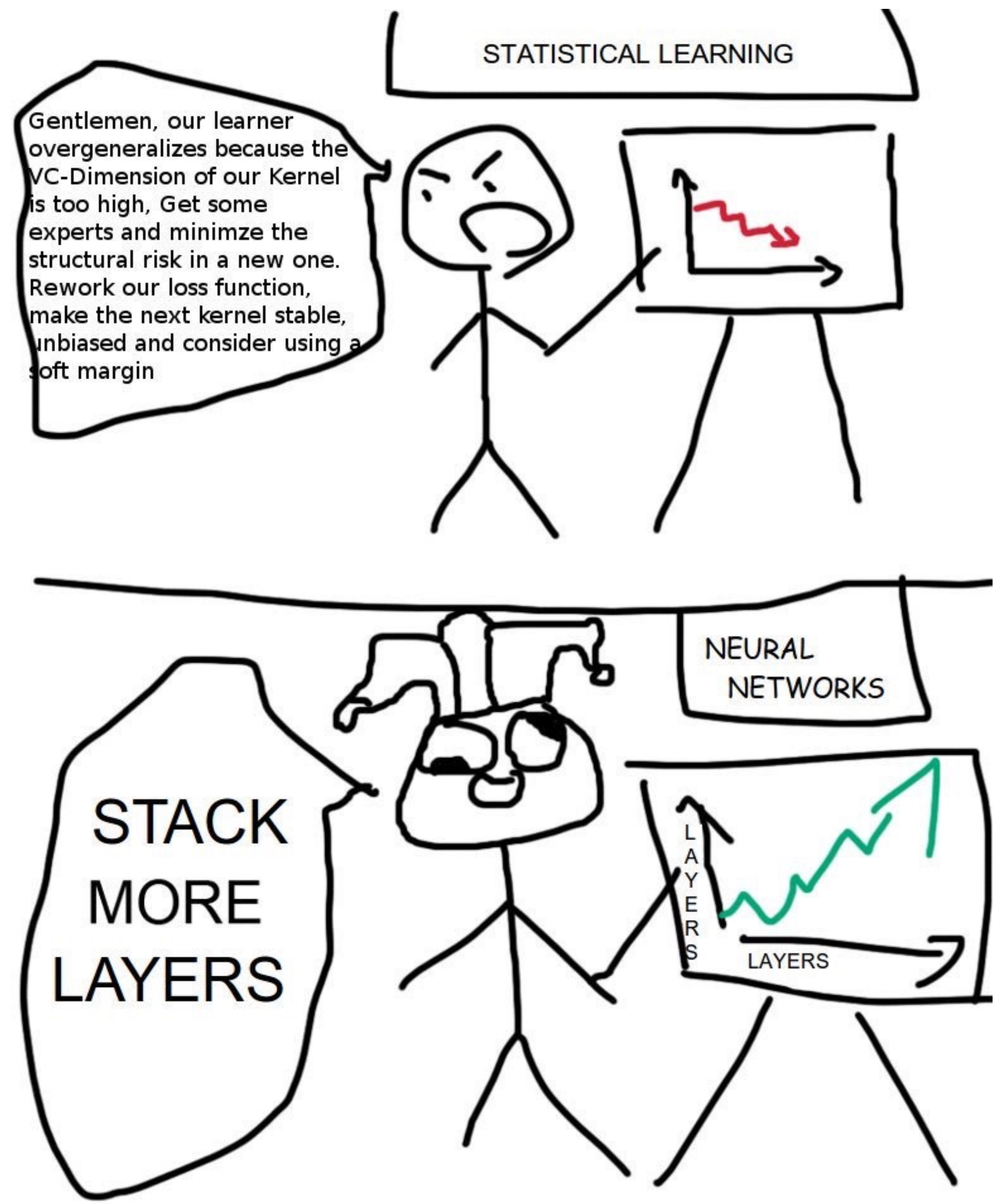
- Model checking by sampling
- Natural way to specify models
- Compact representations
- Semi-Supervised learning
- Understanding factors of variation in data

