Generating & Designing DNA with Deep Generative Models

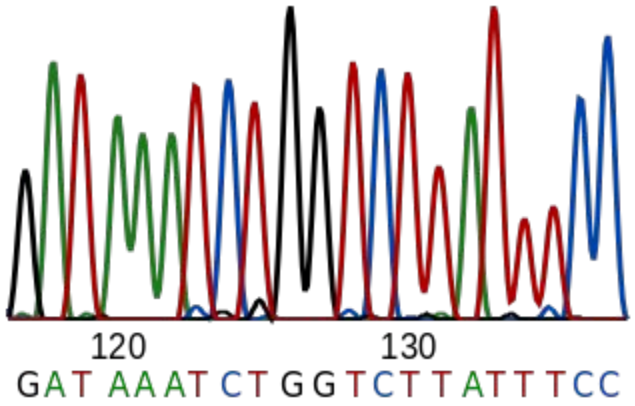
Nathan Killoran, Leo J. Lee, Andrew Delong, David Duvenaud, Brendan J. Fey

Presented by Yomna Omar on February 16th, 2018

What is DNA sequencing?

The process of determining the order of *nucleotides* within a DNA molecule

Basically, we want to figure out the order of the four bases that appear in a strand of DNA: Adenine (A), Guanine (G), Cytosine (C), and Thymine (T)



What is DNA sequencing data similar to?

It can be thought to have similar properties as natural language data and computer vision data:

- Like language, it has discrete sequences of characters (A, G, C, and T)
- Like vision, it has regularly appearing patterns placed on noisy backgrounds

Where do generative models fit in?

They are able to construct DNA sequence data with the desired structure and properties

Examples of models include:

- GANs
- VAEs
- Deep Autoregressive Models
- Activation Maximization
- Style Transfer

3 complementary methods to synthesize DNA

- A GAN-based deep generative network
- A variant of the activation maximization method
- A joint method which combines the previous two in a single architecture

By capturing the underlying structure of the training dataset, these models are able to generate DNA sequences with the desired structure

1. Generative Adversarial Networks (GANs)

to learn the underlying structure of a given dataset of DNA sequences and generate realistic data

First Test: Exploring the Latent Encoding of DNA sequences

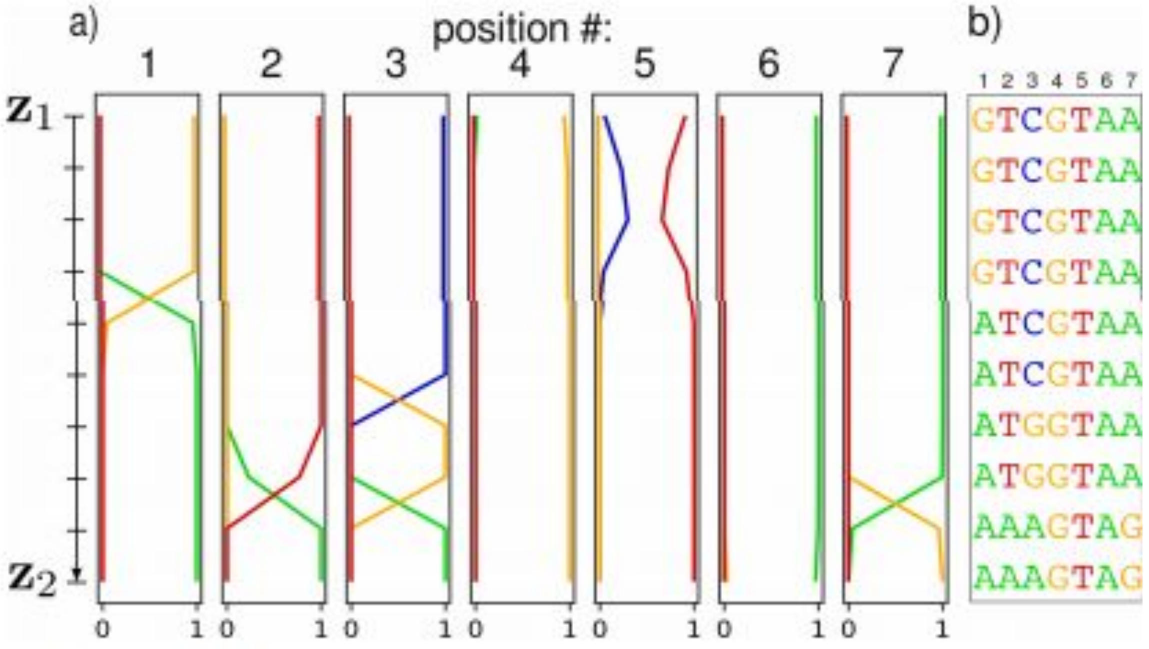
By training your model on a large dataset, you can analyze what it has learned about the latent space through manipulations

