Generative Models and Discrete Optimization

Presented by: Þorsteinn (Thor) Hjörtur Jónsson

Generative Models

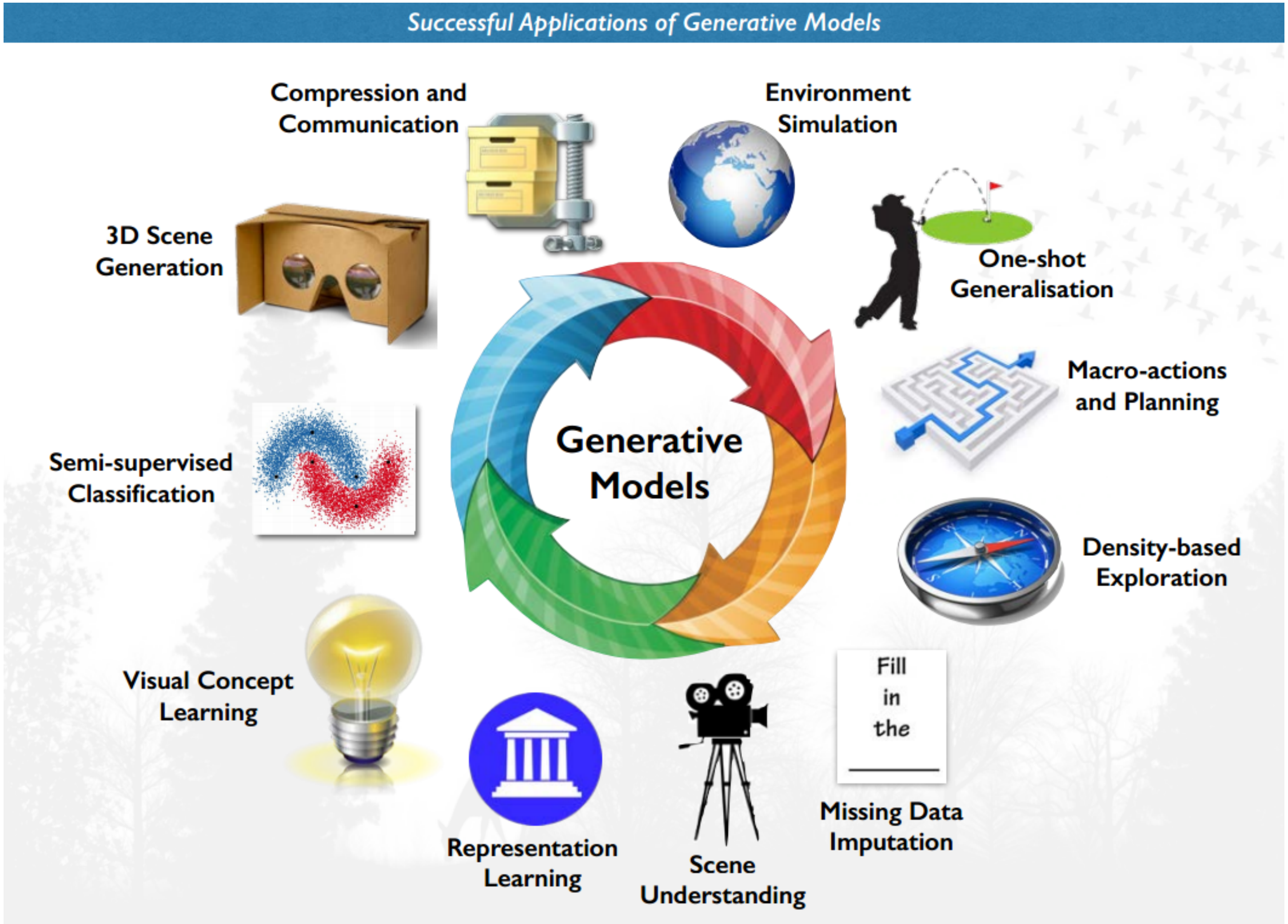
Learning Objectives.

a. Understand for which kinds of tasks generative models can be useful.

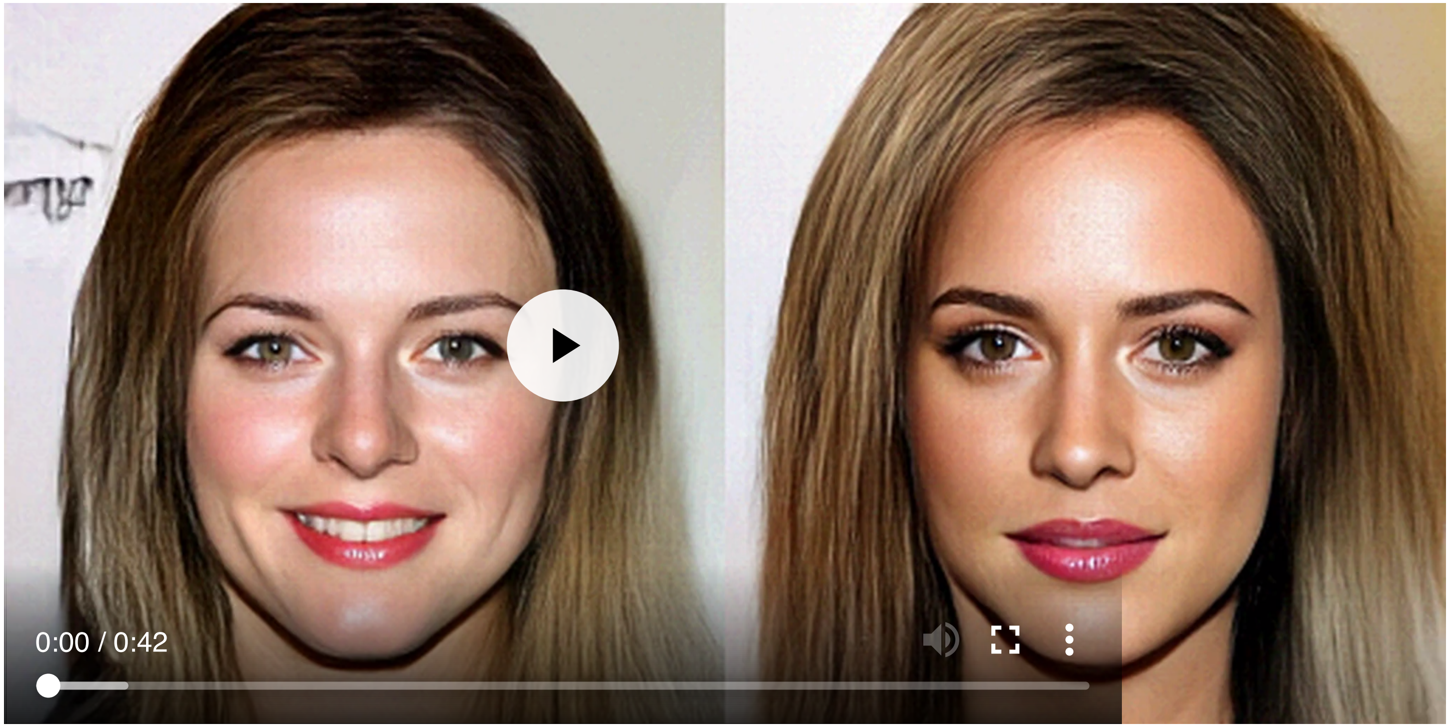
b. Understand how to formulate a latent-variable model.

