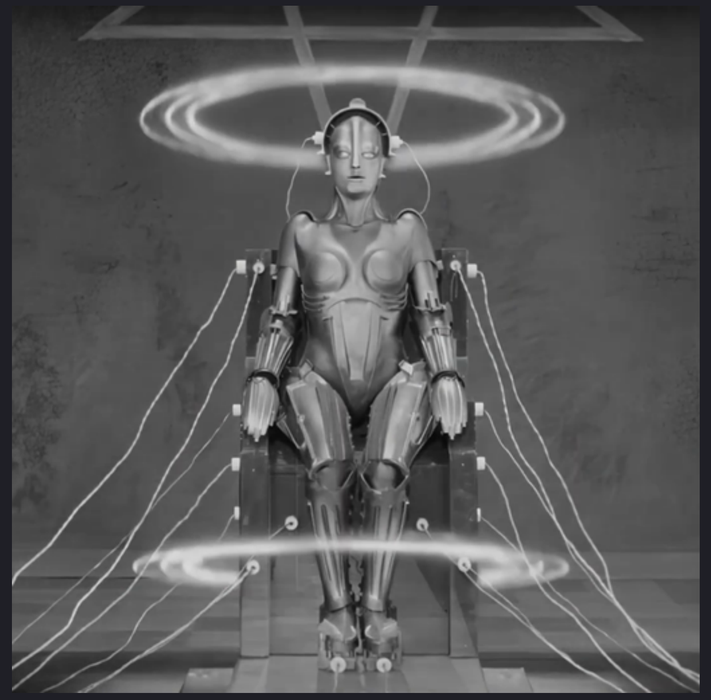
Stable Diffusion, CoreML
and the Future of AI in iOS

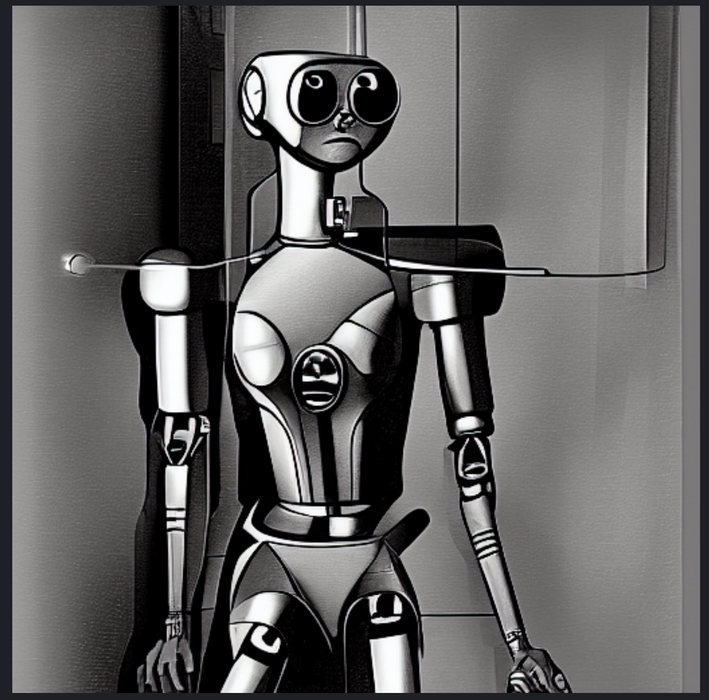
Allison McEntire | Staff Engineer at Urban Outfitters, Nuuly
iOSDevUK 2023

What is Stable Diffusion?

Capabilities



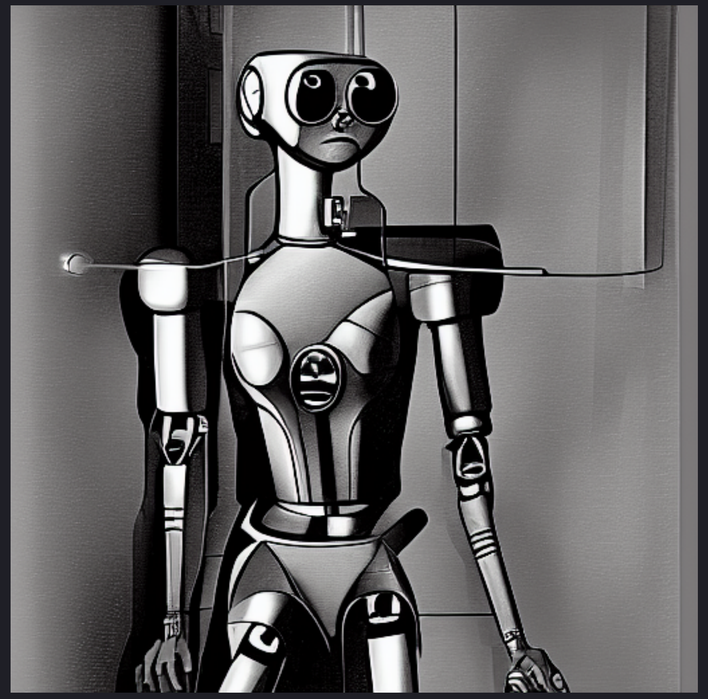
Metropolis, 1927



Text-to-image
Prompt: A female robot in the style of the Fritz Lang movie Metropolis

What is Stable Diffusion?

Capabilities



Text-to-image
Prompt: A female robot in the style of the Fritz Lang movie Metropolis

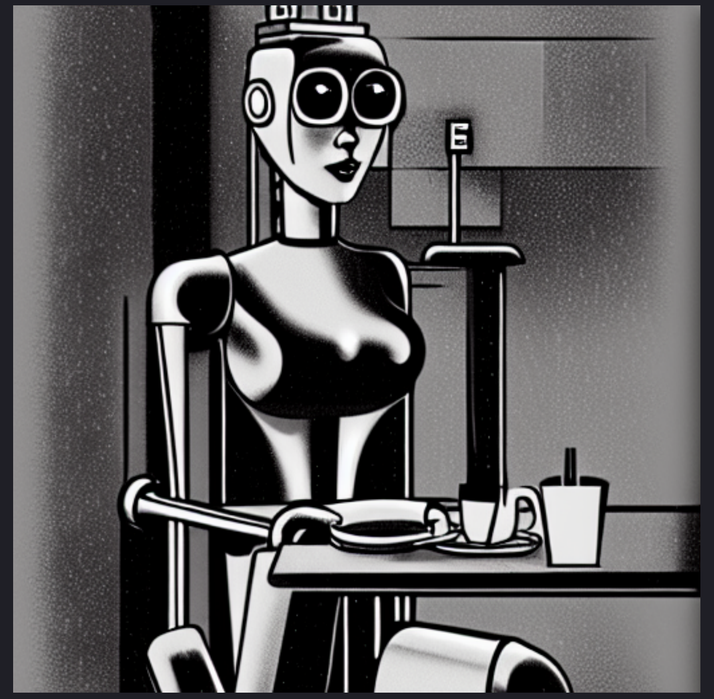
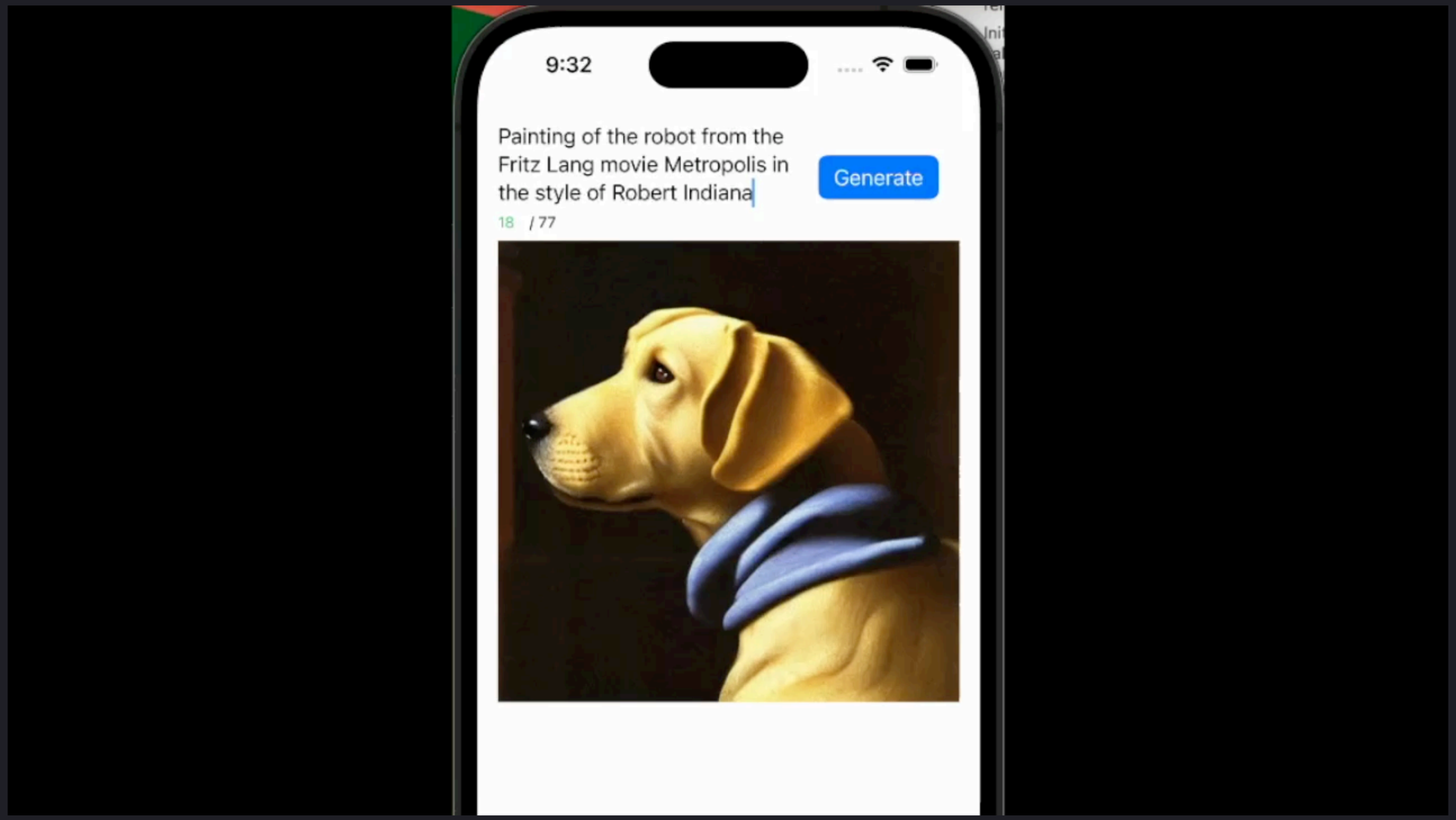


Image-to-image
Prompt: A female robot drinking coffee in the style of the Fritz Lang movie Metropolis

What is Stable Diffusion?

What is a Latent Diffusion Model?

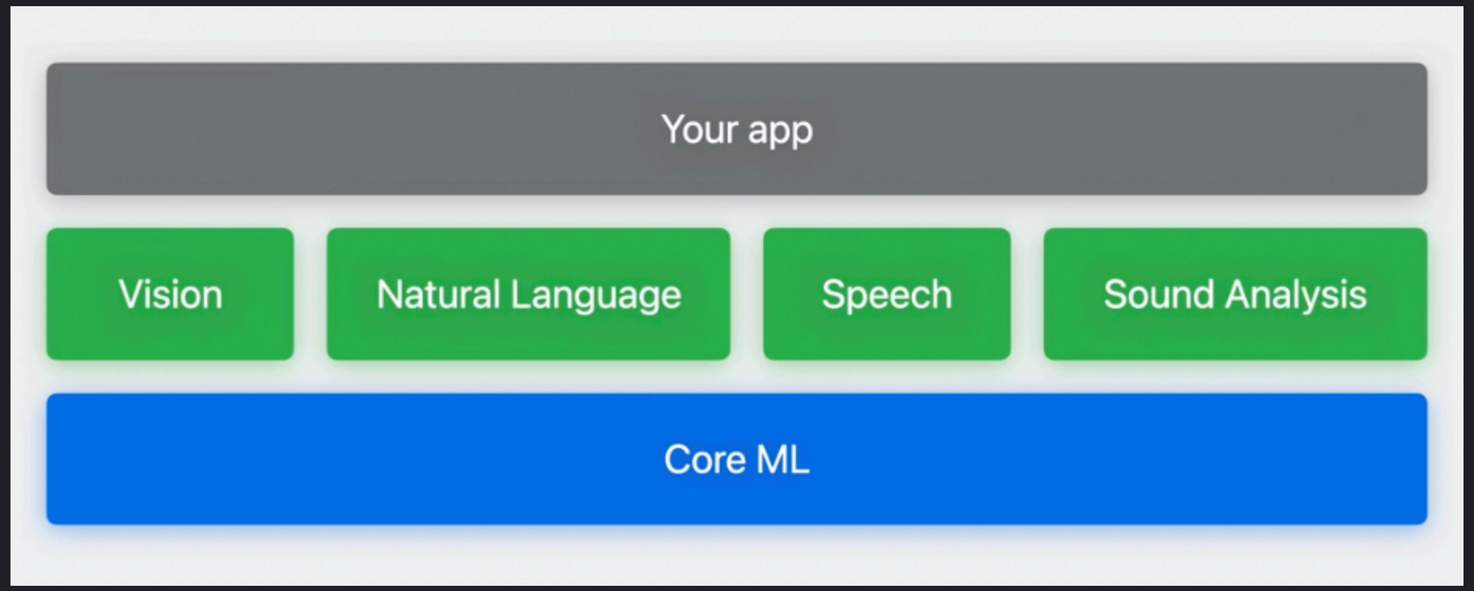


What is Stable Diffusion?

Latent Diffusion Model: Pipeline (image-to-image)

What is Stable Diffusion?