A dataset of 4.6M sequences, each of length 50, has been trained. Each sequence contains the hg38 chromosome

▷ Latent interpolation
▷ Latent complementation
▷ Distance to training sequences

Latent Interpolation

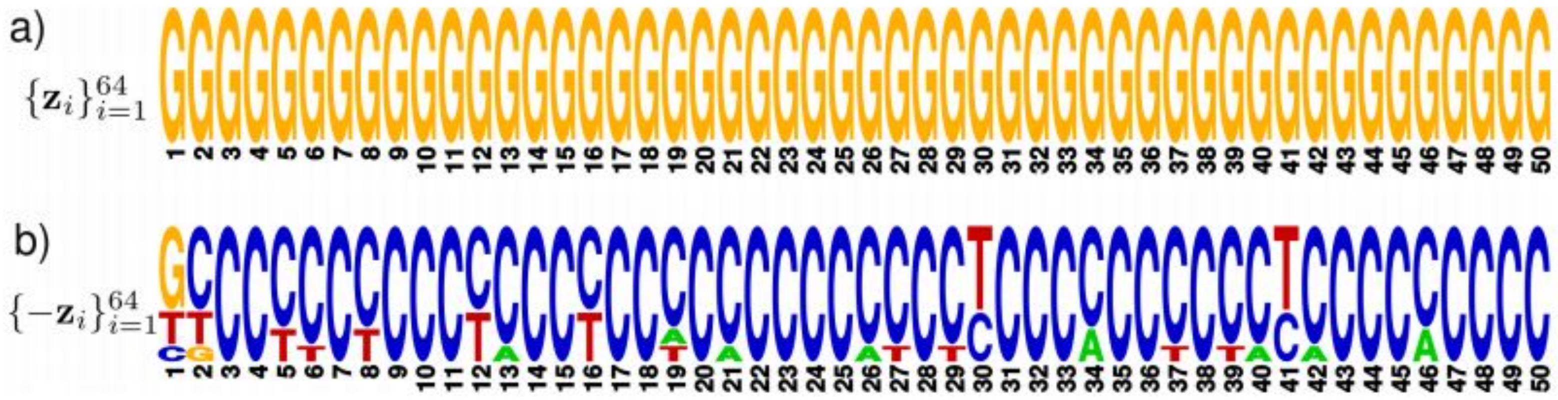
By interpolating points in latent space, it can be shown how the data generated varies between any two points, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$



Latent Complementation

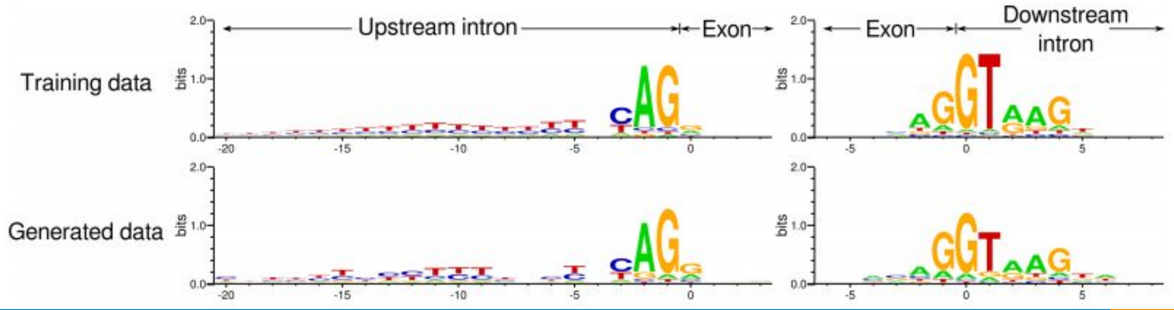
Since we know there are several latent vectors that generate the same data:

1. Fix a DNA sequence ***x*** and 64 different points in the latent space, ***z_1, z_2, …, z_{64}***, that generate ***x***, ***$G(z_i) = x$***
2. Reflect each of the latent points and decode the generated sequence
3. The complement/reflection of ***x*** is ***$G(-z_i)$***



Second Test: Capturing Exon Splice Site Signals

- Used a dataset of 116k sequences, each of length 500
- Each sequence contains exactly one exon
- Each exon varies in length between 50 and 400
- To track exon positions, introduce a flag in the train set
- Any positions that is part of an exon is flagged as 1, otherwise, as 0
- The model then learns the positions of an exon by capturing statistical data about the exon borders (splice sites) from the training samples



2. The Activation Maximization Method & the Joint Method

to design DNA sequences that exhibit a desired property or multiple properties (can be contrasting)

Activation Maximization (AM)

- Using GANs, we can generate realistic-looking data
- Using AM, we will generate DNA sequences that have properties we desire
- That is, AM is an optimization problem and not a modelling problem
- For images, AM uses a classifier network to create images that belong to a certain class ***in reverse***
- In reverse, meaning we generate ***p(x|c)*** using ***p(c|x)*** and ***p(x)***

AM

- A predictor function ***P*** that takes data as input and produces some target property $t = P(x)$ as output
- $P(x)$ is some combination of known, explicitly coded functions f_i and parameterized neural network learned functions $f_{\theta i}$:

$$P(x) = \Sigma_i \alpha_i f_i(x) + \Sigma_j \beta_j f_{\theta j}(x)$$

- α_i and β_j are fixed weights that indicate the influence of each property
- Using any input x, calculate the gradient $\nabla_x t$ with a small step size ϵ:

$$x = x + \epsilon \nabla_x t$$

- The original input x increases/decreases t towards its desired/optimal values

AM for DNA

▷ Obtain a continuous relaxation of one-hot sequence vectors by adding a simple unstructured latent variable ***z*** of shape (***L***, 4), same as the data encoding

▷ Transform z into a sequence using a simple softmax pre-layer:

$$x_{ij} = \frac{exp(z_{ij})}{\Sigma_{k=1}^{4} exp(z_{ik})}$$

▷ x_{ij} is then used as an input to the predictor function ***P***

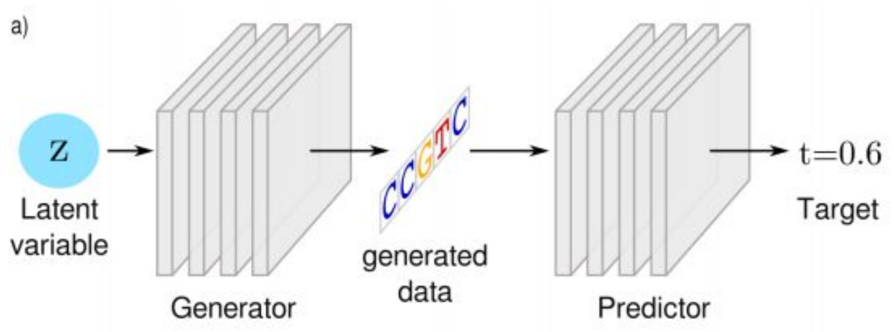
▷ Gradients are calculated wrt ***z*** rather than ***x***:

$$z = z + \epsilon \nabla_z t$$

▷ The derived distributions can be interpreted as discrete sequences by taking the argmax at each position

The Joint Method

- AM is often unrealistic because it favors the optimization of attributes
- **Solution:** Combine AM with a generative model

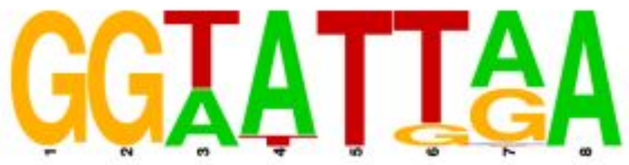


- Generator ***G*** which transforms latent codes ***z*** into synthetic data ***x***
- Predictor ***P*** maps ***x*** to its target attributes ***t = P(x)***
- ***G*** and ***P*** are plugged back-to-back to form $z \longrightarrow x \longrightarrow t$

$$\nabla_{\mathbf{z}}\mathbf{t} = \sum_i \frac{\partial \mathbf{t}}{\partial \mathbf{x}_i}\frac{\partial \mathbf{x}_i}{\partial \mathbf{z}} = \sum_i \frac{\partial P(\mathbf{x})}{\partial \mathbf{x}_i}\frac{\partial G_i(\mathbf{z})}{\partial \mathbf{z}}, \qquad z = z + \epsilon \nabla_z t$$