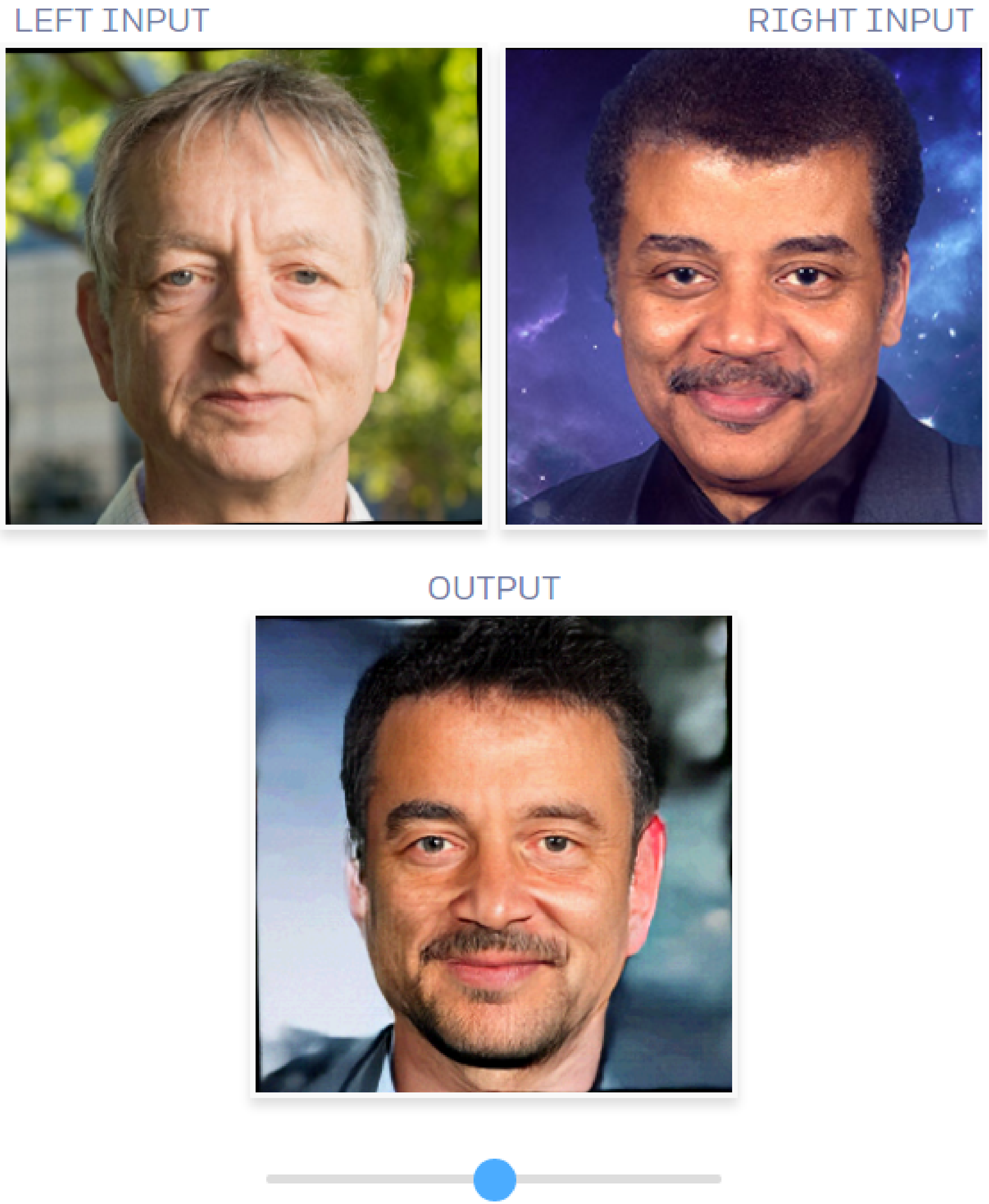
c. Understand how we can use neural networks to define divergences between data-generating distributions.

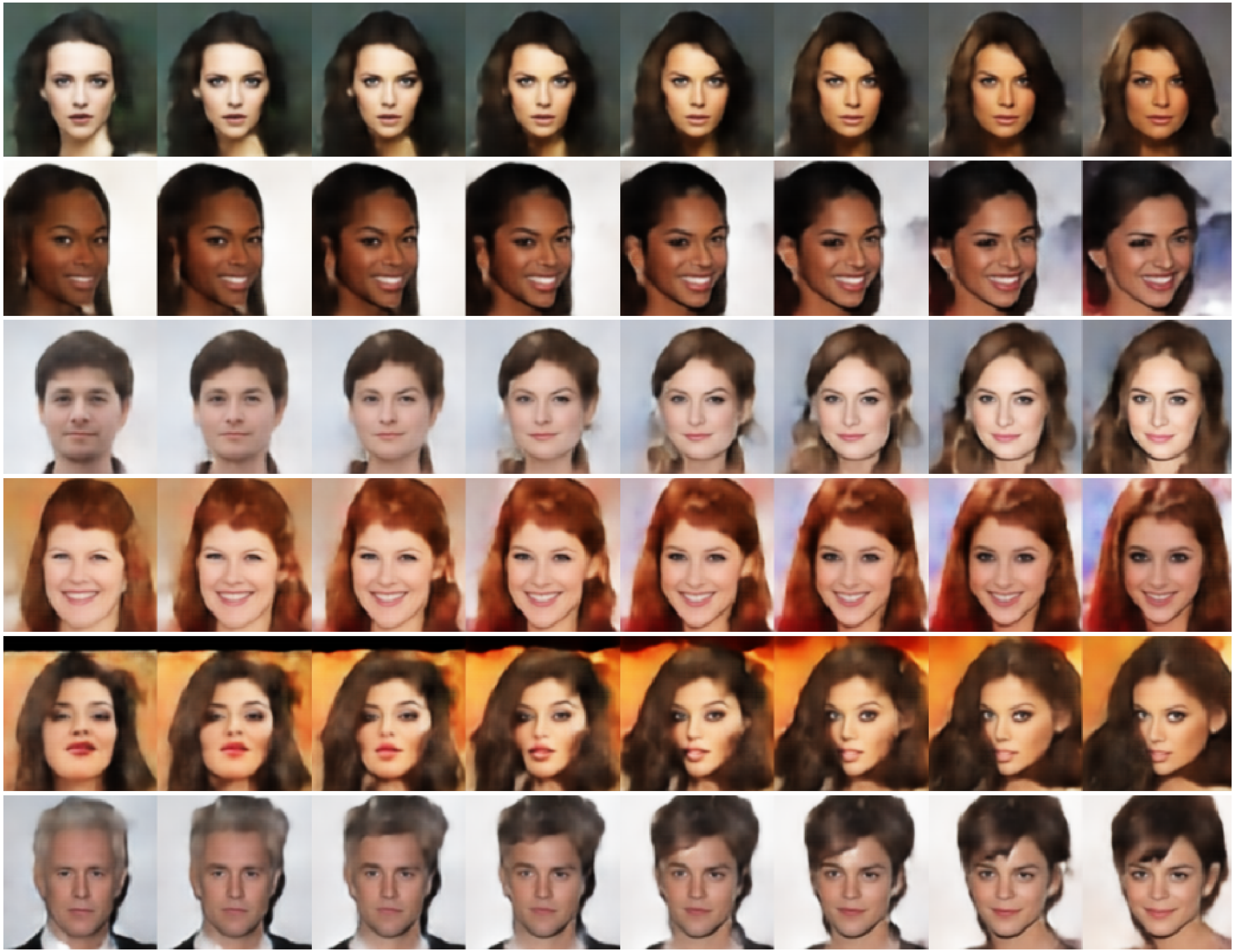
Having a Generative Model can be useful for many things.