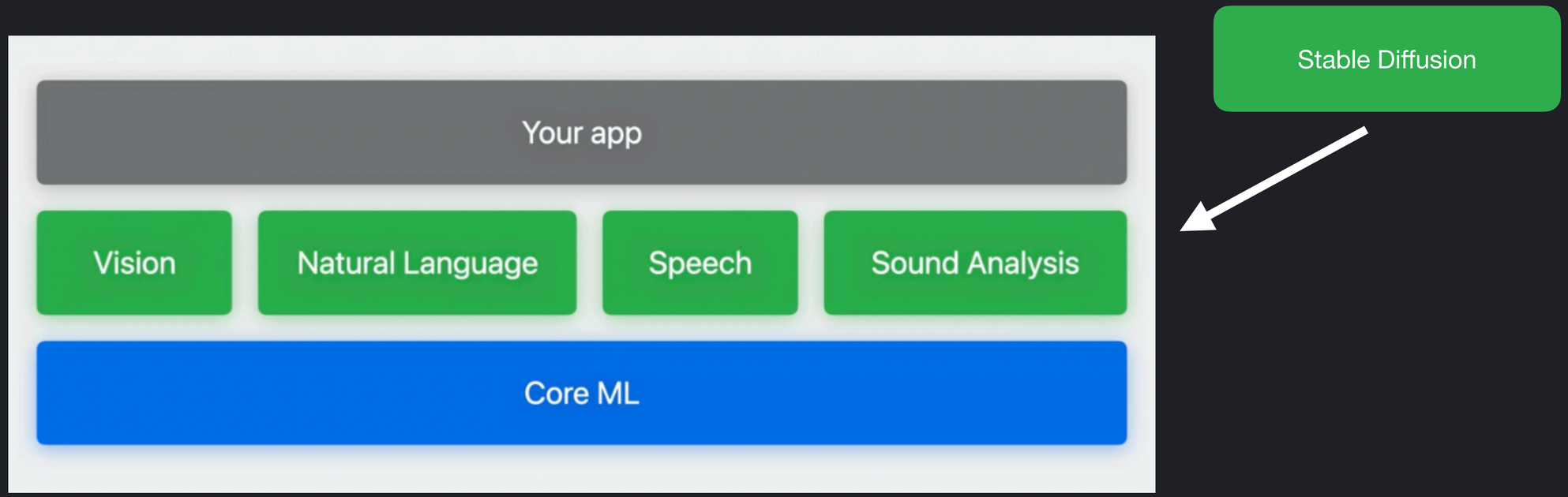
The Swift Package



https://developer.apple.com/videos/play/wwdc2023/10049/

What is Stable Diffusion?

The Swift Package



What is Stable Diffusion?

The Swift Package

ml-stable-diffusion

Reposit github.com/apple/ml-stable-diffusion

Dependency Rule Up to Next Major Version 1.0.0 < 2.0.0

Add to Project Diffusion

Core ML Stable Diffusion

Run Stable Diffusion on Apple Silicon with Core ML

[Blog Post] [BibTeX]

This repository comprises:

- `python_coreml_stable_diffusion`, a Python package for converting PyTorch models to Core ML format and performing image generation with Hugging Face diffusers in Python
- `StableDiffusion`, a Swift package that developers can add to their Xcode projects as a dependency to deploy image generation capabilities in their apps. The Swift package relies on the Core ML model files generated by `python_coreml_stable_diffusion`

If you run into issues during installation or runtime, please refer to the FAQ section. Please refer to the System Requirements section before getting started.

System Requirements

Model Conversion:

macOS	Python	coremltools
13.1	3.8	7.0

Project Build:

macOS	Xcode	Swift
13.1	14.3	5.8

Target Device Runtime:

macOS	iPadOS, iOS
13.1	16.2

Target Device Runtime (With Memory Improvements):

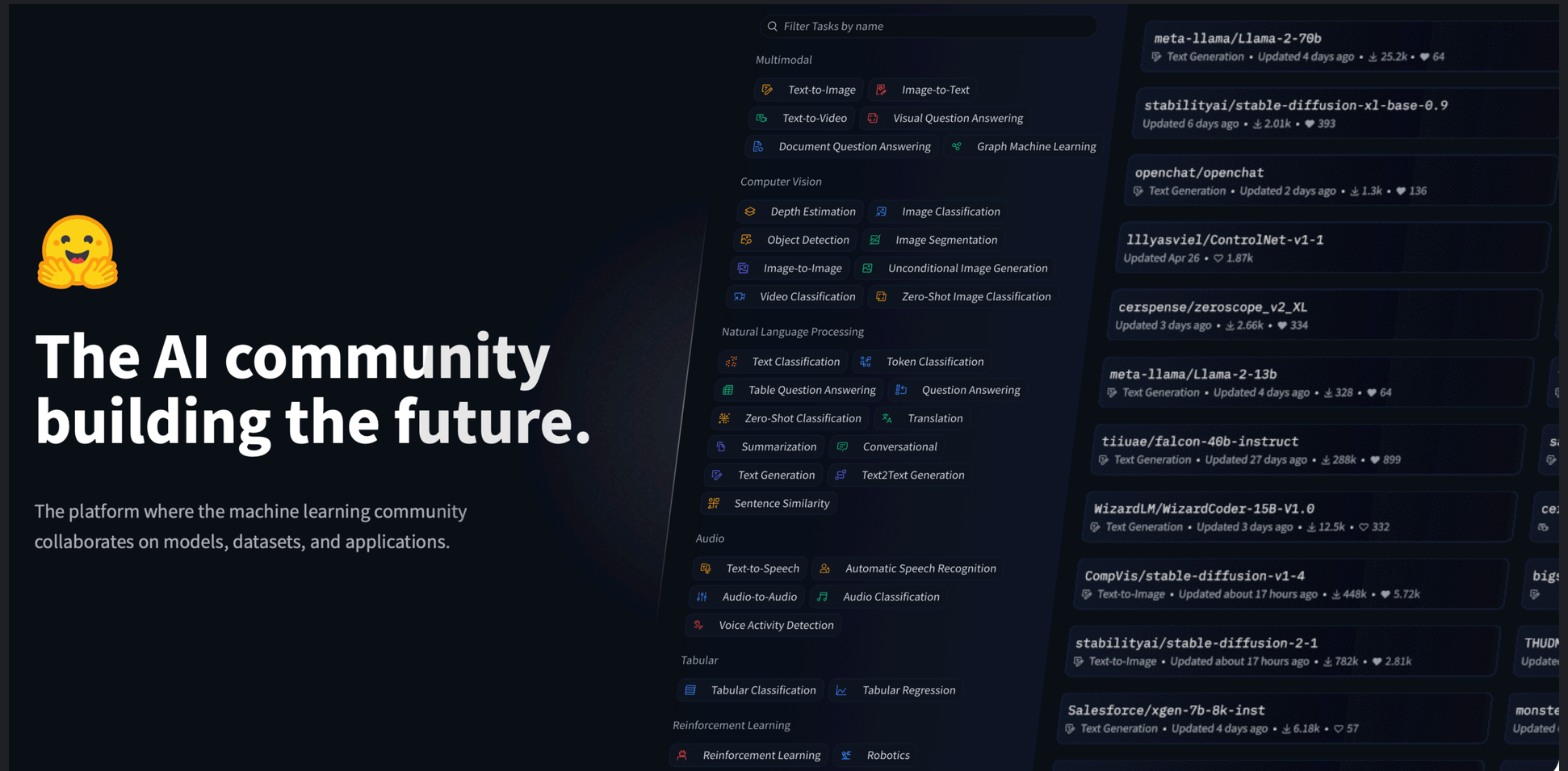
macOS	iPadOS, iOS
14.0	17.0

Target Device Hardware Generation:

Mac	iPad	iPhone
M1	M1	A14

How to Train Your Own Custom Model

Pre-trained Models



How to Train Your Own Custom Model

DreamBooth

DreamBooth is a method to personalize text-to-image models like Stable Diffusion given just a few (3-5) images of a subject. It allows the model to generate contextualized images of the subject in different scenes, poses, and views.

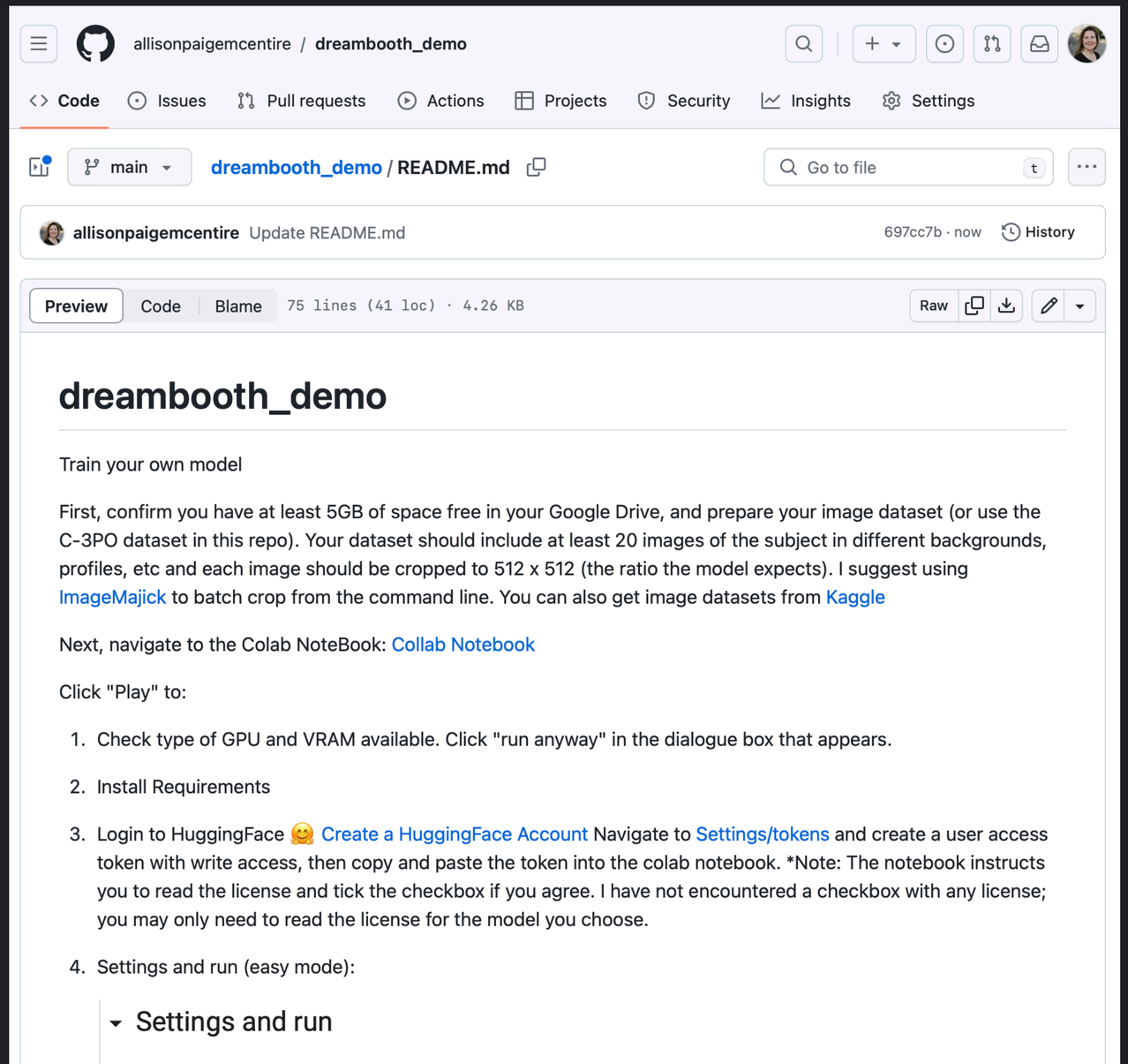
Dreambooth examples from the project's blog.

This guide will show you how to finetune DreamBooth with the `CompVis/stable-diffusion-v1-4` model for various GPU sizes, and with Flax. All the training scripts for DreamBooth used in this guide can be found here if you're interested in digging deeper and seeing how things work.

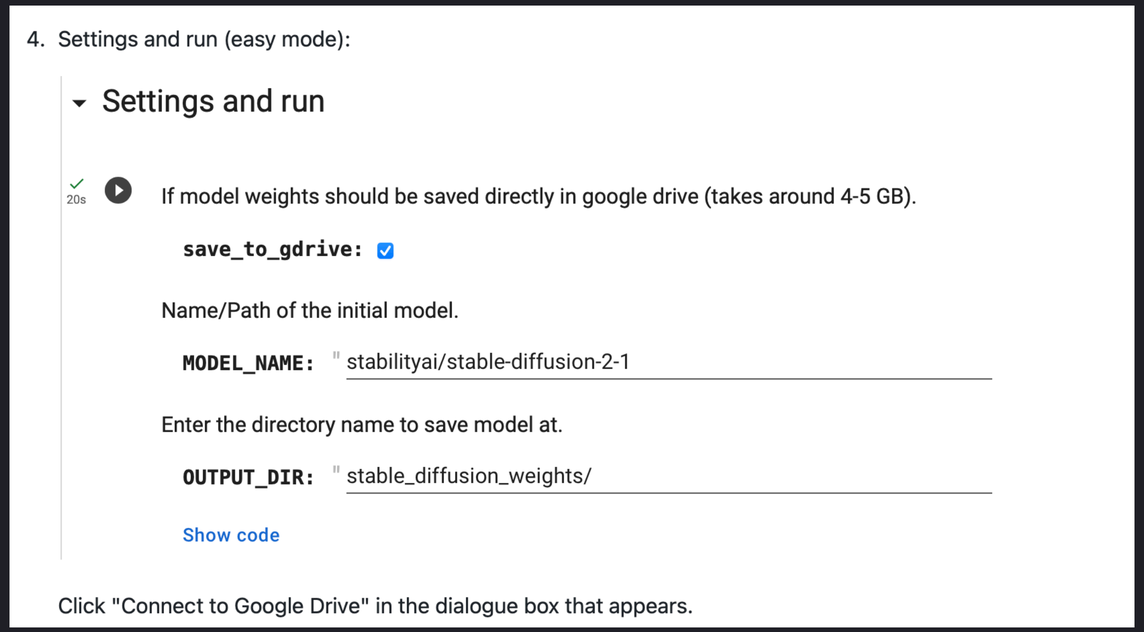
Before running the scripts, make sure you install the library's training dependencies. We also recommend installing 🧨 Diffusers from the `main` GitHub branch:

```
pip install git+https://github.com/huggingface/diffusers
pip install -U -r diffusers/examples/dreambooth/requirements.txt
```

How to Train Your Own Custom Model



How to Train Your Own Custom Model



How to Train Your Own Custom Model

5. Start Training: Edit the concepts list. Choose a unique identifier for the instance prompt, e.g. "robotMaria" and a generic identifier for the class prompt ("robot)". The unique identifier you choose will be the prompt you use on your final model.

