

# Programming Generative AI: From VAEs to Stable Diffusion with PyTorch and Hugging Face



**Jonathan Dinu (@jonathandinu)**

Author, Artist, and Content Creator

**Lesson 1: The What, Why, and How of Generative AI**

**Lesson 2: PyTorch for the Impatient**

**Lesson 3: Latent Space Rules Everything Around Me**

**Lesson 4: Demystifying Diffusion**

**Lesson 5: Generating and Encoding Text with Transformers**

**Lesson 6: Connecting Text and Images**

**Lesson 7: Post-Training Procedures for Diffusion Models**

**Author —** *Programming Generative AI*. Addison-Wesley, 2025

**Artist —** [jonathandinu.com](http://jonathandinu.com)

**YouTube —** <https://www.youtube.com/c/jonathandinu>

- Deep Learning and Generative AI Tutorials

# Getting the Materials

<https://github.com/jonathandinu/programming-generative-ai>

**Lesson 1 is a little slide heavy**

# Disclaimer

**Feel free to jump to lesson 2 if you  
are itching for some Python coding  
(and come back to this lesson as needed)**



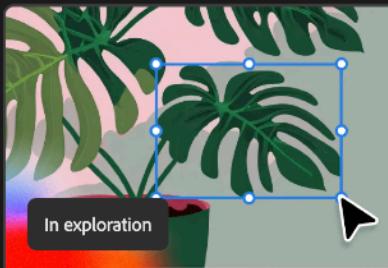
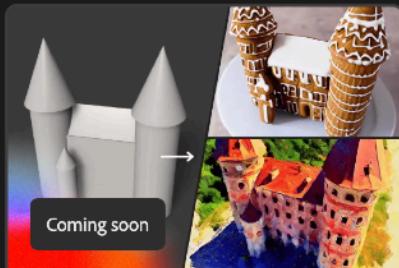
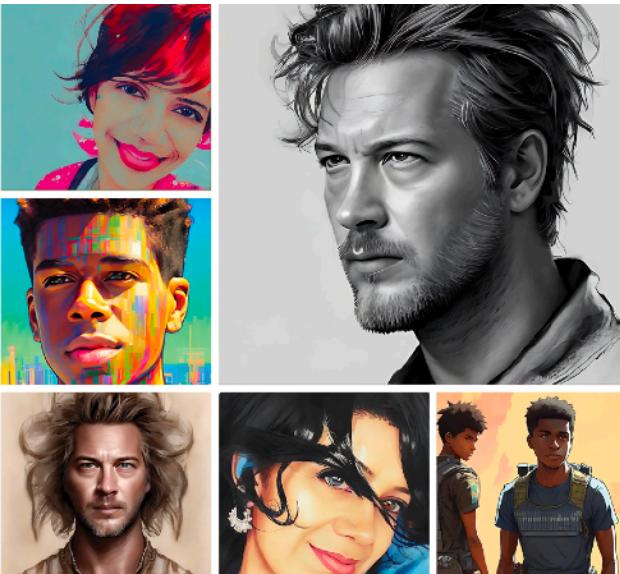
# **Lesson 1: The What, Why, and How of Generative AI**

- 1.1 Generative AI in the Wild**
- 1.2 Defining Generative AI**
- 1.3 Multitudes of Media**
- 1.4 How Machines Create**
- 1.5 Formalizing Generative Models**
- 1.6 Generative versus Discriminative Models**
- 1.7 The Generative Modeling Trilemma**
- 1.8 Introduction to Google Colab**

1.1

## **Generative AI in the Wild**

# Image Synthesis



# Image Editing and Enhancement



**Image to Image**  
Transform any image with a text prompt.



**Expand Image**  
Expand the edges of any image.



**Frame Interpolation**  
Turn a sequence of images into an animated video.



**Erase and Replace**  
Reimagine and remix any part of any image.



**Infinite Image**  
Expand an image by generating outside the original canvas.



**Backdrop Remix**  
Give any photo infinite backgrounds.



**Image Variation**  
Generate new variations of an image.



**3D Texture**  
Generate 3D textures with text prompts.



**Inpainting**  
Remove people and things from videos.



**Text to Color Grade (LUT)**  
Color grade videos with text prompts.



**Super-Slow Motion**  
Turn any video into smooth slow motion.



**Blur Faces**  
Automatically blur faces in videos.

# Image and Video Enhancement

## Upscale & Enhance



Watch Premiere Gal  
upscale in Topaz Video AI



## Stabilization

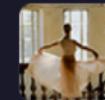


See results of Stabilization  
AI in Topaz Video AI



Learn more →

## Motion Interpolation



See results of Chronos  
AI Model in Topaz Video AI



## Deinterlacing



See results of Dione  
AI Model in Topaz Video AI



# Generative Text

User help me write a short note to introduce myself to my neighbor

ChatGPT Hi there,

My name is [Your Name] and I just moved in next door. I wanted to reach out and introduce myself. I'm looking forward to getting to know you and the neighborhood. If you ever need anything or just want to chat, please don't hesitate to stop by or give me a call.

Best,  
[Your Name]

User can you make it more formal?

ChatGPT Dear Neighbor,

I am writing to introduce myself. My name is [Your Name] and I have recently moved in next door. I wanted to take a moment to reach out and say hello. I am looking forward to getting to know you and the neighborhood.

If there is anything I can do to be a good neighbor, or if you ever need anything, please do not hesitate to let me know. I am happy to help in any way that I can.

Sincerely,  
[Your Name]

User can you send it for me?

ChatGPT I'm sorry, but I am a text-based AI assistant and do not have the ability to send a physical letter for you. I can only provide suggestions and advice on what to write. It is up to you to take that advice and send the letter yourself.

# Generative Code

The screenshot shows the GitHub Copilot interface integrated into a code editor. On the left, the GitHub Copilot chat window displays a message from user 'monalisa' asking for unit test functions for the selected code. The Copilot has generated the following Python code:

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2),
                            float(-34.01), 'USD'),
                           (datetime.datetime(2023, 1, 3),
                            float(2.59), 'DKK'),
                           (datetime.datetime(2023, 1, 3),
                            float(-2.72), 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_comments(self):
        expenses_string = '# This is a comment\n2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2),
                            float(-34.01), 'USD'),
                           (datetime.datetime(2023, 1, 3),
                            float(2.59), 'DKK'),
                           (datetime.datetime(2023, 1, 3),
                            float(-2.72), 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

if __name__ == '__main__':
    unittest.main()
```

A note below the generated code states: "The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format."

On the right, the main code editor window shows the file 'module.py' with the following code:

```
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, amount, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
    2023-01-02 -34.01 USD
    2023-01-03 2.59 DKK
    2023-01-03 -2.72 EUR
    """
    expenses = []

    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                        float(value),
                        currency))
    return expenses

expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''
```

At the bottom, status information includes: Ln 17, Col 3, Spaces: 2, UTF-8, LF, { } TypeScript.

# Voice Cloning

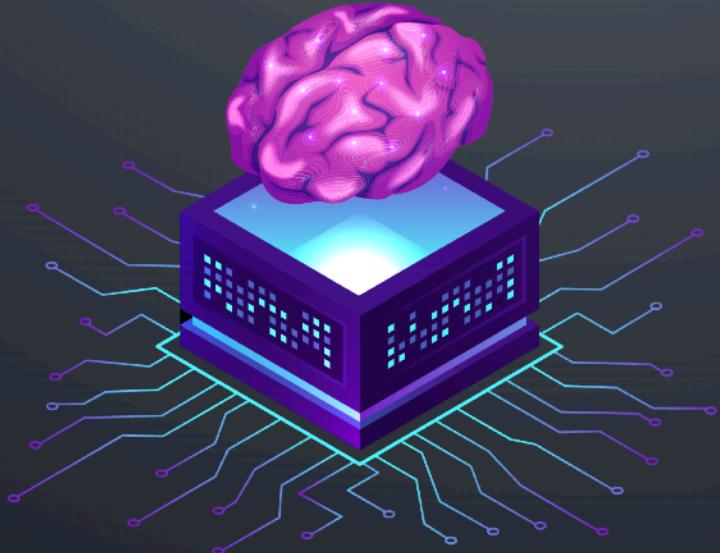
## Custom AI Text to Speech For Streamers

Enhance your Twitch or YouTube livestream with TTS powered by AI.

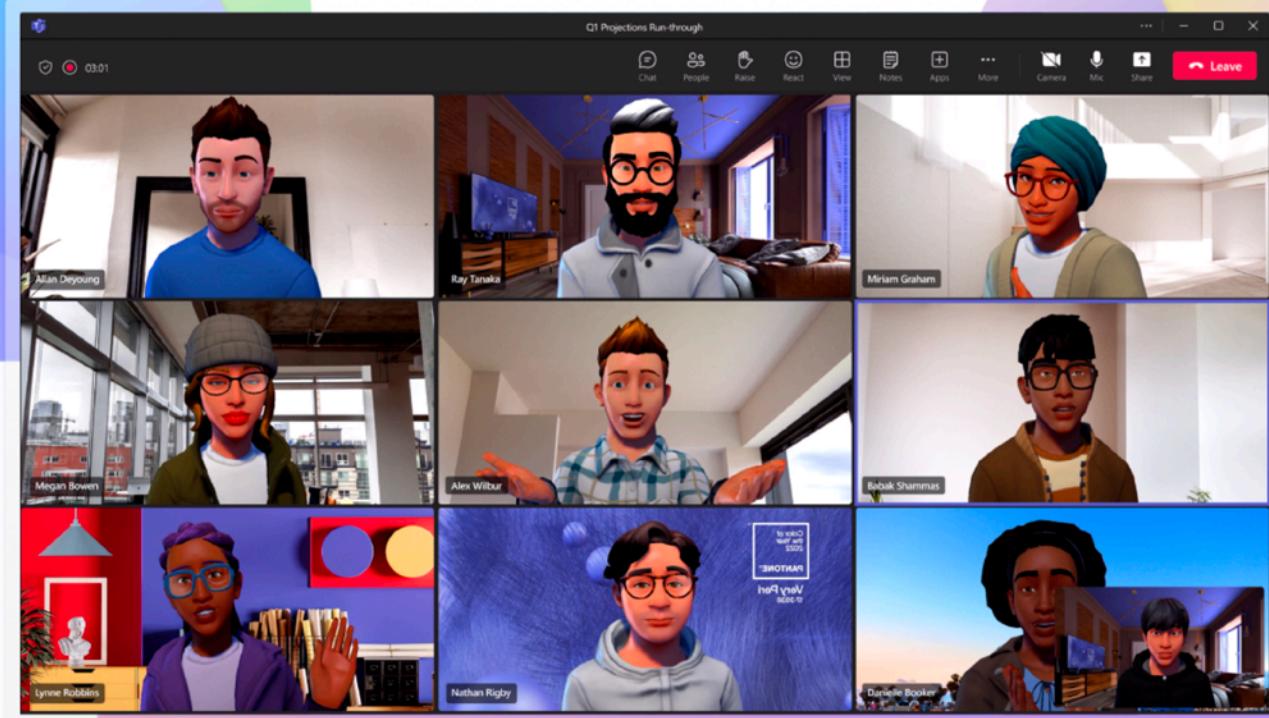
Trusted and used by thousands of creators.

Completely free.

→ Get Started



# Virtual Avatars



# 1.2

## Defining Generative AI



I know it when I see it

# What is Generative AI?

*A computer program that produces unstructured output samples according to a learned data distribution.*

- Jonathan Dinu

# What is Generative AI?

A computer program that **produces** unstructured output samples according to a learned data distribution.

- Jonathan Dinu

# What is Generative AI?

A computer program that **produces unstructured output samples** according to a learned data distribution.