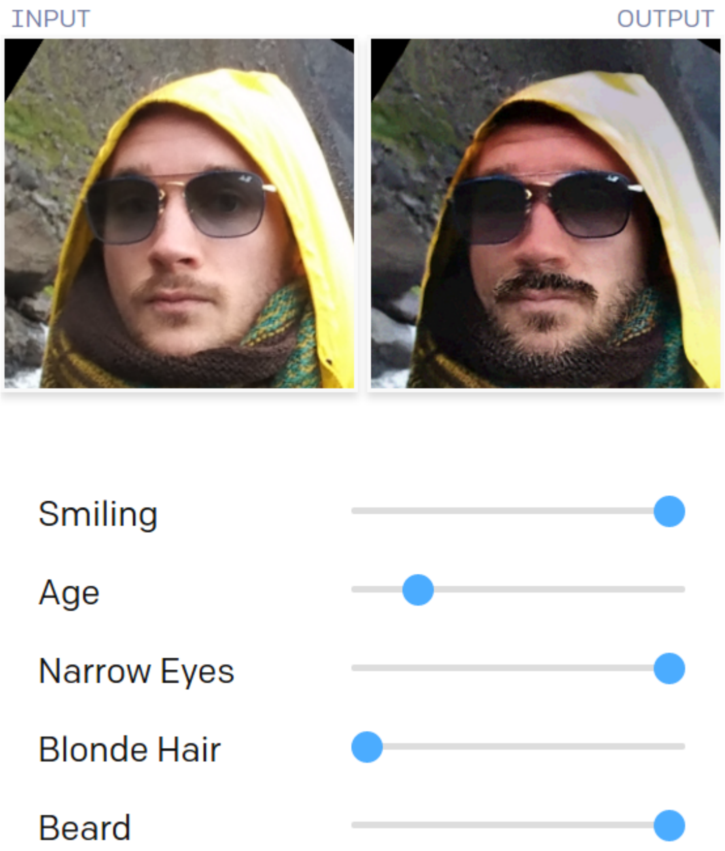


[Source: Shakir Mohamed]