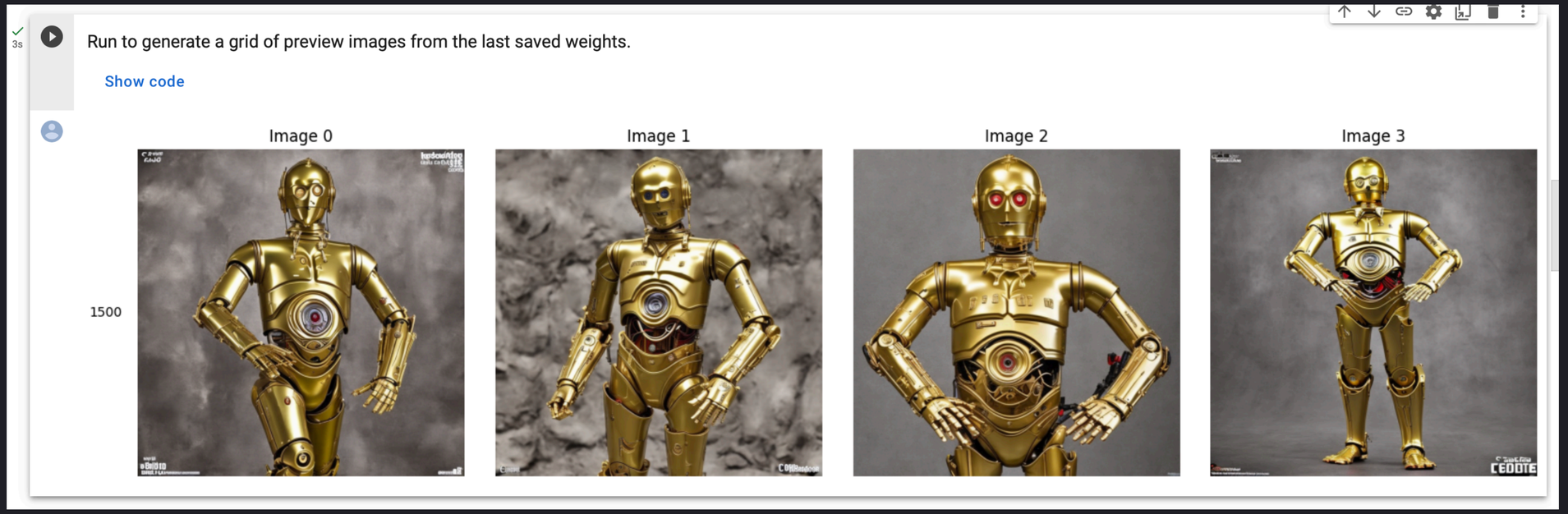
```
# You can also add multiple concepts here. Try tweaking `--max_train_steps` accordingly.

concepts_list = [
    {
        "instance_prompt":      "photo of robotC-3PO robot",
        "class_prompt":         "photo of a robot",
        "instance_data_dir":    "/content/data/robotC-3PO",
        "class_data_dir":       "/content/data/robot"
    },
#     {
#         "instance_prompt":      "photo of ukj person",
#         "class_prompt":         "photo of a person",
#         "instance_data_dir":    "/content/data/ukj",
#         "class_data_dir":       "/content/data/person"
#     }
]
```

How to Train Your Own Custom Model

7. Edit the python command: Seed = changing this number will give you slightly different results if you're unhappy with the output of the model after training Max train steps (iterations the model makes on each images in training) = the number of images you uploaded x 100 Save interval = the number of images you uploaded x 100 Sample prompt = the string used for "instance prompt" in step 6. Click play, then wait 20-30 minutes for training to complete. Keep your browser active to avoid complications/timing out.

```
!python3 train_dreambooth.py \
  --pretrained_model_name_or_path=$MODEL_NAME \
  --pretrained_vae_name_or_path="stabilityai/sd-vae-ft-mse" \
  --output_dir=$OUTPUT_DIR \
  --revision="fp16" \
  --with_prior_preservation --prior_loss_weight=1.0 \
  --seed=1337 \
  --resolution=512 \
  --train_batch_size=1 \
  --train_text_encoder \
  --mixed_precision="fp16" \
  --use_8bit_adam \
  --gradient_accumulation_steps=1 \
  --learning_rate=1e-6 \
  --lr_scheduler="constant" \
  --lr_warmup_steps=0 \
  --num_class_images=50 \
  --sample_batch_size=4 \
  --max_train_steps=1500 \
  --save_interval=1500 \
  --save_sample_prompt="photo of robotC-3PO robot" \
  --concepts_list="concepts_list.json"
```

How to Train Your Own Custom Model



How to Train Your Own Custom Model

Run for generating images.