- Jonathan Dinu

# What is Generative AI?

A computer program that **produces unstructured output samples** according to a **learned data distribution.**

- Jonathan Dinu

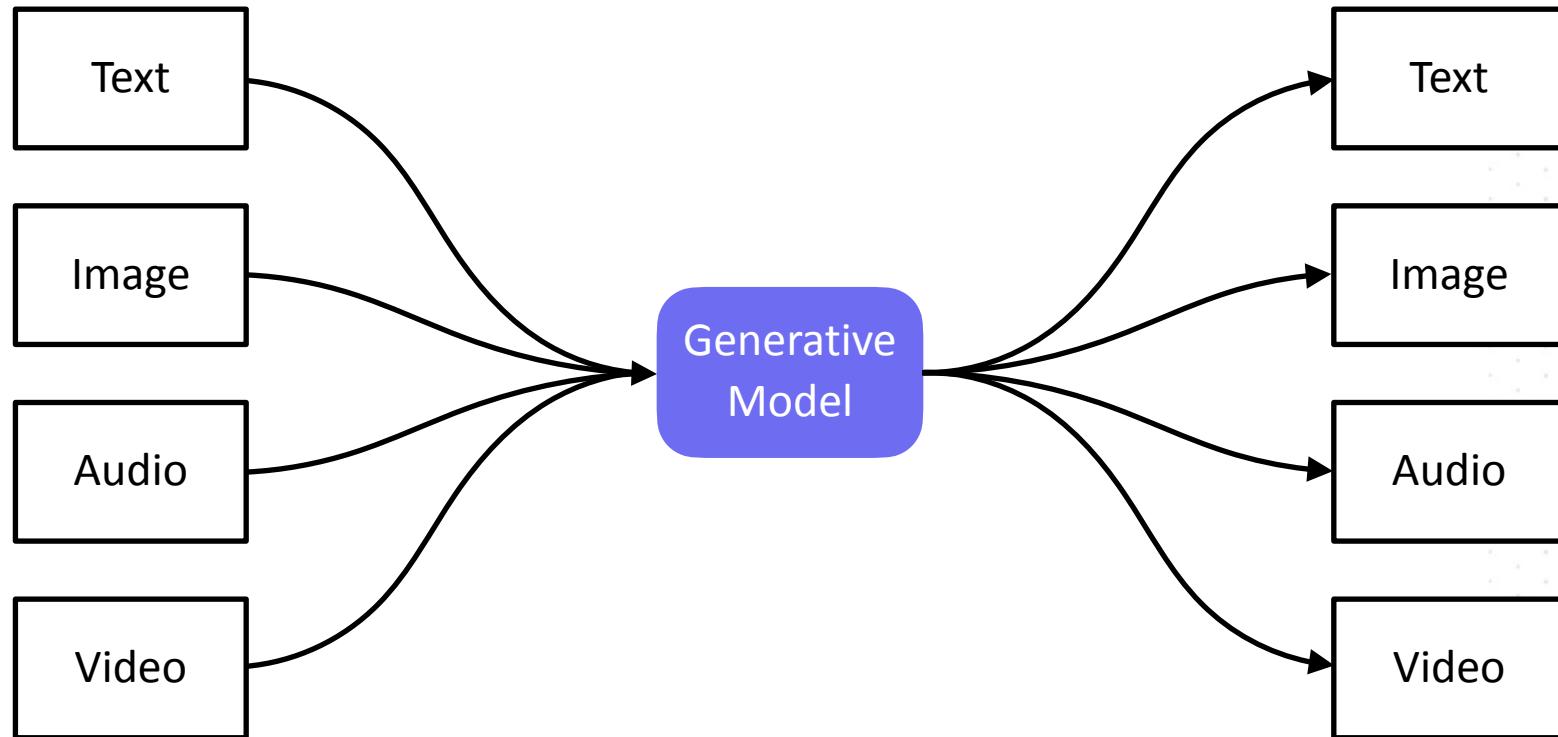
# Why Now?

- New algorithmic methods developed
- Increase in available computation/engineering advances
- Order of magnitude more accessible/processable data

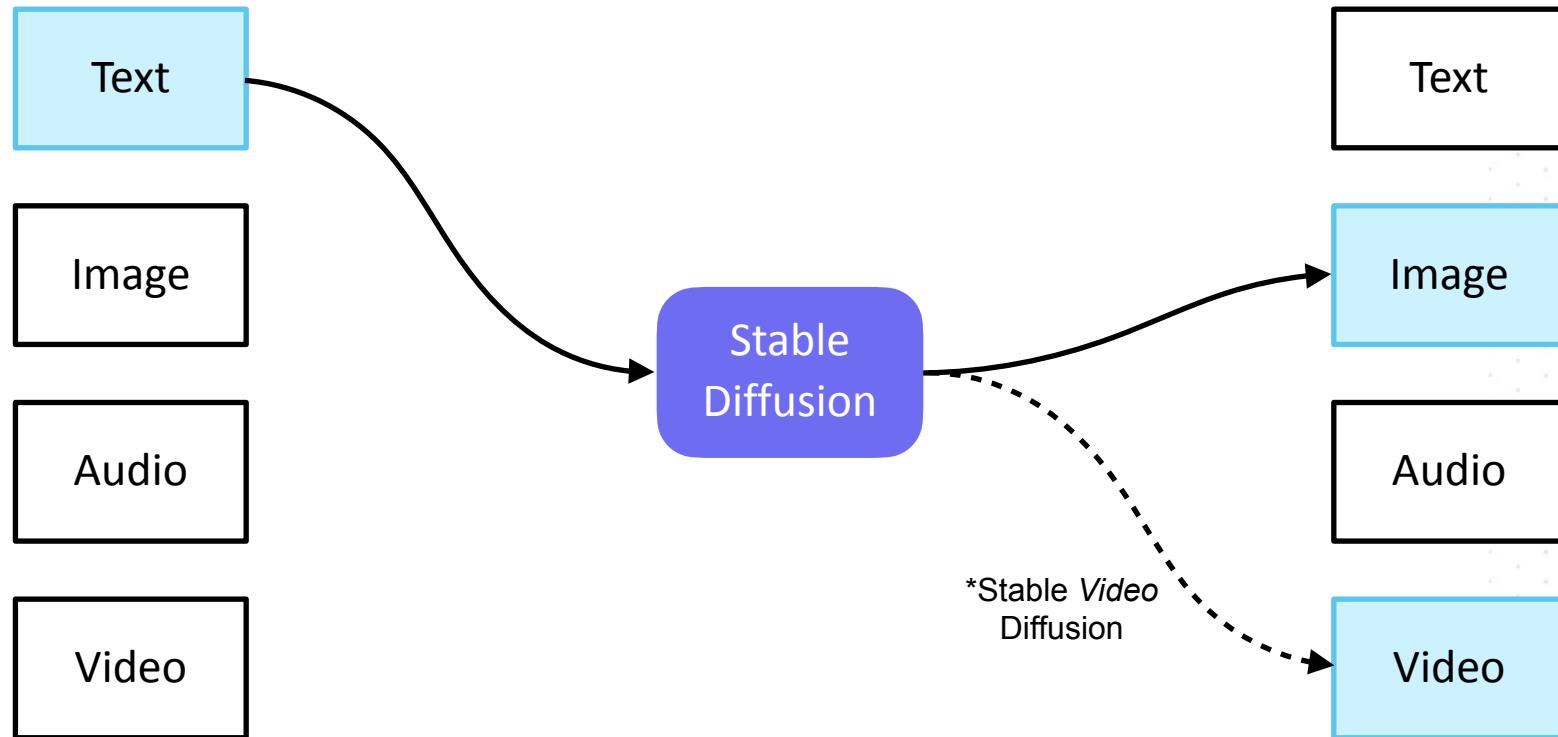
1.3

## Multitudes of Media

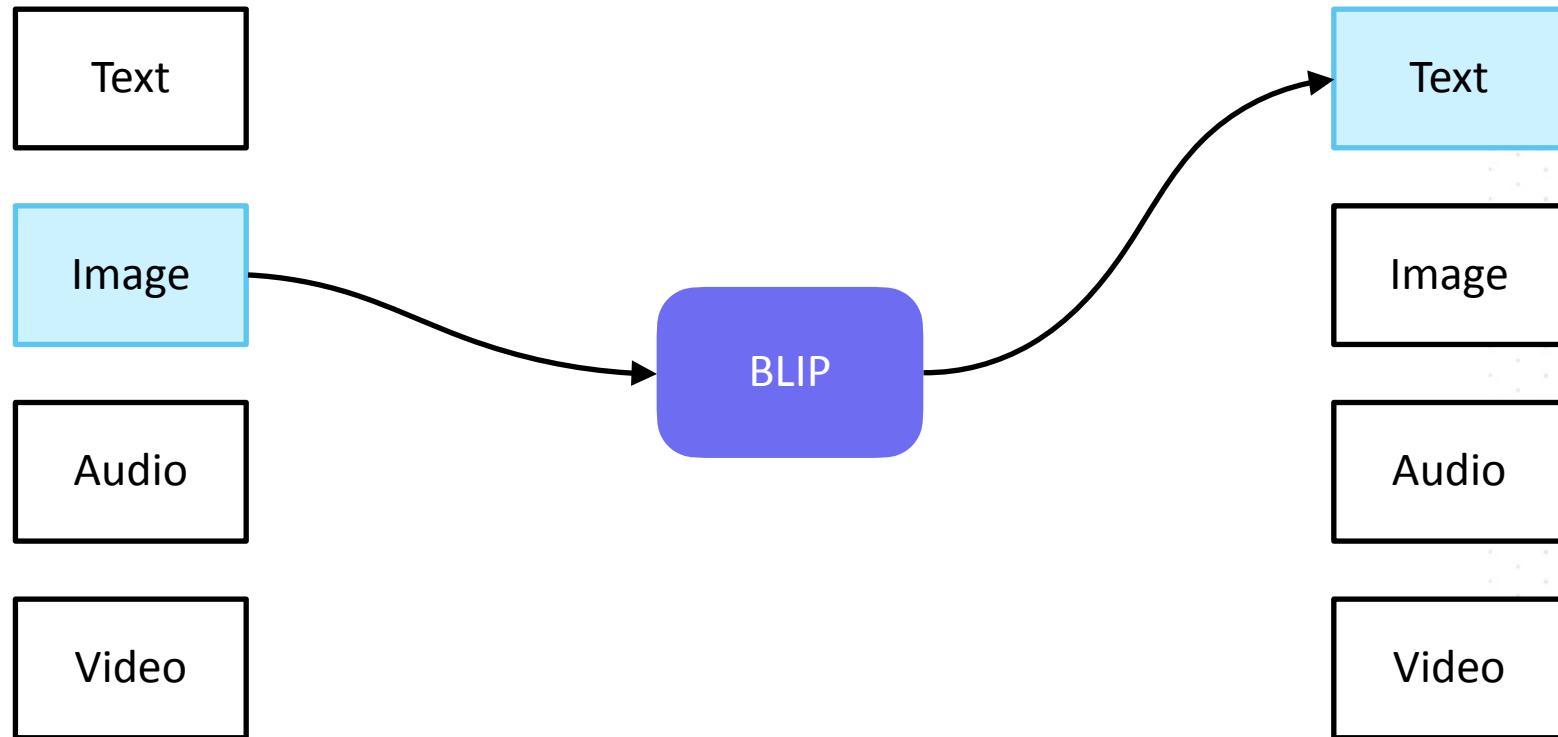
# Multitudes of Media



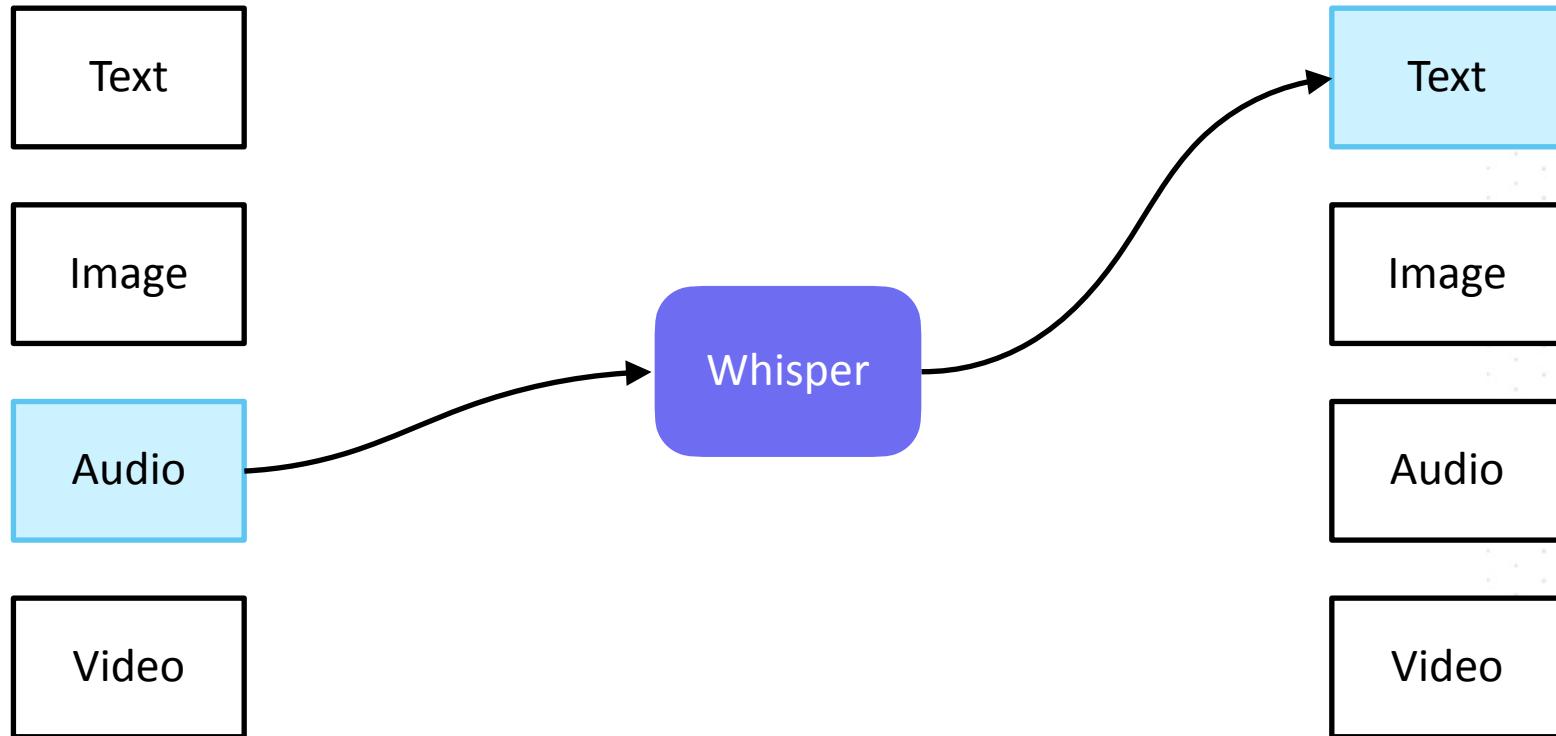
# text-to-image



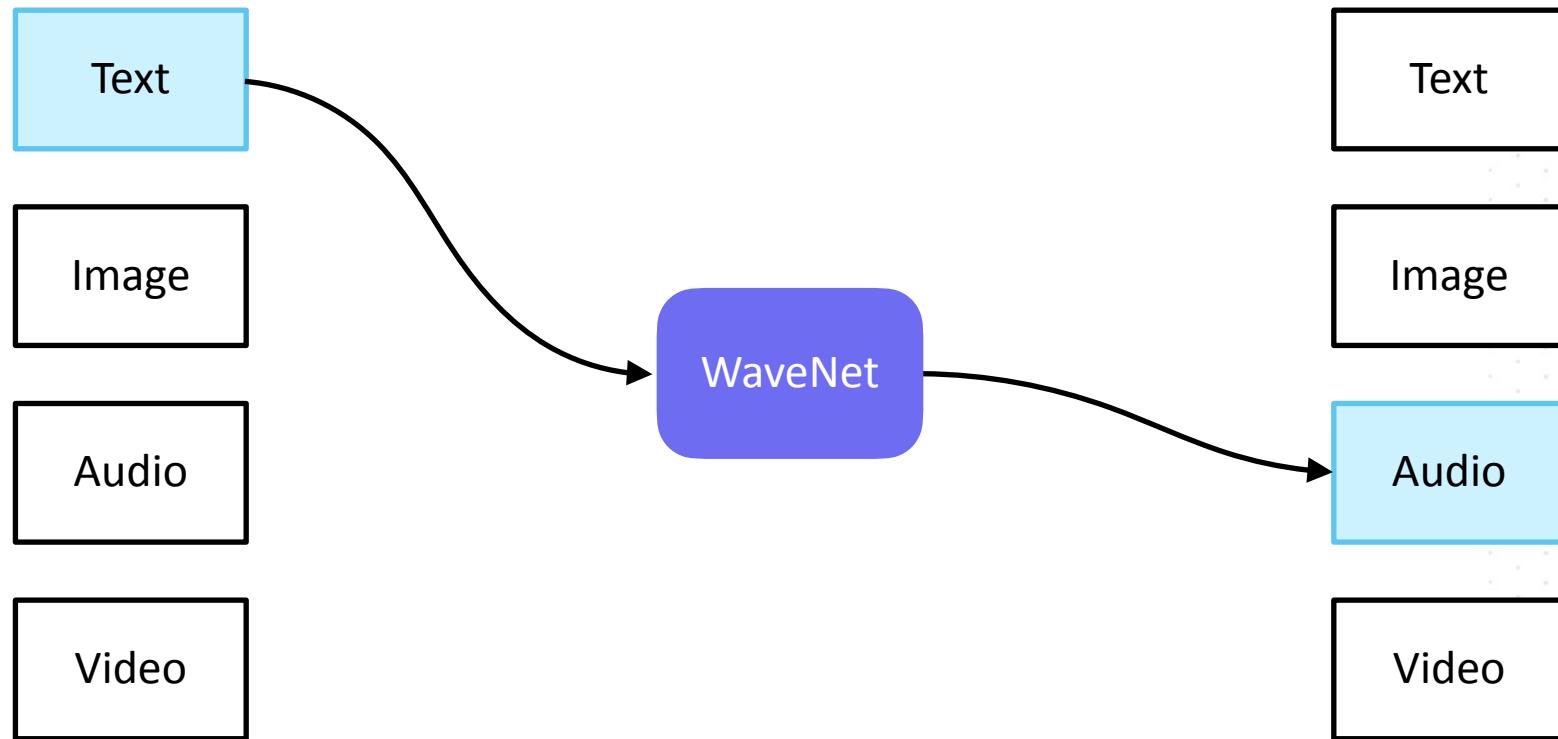
# image-to-text (image captioning)



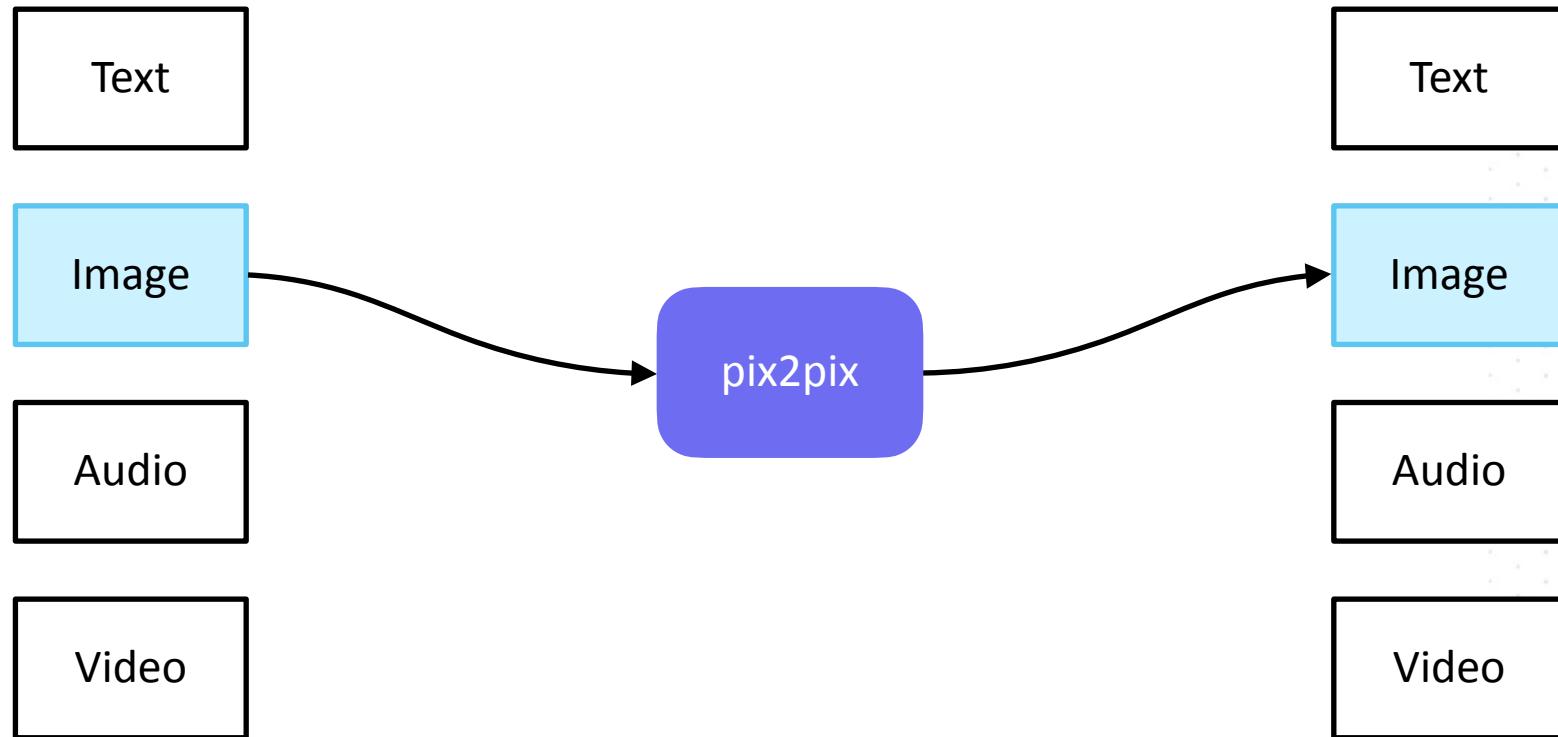
# audio-to-text (transcription)



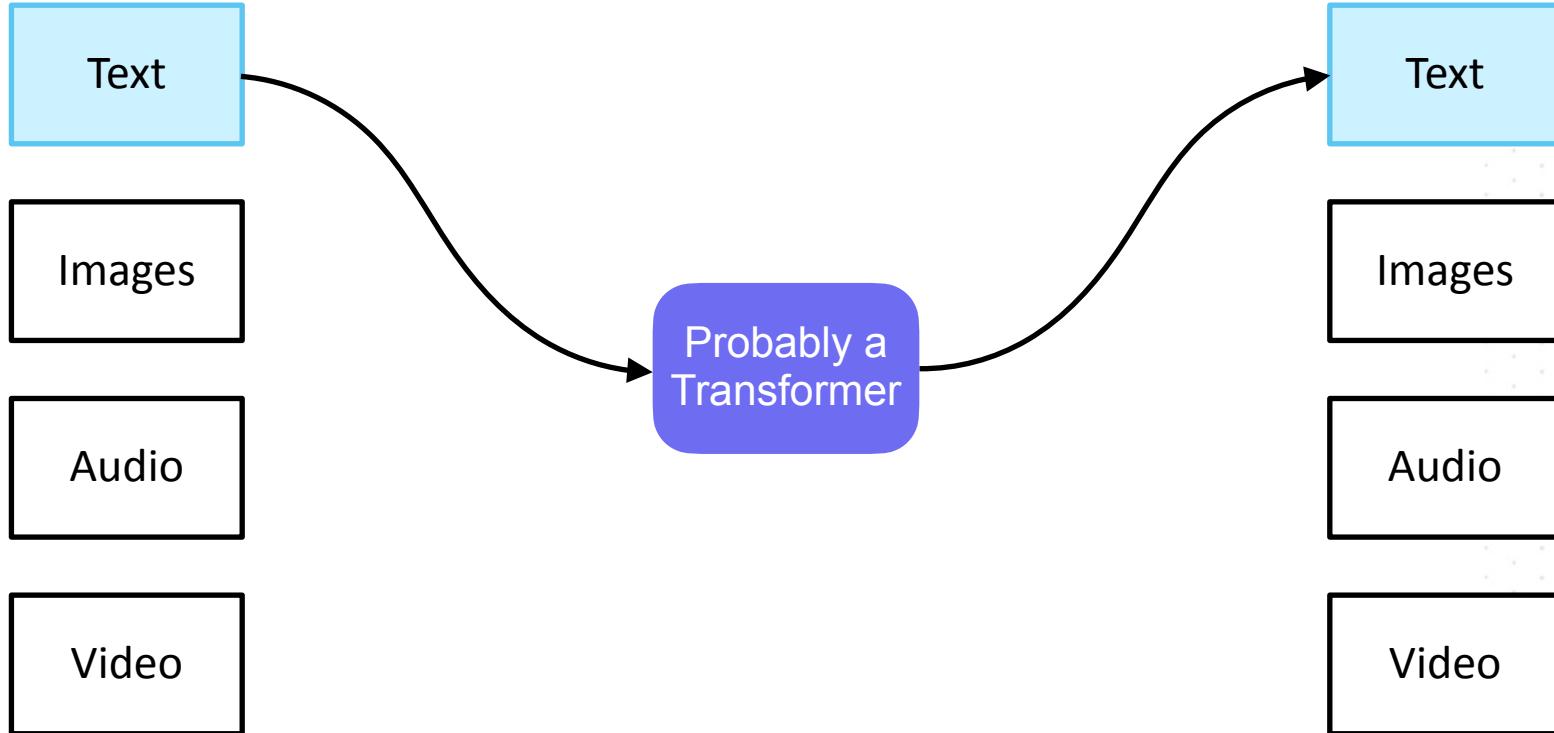
# text-to-audio (TTS)



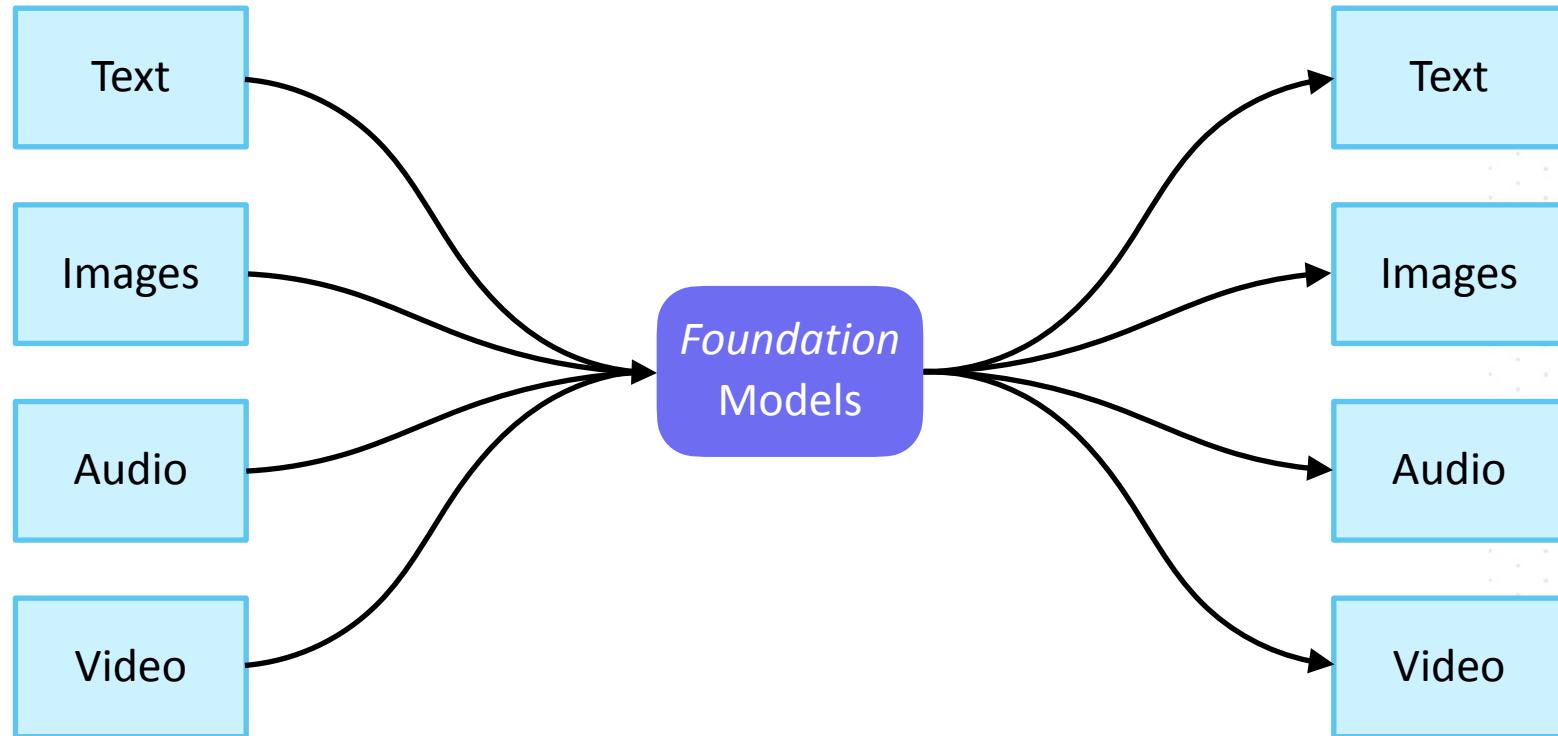
