Hands-on with Generative AI

Instructor: Mahmoud Mohammadi

Introduction

Senior AI Scientist

Senior Applied Scientist

PhD, Computer Science

GAN, University of North Carolina

Welcome to today's class, before we begin, pop into the chat:

Optimize Your Experience

- ✓ Interact with your instructors via Sunday live class.
- ✓ Don’t be shy to speak up and get clarifications !

- ✓ Solve your assignments, MCQs and other assessments and get feedback (Thursday review session)
- ✓ Use your resources! It’s your experience – what you put in is what you’ll get out.

What happened before?

- In past sections you gained exposure to the Python programming language
- Here we will use Python to work hands-on with Generative AI models
- Your past exposure to Python should be sufficient to follow along with the high-level concepts in the examples that we present here
- There will be some details that likely will not be familiar to you…
- …don’t worry, this is just a first look at how to use Python for Generative AI
- …you will have plenty of time to practice in later sections

Recap Quiz

What is a Python module?

A A data type with member variables and methods associated with it.

B An immutable data structure containing a list of values.

C A file containing Python code that can be imported inside a program.

D A computer program that converts Python code into machine code.

Recap Quiz

What is a Python module?

- **A** A data type with member variables and methods associated with it.
- **B** An immutable data structure containing a list of values.
- **C** A file containing Python code that can be imported inside a program.
- **D** A computer program that converts Python code into machine code.

Learning Objectives for Today

- Get a **high-level** overview of generative AI capabilities available today:
  - How text-to-text models work
  - Text-to-text models that are widely used today
  - How to invoke text-to-text models in Python
  - How text-to-image models work
  - Text-to-image models that are widely used today
  - How to invoke text-to-image models in Python
- We try to give a broad view with limited details (esp. on how models work)
- We will dive deeper in later modules

Today's Agenda

	1st Hour (9-10 PT)	2nd Hour (10-11 PT)	3rd Hour (11-12 PT)	4th Hour (12-1 PT)
h:00 - h:15	Introduction	Demonstration of Language Model API and Open Source Model	Overview of Latent Diffusion Models	Demonstration of Text-to-Image API and Open Source Model
h:15 - h:55	Overview of Language Models			
h:55 - h:00	Break	Break	Break	Summary

How is Today's Content Relevant to my Role?

- **PMs:** Understand what types of Generative AI product features could be build with currently available tools.
- **TPMs:** Understand the technical feasibility and requirements of implementing certain generative AI features.
- **SDEs:** Become familiar with how to implement Generative AI capabilities in applications.
- **Engineering Managers:** Become familiar with options available for integrating Generative AI capabilities into the applications that you own.
- **DevOps Engineers:** Get a first look at software and infrastructure requirements for running or accessing Generative AI models.

Today's Agenda

Break

Break

Break

1

Language Models Overview

2

Language Models in Practice

3

Diffusion Models Overview

4

Diffusion Models in Practice

TOPIC #1
Language Models Overview

Overview

- In this section you will:
  - Learn what a *language model* is
  - Understand how language models are used to generate text
  - Understand why building accurate language models is difficult
- We will mention the Transformer architecture, but not cover details yet

Modeling Text

- A *language model* is at the heart of applications like ChatGPT
- A language model is a statistical model that can be used to generate text
- Generating text can be cast as a problem of generating sequences of *tokens*
- Tokens are represented by unique IDs, each corresponding to a word or part of a word in the model's vocabulary

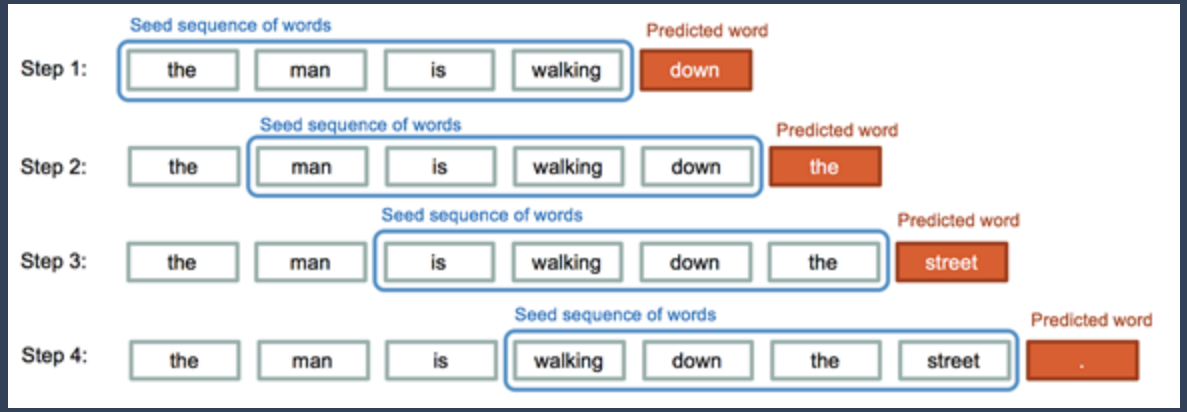
Language Models

- A language model is given by conditional probabilities over tokens:

$$P(w_{n+1} | w_n, \ldots, w_1)$$

- We call the sequence $w_1, \ldots, w_n$ the *context*
- For example, you give the model the context: “This morning I went to the”...
- ...and get probabilities that each token in your vocabulary is the next token:
  - *P* (“office”|previous tokens) = 0.031
  - *P* (“store”|previous tokens) = 0.028
  - *P* (“local”|previous tokens) = 0.027
  - *P* (“gym”|previous tokens) = 0.023
  - ...

Predicting the Next Token in a Sequence



- If we can predict just the next token, we can generate a sequence
- That is, given a context select the next token…
- …then add that token to the context and repeat
- How should we select the next token from the probability model?