prompt: " painting of robotMaria robot in the style of Van Gogh

negative_prompt: " Insert text here

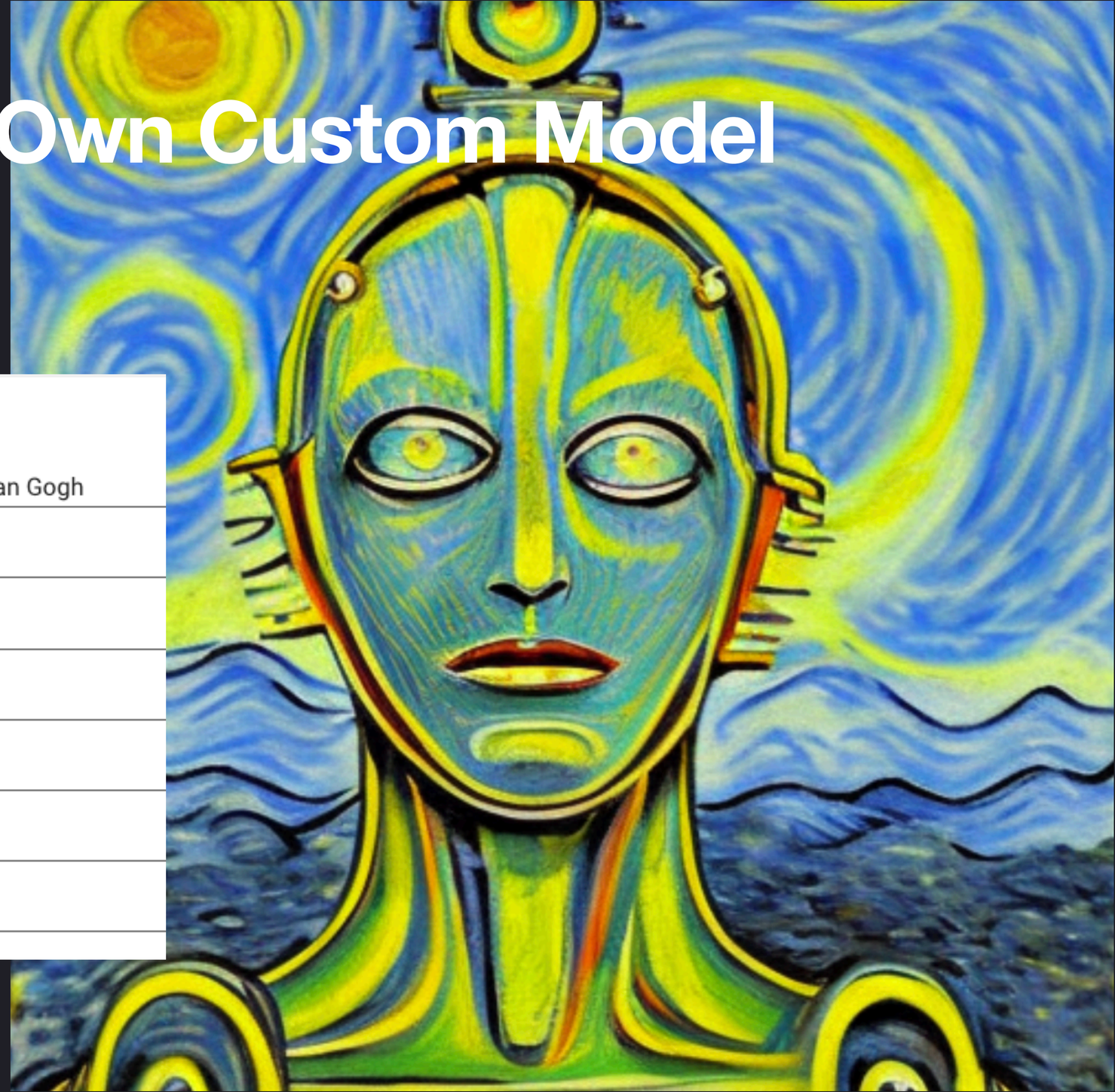
num_samples: 4

guidance_scale: 7.5

num_inference_steps: 24

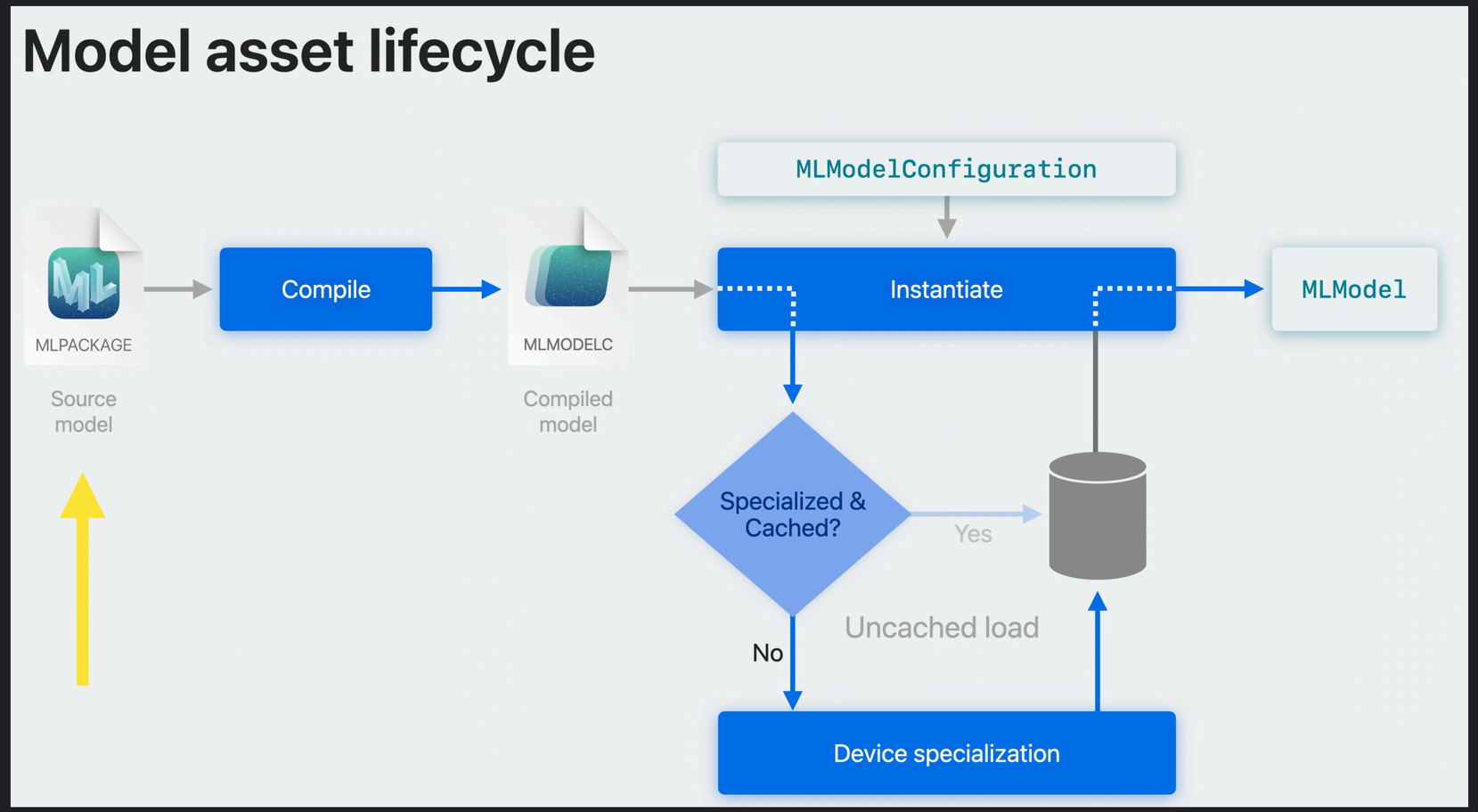
height: 512

width: 512



Optimizing Your Model

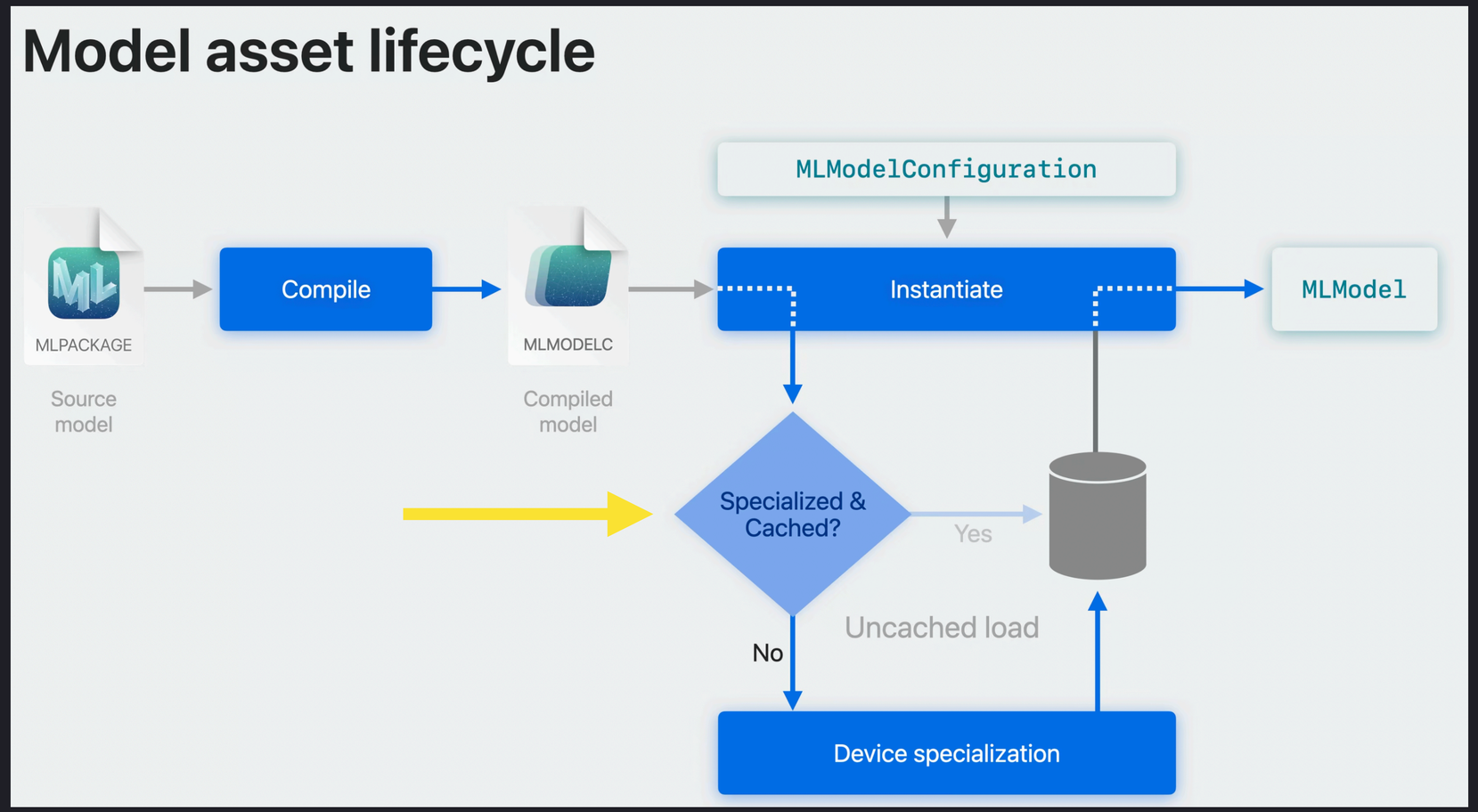
CoreML model Lifecycle



https://developer.apple.com/videos/play/wwdc2023/10049/

Optimizing Your Model

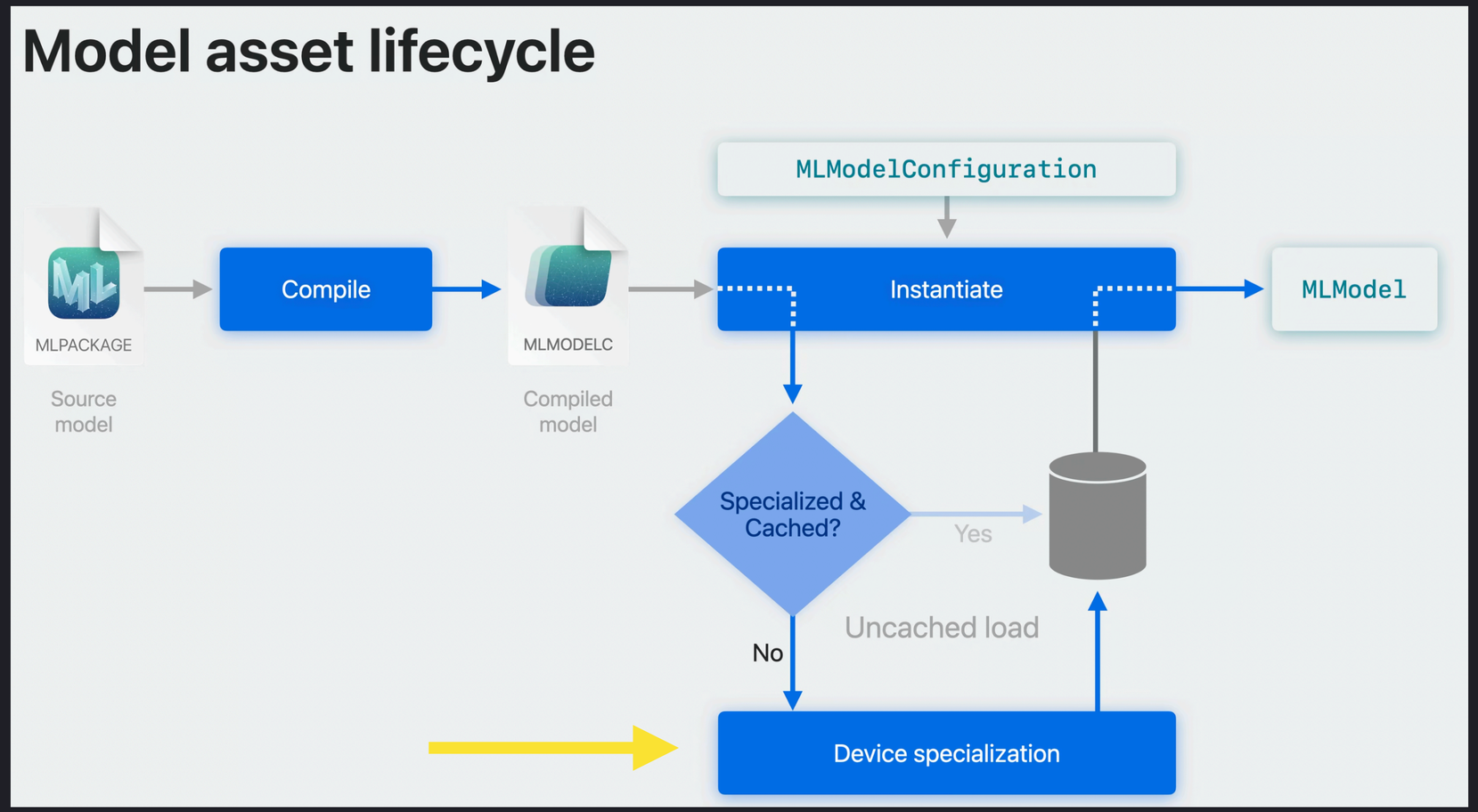
CoreML model Lifecycle



https://developer.apple.com/videos/play/wwdc2023/10049/

Optimizing Your Model

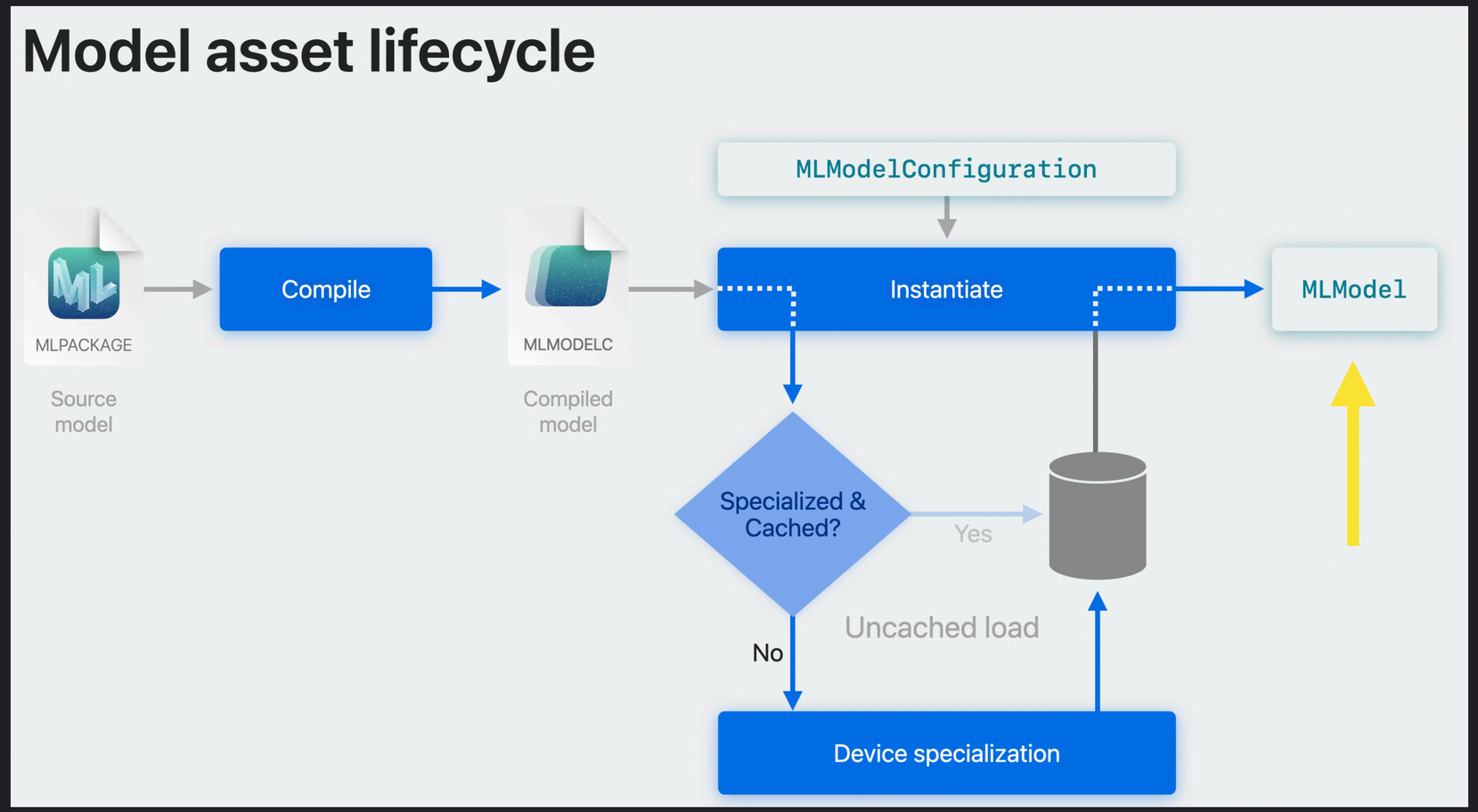
CoreML model Lifecycle



https://developer.apple.com/videos/play/wwdc2023/10049/

Optimizing Your Model

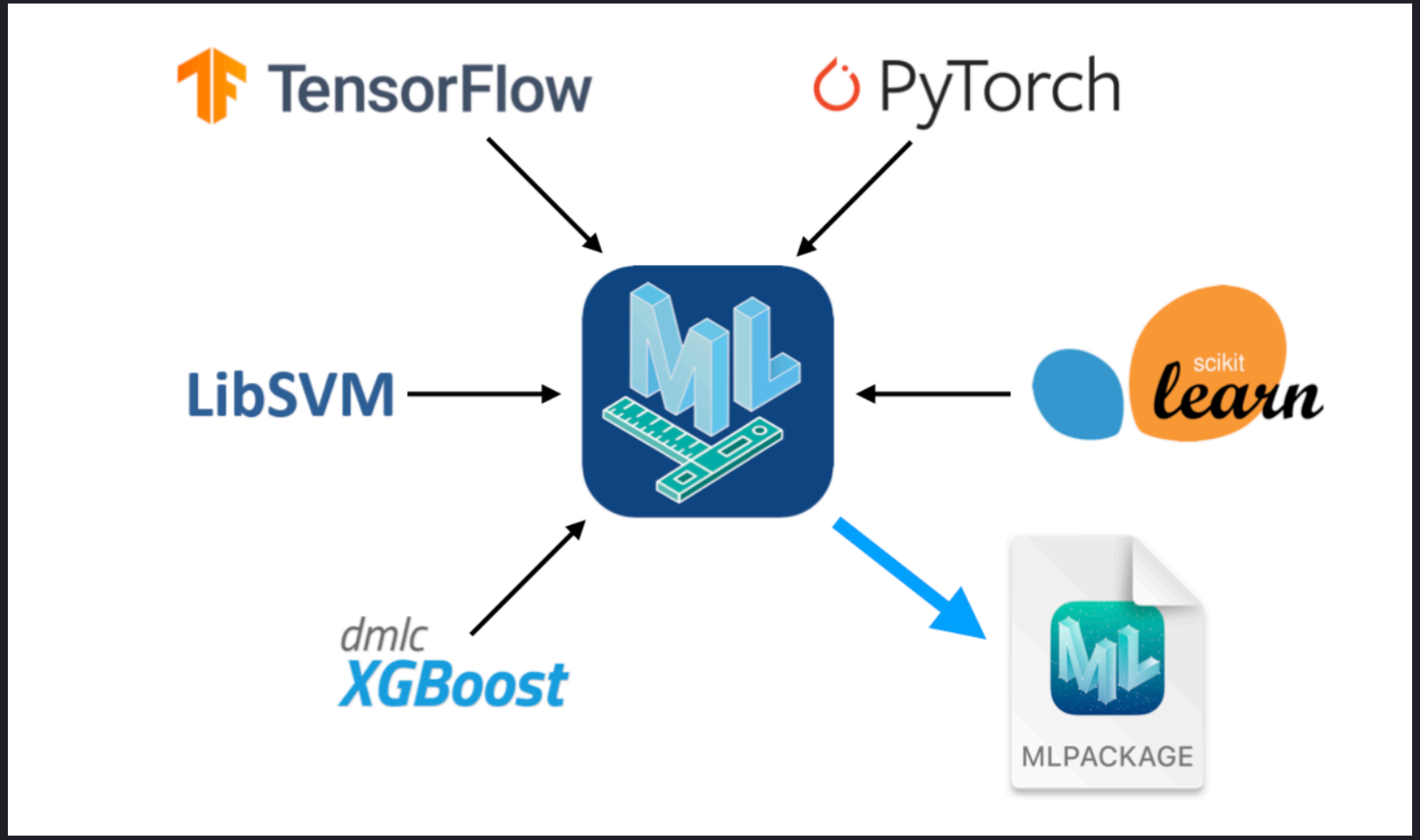
CoreML model Lifecycle



https://developer.apple.com/videos/play/wwdc2023/10049/

Optimizing Your Model

CoreMLTools



Optimizing Your Model

CoreMLTools

Why Reduce Model Size?