0:00 / 0:42


LEFT INPUT
RIGHT INPUT
OUTPUT





INPUT

OUTPUT

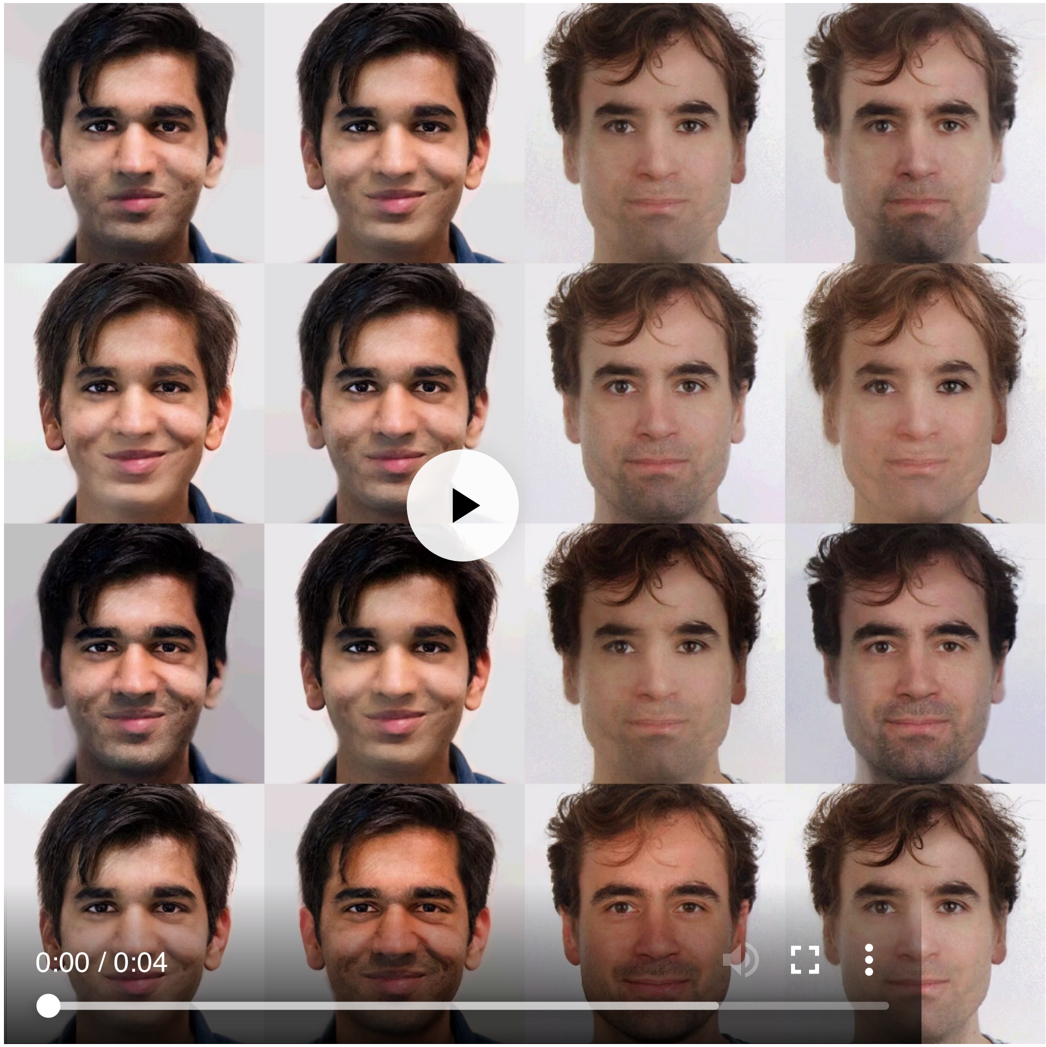
Smiling

Age

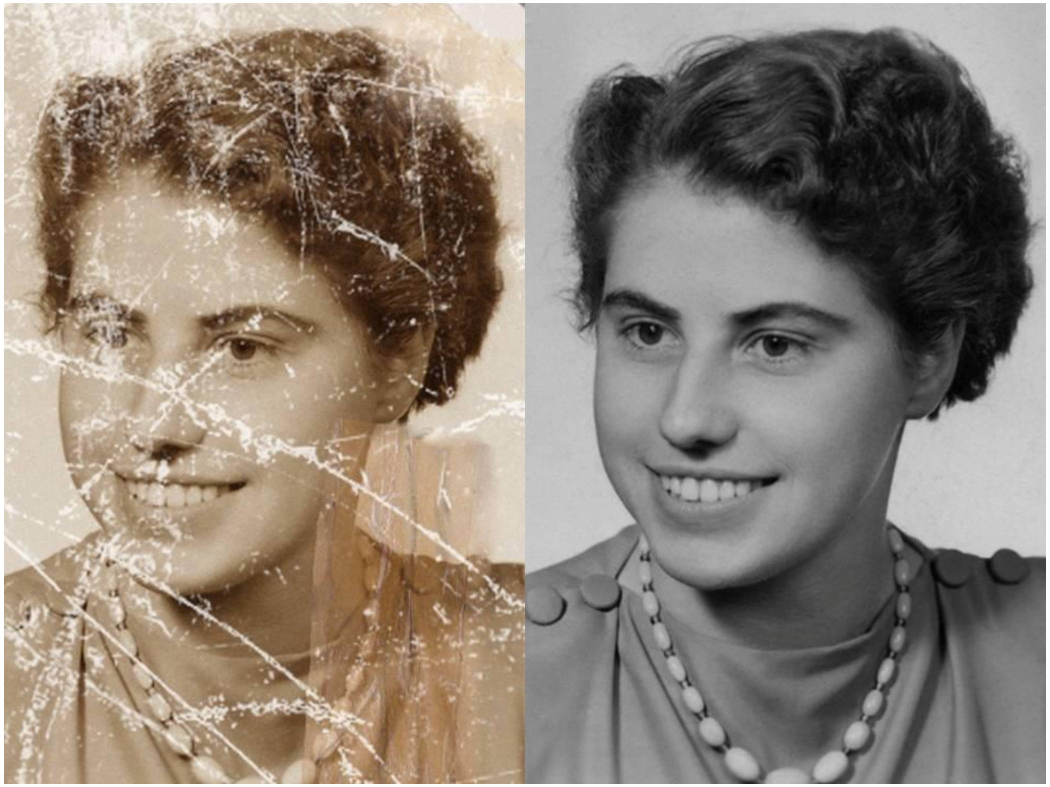
Narrow Eyes

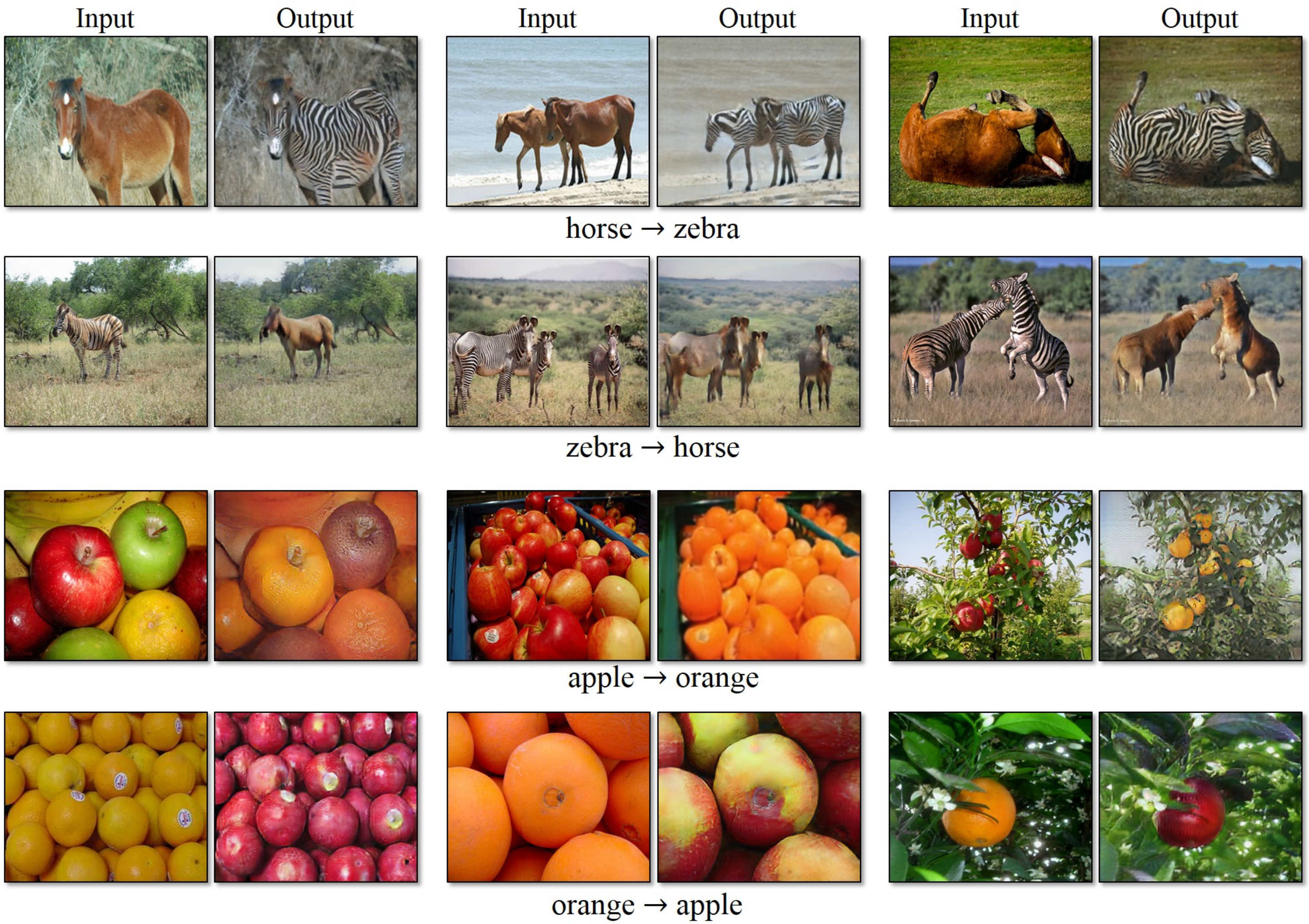
Blonde Hair

Beard


0:00 / 0:04

We can also consider many practical problems defined in terms of learning conditional probability distributions.



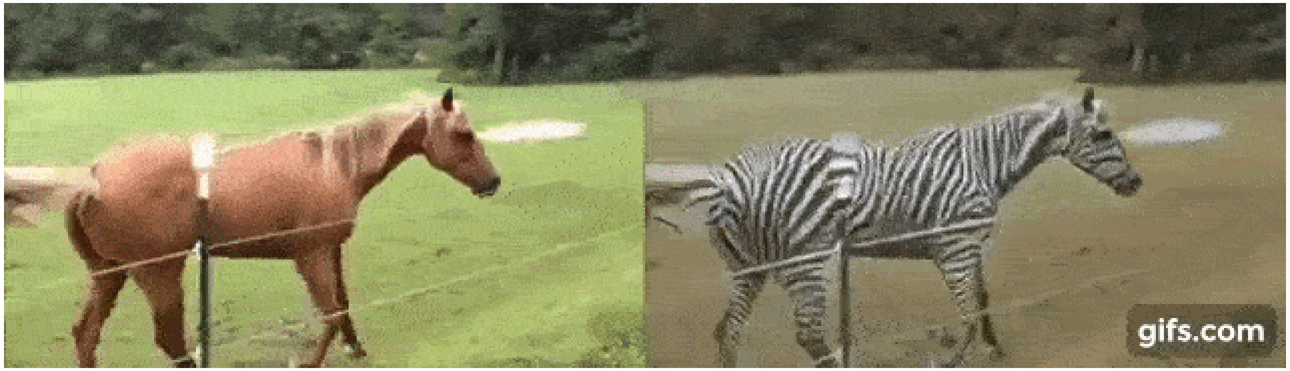


horse → zebra

zebra → horse

apple → orange

orange → apple



gifs.com

Music	do you know a website that you can find people who want to join bands ?
⇒ Science	do you know a website that can help me with science ?
⇒ Politics	do you think that you can find a person who is in prison ?
Music	all three are fabulous artists , with just incredible talent ! !
⇒ Science	all three are genetically bonded with water , but just as many substances , are capable of producing a special case .
⇒ Politics	all three are competing with the government , just as far as i can .
Music	but there are so many more i can &apos;t think of !
⇒ Science	but there are so many more of the number of questions .
⇒ Politics	but there are so many more of the can i think of today .