# image-to-image (upsampling, restoration, inpainting, etc.)



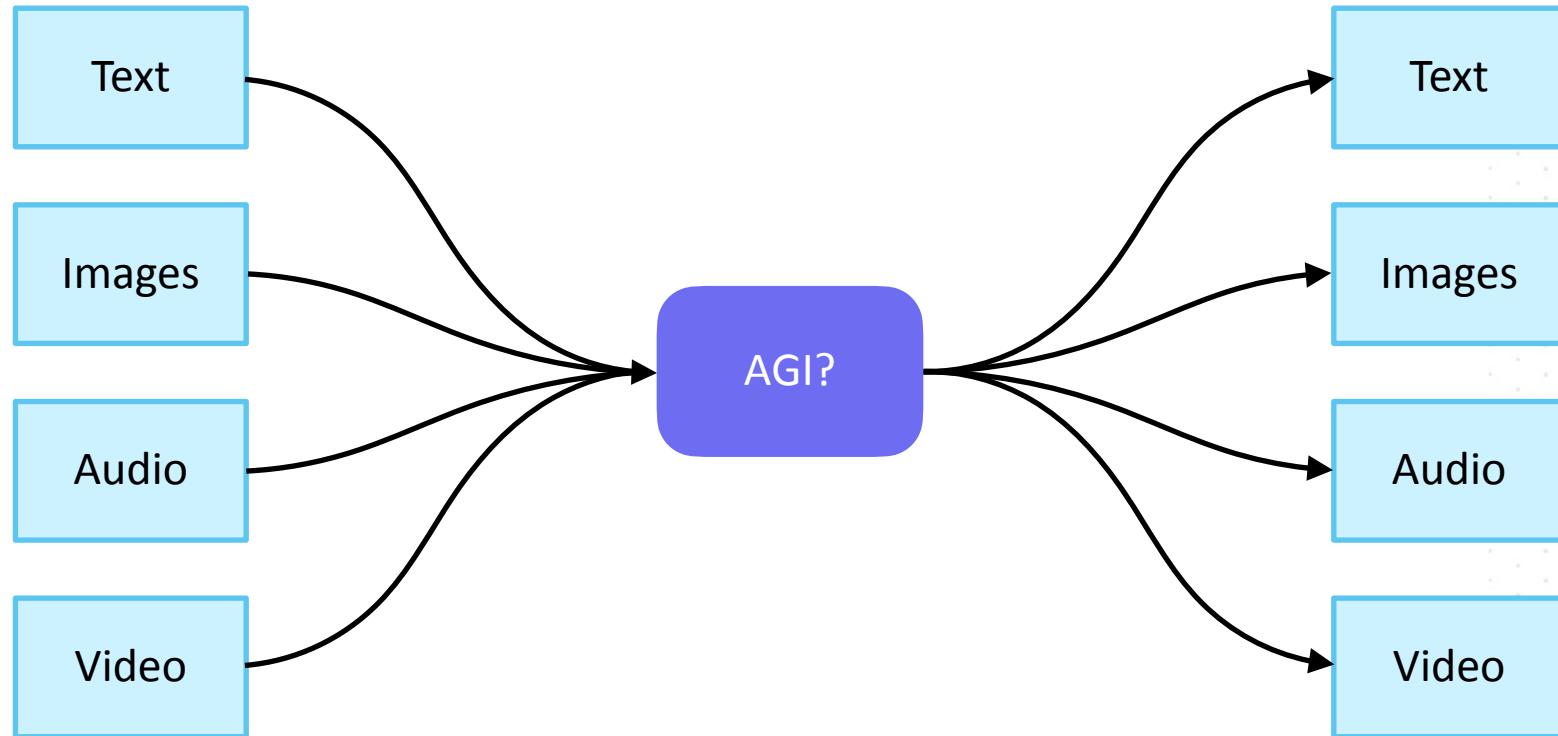
# text-to-text (translation, summarization, etc.)



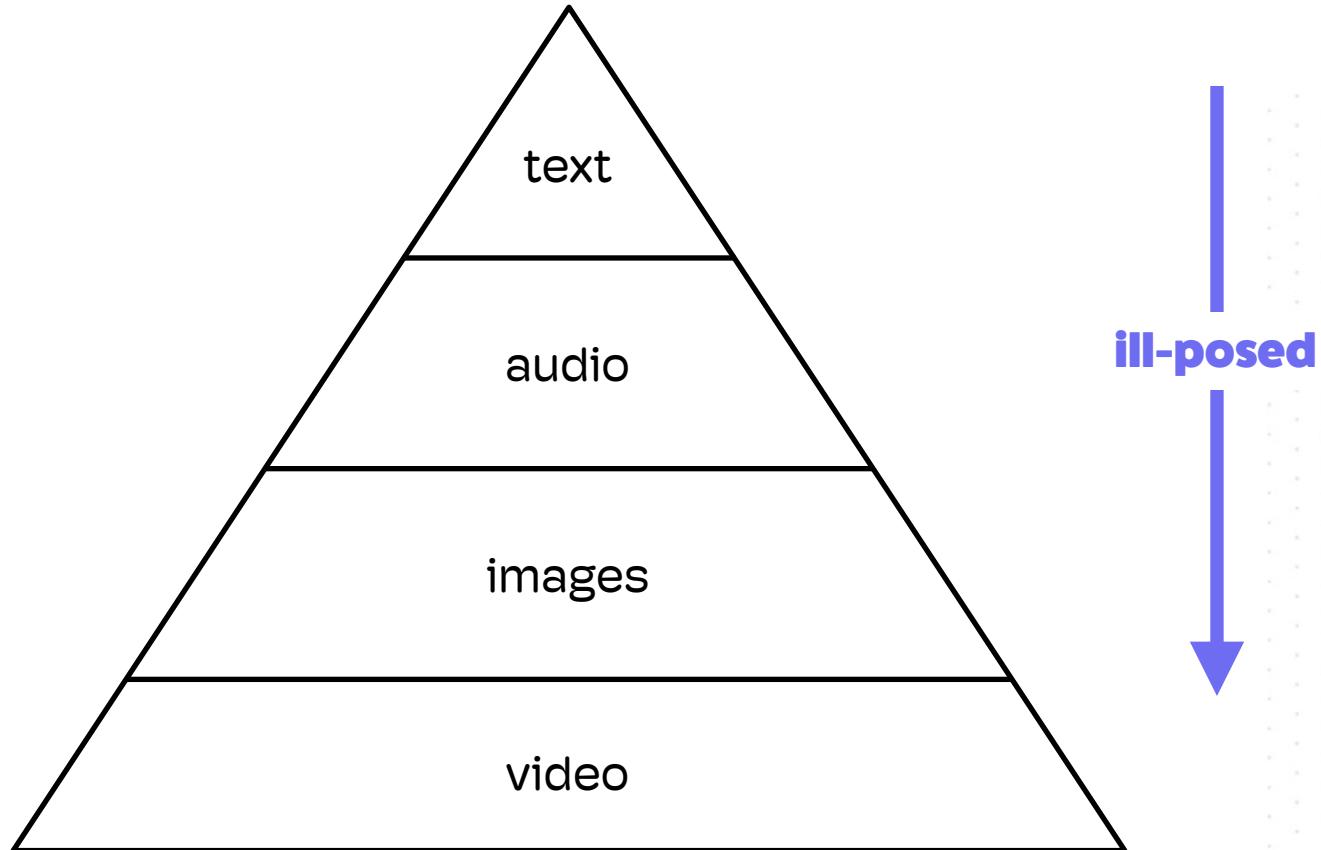
# anything-to-anything



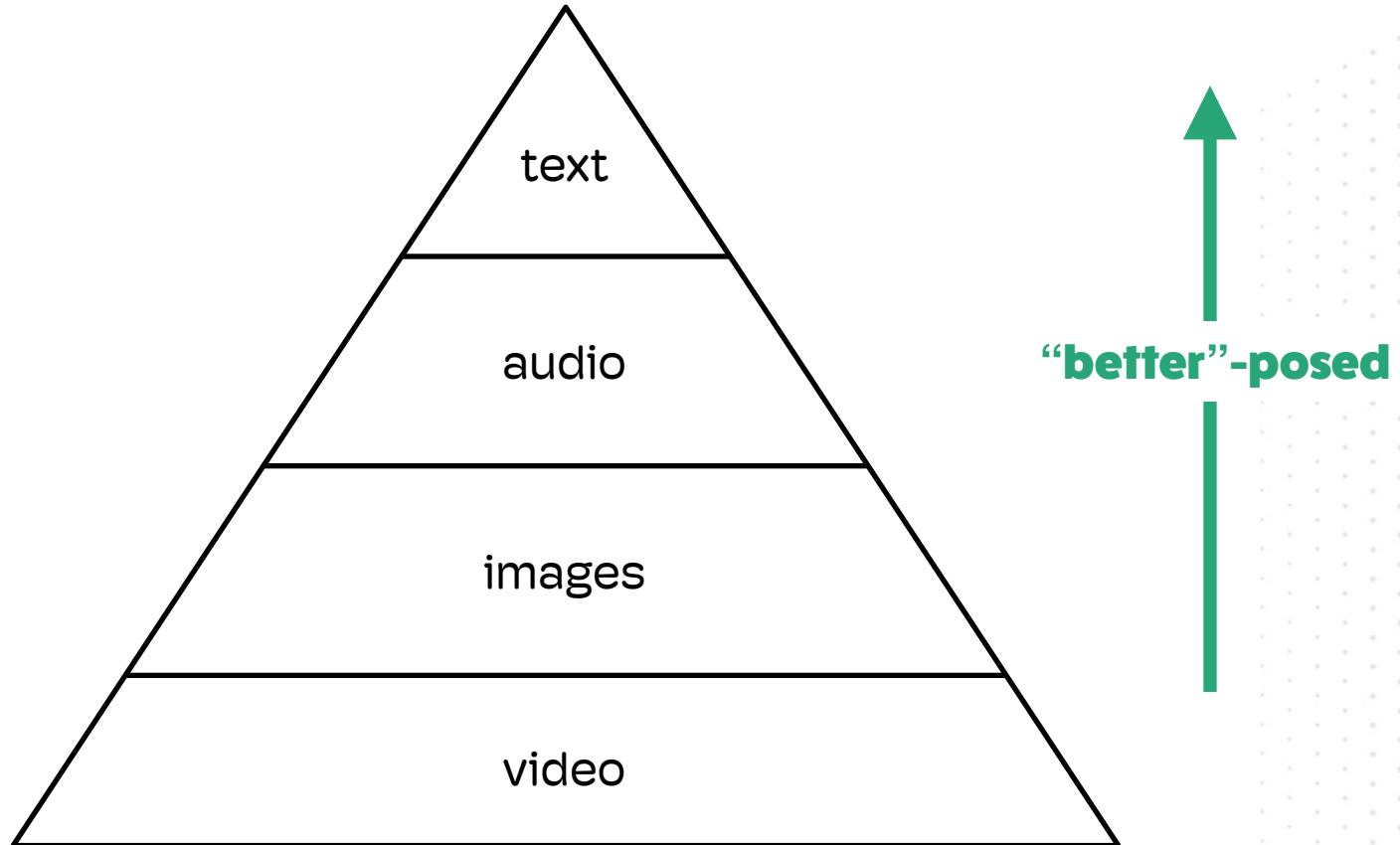
# anything-to-anything



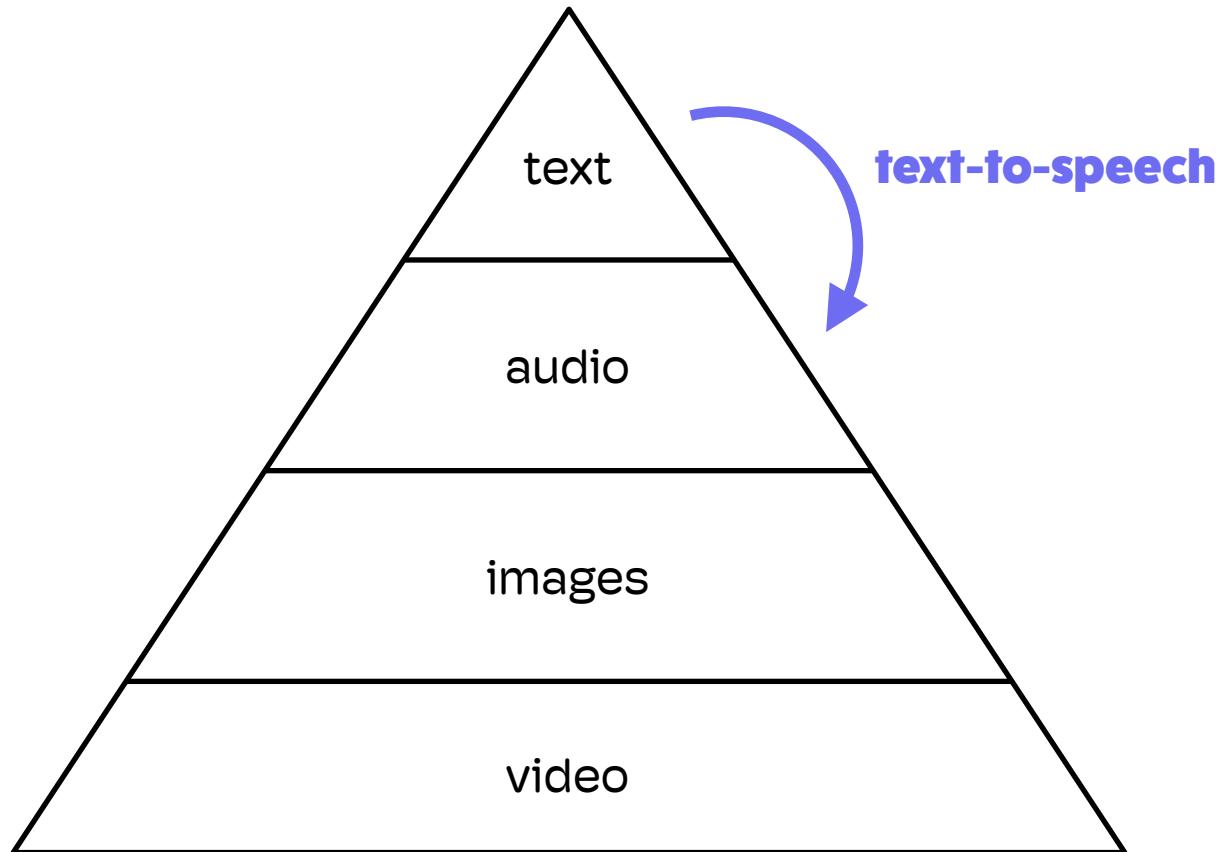
# Hierarchy of Media “richness”



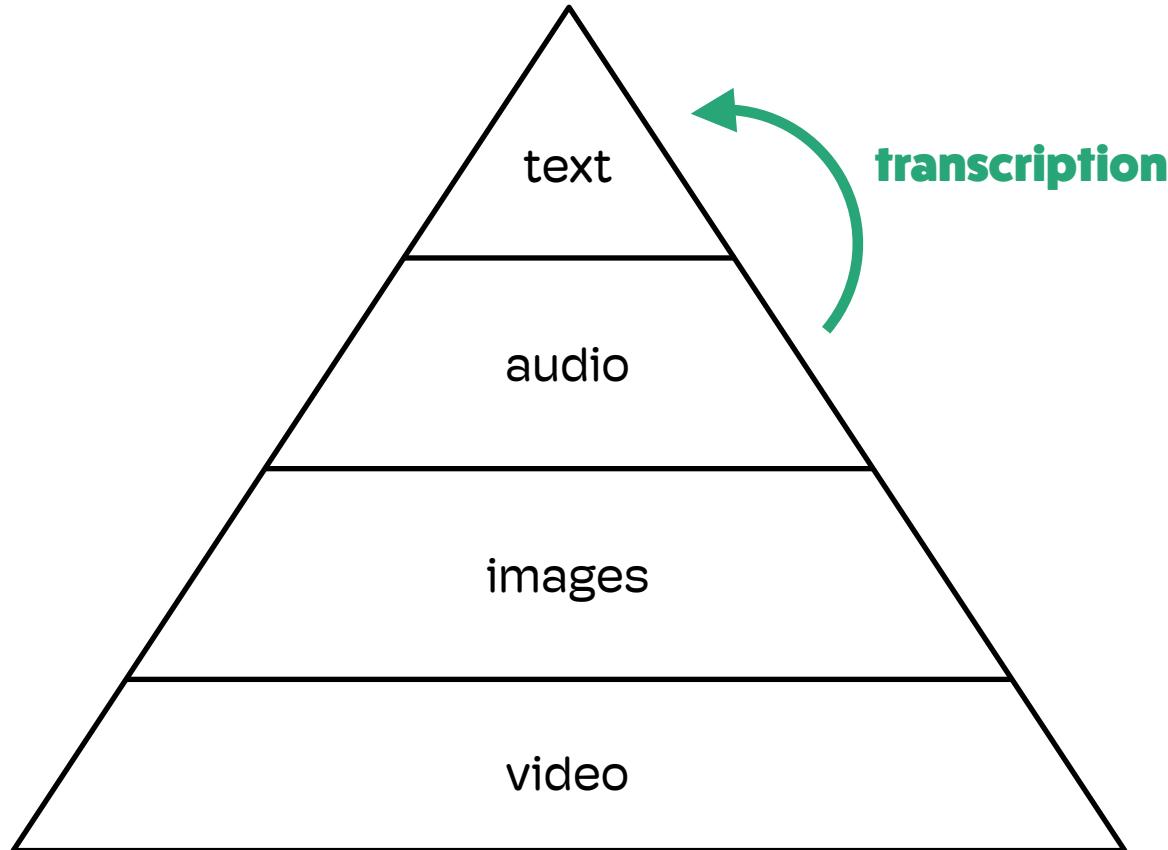
# Hierarchy of Media “richness”



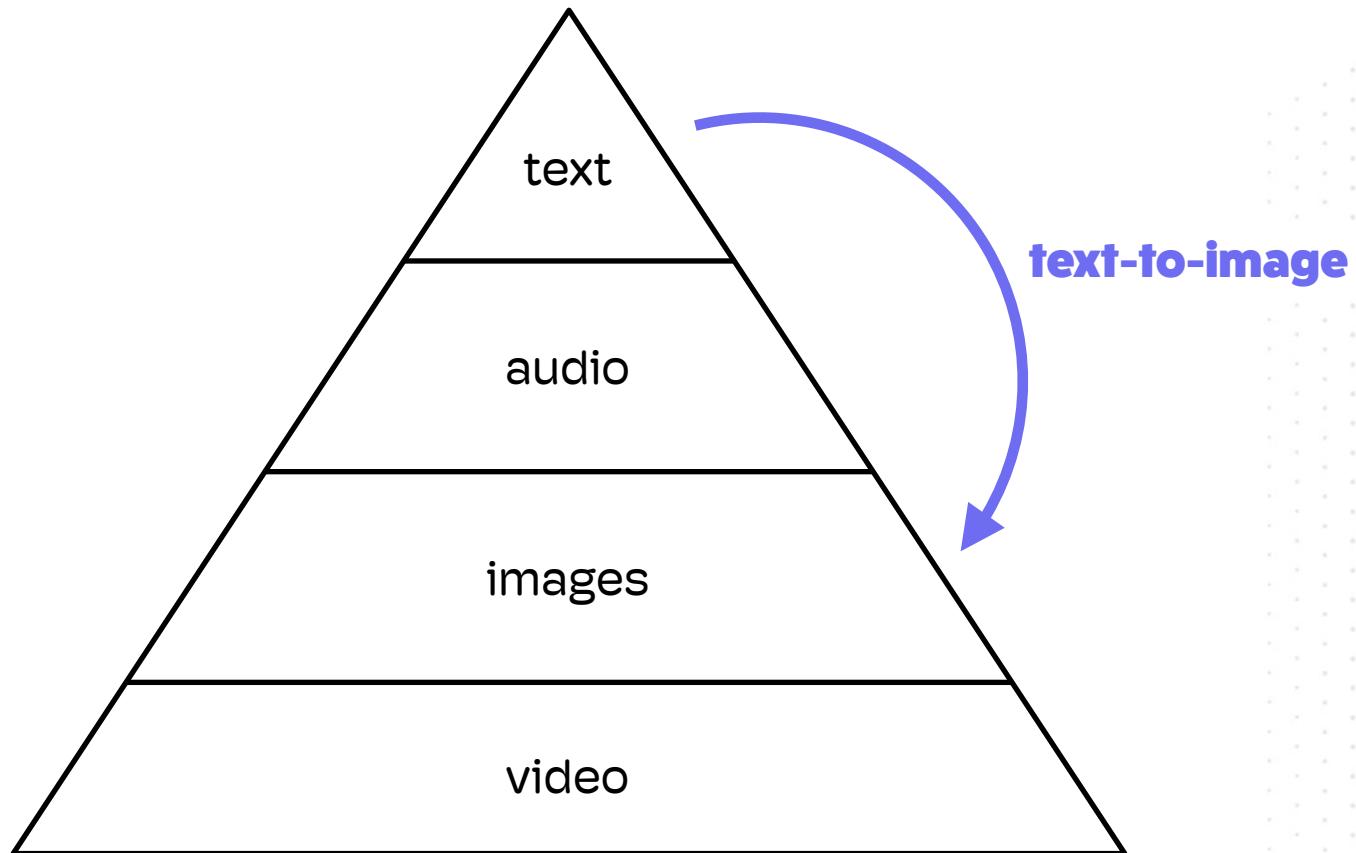
# Hierarchy of Media “richness”



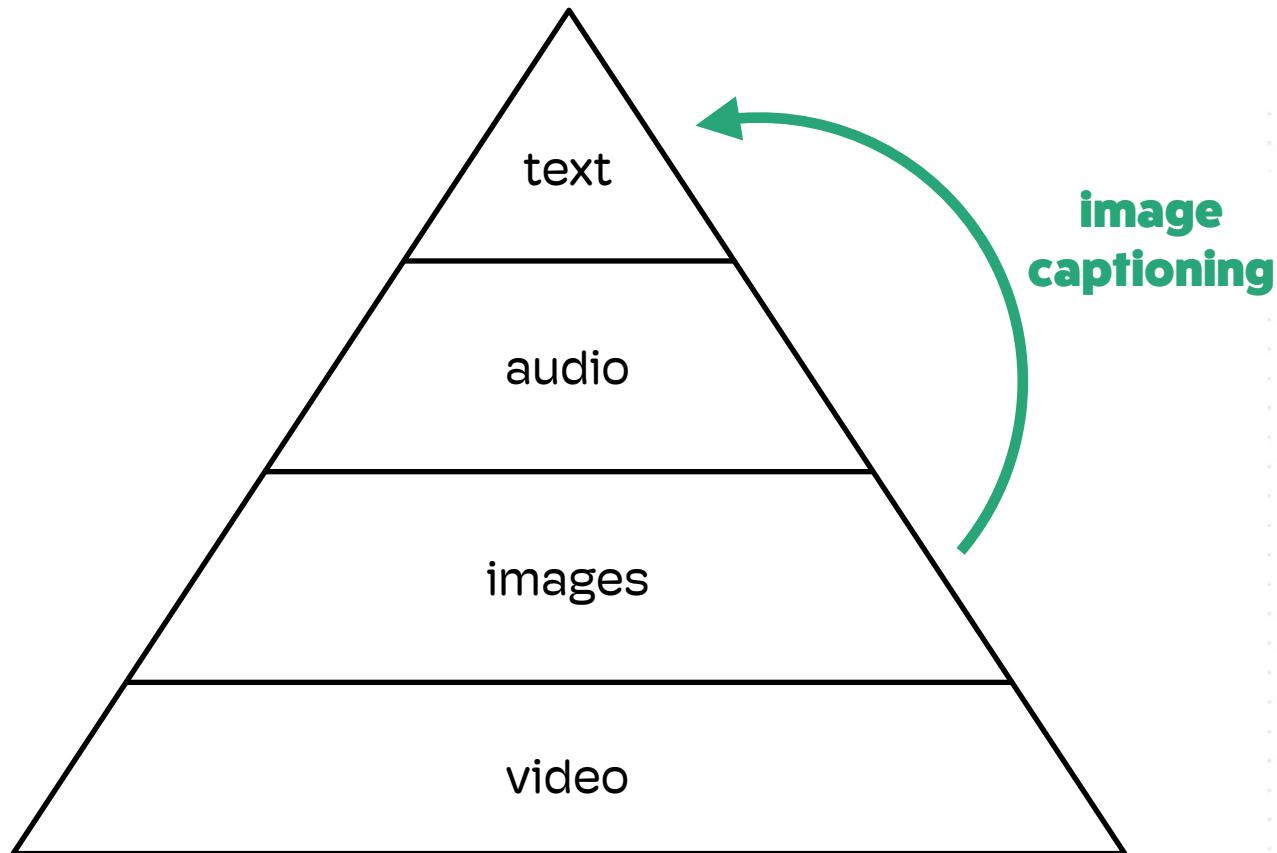
# Hierarchy of Media “richness”



# Hierarchy of Media “richness”



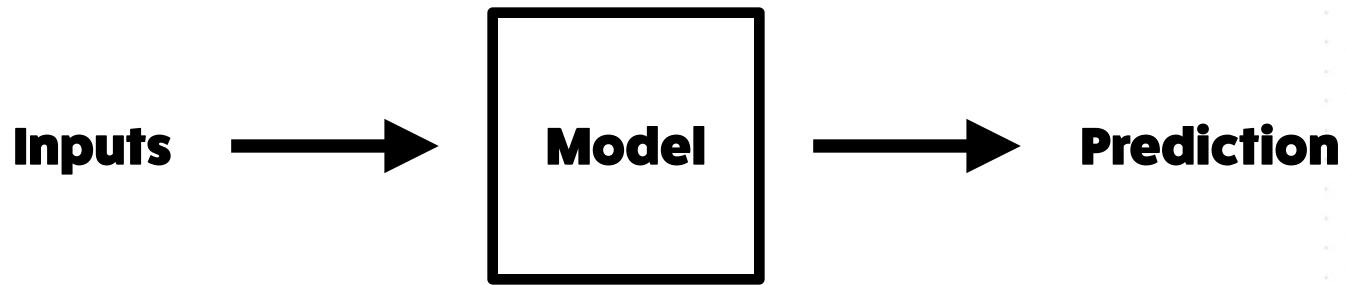
# Hierarchy of Media “richness”



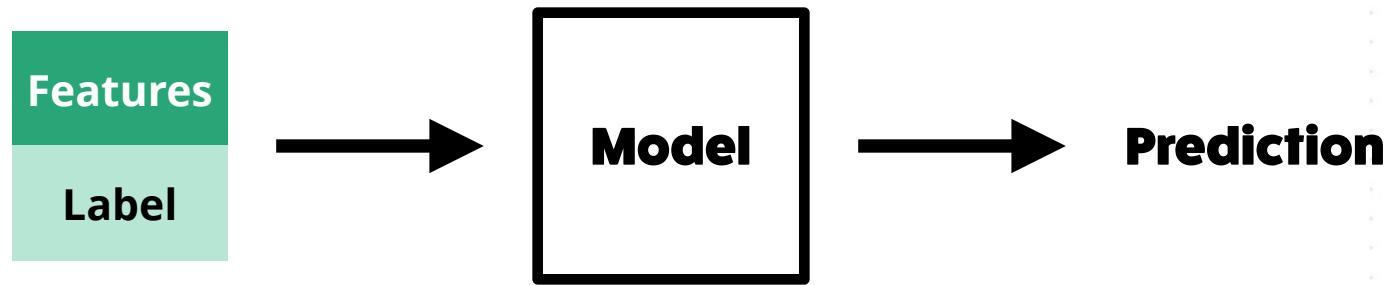
1.4

## How Machines Create

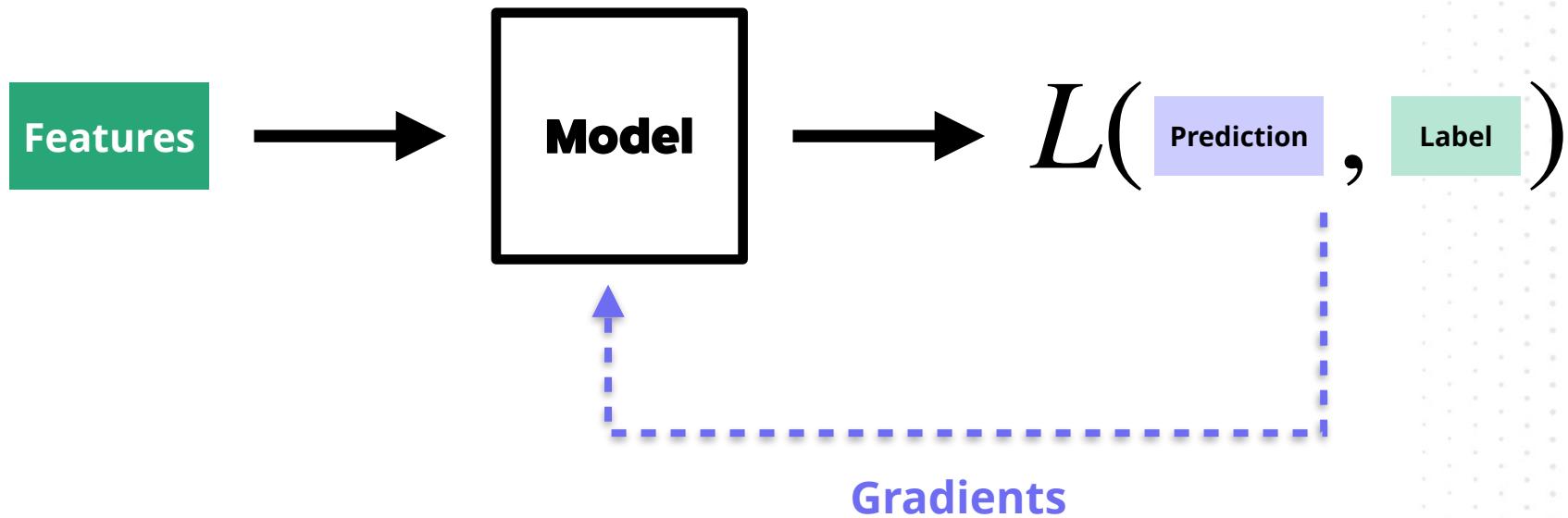
# How Machines Learn



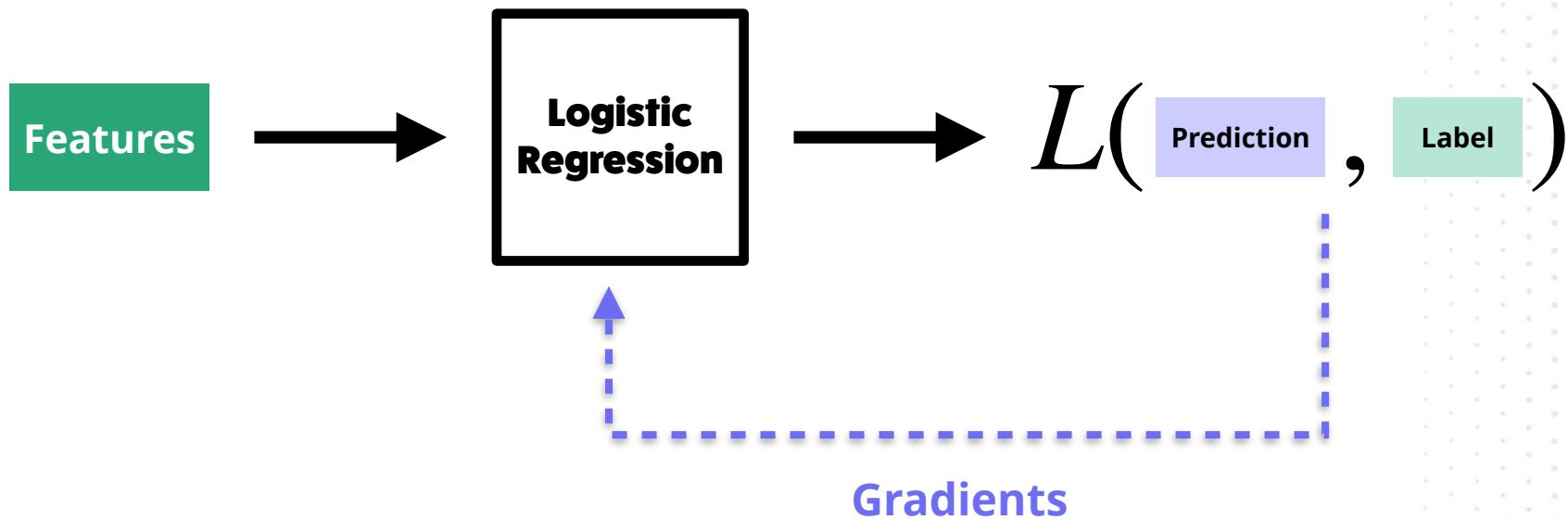