- Smaller models mean you can include more models within the same memory limit.

Optimizing Your Model

CoreMLTools

Why Reduce Model Size?

- Smaller models mean you can include more models within the same memory limit.
- Even larger, powerful models can be included if they're reduced in size.

Optimizing Your Model

CoreMLTools

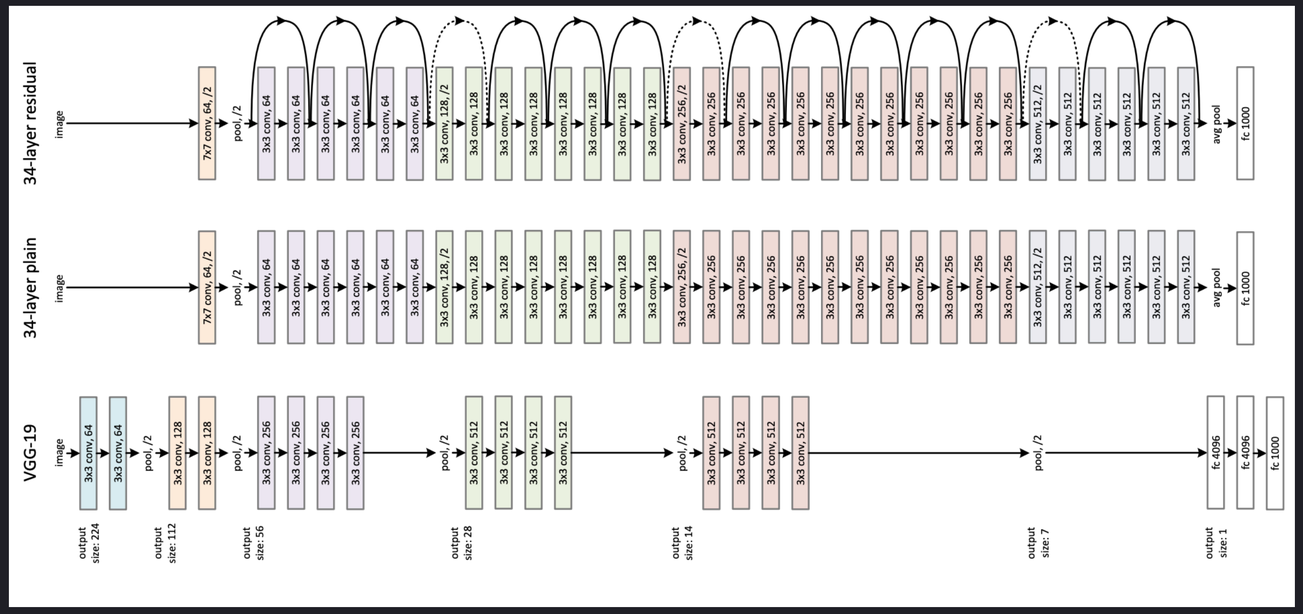
Why Reduce Model Size?

- Smaller models mean you can include more models within the same memory limit.
- Even larger, powerful models can be included if they're reduced in size.
- Speed: Smaller models usually run faster because there's less data to shuttle back and forth.

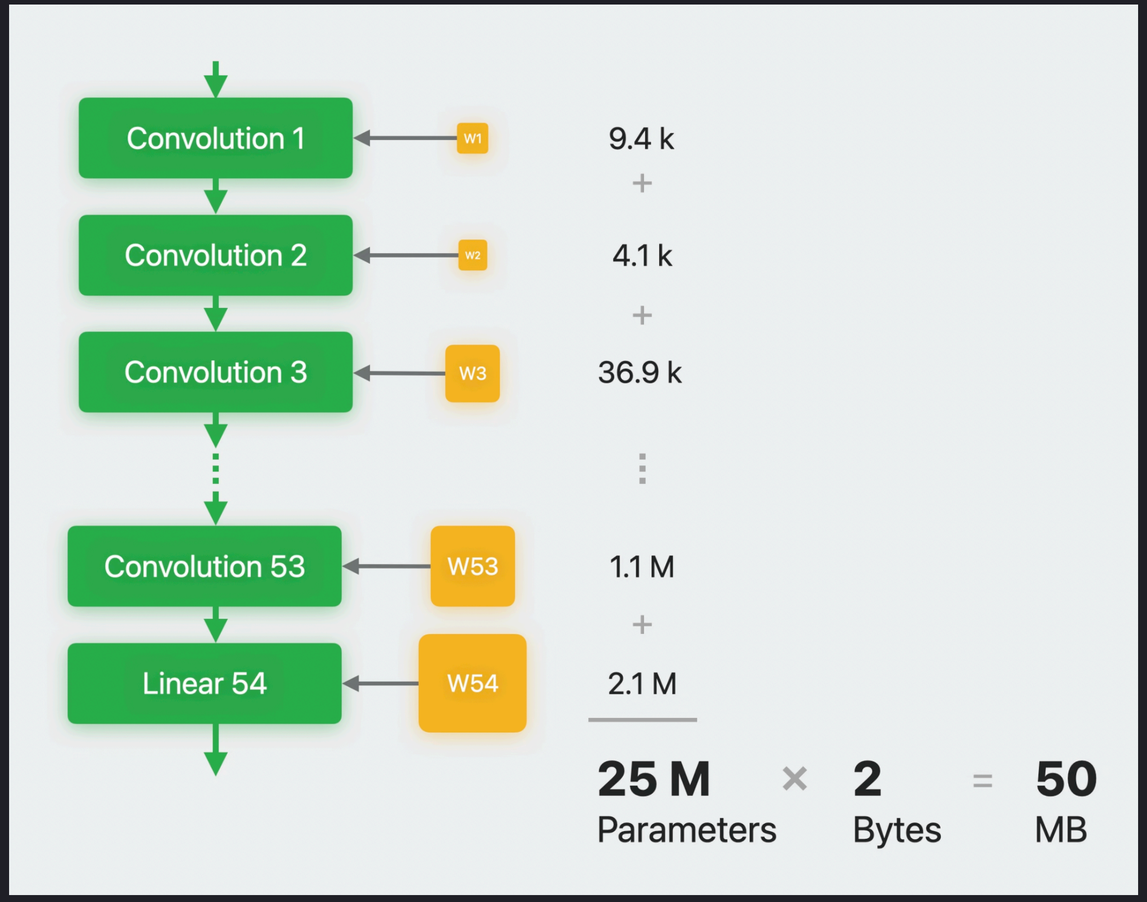
Optimizing Your Model

CoreMLTools

What Makes a Model Large?



https://huggingface.co/microsoft/resnet-50



https://developer.apple.com/videos/play/wwdc2023/10047/

Optimizing Your Model

CoreMLTools

Paths to a Smaller Model:

- **Design a More Efficient Model**

Optimizing Your Model

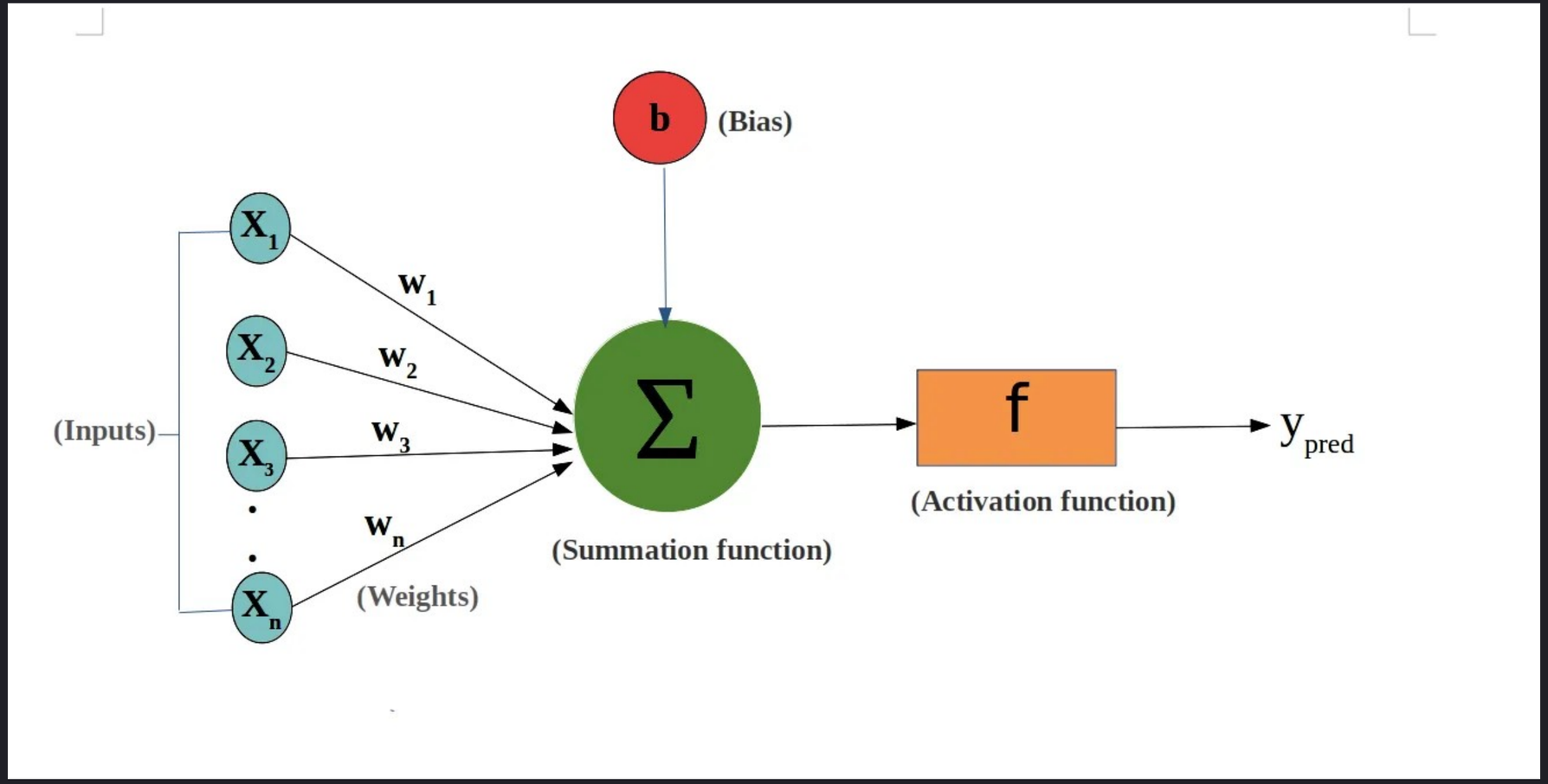
CoreMLTools

Paths to a Smaller Model:

- **Design a More Efficient Model**
- **Compress the Existing Model**

Optimizing Your Model

CoreMLTools



https://towardsdatascience.com/whats-the-role-of-weights-and-bias-in-a-neural-network-4cf7e9888a0f