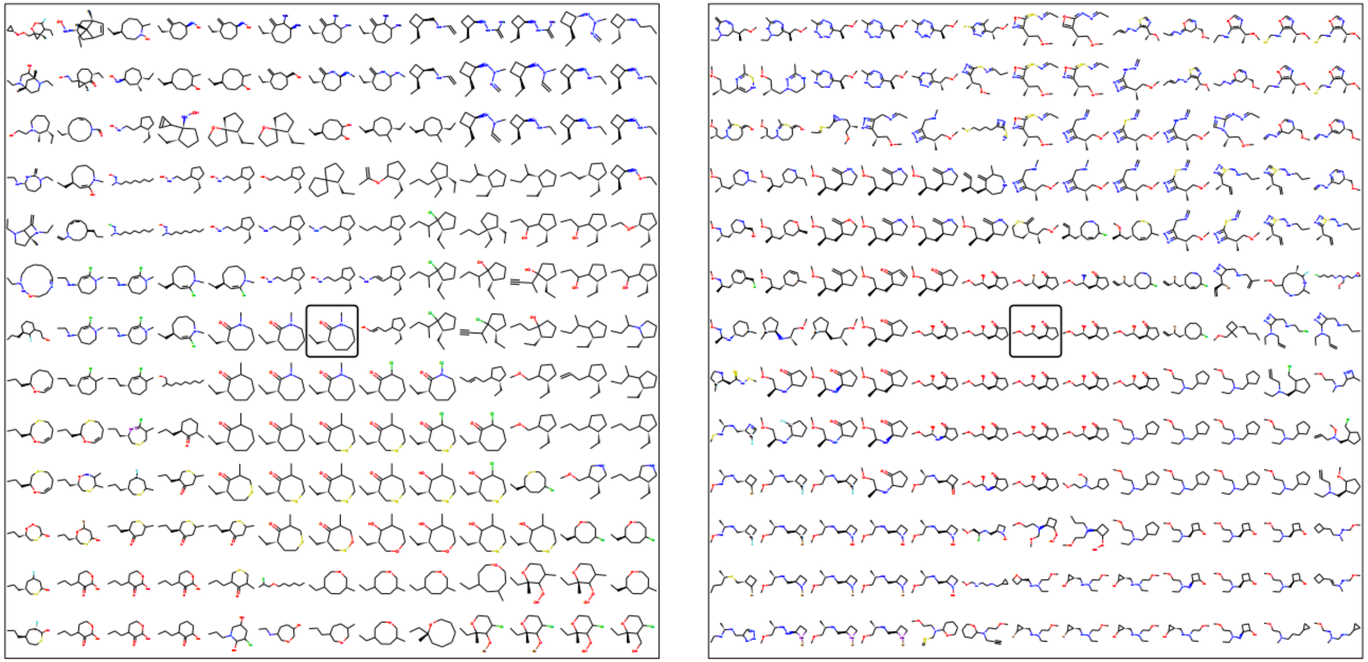


Figure 3. Searching the 56-dimensional latent space of the GVAE, starting at the molecule in the center.

A generative model is a model the approximates some data-generating distribution $P(x)$ where $x \in X$ is from some specified dataset.

A latent variable model is a model that approximates a joint data-generating distribution $P(x, z)$ of datapoints $x \in X$ and latent variables z.

GAN Pseudocode.

Suppose X is a dataset and $P(x)$ is the generating distribution of the dataset.

Let $G_\theta : \mathbb{R}^N \to \mathbb{R}^{3 \times \text{height} \times \text{width}}$ be a neural network with parameters θ.

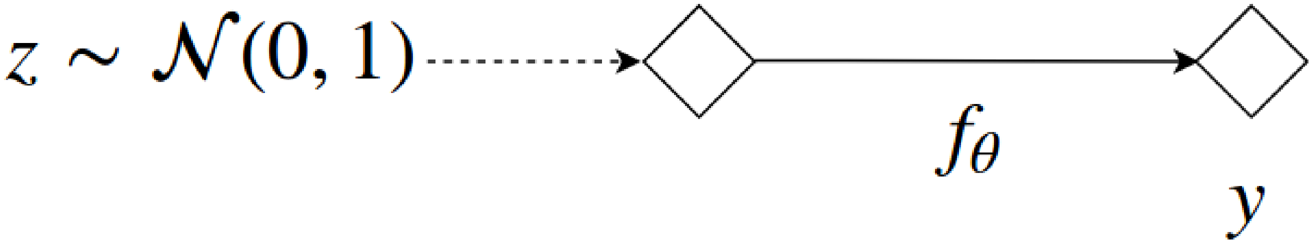
Let $D_\phi : \mathbb{R}^{3 \times \text{height} \times \text{width}} \to]0, 1[$ be a neural network with parameters ϕ.

Sample $z \sim N(0, I)$ and compute $y = G_\theta(z)$.

Sample $x \sim P(x)$.

Train D_ϕ to be good at distinguishing x from y by minimizing $\mathbb{E}_{x \sim P}[\log D_\phi(x)] + \mathbb{E}_{z \sim N(0,I)}[\log(1 - D_\phi(G(z))]$.

Train G_θ to make x more similar to y according to D_ϕ by minimizing $\mathbb{E}_{z \sim N(0,I)}[\log(D_\phi(G(z))]$.


$z \sim \mathcal{N}(0, 1)$
f_θ
y

Latent Variable Model Pseudocode.

Suppose X is a dataset and $P(x, z)$ is the generating distribution of the dataset.

Let $E_\varphi : \mathbb{R}^{3\times\text{height}\times\text{width}} \to \mathbb{R}^N$ be a neural network with parameters θ.

Let $G_\theta : \mathbb{R}^N \to \mathbb{R}^{3\times\text{height}\times\text{width}}$ be a neural network with parameters φ.

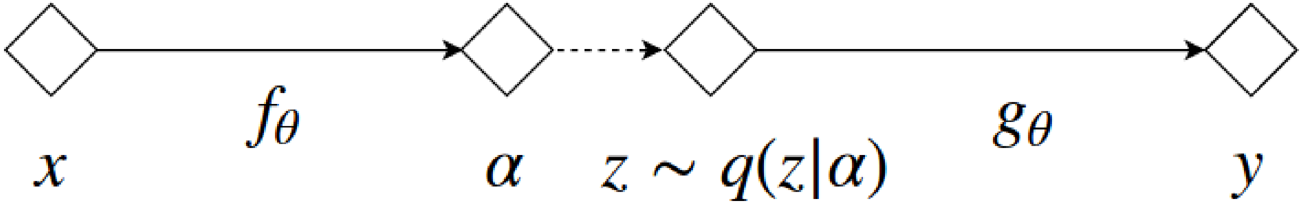
Let $D_\phi : \mathbb{R}^N \to]0, 1[$ be a neural network with parameters ϕ.

Sample $x \sim P(x)$. Sample $z \sim N(0, I)$ and compute $y = G_\theta(E_\varphi(x))$.

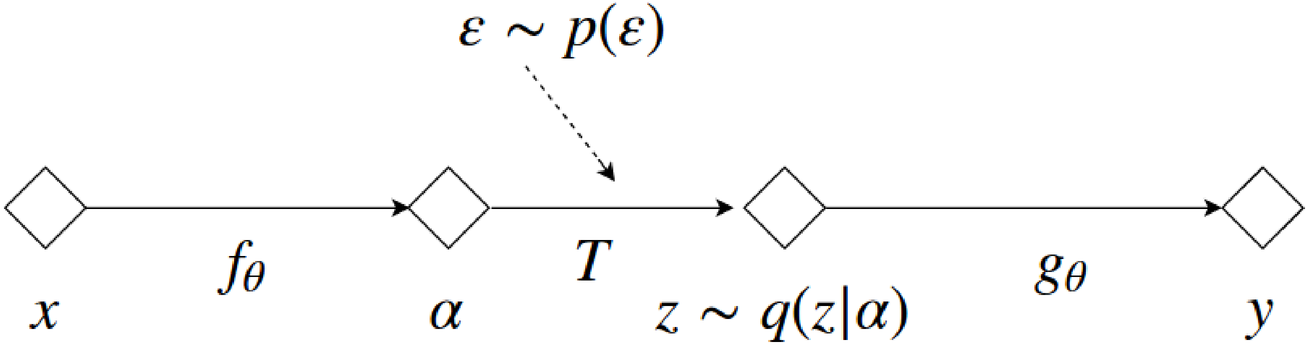
Train D_ϕ to be good at distinguishing z from y by minimizing $\mathbb{E}_{x \sim P}[\log D_\phi(x)] + \mathbb{E}_{z \sim N(0,I)}[\log(1 - D_\phi(G(z))]$.