Generating Text

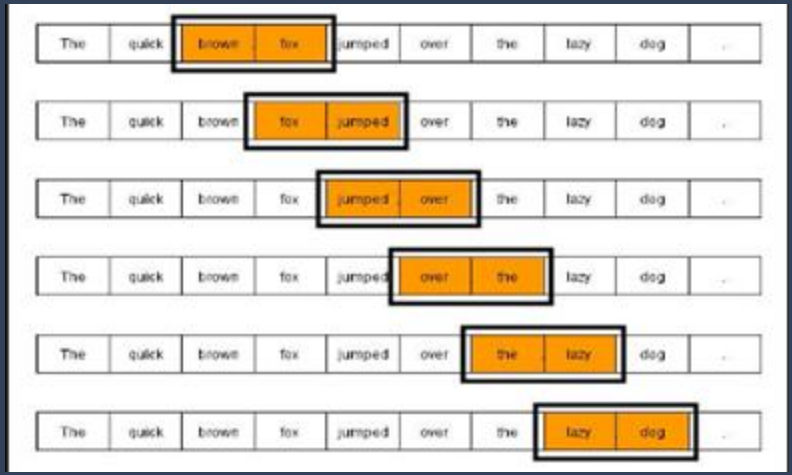
- As an example, this text is generated given this context:
  "This morning I went to the office as I usually do on weekdays."

- The concept of text completion is very versatile:

"Question: What is the largest city in the world? Answer: Tokyo."

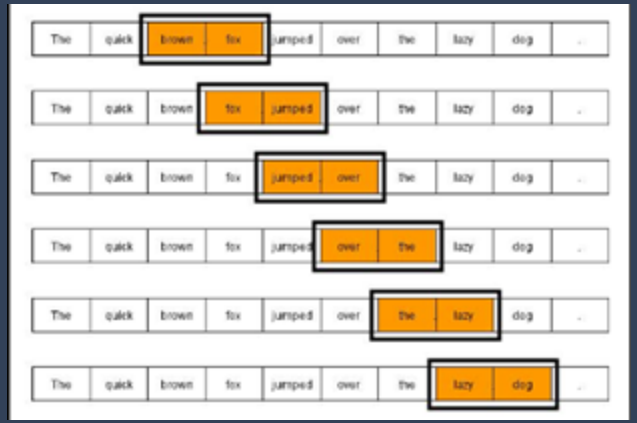
"Human: How are you? Chatbot: I am well, thanks."

Simple Language Models - *n*-grams



- Language models are built by training on a large corpus of text
- Models are trained on the task of predicting the next word given context
- Consider the simple model that only uses the previous word as context
- This is called the *bigram* model

Simple Language Models - *n*-grams (cont.)



$$P(w_i|w_{i-1}) = \frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})}$$

- Training a bigram model is simple...
- ...record the frequency of occurrence of each word for each previous word
- Can generalize this idea to the *n-gram* model
- In general, record word frequencies for each previous *n*-1 words

Challenges with modeling long contexts

- Suppose our vocabulary consists of *V* tokens
- When we build an *n*-gram model we need to specify:
  - For a given token, probabilities for all possible next tokens (*V*-1 values)...
  - ...for each group of *n*-1 context tokens (V^{n-1} combinations)

- **What happens as we increase *n*?**

Long contexts are necessary

- Consider the example, where text is generated with limited context:

"My dog is named Joey. Yesterday Joey and I went for a long walk by the river. The spring weather was beautiful, and many people were out enjoying the day. Joey was happy too. I know this because he told me so."

- The fact that Joey is a dog is important, but that information is outside the context used in this example.

Effectively modeling long contexts

- Various models have been used to capture-range dependencies
- Effective models are generally based on deep neural network architecture
- For example, recurrent neural networks (RNNs) were a commonly used architecture for this task in the past
- Since 2017, the *Transformer* has been the dominant model architecture for language models
- Transformers remove some of the complexity of RNNs
- This results in larger, more performant models that can be trained in less time than previous RNN architectures
- Transformers will be covered in detail in later modules

Summary

01 Language models capture statistical relationships in sequences of tokens

02 Text is generated from a LM by conditioning on prior context

03 Models predict probabilities of the next single token given a context

04 Developing models that capture long-range dependencies is challenging

05 The Transformer effectively models dependencies within long contexts

Topic #1 Quiz

What is the primary purpose of tokenization in language models?

A To convert text into numerical representations.

B To train the model on large dataset.

C To generate new text from the model.