# Supervised Training



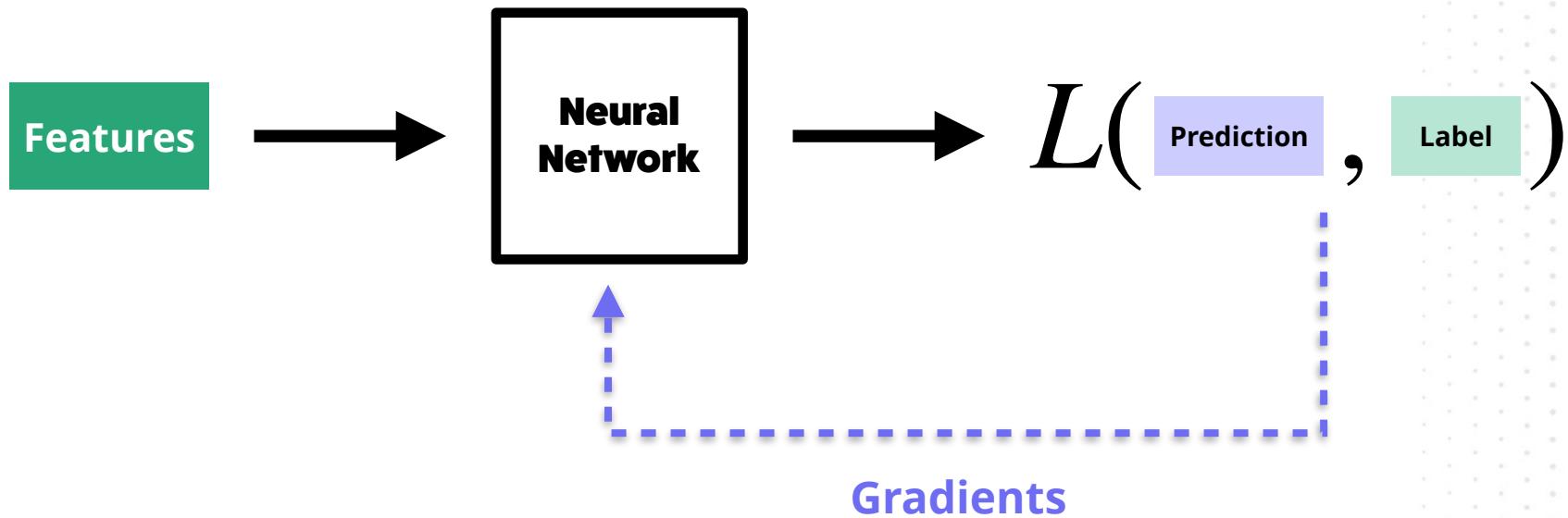
# Supervised Training



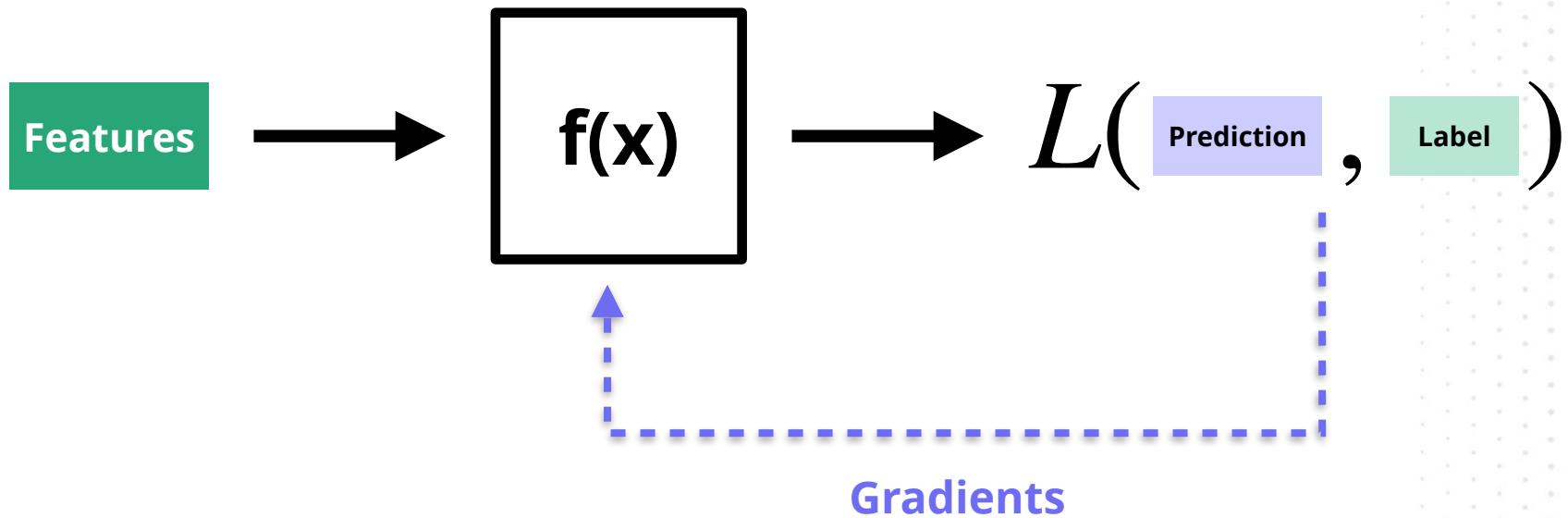
# Supervised Training



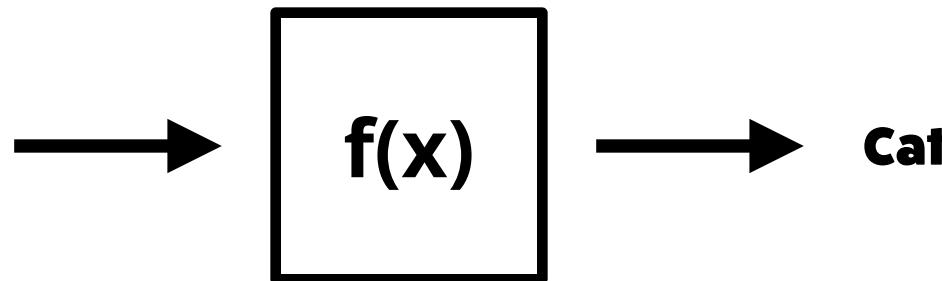
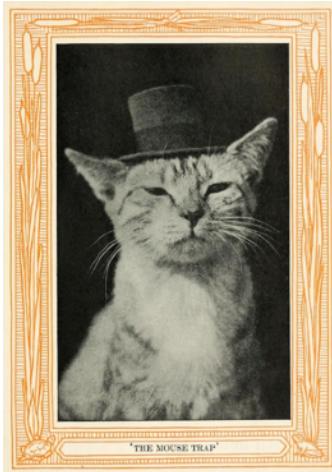
# Supervised Training



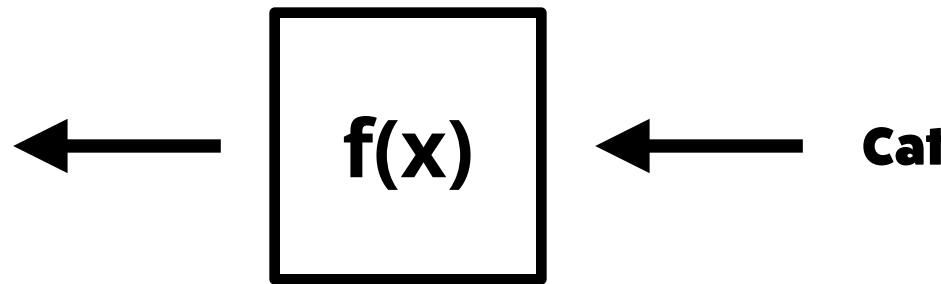
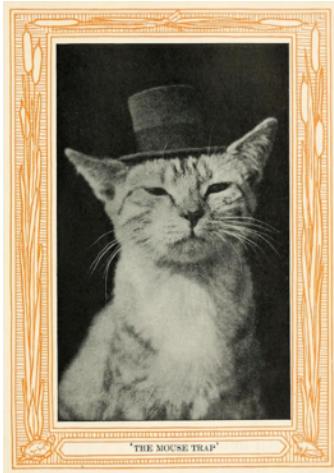
# Supervised Training



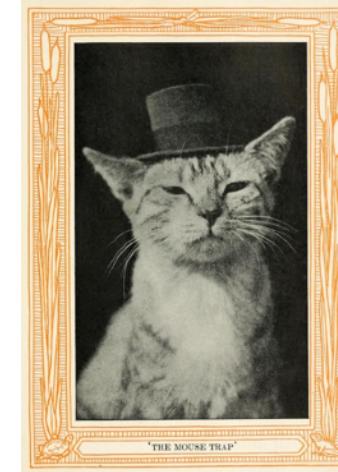
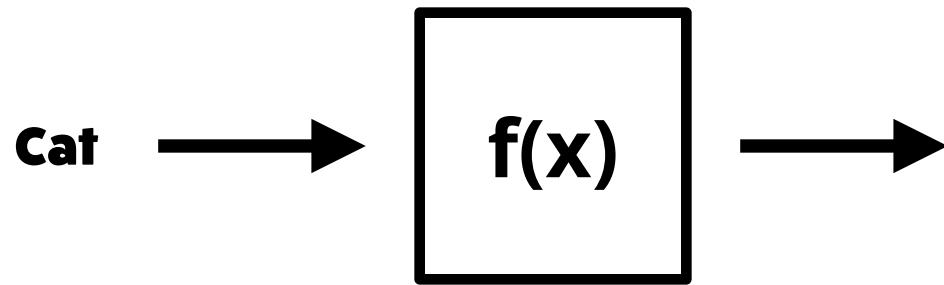
# How Machines Create



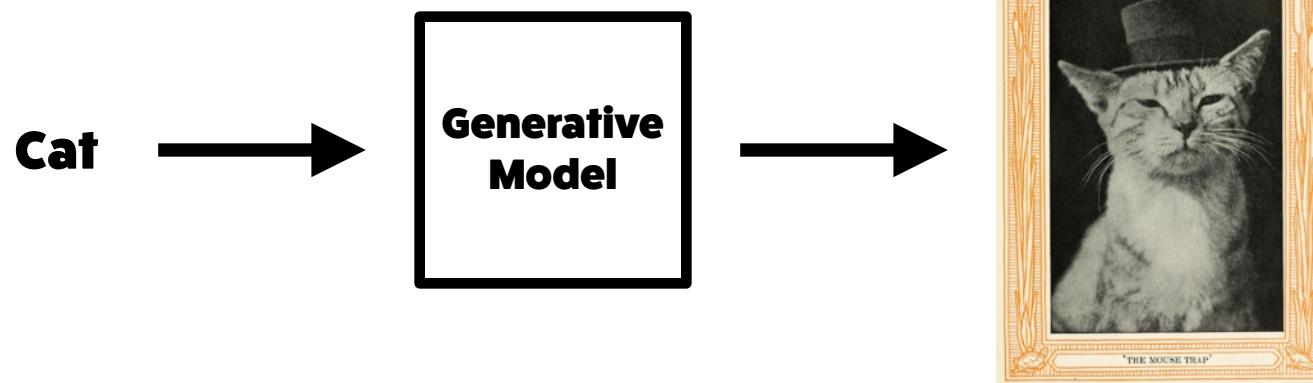
# How Machines Create



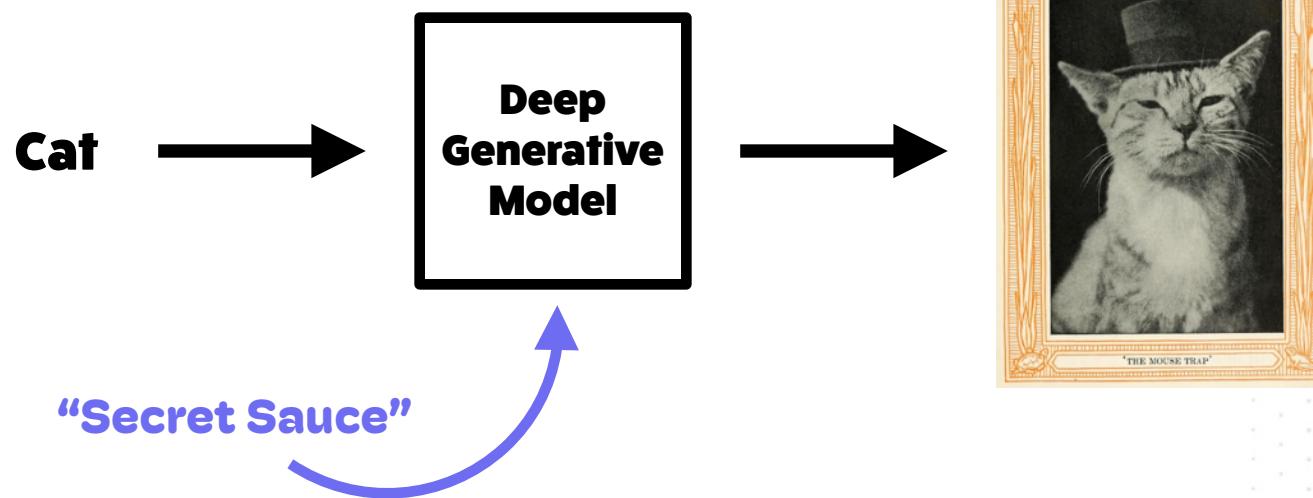
# How Machines *Create*



# How Machines *Create*

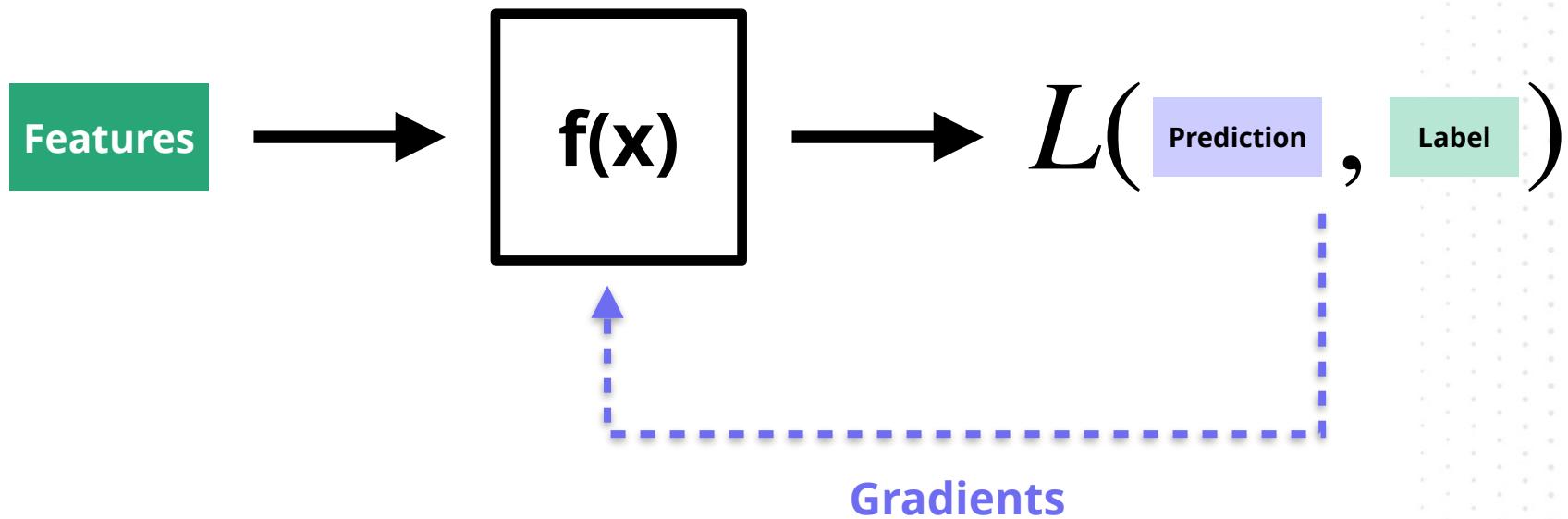


# How Machines *Create*

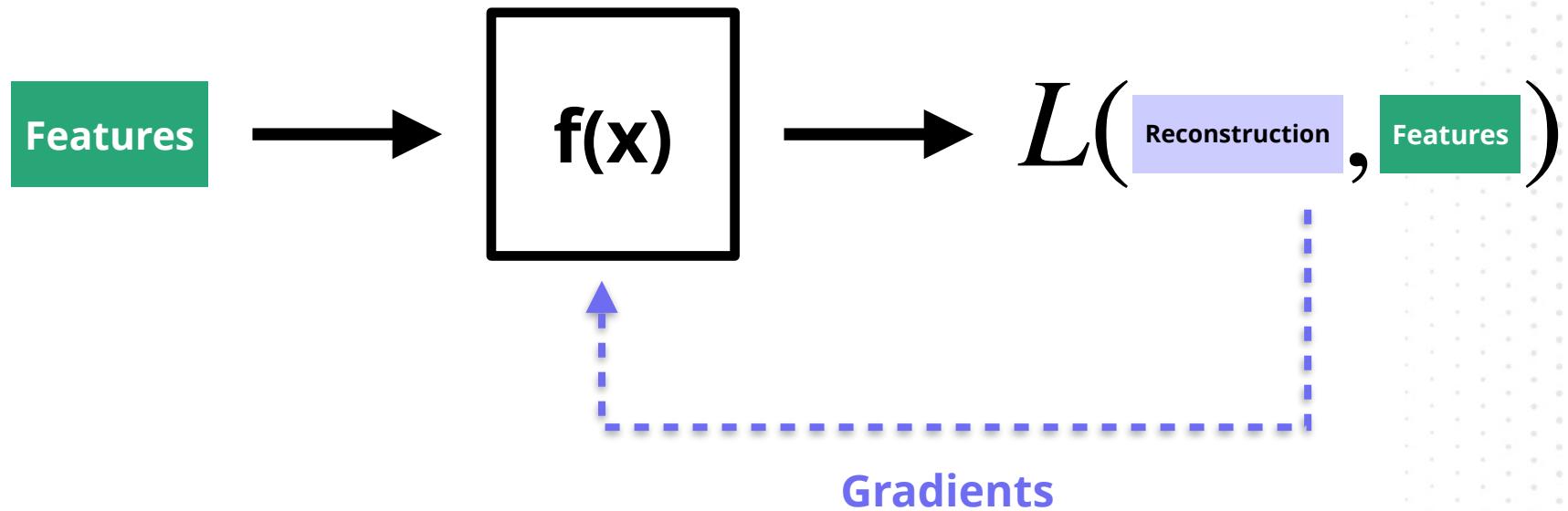


Source: Kittens and Cats: A First Reader (1911)

# Supervised Training



# Generative Training



# Generative AI vs. Generative Models

## Generative AI

- Entire system/pipeline