Train G_θ to make x more similar to y according to a reconstruction loss of your choice.

But sampling is not differentiable!

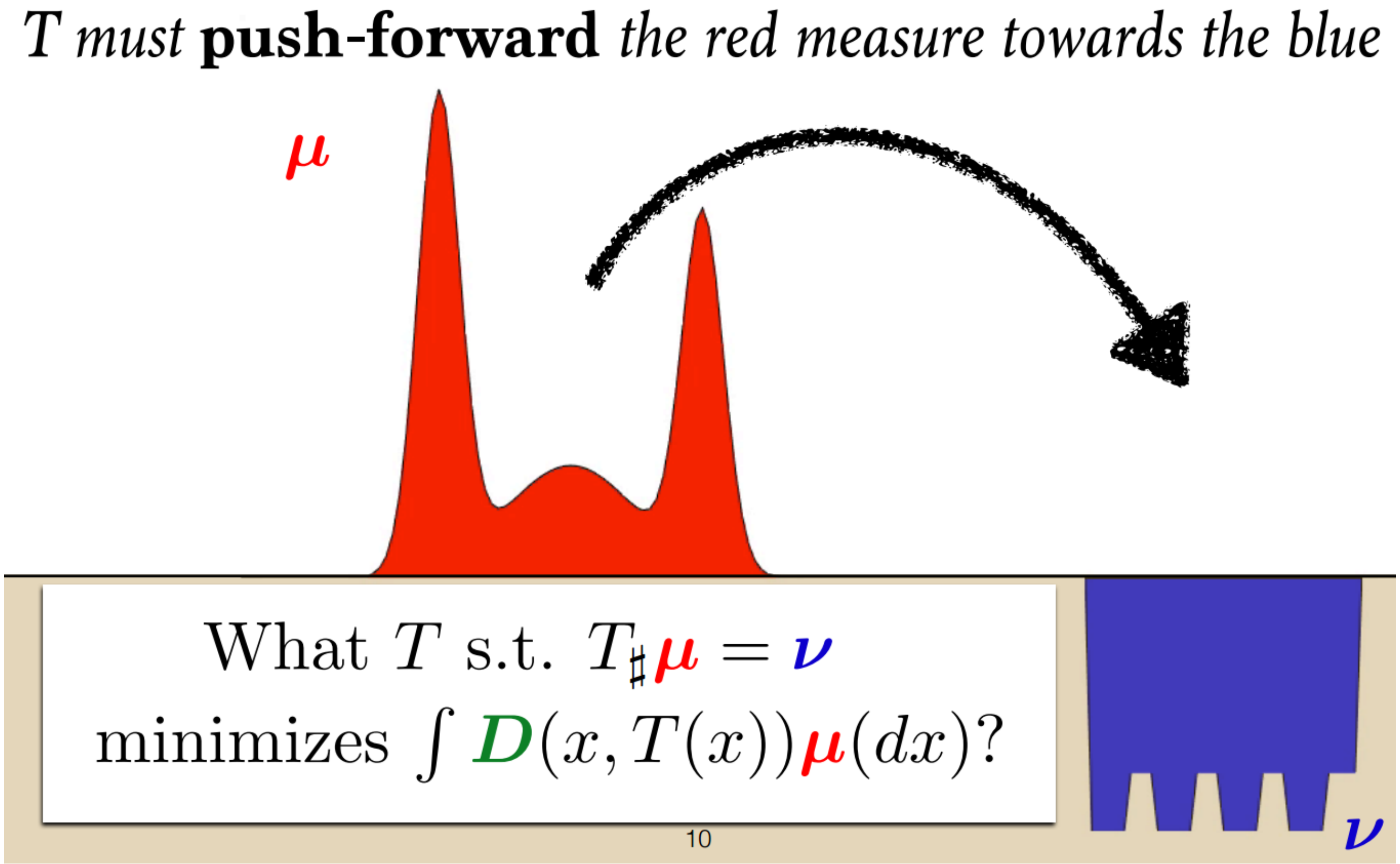


We can reparameterize the sampling procedure!



Why do we want to use Variational Inference?

By describing our neural network in terms of a probability distribution we get a principled way of defining the objective function.



T must **push-forward** *the red measure towards the blue*

What T s.t. $T_\sharp \mu = \nu$

minimizes $\int D(x, T(x))\mu(dx)$?

The amount of transport that needs to be done in the optimal case defines a metric between distributions.

There are many ways of doing variational inference.

One way is to minimize the Wasserstein distance between $q(x, z; \alpha)$ and $p(x, z)$.

$$W_c(p, q) := \inf_{\gamma \in \mathcal{P}(p,q)} \mathbb{E}_{x,y \sim \gamma} \left[\mathcal{L}(x, y) \right]$$

A Reparameterization of the Optimal Transport Problem

When the data-generative model is expressed as a latent-variable model it turns out that we can reparameterize the optimal transport problem to be over the search space of encoding distributions $q(z|x)$, rather than the search space of couplings γ:

$$W_c(p, q) = \inf_{\{q : q(z) = p(z)\}} \mathbb{E}_{z \sim q(z)} \left[\mathcal{L}(x, g_\theta(z)) \right]$$

To obtain a computational solution of this optimization problem, we relax it by introducing a divergence term $D[q(x|z; \alpha) : p(z)]$ to get:

$$W_c(p, q) \approx \inf_{\{q : q(z) = p(z)\}} \mathbb{E}_{z \sim q(z)} \left[\mathcal{L}(x, g_\theta(z)) \right] + \lambda D \left[q(z|x; \alpha) : p(z) \right]$$

Code can be run here

In [ ]:

```
run run_models --xp_id exp0 --dataset mnist
```

In [ ]:

```
run run_models --xp_id exp1 --dataset mnist
```

In [ ]:

```
run run_models --xp_id exp2 --dataset mnist
```

In [ ]:

```
run run_models --xp_id exp3 --dataset mnist
```

In [ ]:

```
run run_models --xp_id exp4 --dataset mnist
```

In [ ]:

```
run run_models --xp_id exp5 --dataset mnist
```

In [ ]:

```
run run_models --xp_id exp6 --dataset mnist
```

In [ ]:

```
run run_models --xp_id exp7 --dataset mnist
```

Limitations

```
- These models are difficult to train on very rich data.
- No obvious way of generalizing to images containing multiple objects.
- Latent variable models are not identifiable.
- Evaluating the models can be difficult.
```

Part II - Discrete Optimization.

Learning Objectives.