Optimizing Your Model

CoreMLTools

Compression technique summary

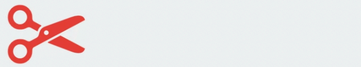
	Pruning	Quantization	Palettization
Representation	Non-zero values + zero indices	`Int8` weights + Float scale + Bias	2,4,6 or 8 bit per weight value
Compression factor (w.r.t. `Float16`)	~1x-10x	2x	2x, 2.67x, 4x, 8x
Parameter	% sparsity (S)	Scale only / Scale + Bias	Number of clusters

https://developer.apple.com/videos/play/wwdc2023/10047/

Optimizing Your Model

CoreMLTools

Compression technique summary



	Pruning	→ Quantization	Palettization
Representation	Non-zero values + zero indices	`Int8` weights + Float scale + Bias	2,4,6 or 8 bit per weight value
Compression factor (w.r.t. `Float16`)	~1x-10x	2x	2x, 2.67x, 4x, 8x
Parameter	% sparsity (S)	Scale only / Scale + Bias	Number of clusters

https://developer.apple.com/videos/play/wwdc2023/10047/

Optimizing Your Model

CoreMLTools

Compression technique summary

Pruning

Quantization

Palettization

	Pruning	Quantization	Palettization
Representation	Non-zero values + zero indices	`Int8` weights + Float scale + Bias	2,4,6 or 8 bit per weight value
Compression factor (w.r.t. `Float16`)	~1x-10x	2x	2x, 2.67x, 4x, 8x
Parameter	% sparsity (S)	Scale only / Scale + Bias	Number of clusters

https://developer.apple.com/videos/play/wwdc2023/10047/

Optimizing Your Model

CoreMLTools

https://developer.apple.com/wwdc23/10047

Optimizing Your Model

CoreMLTools

Core ML Tools 7

NEW

	Post-training compression	Training time compression
Sub-module	`coremltools.optimize.coreml`	`coremltools.optimize.torch`
Pruning	`.prune_weights`	`.pruning.MagnitudePruner`
Quantization	`.linear_quantize_weights`	`.quantization.LinearQuantizer`
Palettization	`.palettize_weights`	`.palettization.DKMPalettizer`

https://developer.apple.com/videos/play/wwdc2023/10047/

Optimizing Your Model

CoreMLTools

Core ML Tools 7

NEW

	Post-training compression	Training time compression
Sub-module	`coremltools.optimize.coreml`	`coremltools.optimize.torch`
Pruning	`.prune_weights`	`.pruning.MagnitudePruner`
Quantization	`.linear_quantize_weights`	`.quantization.LinearQuantizer`
Palettization	`.palettize_weights`	`.palettization.DKMPalettizer`

https://developer.apple.com/videos/play/wwdc2023/10047/

Optimizing Your Model

CoreMLTools

Core ML Tools 7

NEW

	Post-training compression	Training time compression
Sub-module	`coremltools.optimize.coreml`	`coremltools.optimize.torch`
Pruning	`.prune_weights`	`.pruning.MagnitudePruner`
Quantization	`.linear_quantize_weights`	`.quantization.LinearQuantizer`
Palettization	`.palettize_weights`	`.palettization.DKMPalettizer`

https://developer.apple.com/videos/play/wwdc2023/10047/

Optimizing Your Model

CoreMLTools

CoreMLTools 7.0 beta
Xcode 15.0 beta
apple/ml-stable-diffusion repo

Converting Models to Core ML

▼ Click to expand

Step 1: Create a Python environment and install dependencies:

```
conda create -n coreml_stable_diffusion python=3.8 -y
conda activate coreml_stable_diffusion
cd /path/to/cloned/ml-stable-diffusion/repository
pip install -e .
```

Step 2: Log in to or register for your Hugging Face account, generate a User Access Token and set up Hugging Face API access by running `huggingface-cli login` in a Terminal window.

Step 3: Navigate to the version of Stable Diffusion that you would like to use on Hugging Face H Terms of Use. The default model version is CompVis/stable-diffusion-v1-4. The model version r the user as described in the next step.

Step 4: Execute the following command from the Terminal to generate Core ML model files (`.m`

```
python -m python_coreml_stable_diffusion.torch2coreml --convert-unet --convert-t
```

WARNING: This command will download several GB worth of PyTorch checkpoints from Huggin ensure that you are on Wi-Fi and have enough disk space.

This generally takes 15-20 minutes on an M1 MacBook Pro. Upon successful execution, the 4 n models that comprise Stable Diffusion will have been converted from PyTorch to Core ML (`.mlp` saved into the specified `<output-mlpackages-directory>`. Some additional notable argument

- `--model-version`: The model version name as published on the Hugging Face Hub
- `--bundle-resources-for-swift-cli`: Compiles all 4 models and bundles them along with resources for text tokenization into `<output-mlpackages-directory>/Resources` which sh input to the Swift package. This flag is not necessary for the diffusers-based Python pipelin
- `--quantize-nbits`: Quantizes the weights of unet and text_encoder models down to 2, 4, globally optimal k-means clustering algorithm. By default all models are weight-quantized t

Optimizing Your Model

CoreMLTools

```
python -m python_coreml_stable_diffusion.torch2coreml \
    --model-version prompthero/openjourney-v4 \
    --convert-unet \
    --convert-text-encoder \
    --convert-vae-decoder \
    --convert-safety-checker \
    --quantize-nbits 6 \
    --attention-implementation SPLIT_EINSUM_V2 \
    --compute-unit ALL \
    --bundle-resources-for-swift-cli \
    --check-output-correctness \
    -o models/split_einsum_v2/openjourney-6-bit
```

Optimizing Your Model

CoreMLTools

```
python -m python_coreml_stable_diffusion.torch2coreml \
    --model-version prompthero/openjourney-v4 \
    --convert-unet \
    --convert-text-encoder \
    --convert-vae-decoder \
    --convert-vae-encoder \
    --quantize-nbits 6 \
    --attention-implementation SPLIT_EINSUM_V2 \
    --compute-unit ALL \
    --bundle-resources-for-swift-cli \
    --check-output-correctness \
    -o models/split_einsum_v2/openjourney-6-bit
```

Optimizing Your Model

CoreMLTools

```
python -m python_coreml_stable_diffusion.torch2coreml \
    --model-version prompthero/openjourney-v4 \
    --convert-unet \
    --convert-text-encoder \
    --convert-vae-decoder \
    --convert-vae-encoder \
    --quantize-nbits 6 \
    --attention-implementation SPLIT_EINSUM_V2 \
    --compute-unit ALL \
    --bundle-resources-for-swift-cli \
    --check-output-correctness \
    -o models/split_einsum_v2/openjourney-6-bit
```

Optimizing Your Model

CoreMLTools

```
python -m python_coreml_stable_diffusion.torch2coreml \
    --model-version prompthero/openjourney-v4 \
    --convert-unet \
    --convert-text-encoder \
    --convert-vae-decoder \
    --convert-vae-encoder \
    --quantize-nbits 6 \
    --attention-implementation SPLIT_EINSUM_V2 \
    --compute-unit ALL \
    --bundle-resources-for-swift-cli \
    --check-output-correctness \
    -o models/split_einsum_v2/openjourney-6-bit
```

Optimizing Your Model

CoreMLTools

```
python -m python_coreml_stable_diffusion.torch2coreml \
    --model-version prompthero/openjourney-v4 \
    --convert-unet \
    --convert-text-encoder \
    --convert-vae-decoder \
    --convert-vae-encoder \
    --quantize-nbits 6 \
    --attention-implementation SPLIT_EINSUM_V2 \
    --compute-unit ALL \
    --bundle-resources-for-swift-cli \
    --check-output-correctness \
    -o models/split_einsum_v2/openjourney-6-bit
```

Optimizing Your Model

CoreMLTools

In order to make it easy for everyone to take advantage of these improvements, we have converted the four official Stable Diffusion models and pushed them to the Hub. These are all the variants:

Model	Uncompressed	Palettized
Stable Diffusion 1.4	Core ML, `float16`	Core ML, 6-bit palettized
Stable Diffusion 1.5	Core ML, `float16`	Core ML, 6-bit palettized
Stable Diffusion 2 base	Core ML, `float16`	Core ML, 6-bit palettized
Stable Diffusion 2.1 base	Core ML, `float16`	Core ML, 6-bit palettized

In order to use 6-bit models, you need the development versions of iOS/iPadOS 17 or macOS 14 (Sonoma) because those are the ones that contain the latest Core ML framework. You can download them from the Apple developer site if you are a registered developer, or you can sign up for the public beta that will be released in a few weeks.

- https://huggingface.co/blog/fast-diffusers-coreml

Hugging Face

Search models, datasets, users...

AllisonMcEntire / **robotMaria**

like 0

License: mit

Model card | Files and versions | Community | Settings

Dreambooth'd model trained to generate images of the robot from Metropolis with the prompt "robotMaria robot"

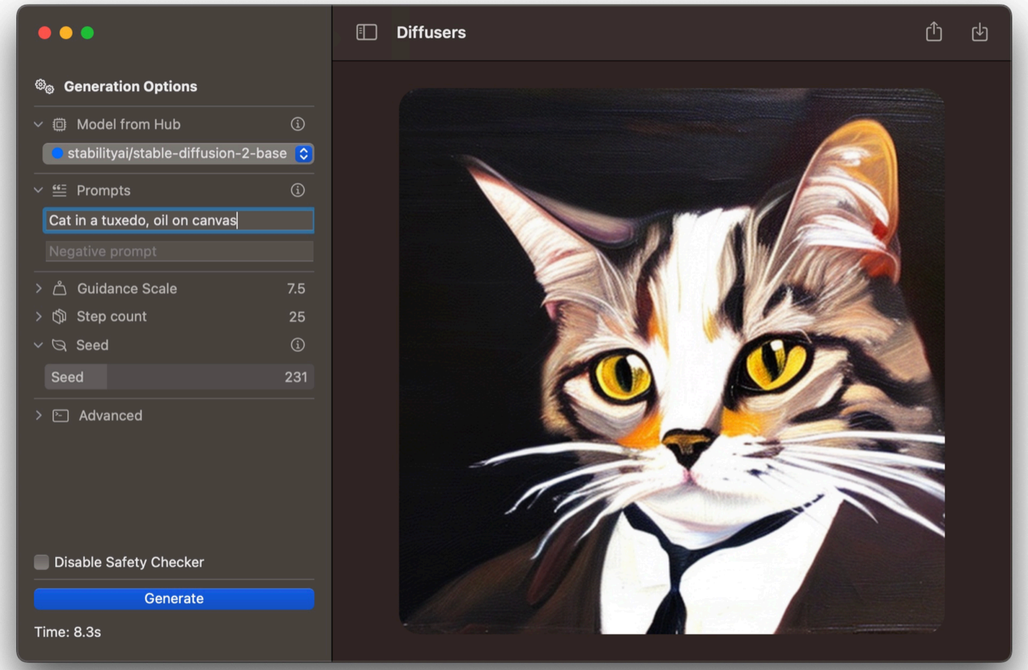
How to Run Stable Diffusion on a Mac

MacOS Ventura 14, iOS/iPadOS 17

Swift Core ML Diffusers 🧨

This is a native app that shows how to integrate Apple's Core ML Stable Diffusion implementation in a native Swift UI application. The Core ML port is a simplification of the Stable Diffusion implementation from the diffusers library. This application can be used for faster iteration, or as sample code for any use cases.

This is what the app looks like on macOS:



How to Run Stable Diffusion on an iPhone

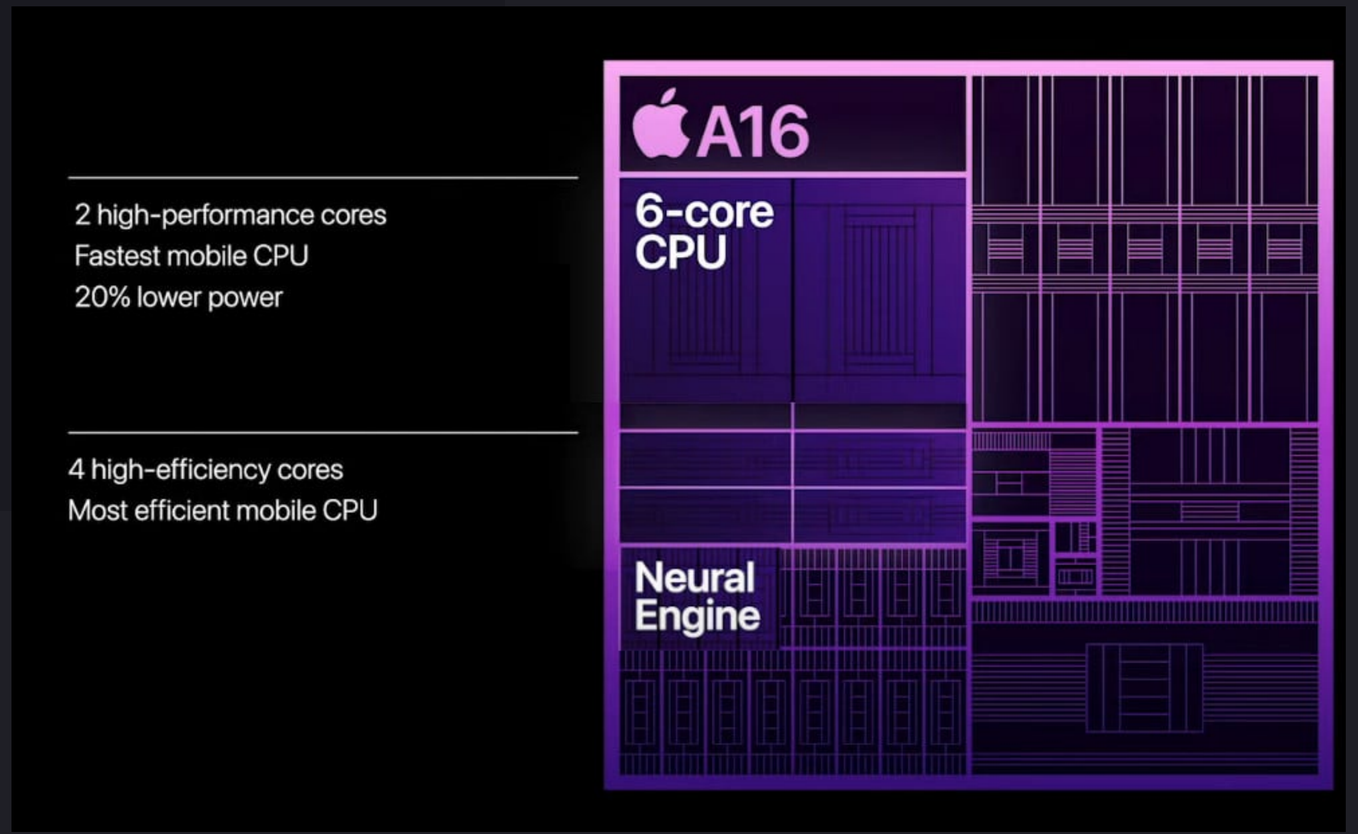
Demo Code Walkthrough

```
import SwiftUI

@main
struct DiffusionApp: App {
    var body: some Scene {
        WindowGroup {
            LoadingView()
        }
    }
}

// Different devices report different amounts, so approximate
let deviceHas6GBOrMore = ProcessInfo.processInfo.physicalMemory > 5924000000

let deviceSupportsQuantization = {
    if #available(iOS 17, *) {
        true
    } else {
        false
    }
}()
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
@available(iOS 12.0, *)
public enum MLComputeUnits : Int, @unchecked Sendable {


    case cpuOnly = 0

    case cpuAndGPU = 1

    case all = 2

    @available(iOS 16.0, *)
    case cpuAndNeuralEngine = 3
}
```