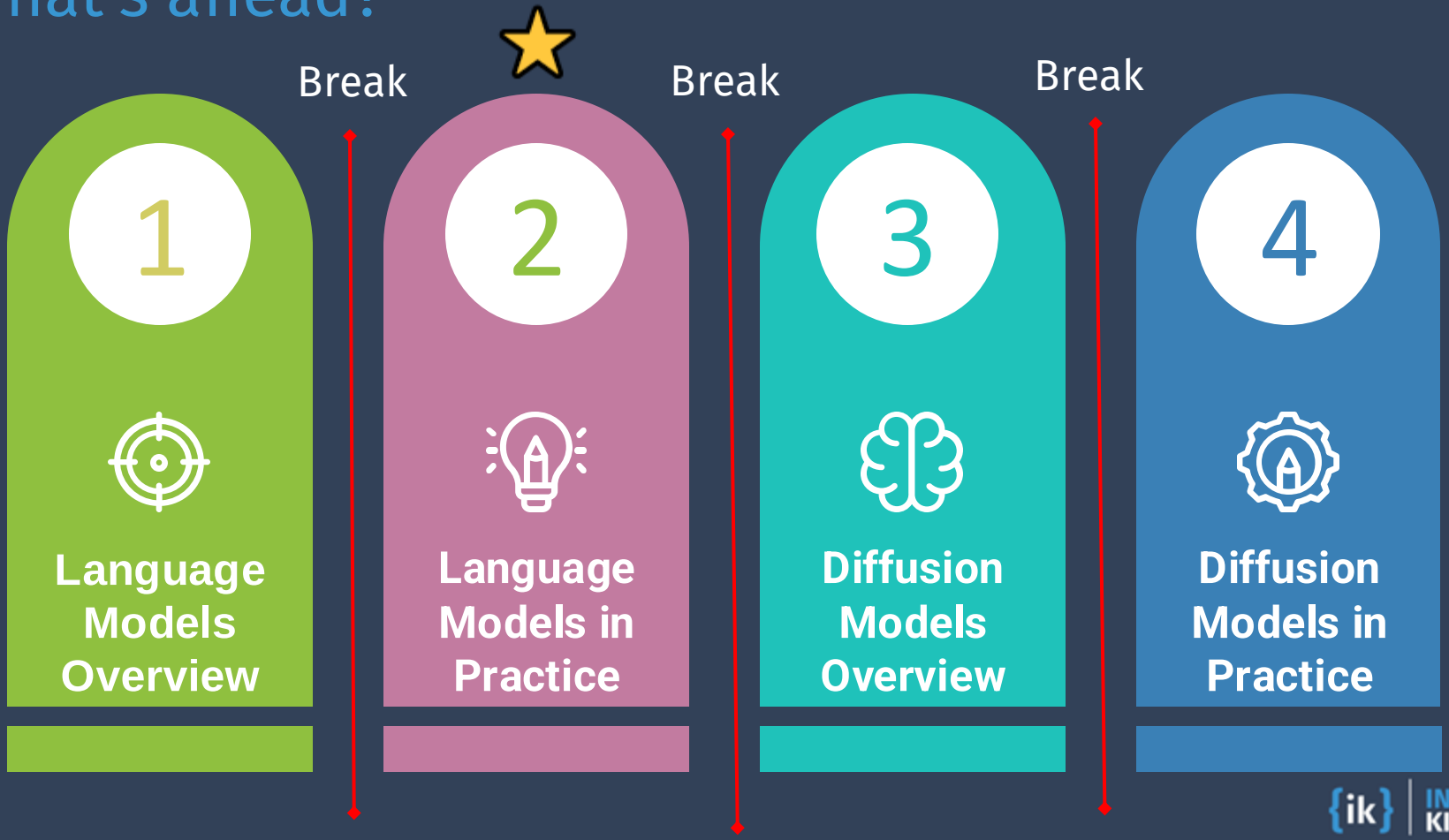
D To optimize the model's performance.

Topic #1 Quiz - Solution

What is the primary purpose of tokenization in language models?

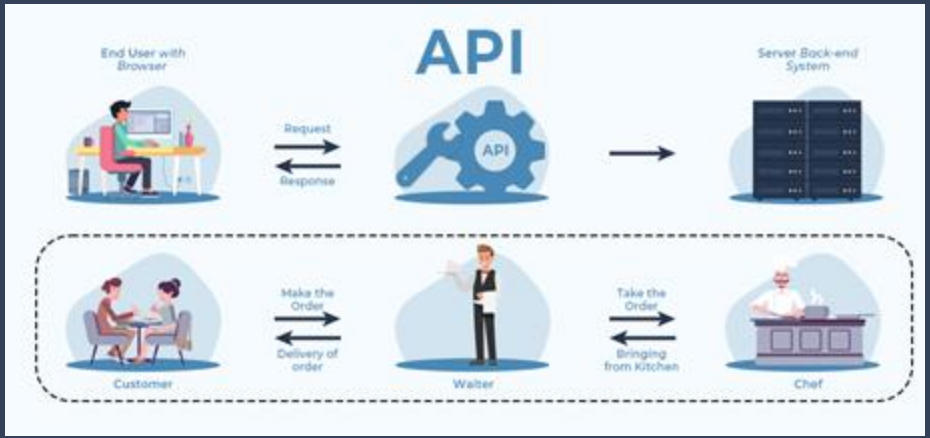
- **A** To convert text into numerical representations.
- **B** To train the model on large dataset.
- **C** To generate new text from the model.
- **D** To optimize the model's performance.

What's ahead?



TOPIC #2
Language Models in Practice

Background: What is an API?



Source: geeksforgeeks.org

- A *web API* (application programming interface) enables applications to access functionality that is hosted on a remote system
- Applications make calls that are send over the internet to a remote system...
- ...the remote system produces a result and sends back to the application

Background: What is a GPU?

- A GPU (graphics processing unit) is a processor initially designed for parallelizing certain graphics-related computations
- The same types of computations are performed by machine learning models
- GPUs can be used to accelerate these computations, which is key for the large models used in Generative AI

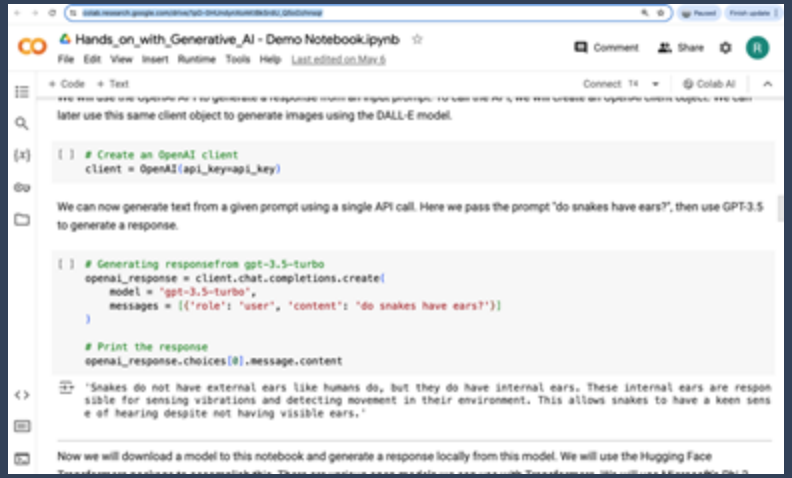
Survey of Large Language Models

- ChatGPT drew enormous public attention to the capabilities of Large Language Models (LLMs)
- There are now many capable LLMs available, and the landscape is rapidly evolving
- We will cover some examples of both closed and open-source models
- Closed models are generally only accessible through APIs, and weights and details of model structure are not publicly known
- The weights and source code for open-source models are freely available and can be modified and customized by developers