- Does the “creating”
- Multiple interacting components

## Generative Model

- Mathematical model
- Does the “learning”
- VAE, GAN, diffusion, etc.

1.5

## **Formalizing Generative Models**

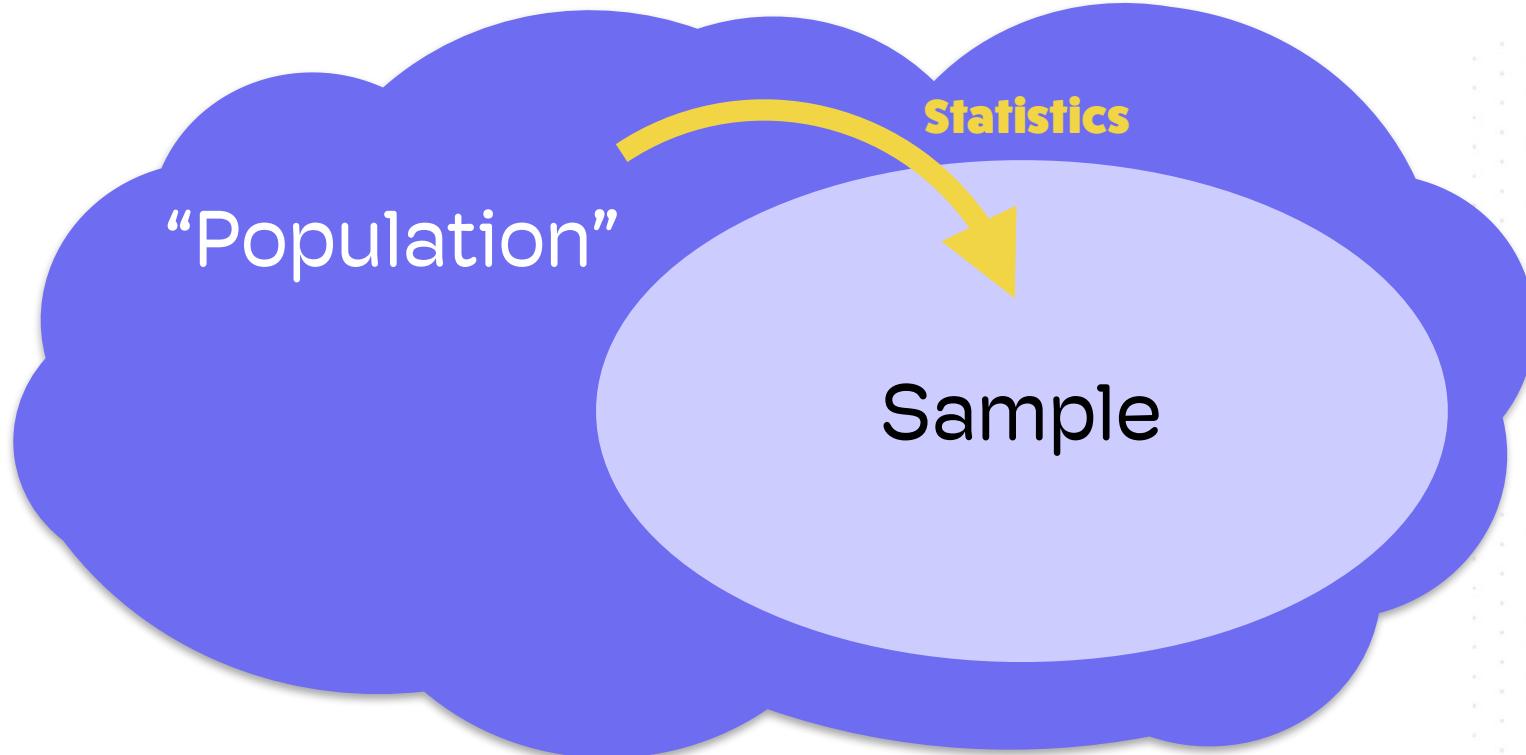
# Generative Processes

The diagram features two overlapping cloud-shaped bubbles. The larger, darker blue bubble on the left contains the text "Generative Process". The smaller, lighter blue bubble on the right contains the text "Sample". Both bubbles have a white outline and a slight shadow.

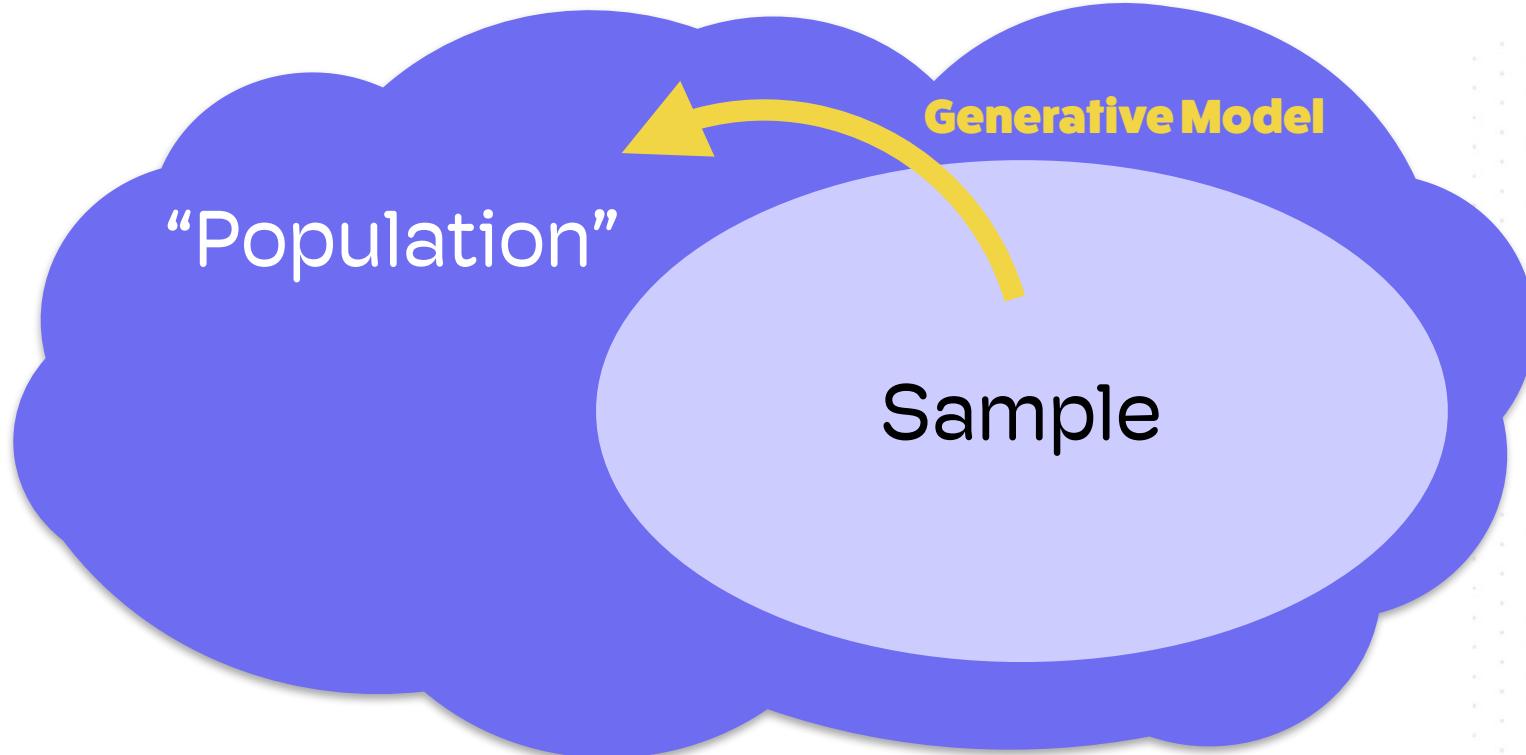
Generative  
Process

Sample

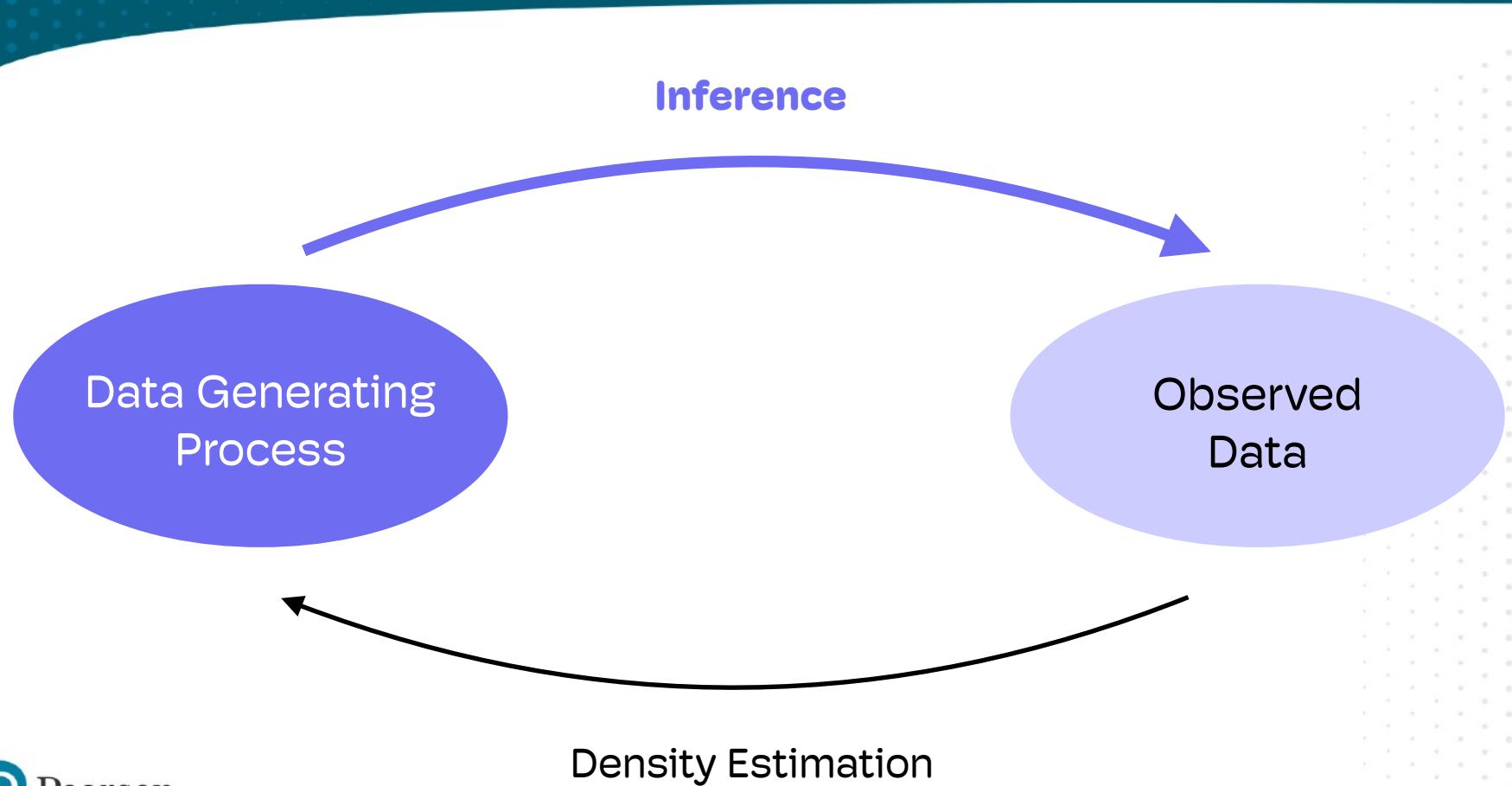
# Generative Processes



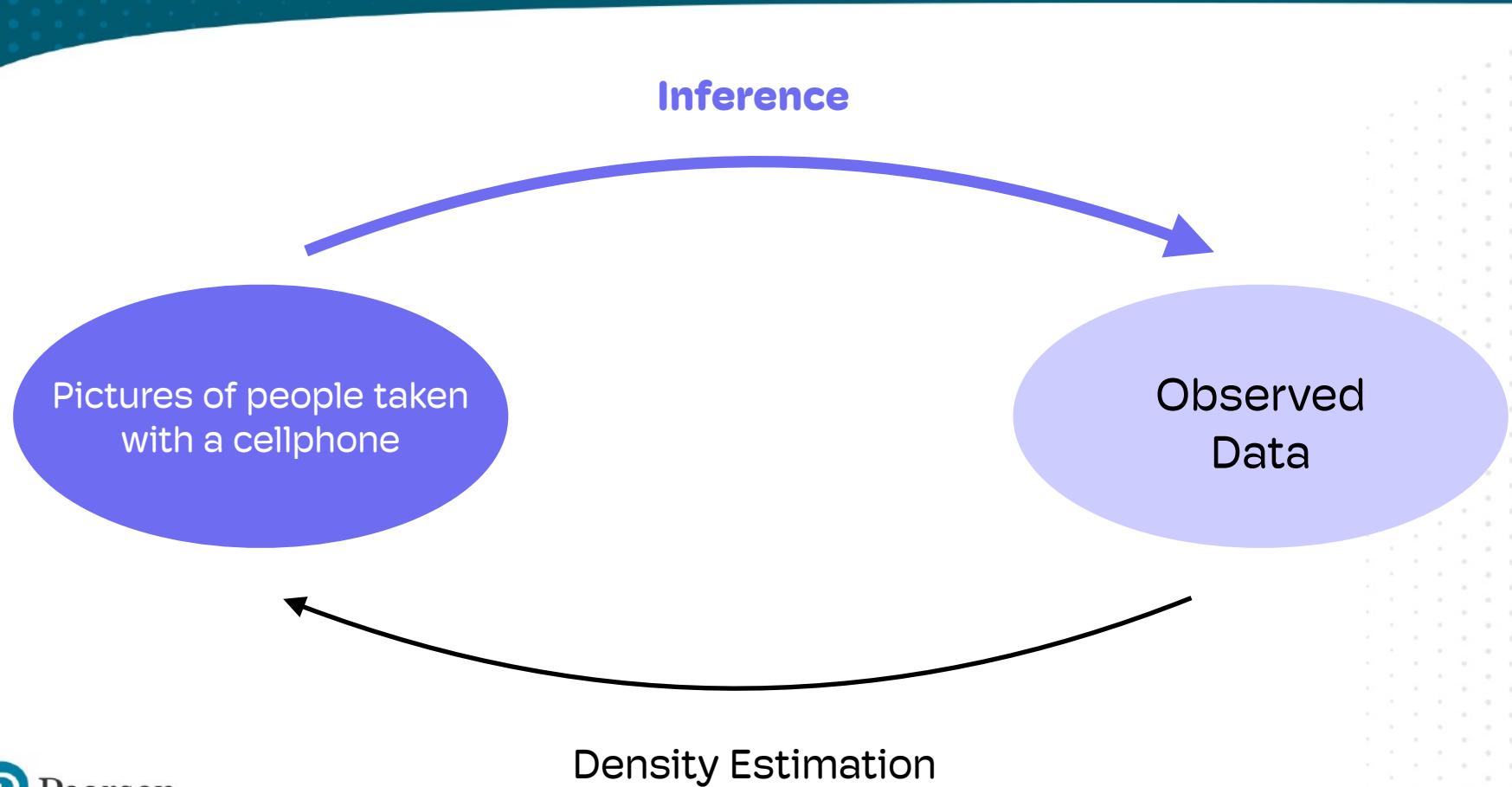
# Generative Processes



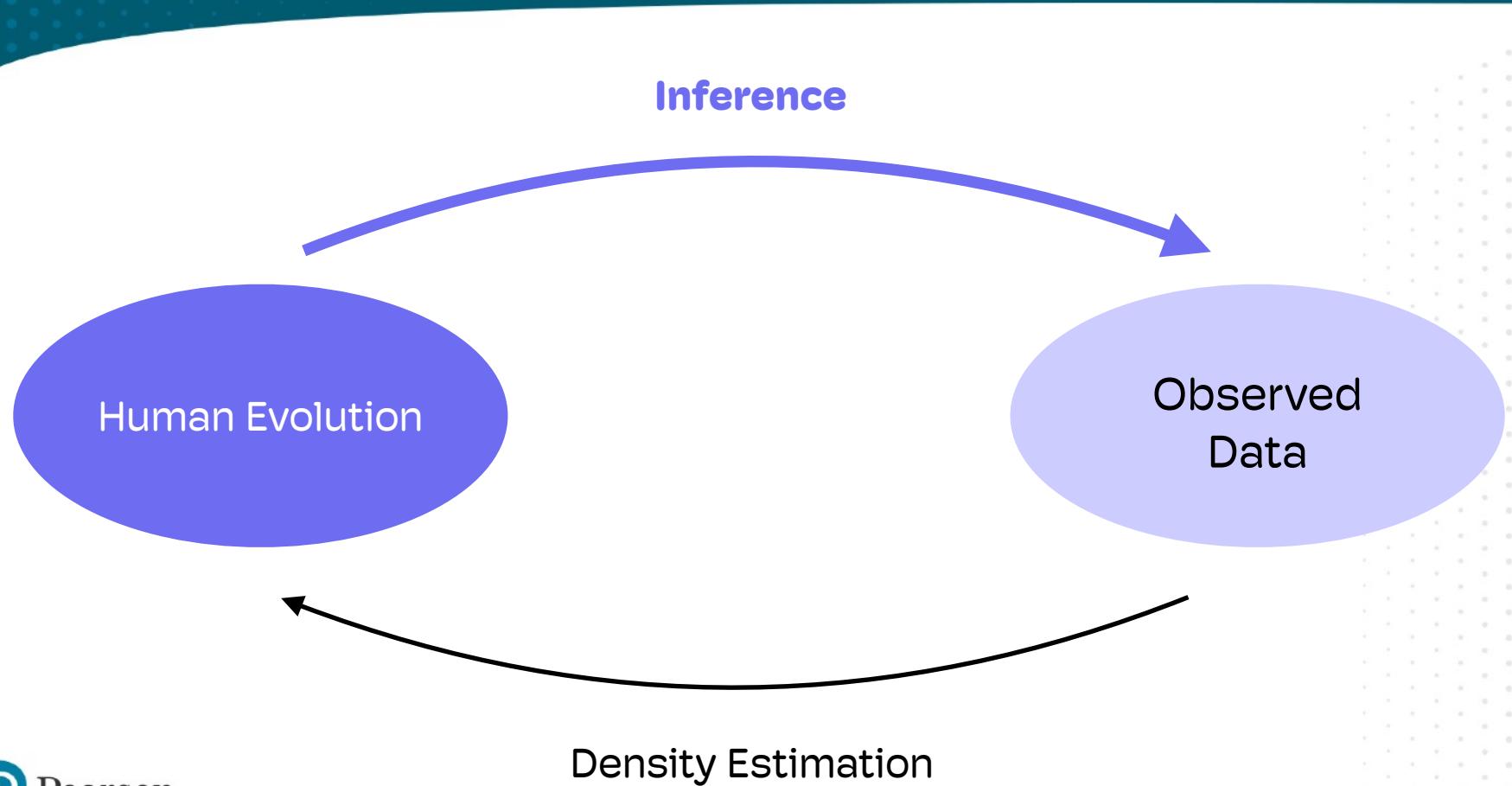
# Generative Processes



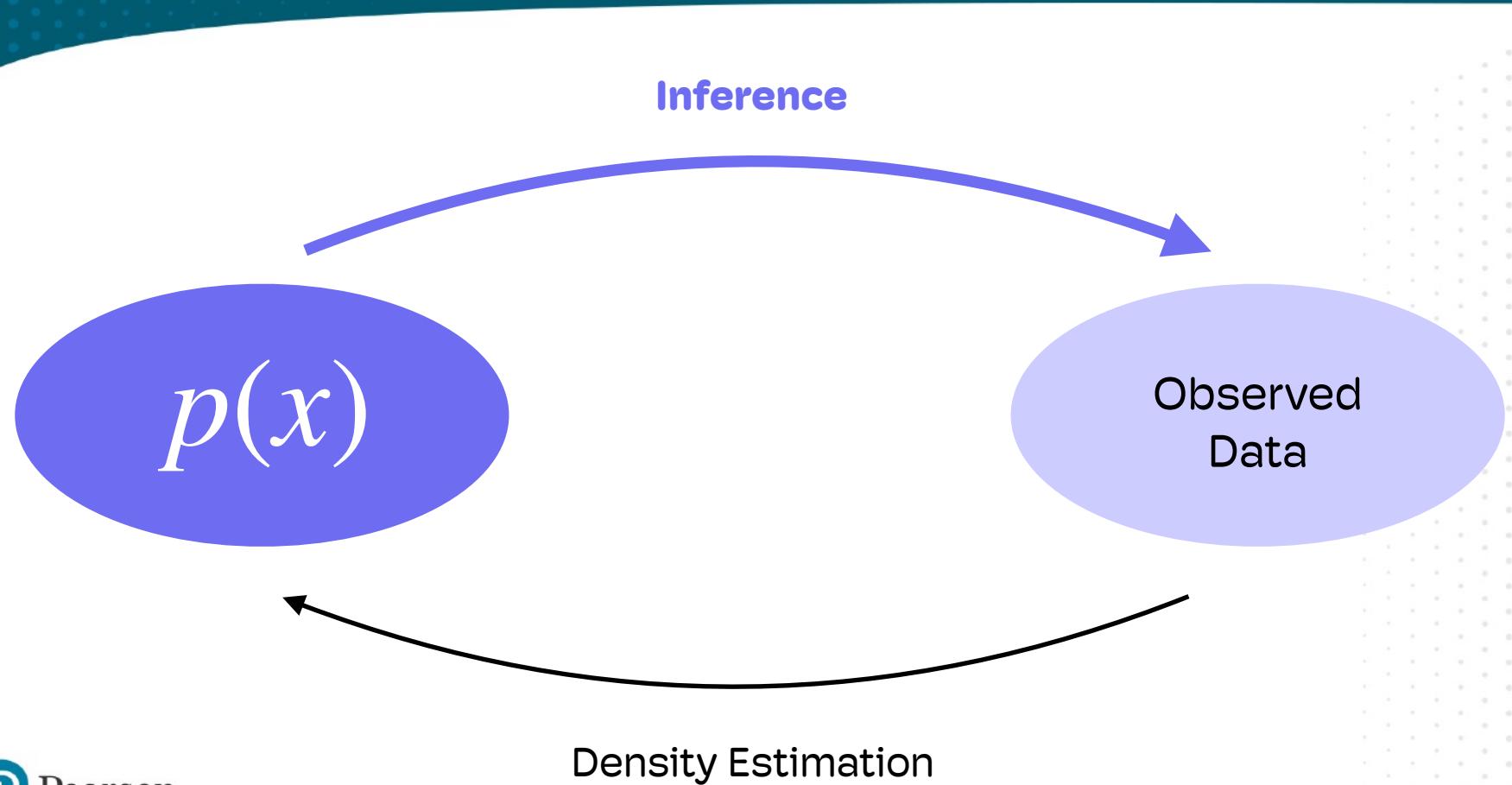
# Generative Processes



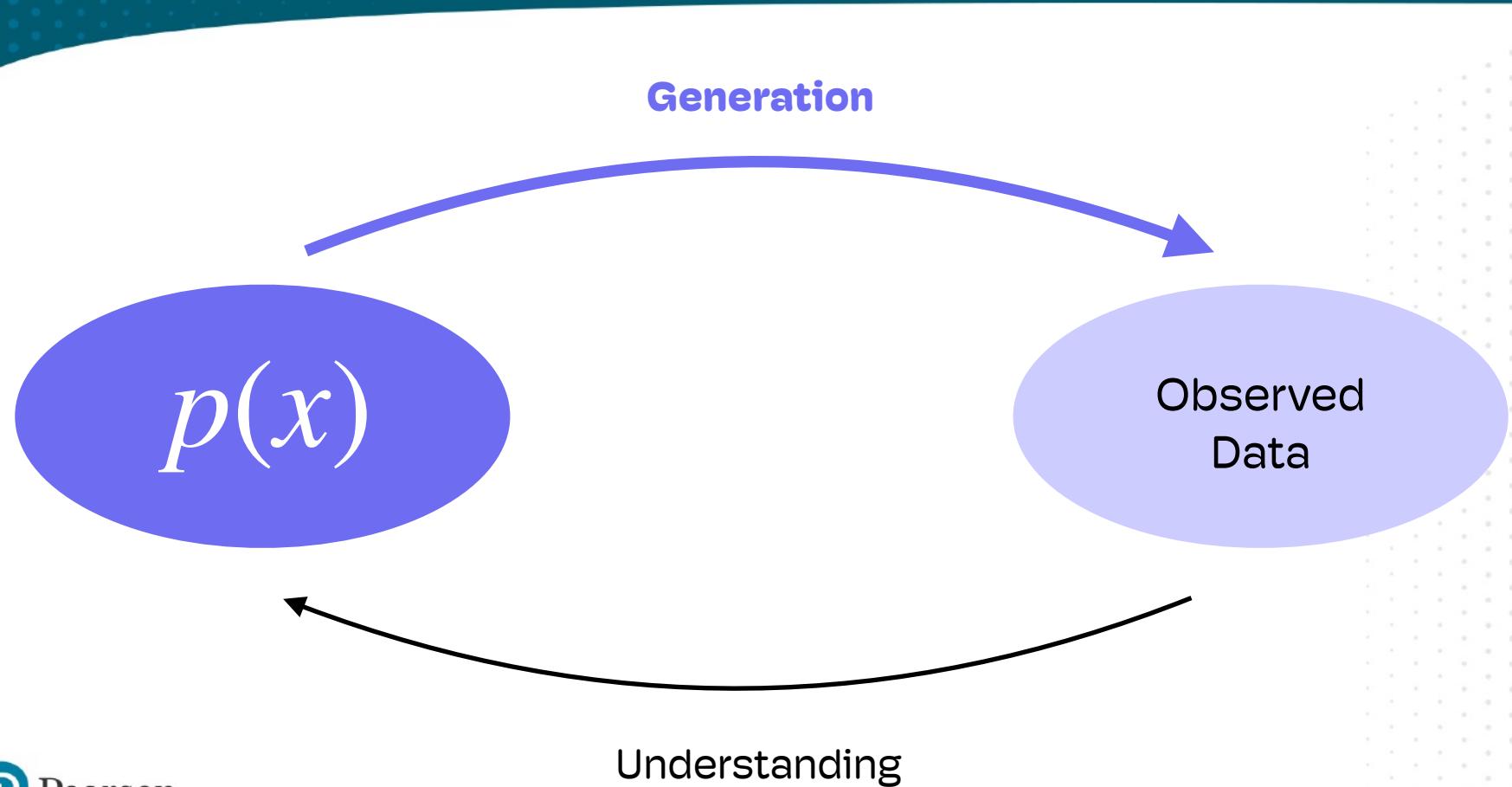