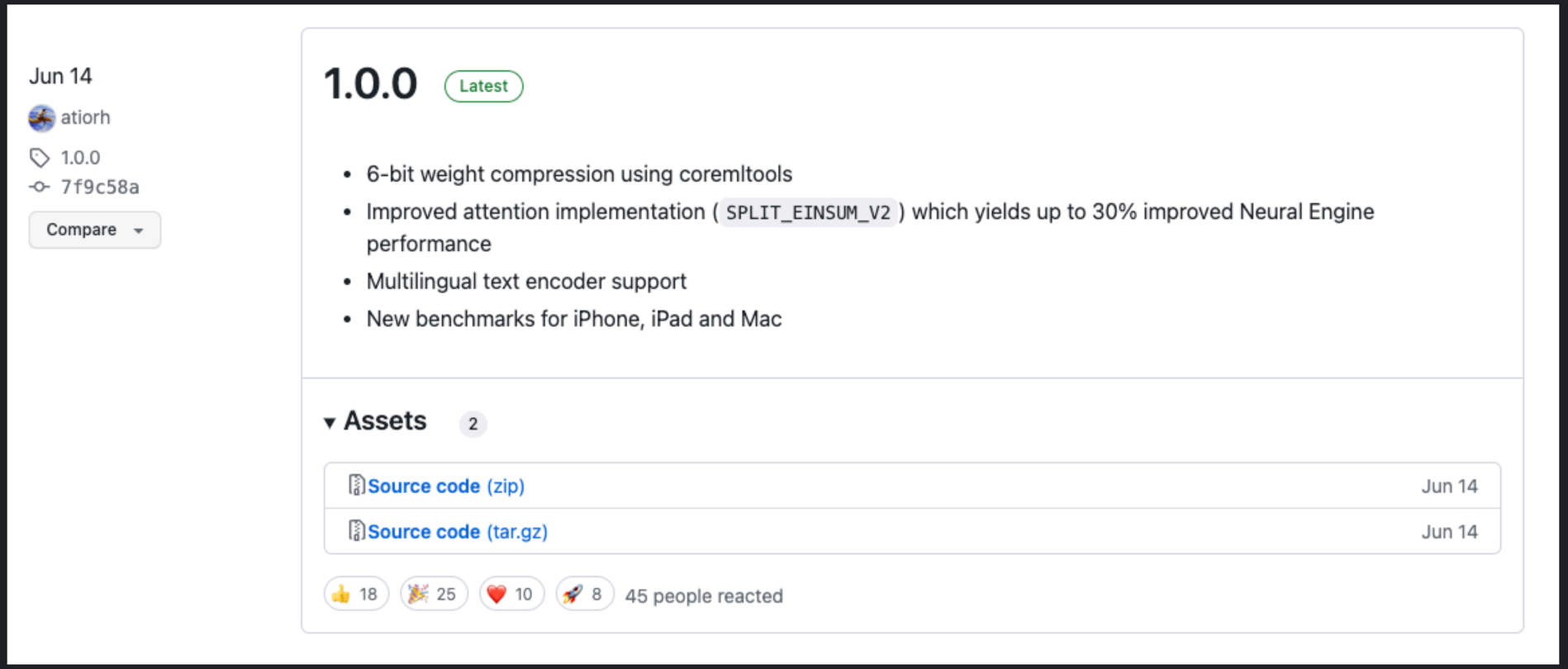
How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```swift
import CoreML

enum AttentionVariant: String {
    case original
    case splitEinsum
    case splitEinsumV2
}

extension AttentionVariant {
    var defaultComputeUnits: MLComputeUnits { self == .original ? .cpuAndGPU : .cpuAndNeuralEngine }
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough



https://github.com/apple/ml-stable-diffusion

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func modelURL(for variant: AttentionVariant) -> URL {
    // Pattern:
        https://huggingface.co/pcuenq/coreml-stable-diffusion/resolve/main/coreml-stable-diffusion-v1-5_original_compiled.zip
    let suffix: String
    switch variant {
    case .original: suffix = originalAttentionSuffix
    case .splitEinsum: suffix = splitAttentionSuffix
    case .splitEinsumV2: suffix = splitAttentionV2Suffix
    }
    let repo = modelId.split(separator: "/").last!
    return URL(string: "https://huggingface.co/\(modelId)/resolve/main/\(repo)_\(suffix)")!
}
```

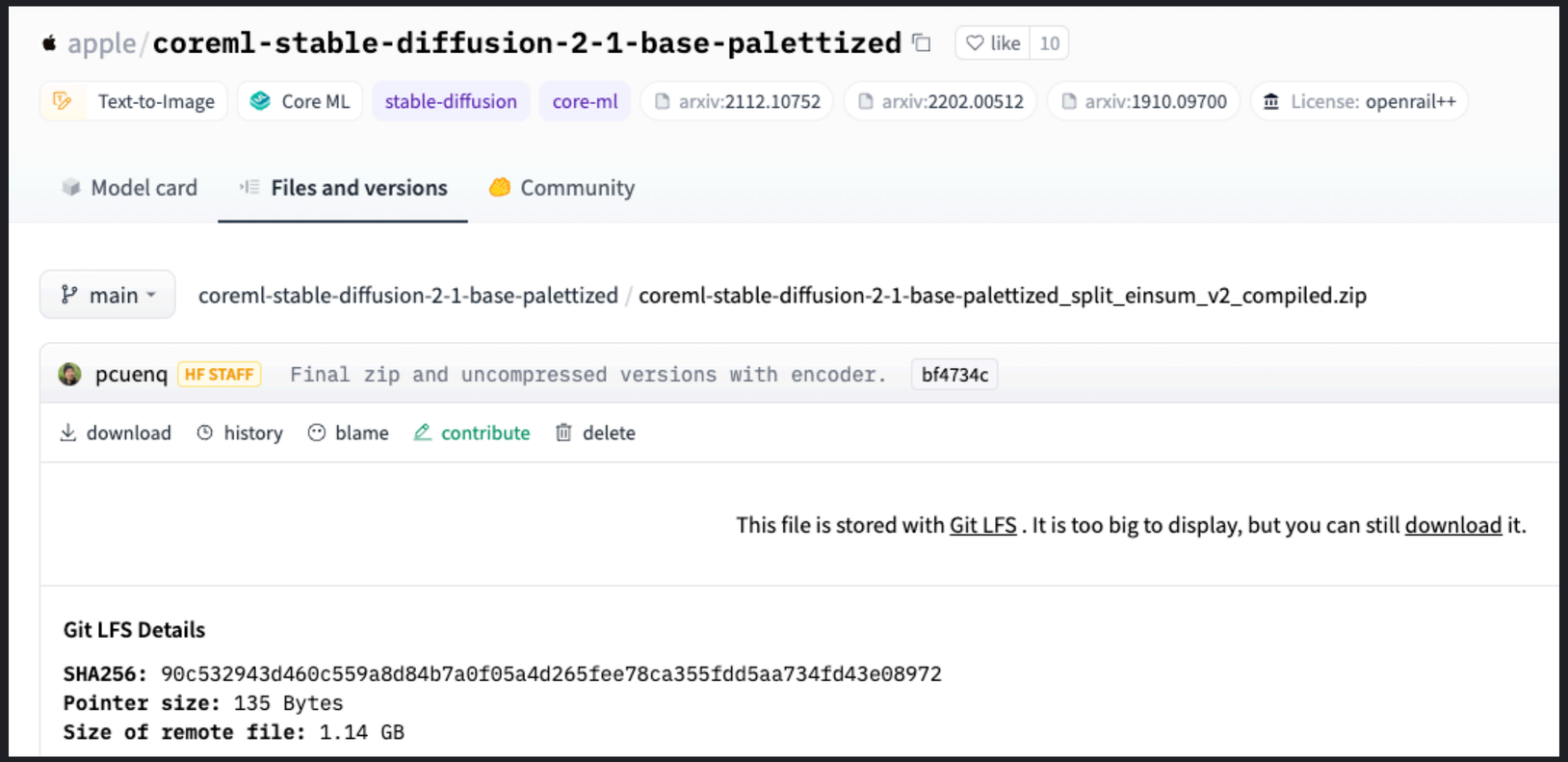
How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
static let v21Palettized = ModelInfo(
    modelId: "apple/coreml-stable-diffusion-2-1-base-palettized",
    modelVersion: "StabilityAI SD 2.1 [6 bit]",
    supportsEncoder: true,
    supportsAttentionV2: true,
    quantized: true
)
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough



How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func prepare() async throws -> Pipeline {
    do {
        do {
            try FileManager.default.createDirectory(atPath: PipelineLoader.models.path,
                withIntermediateDirectories: true, attributes: nil)
        } catch {
            print("Error creating PipelineLoader.models path: \(error)")
        }

        try await download()
        try await unzip()
        let pipeline = try await load(url: compiledURL)
        return Pipeline(pipeline, maxSeed: maxSeed)
    } catch {
        state = .failed(error)
        print("trm error", error)
        throw error
    }
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func prepare() async throws -> Pipeline {
    do {
        do {
            try FileManager.default.createDirectory(atPath: PipelineLoader.models.path,
                withIntermediateDirectories: true, attributes: nil)
        } catch {
            print("Error creating PipelineLoader.models path: \(error)")
        }

        try await download()
        try await unzip()
        let pipeline = try await load(url: compiledURL)
        return Pipeline(pipeline, maxSeed: maxSeed)
    } catch {
        state = .failed(error)
        print("trm error", error)
        throw error
    }
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func setInitialState() {
    if ready {
        state = .readyOnDisk
        return
    }
    if downloaded {
        state = .downloaded
        return
    }
    state = .waitingToDownload
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func load(url: URL) async throws -> StableDiffusionPipelineProtocol {
    let beginDate = Date()
    let configuration = MLModelConfiguration()
    configuration.computeUnits = computeUnits
    let pipeline: StableDiffusionPipelineProtocol
    if model.isXL {
        if #available(macOS 14.0, iOS 17.0, *) {
            pipeline = try StableDiffusionXLPipeline(resourcesAt: url,
                                                     configuration: configuration,
                                                     reduceMemory: model.reduceMemory)
        } else {
            throw "Stable Diffusion XL requires macOS 14"
        }
    } else {
        pipeline = try StableDiffusionPipeline(resourcesAt: url,
                                               controlNet: [],
                                               configuration: configuration,
                                               disableSafety: false,
                                               reduceMemory: model.reduceMemory)
    }
    try pipeline.loadResources()
    print("Pipeline loaded in \(Date().timeIntervalSince(beginDate))")
    state = .loaded
    return pipeline
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func generate() async throws -> GenerationResult {
    guard let pipeline = pipeline else { throw "No pipeline" }
    return try pipeline.generate(
        prompt: positivePrompt,
        negativePrompt: negativePrompt,
        scheduler: scheduler,
        numInferenceSteps: Int(steps),
        seed: seed,
        numPreviews: Int(previews),
        guidanceScale: Float(guidanceScale),
        disableSafety: disableSafety
    )
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func generate(
    prompt: String,
    negativePrompt: String = "",
    scheduler: StableDiffusionScheduler,
    numInferenceSteps stepCount: Int = 50,
    seed: UInt32 = 0,
    numPreviews previewCount: Int = 5,
    guidanceScale: Float = 7.5,
    disableSafety: Bool = false
) throws -> GenerationResult {
    let beginDate = Date()
    canceled = false
    let theSeed = seed > 0 ? seed : UInt32.random(in: 1...maxSeed)
    let sampleTimer = SampleTimer()
    sampleTimer.start()

    var config = StableDiffusionPipeline.Configuration(prompt: prompt)
    config.negativePrompt = negativePrompt
    config.stepCount = stepCount
    config.seed = theSeed
    config.guidanceScale = guidanceScale
    config.disableSafety = disableSafety
    config.schedulerType = scheduler.asStableDiffusionScheduler()
    config.useDenoisedIntermediates = true
    if isXL {
        config.encoderScaleFactor = 0.13025
        config.decoderScaleFactor = 0.13025
    }
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
var config = StableDiffusionPipeline.Configuration(prompt: prompt)
config.negativePrompt = negativePrompt
config.stepCount = stepCount
config.seed = theSeed
config.guidanceScale = guidanceScale
config.disableSafety = disableSafety
config.schedulerType = scheduler.asStableDiffusionScheduler()
config.useDenoisedIntermediates = true
if isXL {
    config.encoderScaleFactor = 0.13025
    config.decoderScaleFactor = 0.13025
}
```

```
/// Schedulers compatible with StableDiffusionPipeline. This is a local implementation of the StableDiffusionScheduler enum as a String represetation to allow for compliance with NSSecureCoding.
public enum StableDiffusionScheduler: String {
    /// Scheduler that uses a pseudo-linear multi-step (PLMS) method
    case pndmScheduler
    /// Scheduler that uses a second order DPM-Solver++ algorithm
    case dpmSolverMultistepScheduler
```