Survey of Closed Models

- As of Feb 2025, some of the most popular and performant models are:
  - OpenAI's GPT-4o
  - Anthropic's Claude 3.7 Sonnet
  - Google's Gemini 2.0 Flash
  - X's Grok 3
- These models can be integrated into applications by accessing via APIs
- Weights for the open source models can be accessed from the Hugging Face platform
- Models can be loaded and integrated into Python applications using the Hugging Face "transformers" Python library

GPT-4 Demonstration

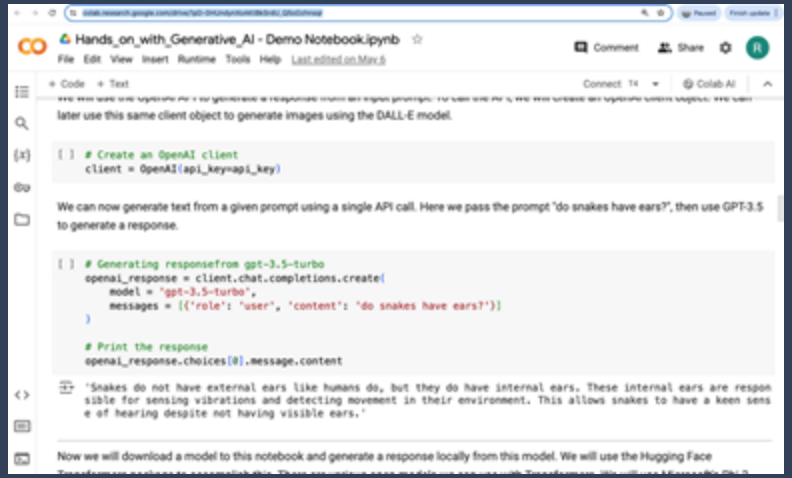


- Demo Notebook
- Here we will demonstrate how to access GPT-3.5 via the OpenAI API.

Survey of Open-Source Models

- Some of the most popular and performant models are:
  - Meta's LLaMA models (LLaMA 3 released in April 2024)
  - Mistral AI's Mistral 7B, Mixtral 8x7B, and Mixtral 8x22B models
  - Microsoft's Phi-3 "small" language models
  - TII's Falcon models
- Weights for these models can be accessed from the Hugging Face platform
- Models can be loaded and integrated into Python applications using the Hugging Face "transformers" Python library

Open Source Model Demonstration



- Demo Notebook
- Here we will demonstrate how a model can be loaded and invoked using the transformers library.

Summary

01 Closed models can be used within applications via their APIs

02 Source code and model weights are available for several open-source models

03 OpenAI, Anthropic, Google, and others all provide access to LLMs via APIs

04 Popular performant open models include LLaMA, Mistral/Mixtral, and Falcon

05 Hugging Face provides a platform and tools for easily working with open models

Topic #2 Quiz

Which company developed the GPT-3 language model?

A Google.

B OpenAI.

C Anthropic.

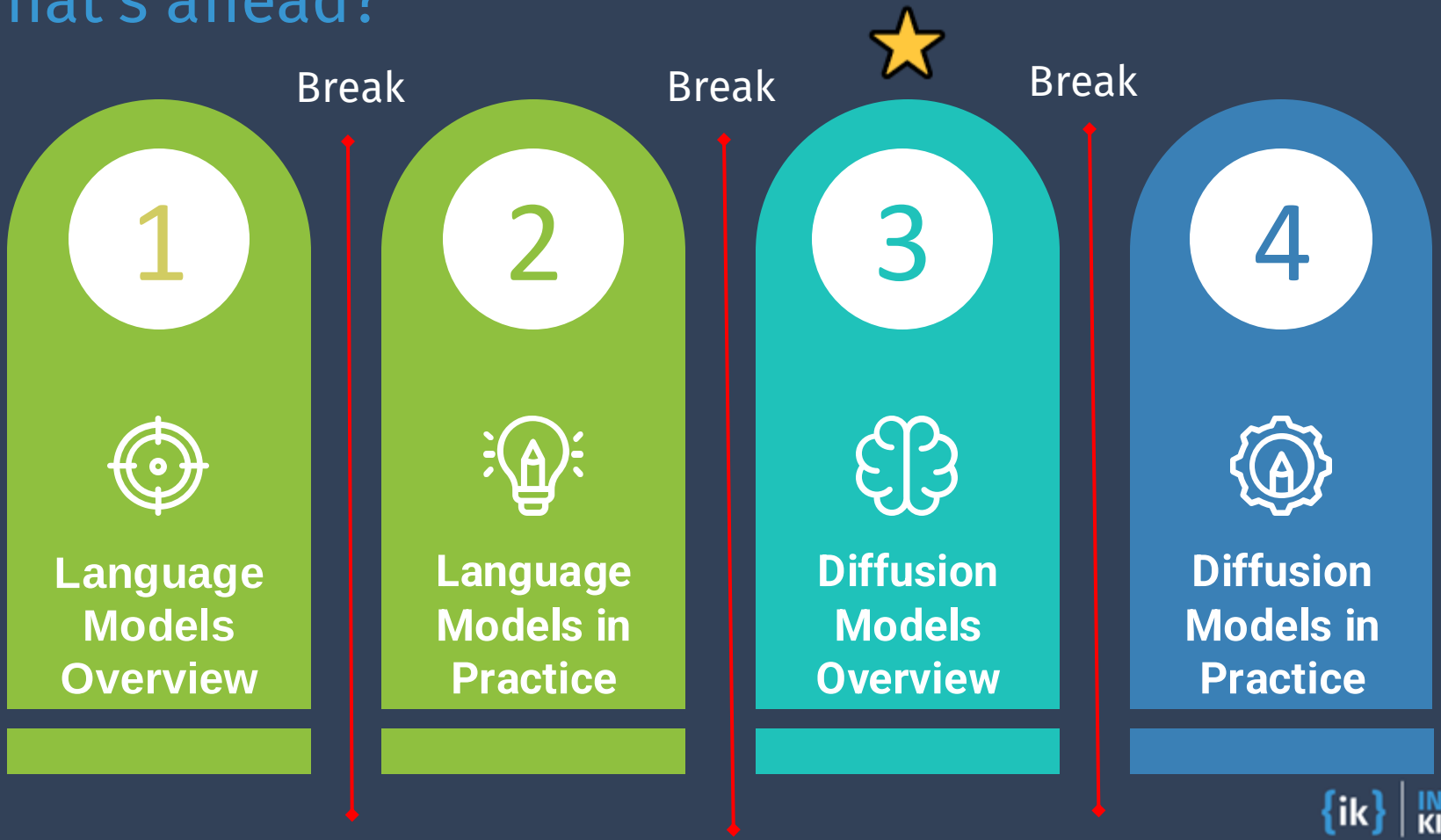
D DeepMind.