- Understand how to identify discrete optimization tasks?
- How can we use neural networks to solve tasks involving discrete variables?
- Understand what Reinforcement Learning is and why it is a difficult problem

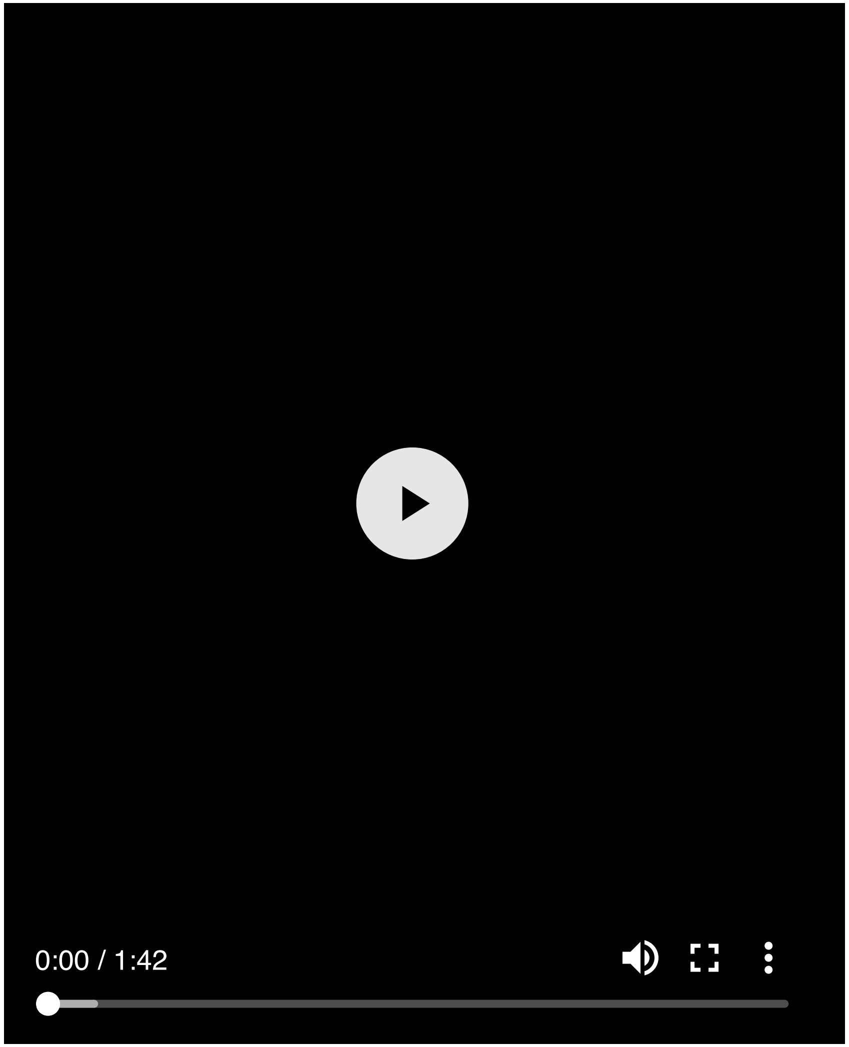
What is discrete optimization?

By discrete optimization we refer to optimization problems that involve discrete variables.

In discrete optimization we consider finding the optimal discrete mapping for some given problem.

Examples.

- Playing Atari Breakout.
- Buying or selling stocks.
- Solving the Traveling Salesman Problem.


0:00 / 1:42

We can think about these kinds of problems as problems involving probability distributions that are called decisision processes.

Let $\mathcal{S}$ be a set of states and let $\mathcal{A}$ be a set of actions.

Reinforcement Learning - Definition

Reinforcement Learning is the problem of optimizing the parameters of some probabilistic model to approximate a policy distribution with respect to a sum of expected reward specified by some reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

Why are these problems discrete?

In all the above cases we want to learn a mapping from some sequence of observations $(X_t)_{t=1}^N$ to a finite set of actions.

Atari Breakout

$$f : (x_t)_{t=1}^N \rightarrow \{\text{Left, Wait, Right}\}$$

Buy or sell stocks

$$f : (x_t)_{t=1}^N \rightarrow \{\text{Buy, Wait, Sell}\}$$

Travelling Salesman

Let
$A = \{\text{Reykjavík, Akureyri, Egilsstaðir, Stykkishólmur, Melrakkaslétta, Landmannalau}$
be a set of places to visit.

Given a sequence of places ${x_t}_{i=1}^{N} \in A$ we want to learn the function.

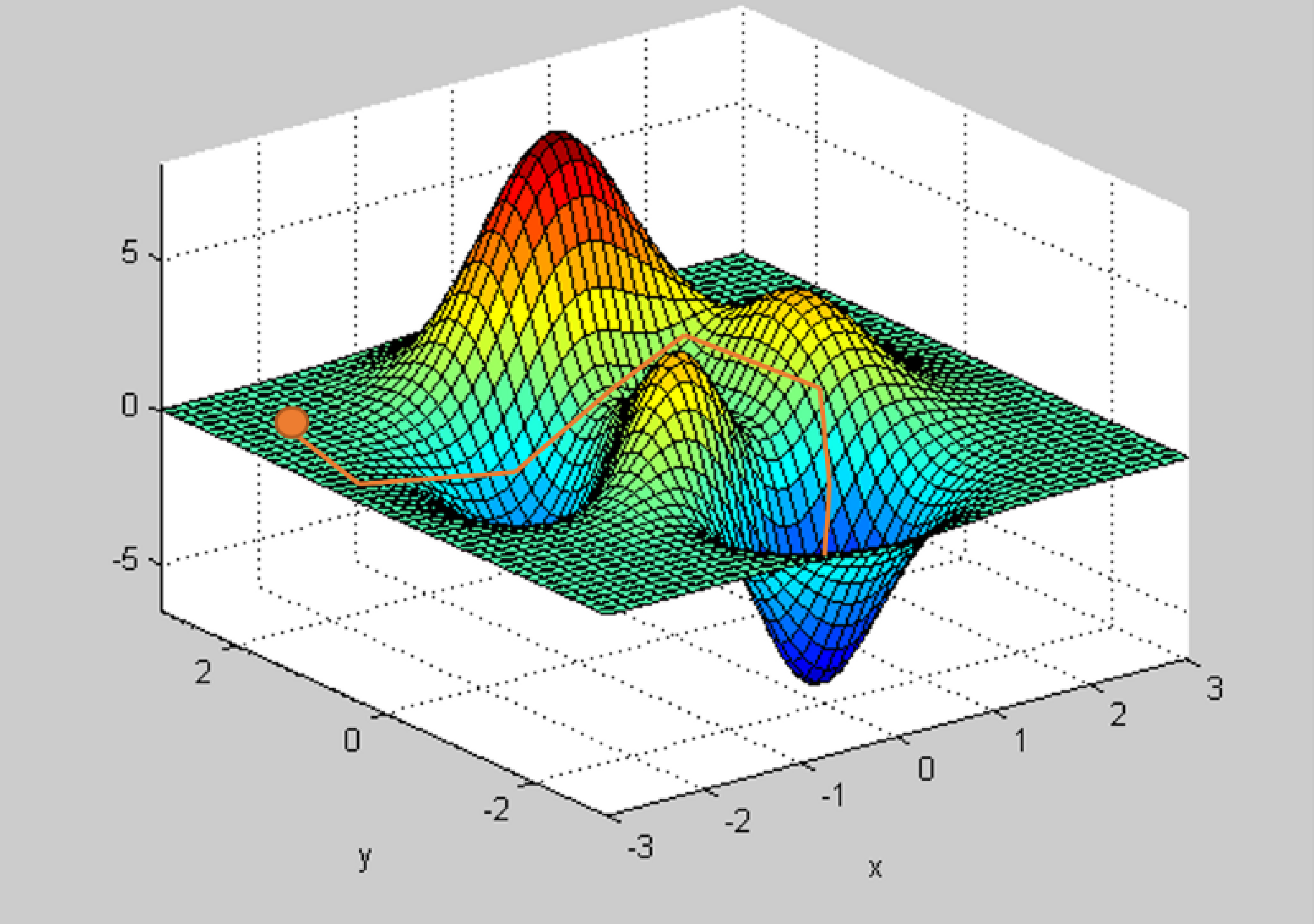
$$f : (x_t)_{t=1}^{N} \to x \in A \backslash \{x_1, \ldots x_N\}$$

Such that the total distance of our journey is as small as possible, i.e.

$$\sum_{t=1}^{N} d(x_t, x_{t+1})$$

Why are these problems challenging for neural networks to solve?