Explicit Predictor: Motif Matching I

▷ A motif of length ***K*** is represented by a ***K x 4*** position weight matrix (PWM)

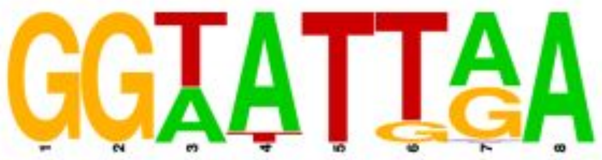


▷ The predictor function has two stages:

a. Scans across the data using a 1D convolution, computing the inner product of a fixed PWM with every length-K subsequence of the data
b. Selects the single convolutional output with the highest value to obtain the sequence's final score

▷ A sequence will have high score as long as it has a single subsequence which has a large inner product with the chosen PWM

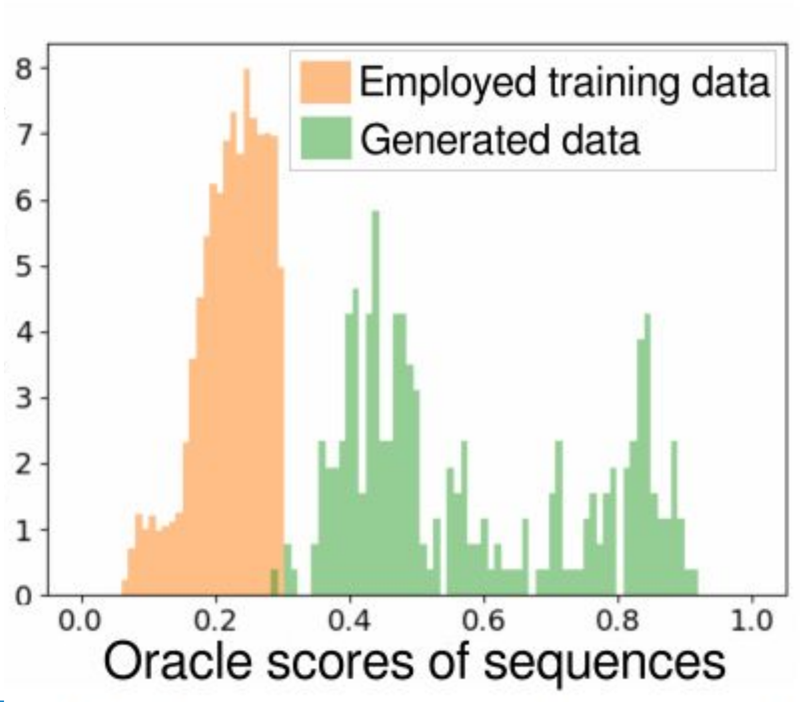
Explicit Predictor: Motif Matching II



```
TGAGAGTGATGTATTGGAATTGATGCCTCACCTCTGCTTGCAGACTGTCA
GGAATGAACTGGGGAGACAGGCCCAGAGGAATTGAGAAAGTAATGAGCAC
GCCCTGGGTTTTAAGAAATACTGTTGCATCAGGGCAAATGTAAGATTTTG
TTTTGTTTGAGATCTGTGGGGTATGCTGGAATTAAAGTCTGGACTACCAC
CTGATACTGAATGCAGATTTGAAGAACAAAGGGTATTAAAACACATGCTT
GATCCCCAAGTGTGGAATTGAGAAGGAAGCTGGAGAATCCCCAAACTCTG
CAGCCACATCAGCTTACCTAAGGAAGTGATGTGTTTTAAAACCAGCTTTG
TAGAATTTTTCTTGGTATTAATGATGATCTAGGCTTACACAGGGACATCA
GACATTGCTTAGTCTGAGGGATACAGTGGGGAGTGGGTATTAAAATCTCC
ACATGCCTGAGACATTCCTGCTCTTGAATCTGAGGAATTATGCTTAATCC
```

Learned Predictor: Protein Binding

- Use an oracle model trained on the original data to query newly designed sequences and determine their binding score
- Dataset contained the original sequences along with their oracle-based scores