Topic #2 Quiz - Solution

Which company developed the GPT-3 language model?

- **A** Google.
- **B** OpenAI.
- **C** Anthropic.
- **D** DeepMind.

What's ahead?

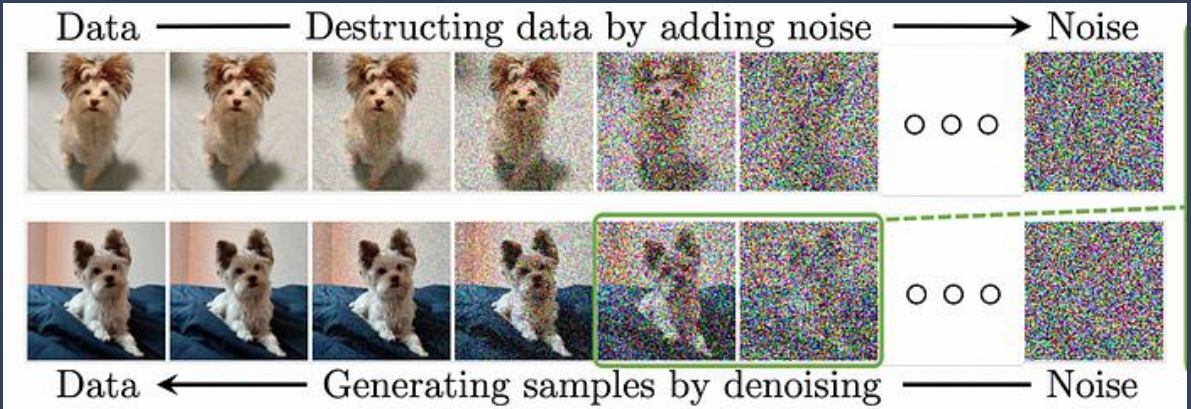


TOPIC #3
Diffusion Models Overview

Overview

- In this section you will:
  - Learn what a *diffusion model* is
  - Understand how diffusion models are used to generate images
  - Understand how generation can be conditioned on text prompts
  - Understand how *latent diffusion models* make generation more efficient
- This is a high-level presentation, with details to come in later modules

Generating Images with Diffusion Models



https://arxiv.org/pdf/2209.00796

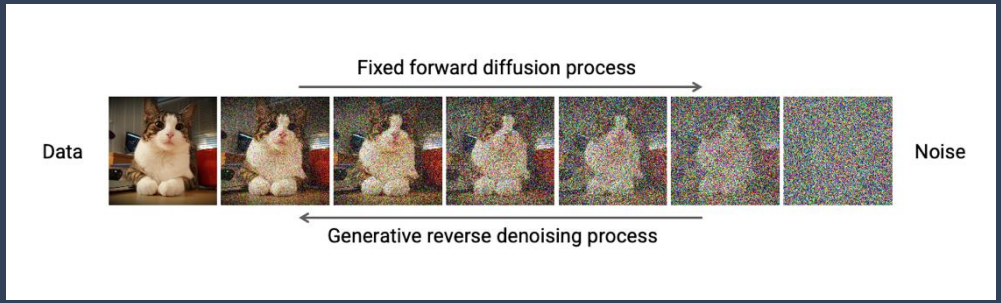
[Training, Multiple steps for each image]

1. Start with real image from training dataset
2. Add noise (forward process)
3. Train Model to predict noise

[Sampling (Generation)]

1. Start with random noise
2. Predict noise (using U-Net)
3. Remove predicted noise
4. Repeat until clean image is formed