# Generative Processes



# Generative Processes



# Generative Processes



**Inputs (data)**

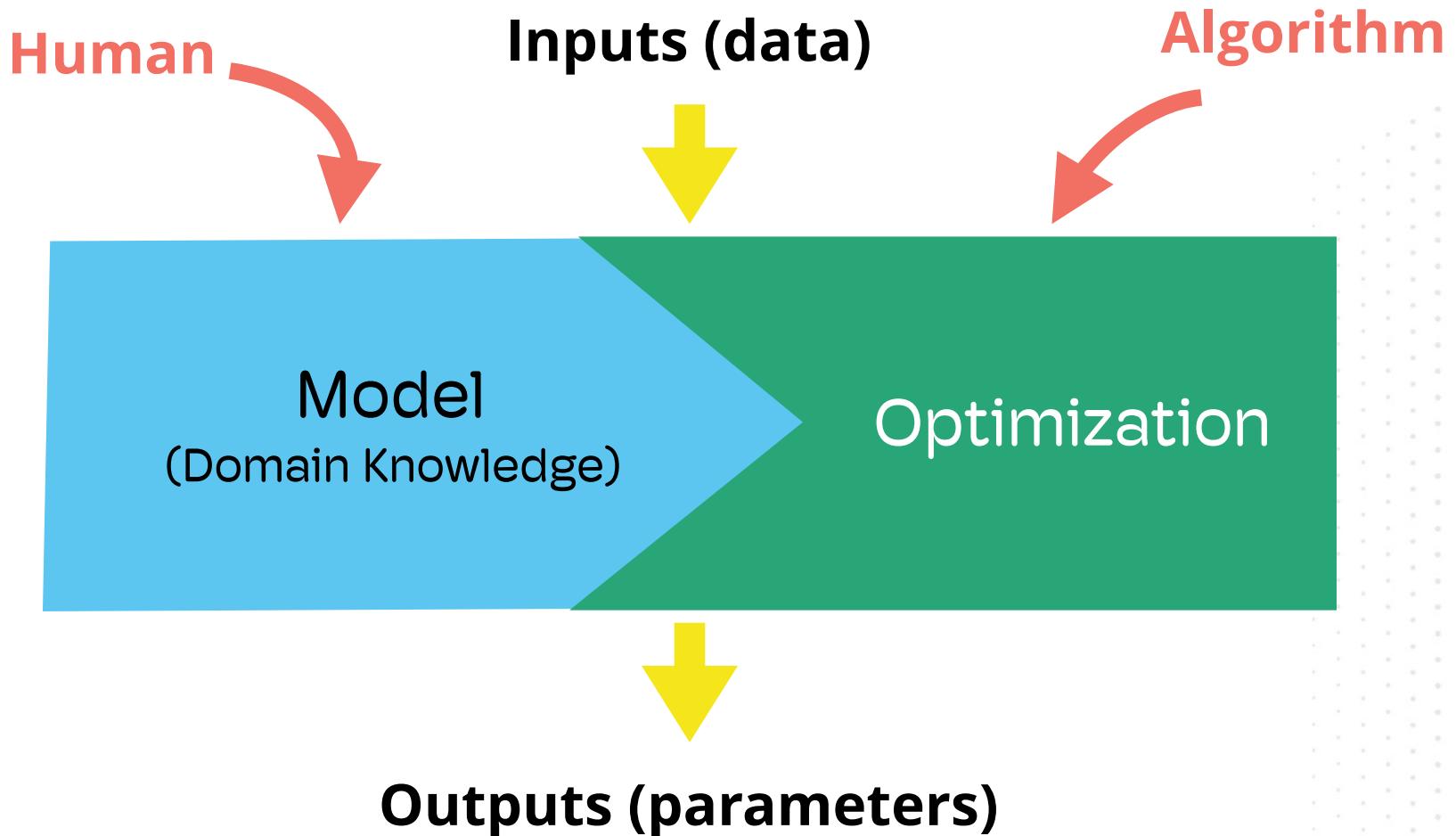


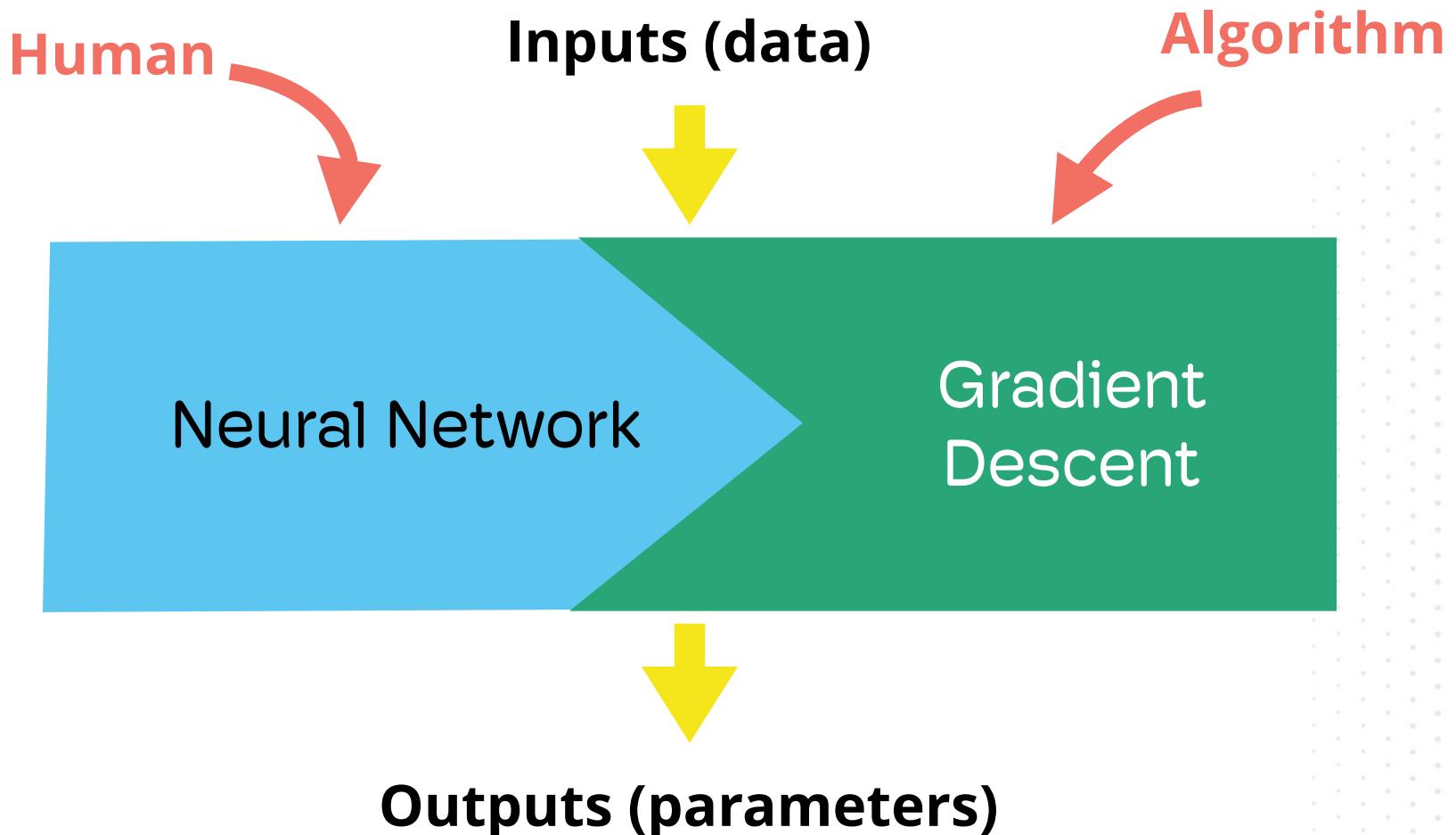
Model  
(Domain Knowledge)

Optimization



**Outputs (parameters)**





# Representation Learning

1. How to best represent  $p(x)$
2. How to learn this representation

# Representation Learning

1. How to best represent  $p(x)$
2. How to learn this representation

Multivariate  
probability  
distribution



**What is the most likely parameterization  
given the training data?**

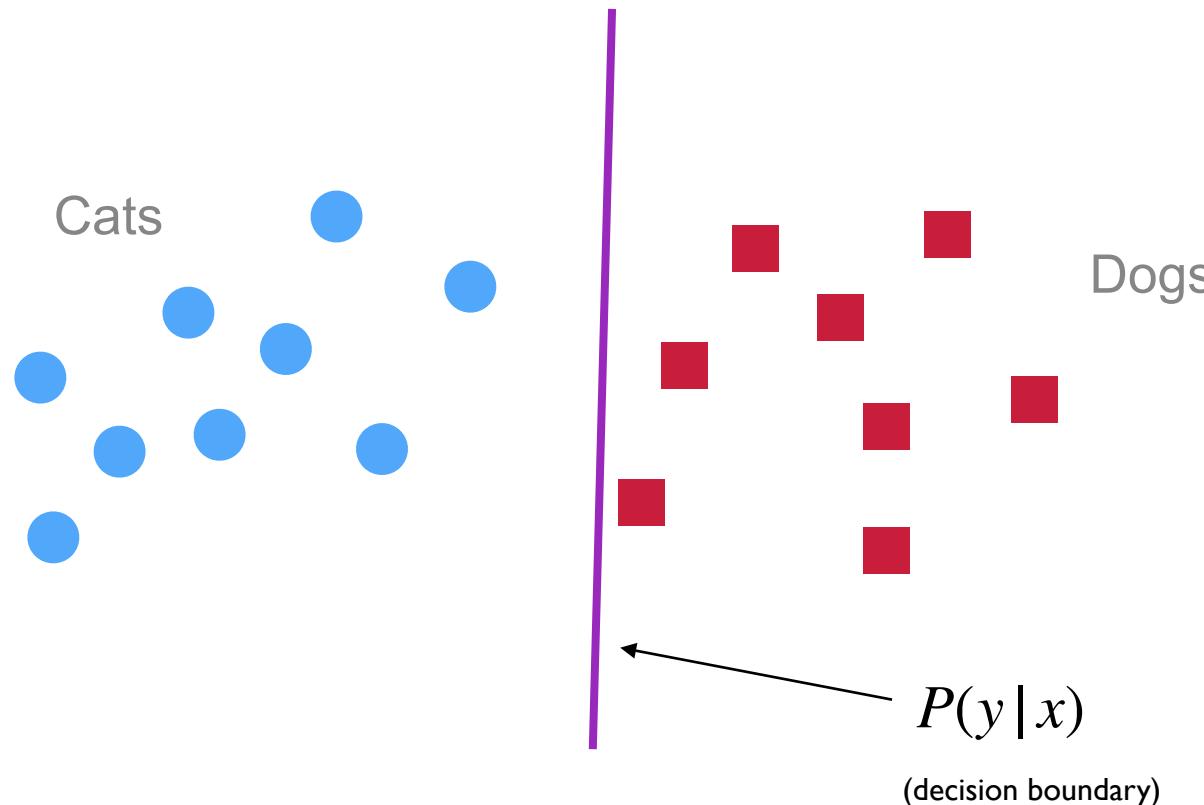
# Representation Learning

$$p_{\theta}(x)$$

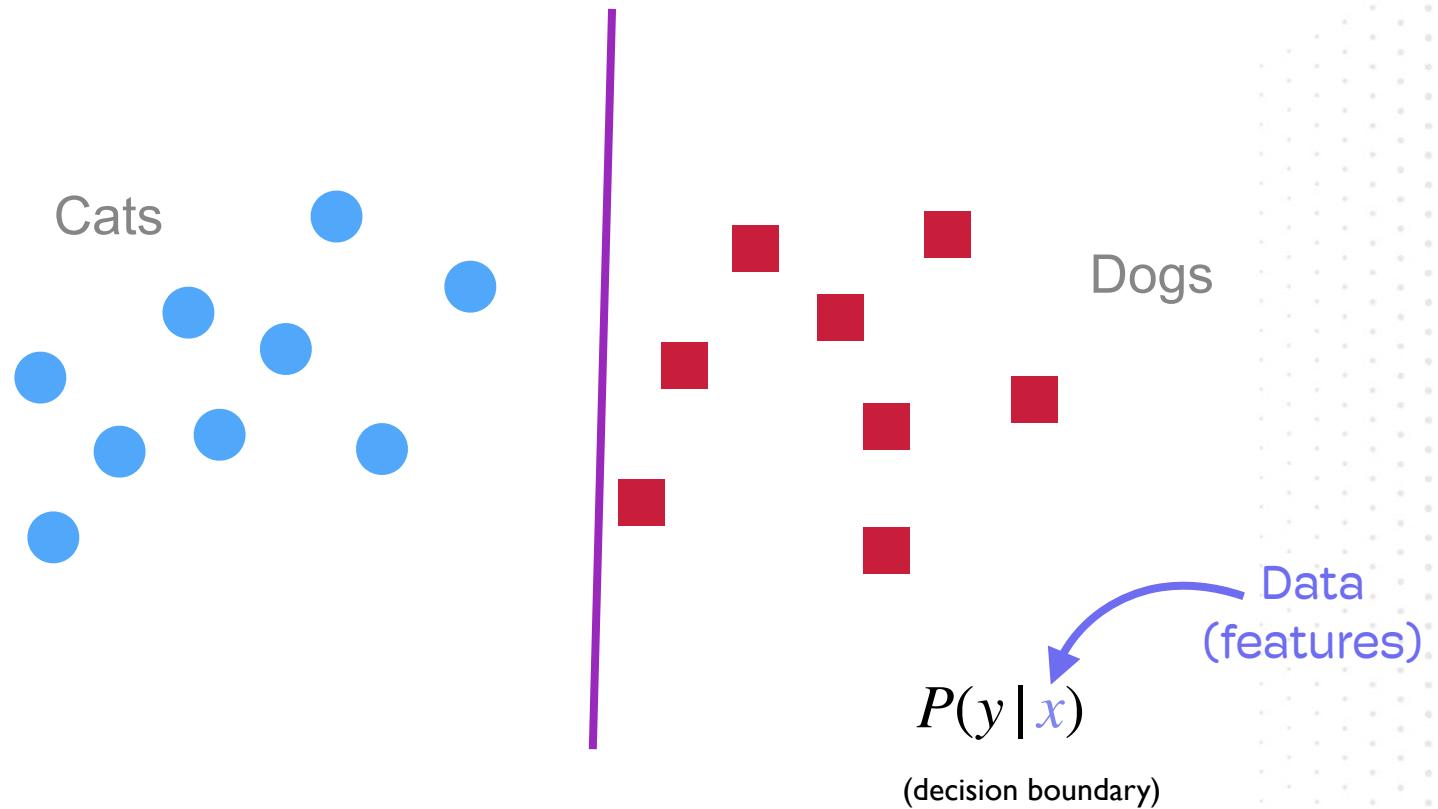
# 1.6

## **Generative versus Discriminative Models**

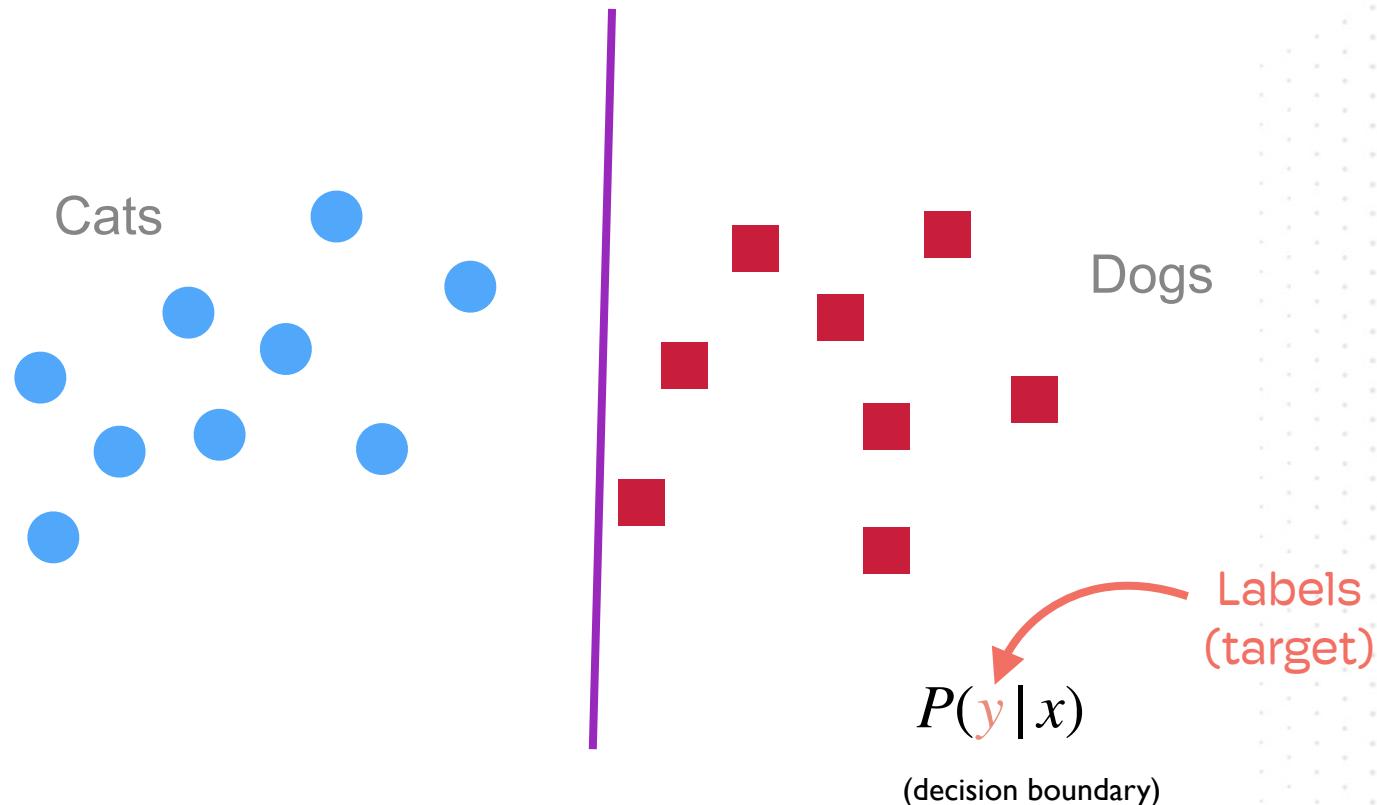
# Discriminative Classifier



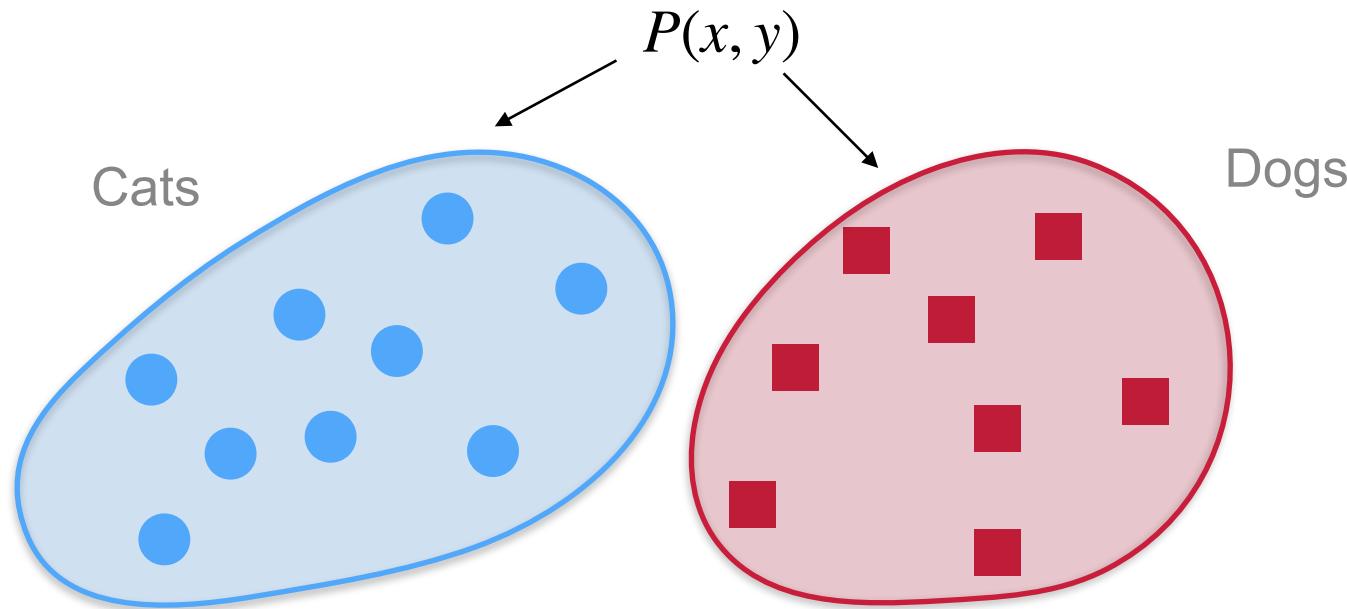
# Discriminative Classifier



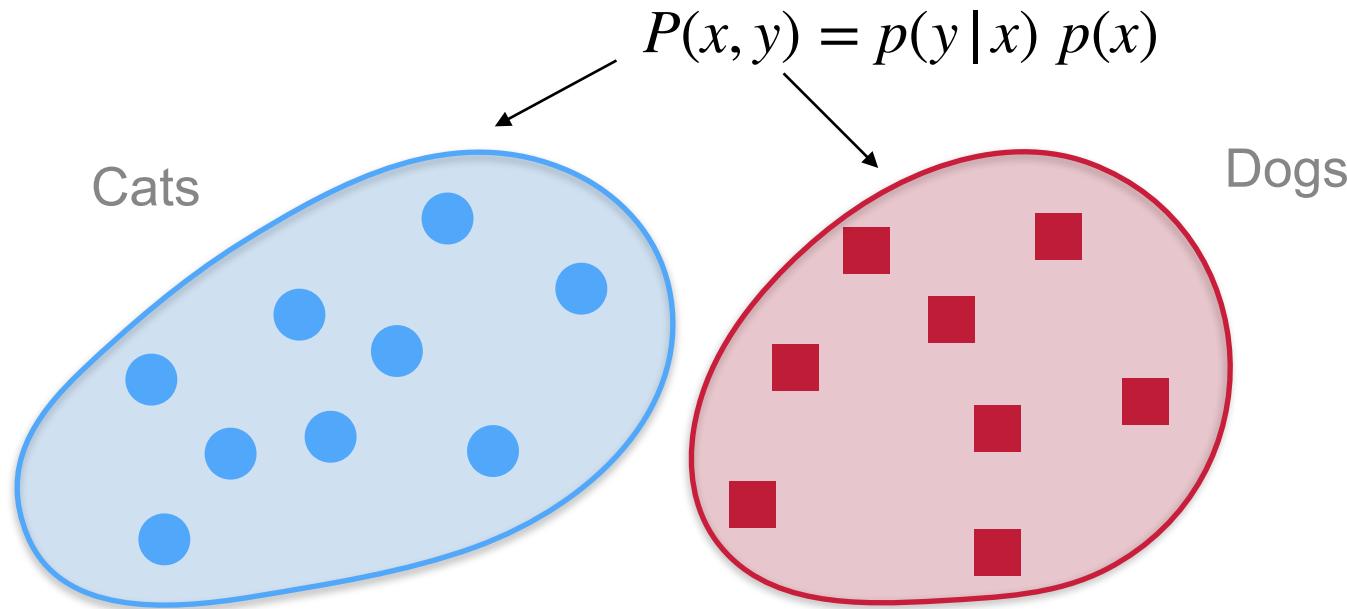