Optimizing Multiple Properties

- **Goal:** design DNA sequences that bind to one protein in a family but not the other
- Many sequences can be designed with characteristics that generalize well beyond the explicit content of the training data
- This is because the two predictors used can capture the same structure, differing only in subtle ways
- Simply put, the joint model explores the subtle differences between the desired properties and uses that to generate the required DNA sequences

Generating & Designing DNA with Deep Generative Models

Thank you

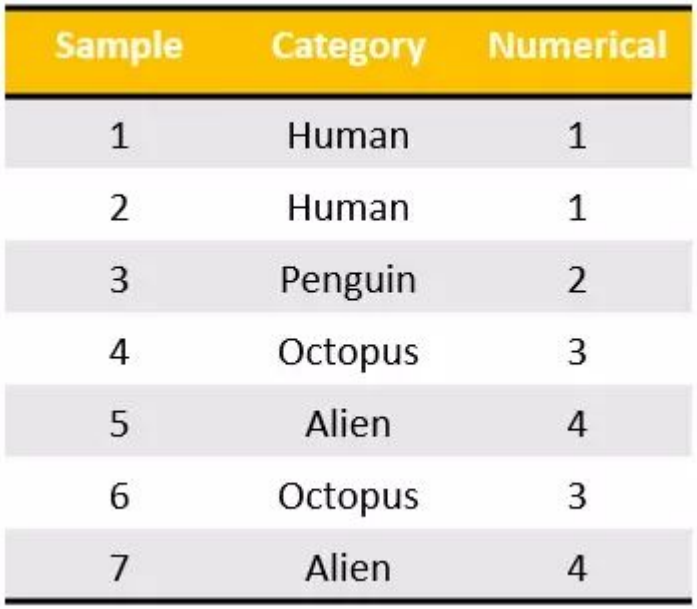
References

[1] N. Killoran, L. Lee, A. Delong, D. Duvenaud and B. Frey, “Generating and designing DNA with deep generative models”, 2017. From: arxiv.org/pdf/1712.06148.pdf

[2] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. “Predicting the sequence specificities of dna-and rna-binding proteins by deep learning”. *Nature biotechnology*, 33(8):831–838, 2015

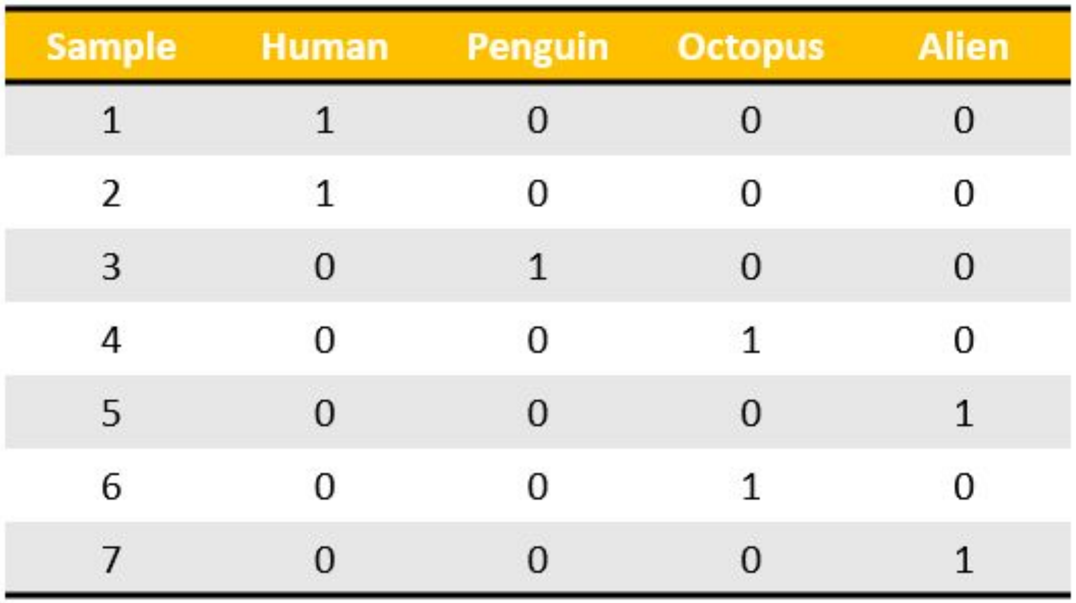
Appendix: One-Hot Encoding

Nominal Encoding



Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

One-Hot Encoding



Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1