Unconditional Generation



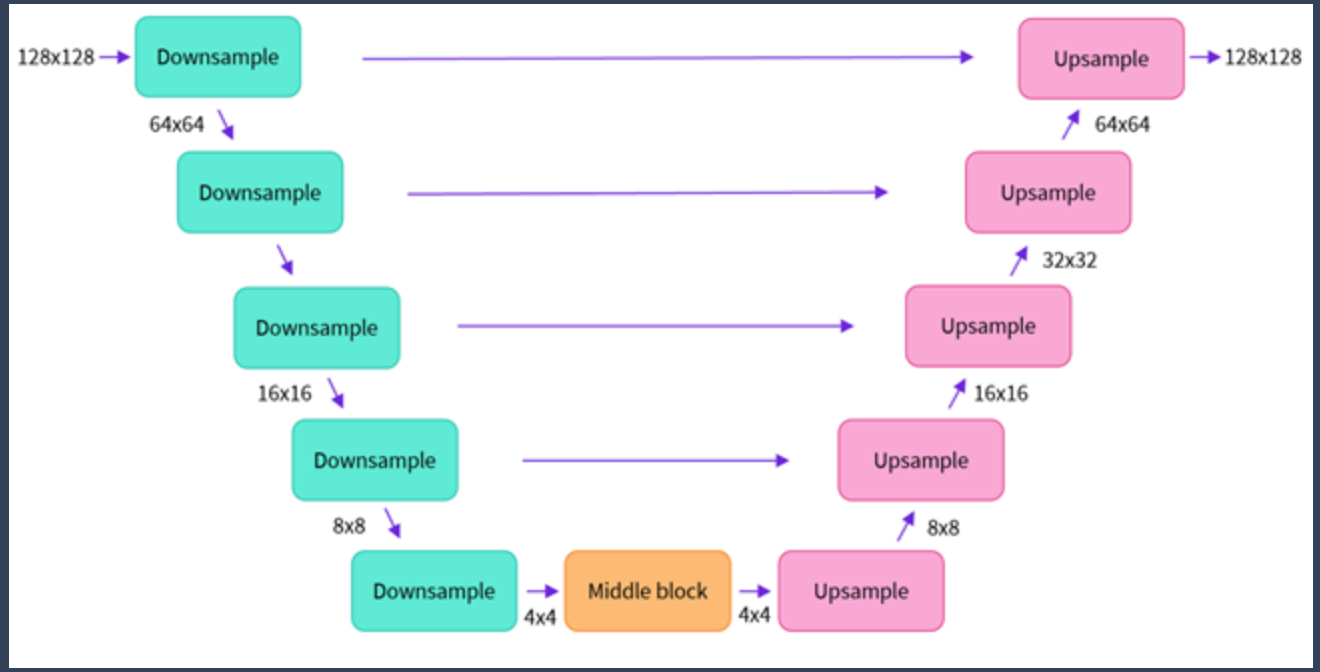
https://cvpr2023-tutorial-diffusion-models.github.io/

- We can train a model that generates random images from some distribution of images (e.g., cats, churches, celebrities)
- To achieve this, a diffusion model is trained on sample images from the desired distribution
- To train the model, we add noise to training images and learn to approximately recover the input images
- This allows us to use the model to feed in random noise, and recover a random image from the input distribution
- The model architecture for noise prediction most often used is the *U-Net*

Conditional Generation

- Alternatively, we can train a model on a broader collection of images, then restrict the image content when generating
- The model can be augmented to allow for additional external inputs for guiding image generation (e.g., text, other images)
- A text-to-image model of this form is composed of a text encoder coupled with a U-Net

The Diffusion and U-Net Architectures



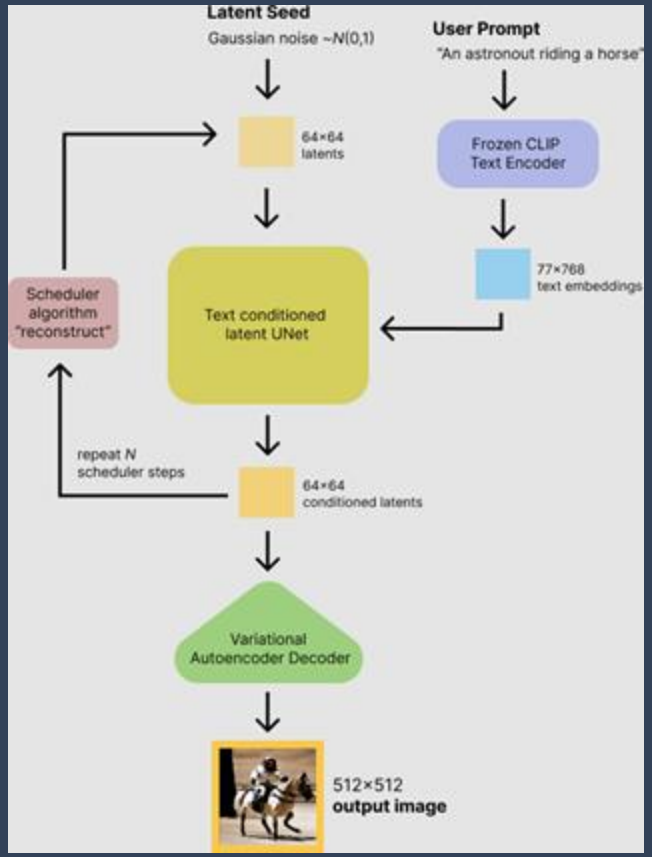
Challenges with Using Diffusion Models

- The input and output of the U-Net are images of the size and resolution of the images that we want to generate
- Generating large, full-color images can be slow, and training a model has high computational and memory requirements
- As a result, it is more efficient to work with a compressed representation of the images
- When generating an image, the U-Net generates a compressed representation of the image (in the “latent” image space), then this latent representation expanded into a full image

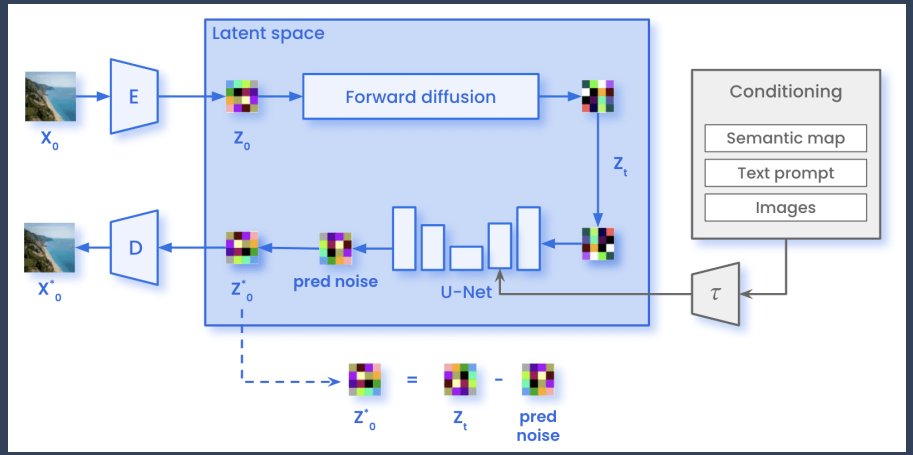
Latent Diffusion Models

The final model contains the following components:

- A **text encoder** for creating text embeddings
- A **text-conditioned U-Net** for generating images in the latent space
- A **variational autoencoder** for converting the latent image into a full image
- A **scheduler** for controlling the iterative generation process

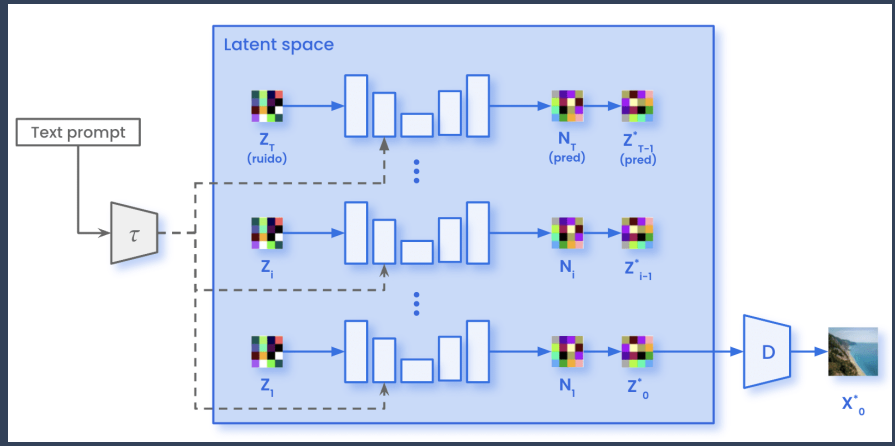


Conditional Generation



https://blog.marvik.ai

Training



Generating

Summary

01	Diffusion models generate images by denoising random noise
02	Denoising is performed by a scheduler that invokes the denoising model
03	Denoising can be conditioned on a text prompt
04	The diffusion process can be slow and inefficient when applied directly to images
05	Diffusion can be performed in a latent space, then an image is generated from VAE

Topic #3 Quiz

Which of the following is a key challenge in diffusion models?

A Overfitting.

B Underfitting.

C Slow sampling process.

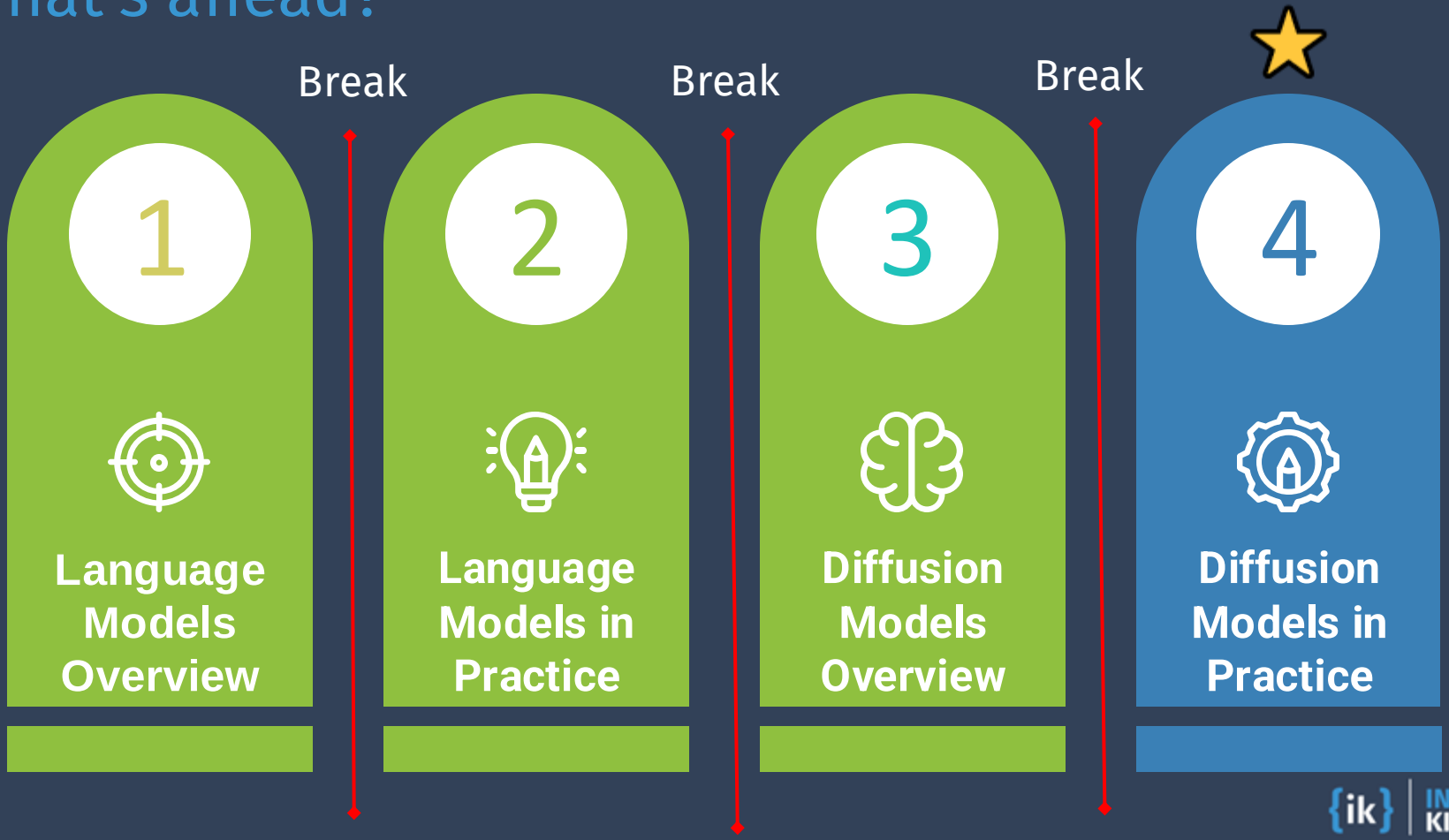
D Difficulty in training.

Topic #3 Quiz - Solution

Which of the following is a key challenge in diffusion models?