# State of the field

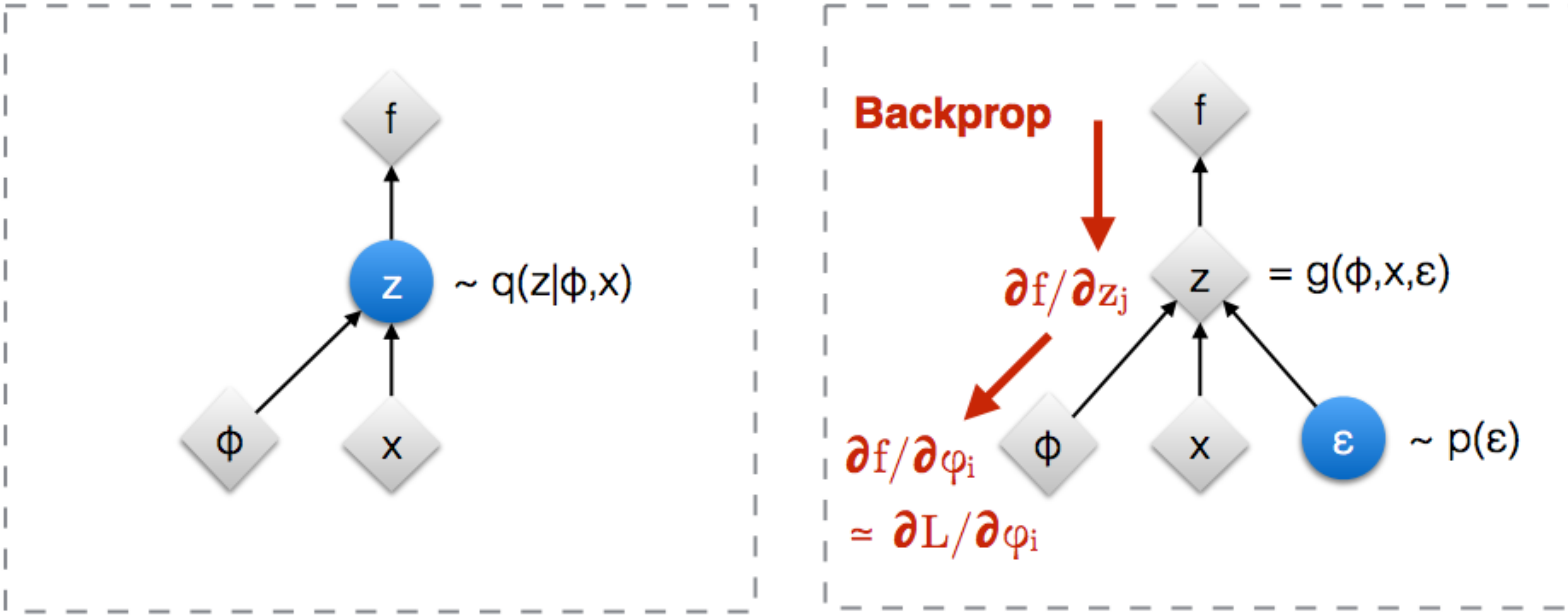
- Big lesson of deep learning: stochastic gradient-based optimization scales well to millions of parameters

- Easy to train supervised and unsupervised models this way, if everything is continuous which allows reparameterization.

- Now, we're hitting the limits of this modeling style



Original form | Reparameterised form




: Deterministic node

: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Source: Kingma's NIPS 2015 workshop slides

# SCORE-FUNCTION ESTIMATOR (“REINFORCE”, WILLIAMS 1992)

$$\frac{\partial}{\partial \theta}\mathbb{E}_{p(b|\theta)} f(b) = \int \frac{\partial}{\partial \theta} p(b|\theta) f(b) d\theta$$

These slides by Geoff Roeder

# SCORE-FUNCTION ESTIMATOR (“REINFORCE”, WILLIAMS 1992)

$$\frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)} f(b) = \int \boxed{\frac{\partial}{\partial\theta} p(b|\theta)} f(b) d\theta$$

$$= \mathbb{E}_{p(b|\theta)}\left[ f(b) \boxed{\frac{\partial}{\partial\theta}\log p(b|\theta)} \right]$$