# Discriminative Classifier



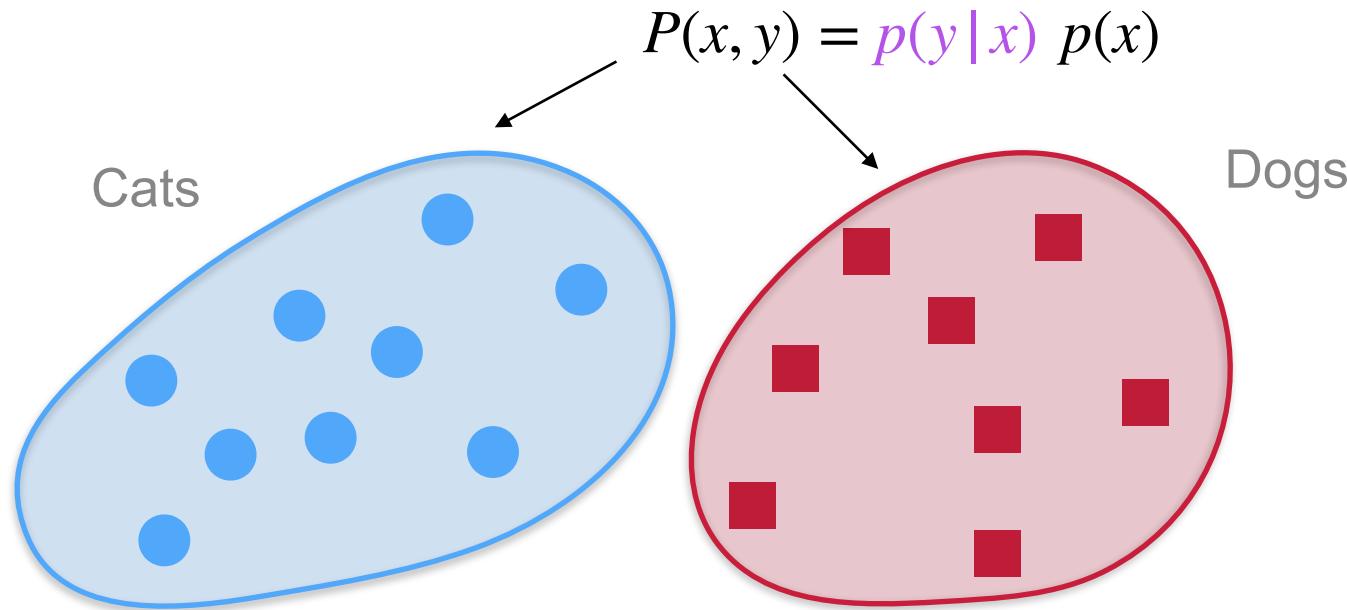
# Generative Model



# Generative Model

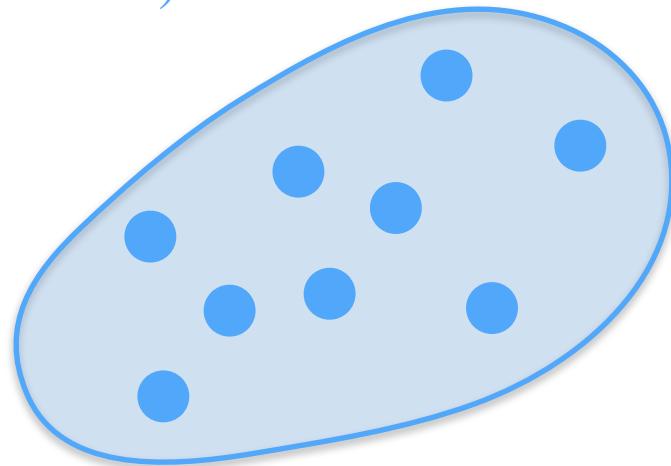


# Generative Model

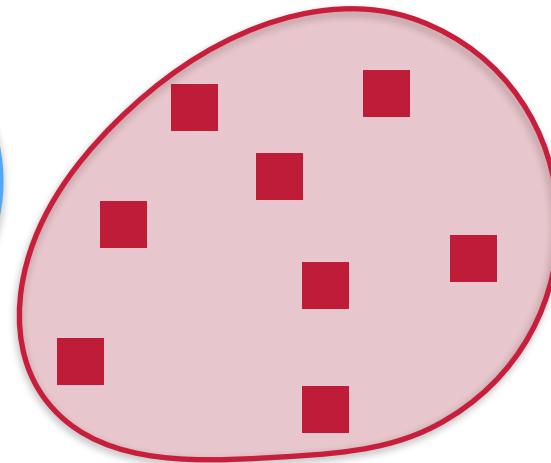


# Generative Model

$P(x, y = \text{cat})$

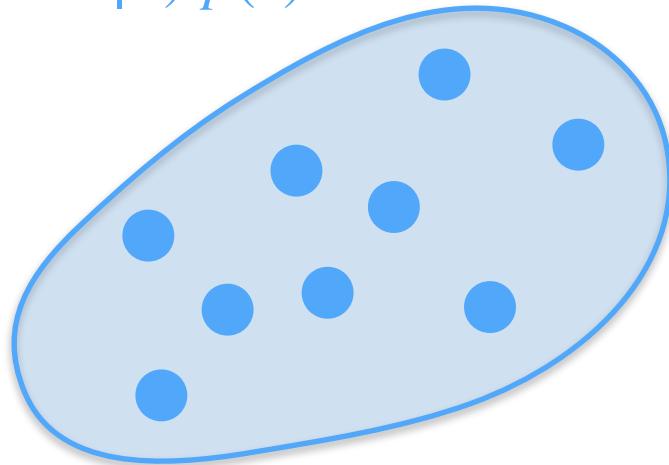


$P(x, y = \text{dog})$

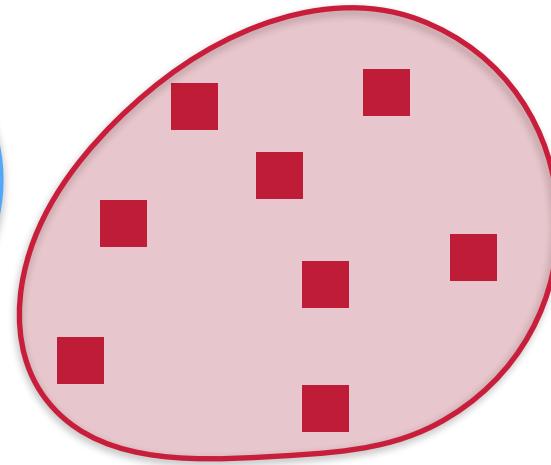


# Generative Model

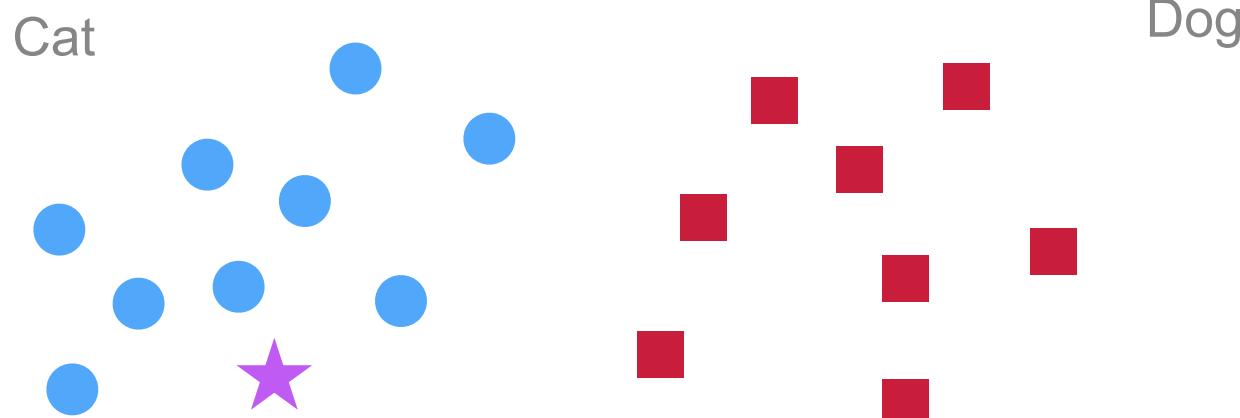
$P(y = \text{cat} | x) p(x)$



$P(y = \text{dog} | x) p(x)$

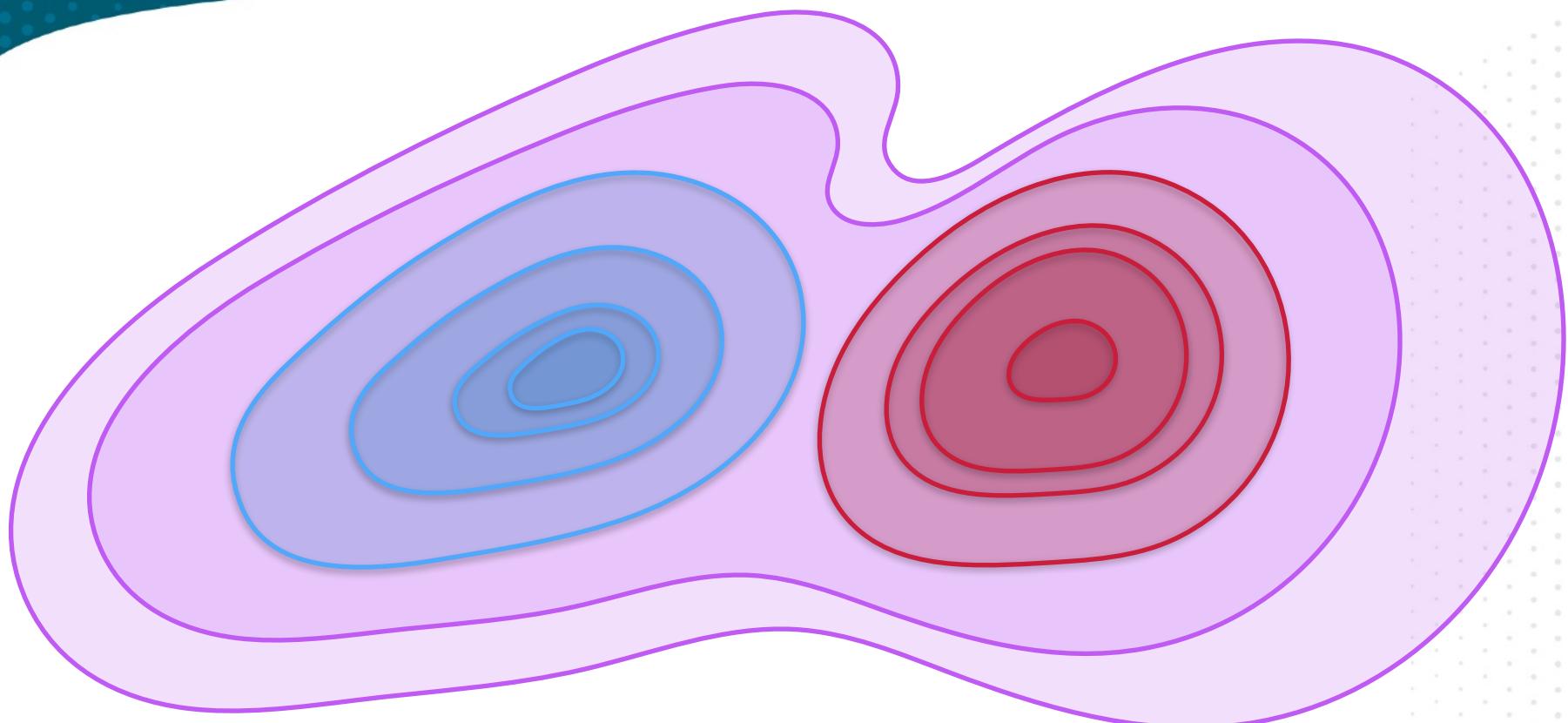


# Conditional Generative Model



$$p(x|y = \text{cat}) = \frac{p(x, y)}{p(y)}$$

# Multivariate Distributions



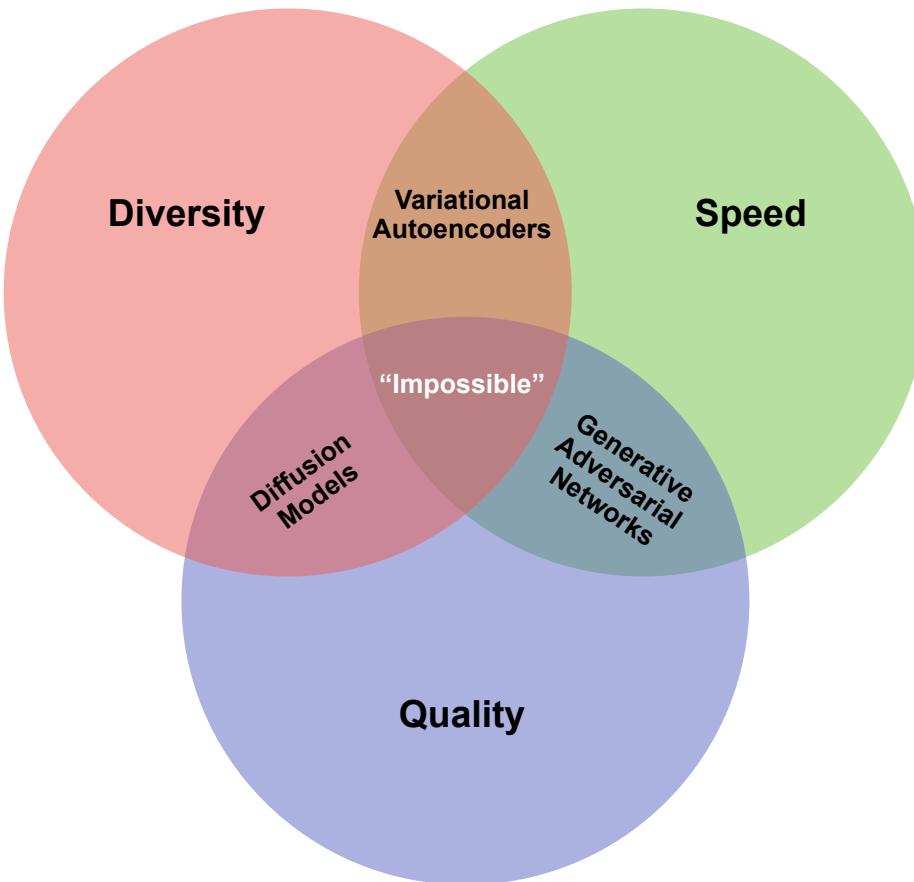
# Drawing Samples

$$x' \sim p(\mathbf{x})$$

1.7

## The Generative Modeling Trilemma

# Generative Modeling Trilemma



1.8

## Introduction to Google Colab

# Live Coding