- **A** Overfitting.
- **B** Underfitting.
- **C** Slow sampling process.
- **D** Difficulty in training.

What's ahead?



TOPIC #4
Diffusion Models in Practice

Survey of Large Language Models

- ChatGPT drew enormous public attention to the capabilities of Large Language Models (LLMs)
- There are now many capable LLMs available, and the landscape is rapidly evolving
- We will cover some examples of both closed and open-source models
- Closed models are generally only accessible through APIs, and weights and details of model structure are not publicly known
- The weights and source code for open-source models are freely available and can be modified and customized by developers

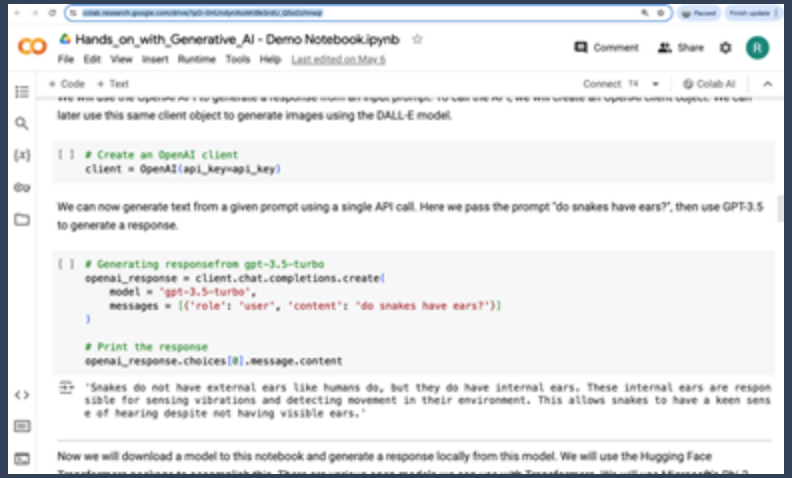
Survey of Text-to-Image Models

- As with language models, there are multiple capable text-to-image models
- We will cover some examples of both closed and open-source models
- Closed models are generally only accessible through APIs, and weights and details of model structure are not publicly known
- The weights and source code for open-source models are freely available and can be modified and customized by developers

Survey of Closed Models

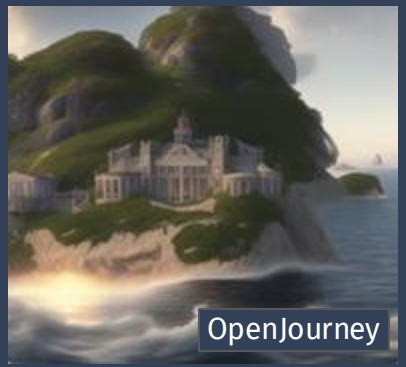
- Some of the most popular and performant models are:
  - OpenAI's DALL-E model
  - Midjourney
  - Google's Imagen model
- These models can be integrated into applications by accessing via APIs

DALL-E Demonstration



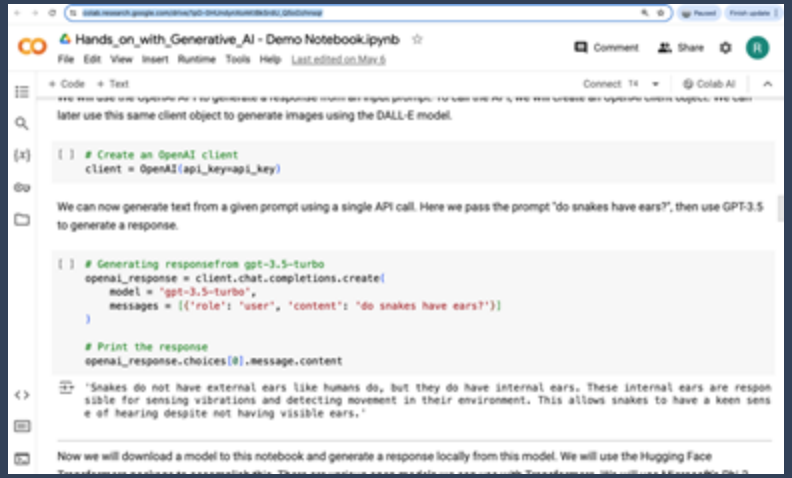
- Demo Notebook
- Here we will demonstrate how to access DALL-E via the OpenAI API.

Survey of Open-Source Models



- Some of the most popular and performant models are:
  - Stability AI's Stable Diffusion model
  - PromptHero's OpenJourney model
- Weights for these models can be accessed from the Hugging Face platform
- Models can be loaded and integrated into Python applications using the Hugging Face “diffusers” Python library

Open Source Model Demonstration



- Demo Notebook
- Here we will demonstrate how Stable Diffusion 3 can be loaded and invoked using the diffusers library.

Summary

01 Closed models can be used within applications via their APIs

02 Source code and model weights are available for several open-source models

03 OpenAI and Google's Imagen provide access to text-to-image models via APIs

04 Stable Diffusion is a popular performant open model

05 Hugging Face provides a platform and tools for easily working with open models

Topic #4 Quiz

Which of the following is an open-source text-to-image model?

A DALL-E 2.

B Stable Diffusion.

C DALL-E.

D Imagen.

Topic #4 Quiz - Solution

Which of the following is an open-source text-to-image model?

- **A** DALL-E 2.
- **B** Stable Diffusion.
- **C** DALL-E.
- **D** Imagen.

Class Summary

Topic 1	Language Models are probabilistic models of text sequences	Modeling long-range dependencies is challenging	The Transformer model is SOA
Topic 2	There are capable closed and open models	Closed models include GPT-4, Claude, and Gemini	Open models include LLaMA, Mistral, and Falcon
Topic 3	Diffusion Models are used to sample images	Model output can be conditioned on text prompts	Latent Diffusion Models use work with compressed images
Topic 4	There are capable closed and open models	Closed models include DALL-E, Midjourney, and Imagen	Open models include Stable Diffusion and OpenJourney