- **Log-derivative trick** allows us to rewrite gradient of expectation as expectation of gradient (under weak regularity conditions)
- We can es[illegible] with Monte Carlo integration
- High variance, co[illegible] good solution challenging

# SCORE-FUNCTION ESTIMATOR ("REINFORCE", WILLIAMS 1992)

$$\frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}f(b) = \int \boxed{\frac{\partial}{\partial\theta}p(b|\theta)} f(b)d\theta$$

$$= \mathbb{E}_{p(b|\theta)}\left[f(b)\boxed{\frac{\partial}{\partial\theta}\log p(b|\theta)}\right]$$

$$\hat{g}_{SF} = f(b)\frac{\partial}{\partial\theta}\log p(b|\theta)$$

- **Log-derivative trick** allows us to rewrite gradient of expectation as expectation of gradient (under weak regularity conditions)
- Yields unbiased, but high variance estimator

# REPARAMETERIZATION TRICK

$$\boxed{g_{REP}\left[f(b)\right] = \frac{\partial}{\partial\theta} f(b) = \frac{\partial f}{\partial \mathcal{T}} \frac{\partial \mathcal{T}}{\partial \theta}, b = \mathcal{T}(\theta, \epsilon), \epsilon \sim p(\epsilon)}$$

- Requires function to be known and differentiable
- Requires distribution $p(b|\theta)$ to be reparameterizable through a transformation $\mathcal{T}(\theta, \epsilon)$
- Unbiased; lower variance empirically

# CONCRETE REPARAMETERIZATION (MADDISON ET AL. 2016)

$$\boxed{g_{CON}\left[f(b)\right] = \frac{\partial}{\partial \theta} f(b) = \frac{\partial f}{\partial \sigma_\lambda(z)} \frac{\partial \sigma_\lambda(z)}{\partial \theta}, z = \mathcal{T}(\theta, \epsilon), \epsilon \sim p(\epsilon)}$$

- Works well with careful hyper parameter choices
- Lower variance than score-function estimator due to reparameterization

- Biased estimator
- Temperature parameter $\lambda$
- Requires $f$ to be known and differentiable
- Requires $p(b|\theta)$ to be reparamaterizable

# REBAR (TUCKER ET AL. 2017)

- Improves over concrete distribution (*rebar* is stronger than *concrete*)
- Uses continuous relaxation of discrete random variables (concrete) to build unbiased, lower-variance gradient estimator
- Using the reparameterization from the Concrete distribution, construct a control variate for the score-function estimator
- Show how tune additional parameters of the estimator (e.g., temperature $\lambda$) online

**Digression**: control variates for Monte Carlo estimators

# CONTROL VARIATES: DIGRESSION

$$\hat{g}_{new}(b) = \hat{g}(b) + \eta\left(c(b) - \mathbb{E}_{p(b)}[c(b)]\right)$$

$$\eta^\star = -\frac{\mathrm{Cov}[\hat{\mathrm{g}}, \mathrm{c}]}{\mathrm{Var}[\hat{g}]}$$

- New estimator is equal in expectation to old estimator (bias is unchanged)
- Variance is reduced when |corr(c, g)| > 0
- We exploit the difference between the function c and its known mean during optimization to "correct" the value of the estimator

# CONTROL VARIATES: FREE-FORM

$$\hat{g}_{new}(b) = \hat{g}(b) - c_\phi(b) + \mathbb{E}_{p(b)}\left[c_\phi(b)\right]$$

- If we choose a neural network as our parameterized differentiable function, then the above formulation can be simplified to the above
- The scaling constant will be absorbed into the weights of the network, and optimality is determined by training
- How should we update the weights of the free-form control variate?

# High-dimensional Bayesopt?

- Bayesian optimization doesn't really work in 50 dimensions
    - BNN instead of GP?