The success of neural networks relies on the fact that we can create good gradient estimators for continuous functions.


5
0
-5
2
0
-2
y
3
2
1
0
-1
-2
-3
x

This allows us to update the parameters of our model towards optimal values of our objective function.

$$\theta_{t+1} = \theta_t + \eta \nabla_{\theta_t} \mathbb{E}_{y \sim P_x, \hat{y} \sim Q}[\mathcal{L}(y, \hat{y})]$$

The challenge in making our neural networks approximate discrete functions is that discrete functions are not differentiable.

However discrete distributions have continuous parameter spaces!

Example:

Let $p \in [0, 1]$.

We say that $X \sim Ber(p)$ if $X = 1$ with probability p and $X = 0$ otherwise.

What if we could get gradients as easily as we do for continuous latent-variables?

We could answer discrete questions, such as.

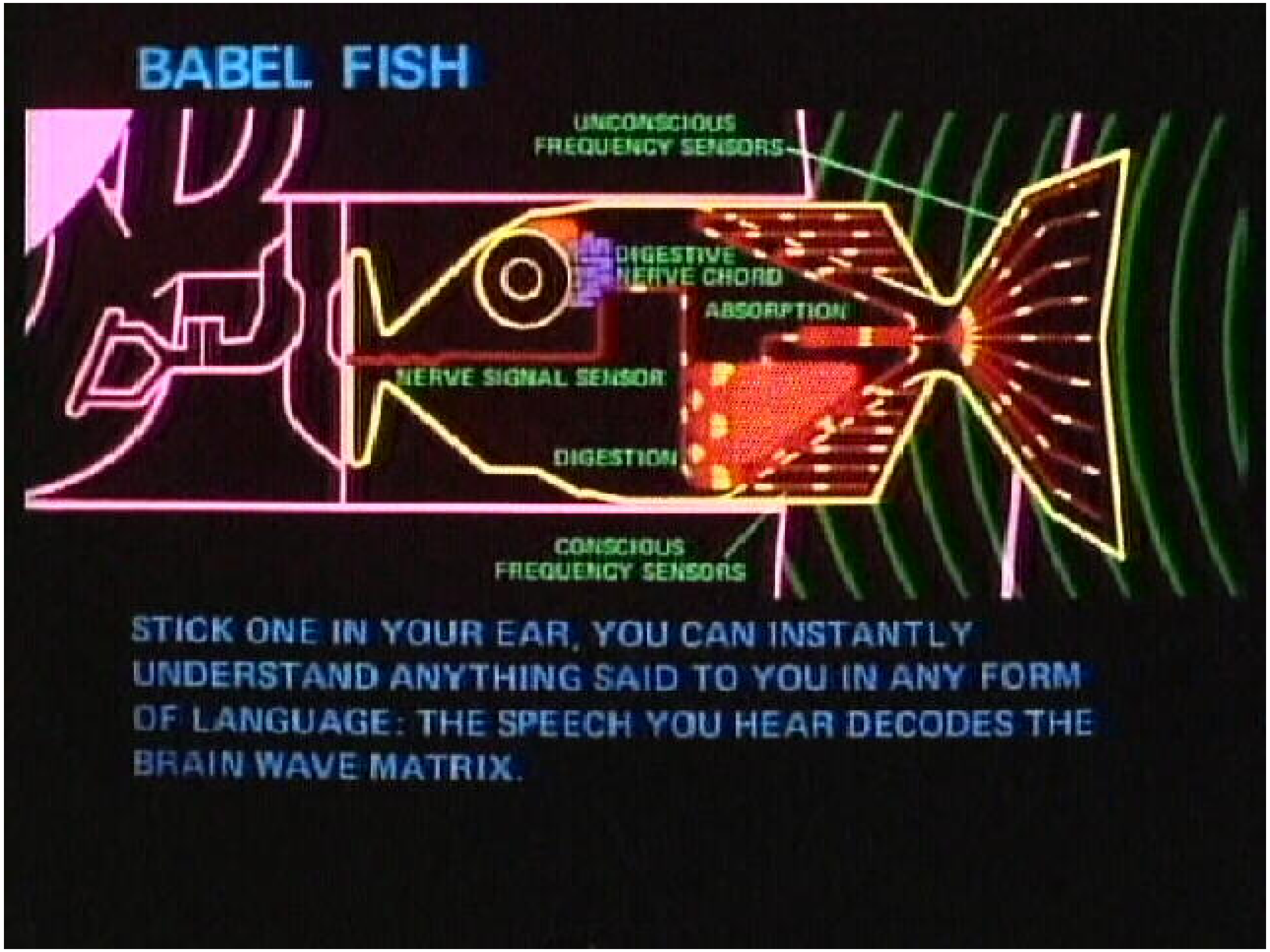
```
- "In which pixel coordinates should I form a bounding box?".

- "How many objects are there in my image?".

- "As a program which of these functions should I try to execute?".

- "What type of a graph should I construct?"
```

- "Should I stop decoding this phrase and wait for further input?".



STICK ONE IN YOUR EAR, YOU CAN INSTANTLY UNDERSTAND ANYTHING SAID TO YOU IN ANY FORM OF LANGUAGE: THE SPEECH YOU HEAR DECODES THE BRAIN WAVE MATRIX.

(If time allows)

A gradient estimator is a functional $\hat{g}$ which has the property of being an estimate of,

$$\nabla_\theta \mathbb{E}_{x \sim p}\left[f(x)\right]$$

i.e.

$$\hat{g}\left[f(x)\right] \approx \nabla_\theta \mathbb{E}_{x \sim p}\left[f(x)\right]$$

We say that a gradient estimator is unbiased if,

$$\mathbb{E}_{x \sim p}\left[\hat{g}\left[f(x)\right]\right] = \nabla_\theta \mathbb{E}_{x \sim p}\left[f(x)\right]$$

The most simple way (REINFORCE). Doesn't assume anything about the function f.

$$\nabla_\theta \mathbb{E}_{p(x|\theta)}\left[f(x)\right] = \int_{\mathcal{X}} \nabla_\theta p(x|\theta) f(x)\, dx$$

$$= \int_{\mathcal{X}} p(x|\theta) \nabla_\theta \log p(x|\theta) f(x) dx$$

$$= \mathbb{E}_{p(x|\theta)}[\nabla_\theta \log p(x|\theta) f(x)]$$

$$= \mathbb{E}_{p(x|\theta)}[\nabla_\theta \log p(x|\theta) f(x)]$$

$$= \mathbb{E}_{p(x|\theta)}\left[f(x) \nabla_\theta \log p(x|\theta)\right]$$

.

And the last equality is obtained since f does not depend on θ.

This gradient estimator is one of the most basic estimators we could define and is known for having high variance.

The pathwise derivative to the rescue!

Define a reparameterization of the samples z in terms of an invertible differentiable function, T, and sample $\epsilon \sim p_\epsilon$ by,

$$z = T(\epsilon, \alpha), \qquad \epsilon \sim p_\epsilon$$

The choice of the function T depends on the functional form of the variational distribution.

This allows us to get an unbiased gradient estimator,

$$\nabla_\alpha \mathbb{E}_{z \sim q(z|x:\alpha)}[f(z)] = \mathbb{E}_{\epsilon \sim p_\epsilon}[\nabla_\alpha f(T(\epsilon, \alpha))].$$

Example: If q is a normal distribution then the variational parameters $\alpha = \mu, \sigma^2$, and we can choose

$$T(\alpha, \epsilon) = \mu + \epsilon \odot \sigma$$

Thanks!