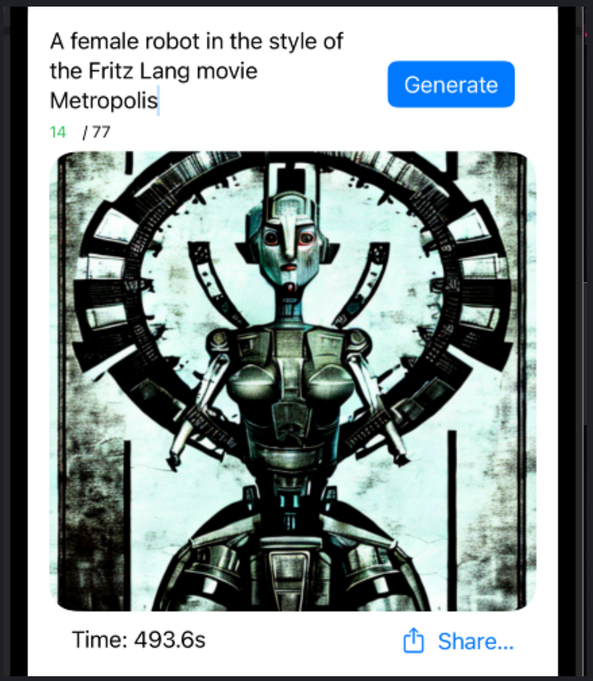
How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
var config = StableDiffusionPipeline.Configuration(prompt: prompt)
config.negativePrompt = negativePrompt
config.stepCount = stepCount
config.seed = UInt32(2347)
config.rngType = .torchRNG
config.guidanceScale = guidanceScale
config.disableSafety = disableSafety
config.schedulerType = scheduler.asStableDiffusionScheduler()
config.useDenoisedIntermediates = true
if isXL {
    config.encoderScaleFactor = 0.13025
    config.decoderScaleFactor = 0.13025
}
```

How to Run Stable Diffusion on an iPhone

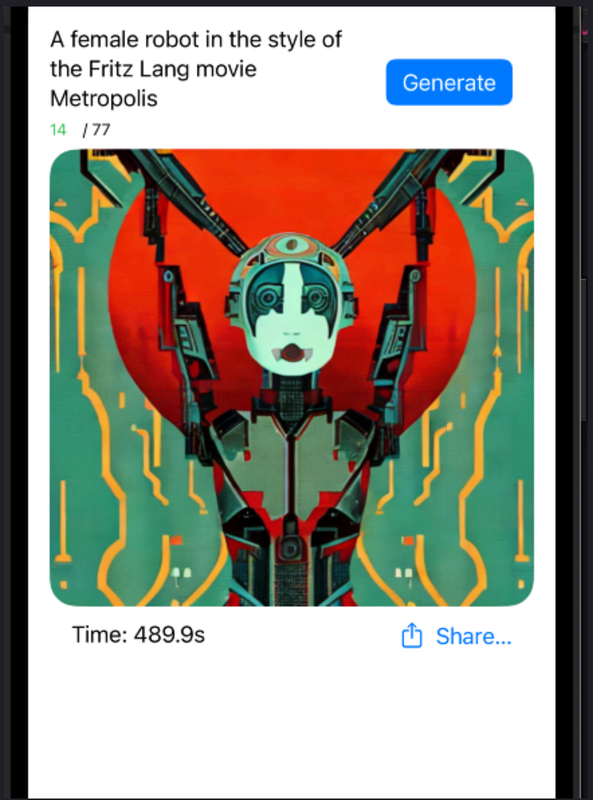
Demo Code Walkthrough




```
var config = StableDiffusionPipeline.Configuration(prompt: prompt)
config.negativePrompt = negativePrompt
config.stepCount = stepCount
config.seed = UInt32(2347)
config.rngType = .numpyRNG
config.guidanceScale = guidanceScale
config.disableSafety = disableSafety
config.schedulerType = scheduler.asStableDiffusionScheduler()
config.useDenoisedIntermediates = true
if isXL {
    config.encoderScaleFactor = 0.13025
    config.decoderScaleFactor = 0.13025
}
```

How to Run Stable Diffusion on an iPhone

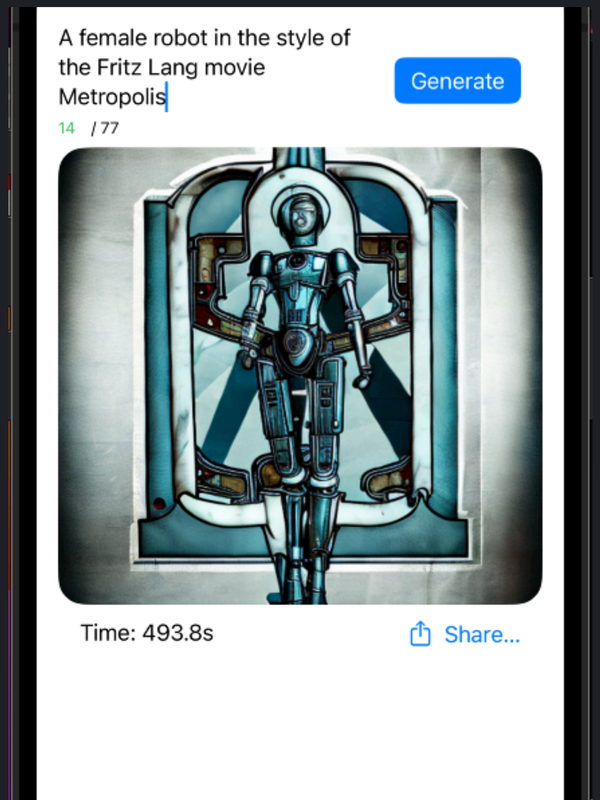
Demo Code Walkthrough




```
var config = StableDiffusionPipeline.Configuration(prompt: prompt)
config.negativePrompt = negativePrompt
config.stepCount = stepCount
config.seed = UInt32(2346)
config.guidanceScale = guidanceScale
config.disableSafety = disableSafety
config.schedulerType = scheduler.asStableDiffusionScheduler()
config.useDenoisedIntermediates = true
if isXL {
    config.encoderScaleFactor = 0.13025
    config.decoderScaleFactor = 0.13025
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

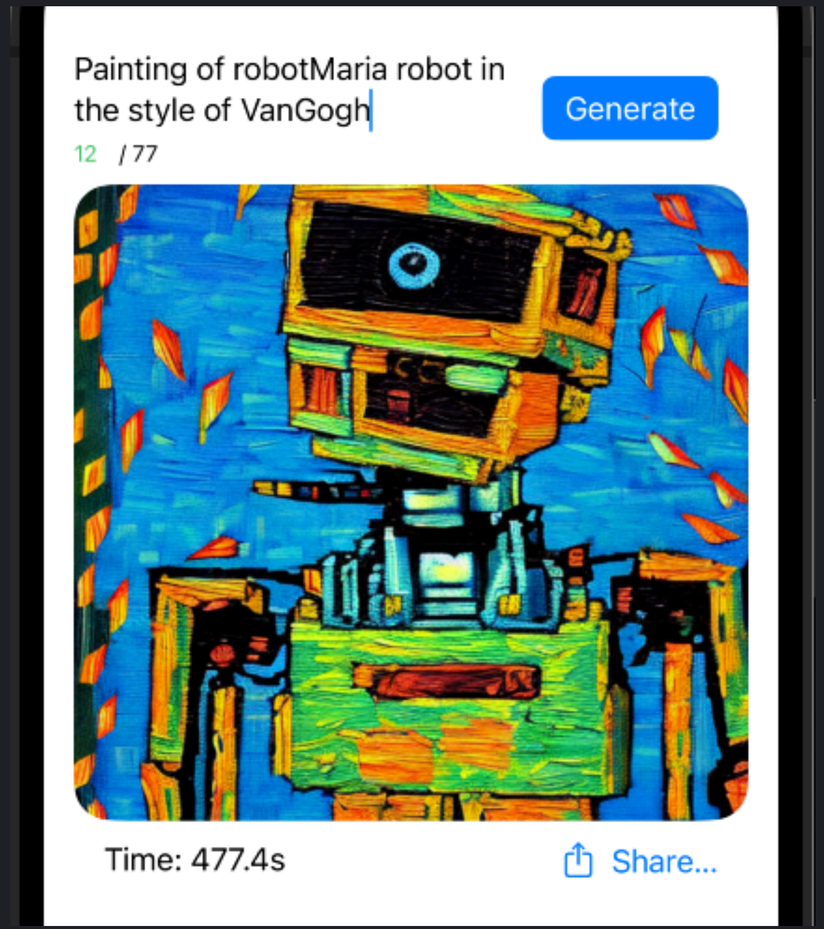



```
var config = StableDiffusionPipeline.Configuration(prompt: prompt)
config.negativePrompt = negativePrompt
config.stepCount = stepCount
config.seed = UInt32(2347)
config.guidanceScale = guidanceScale
config.disableSafety = disableSafety
config.schedulerType = scheduler.asStableDiffusionScheduler()
config.useDenoisedIntermediates = true
if isXL {
    config.encoderScaleFactor = 0.13025
    config.decoderScaleFactor = 0.13025
}
```

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
extension ModelInfo {

    static let v21Palettized = ModelInfo(
        modelId: "apple/coreml-stable-diffusion-2-1-base-palettized",
        modelVersion: "StabilityAI SD 2.1 [6 bit]",
        supportsEncoder: true,
        supportsAttentionV2: true,
        quantized: true
    )
```



How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough

```
func modelURL(for variant: AttentionVariant) -> URL {
    // Pattern:
        https://huggingface
        .co/pcuenq/coreml-stable-diffusion/resolve/main/coreml-stable-diffusion-v1-5_original_compiled
    let suffix: String
    switch variant {
    case .original: suffix = originalAttentionSuffix
    case .splitEinsum: suffix = splitAttentionSuffix
    case .splitEinsumV2: suffix = splitAttentionV2Suffix
    }
    let repo = modelId.split(separator: "/").last!
    return URL(string: "https://huggingface.co/\(modelId)/resolve/main/\(repo)_\(suffix).zip")!
}
```

Hugging Face | Search models, datasets, users...

AllisonMcEntire / **robotMaria** | like 0

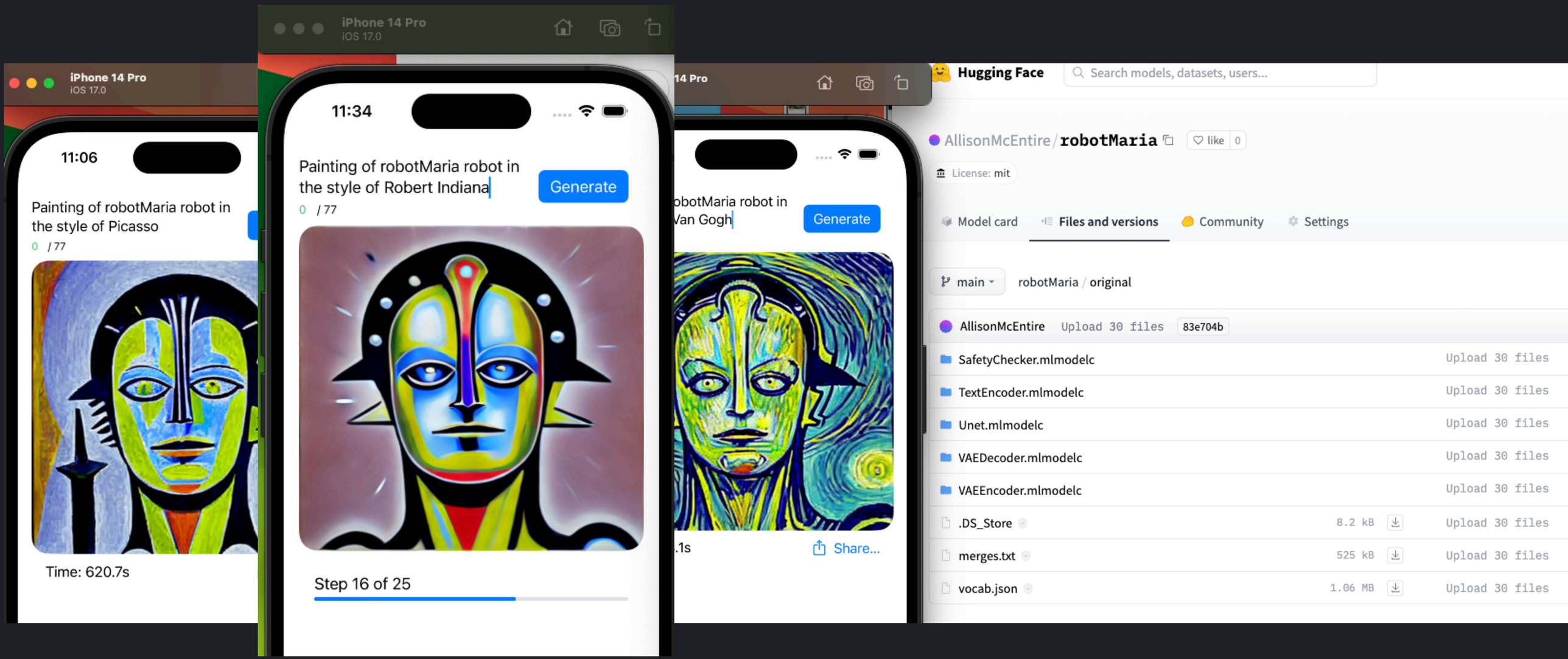
License: mit

Model card | Files and versions | Community | Settings

Dreambooth'd model trained to generate images of the robot from Metropolis wi
the prompt "robotMaria robot"

How to Run Stable Diffusion on an iPhone

Demo Code Walkthrough



The Future of AI in iOS

Emerging trends and possibilities

prompt: “The Future of Artificial Intelligence”

The Future of AI in iOS

is on-device

“We emphasize deploying neural networks on the user’s device for local and privacy-preserving computation.”

https://machinelearning.apple.com/research/on-device-scene-analysis

The Future of AI in iOS

is on-device

“First, the privacy of the end user is protected because any data the user provided as input to the model stays on the user's device. Second, after initial download, users don’t require an internet connection to use the model. Finally, locally deploying this model enables developers to reduce or eliminate their server-related costs.”

https://machinelearning.apple.com/research/stable-diffusion-coreml-apple-silicon

The Future of AI in iOS

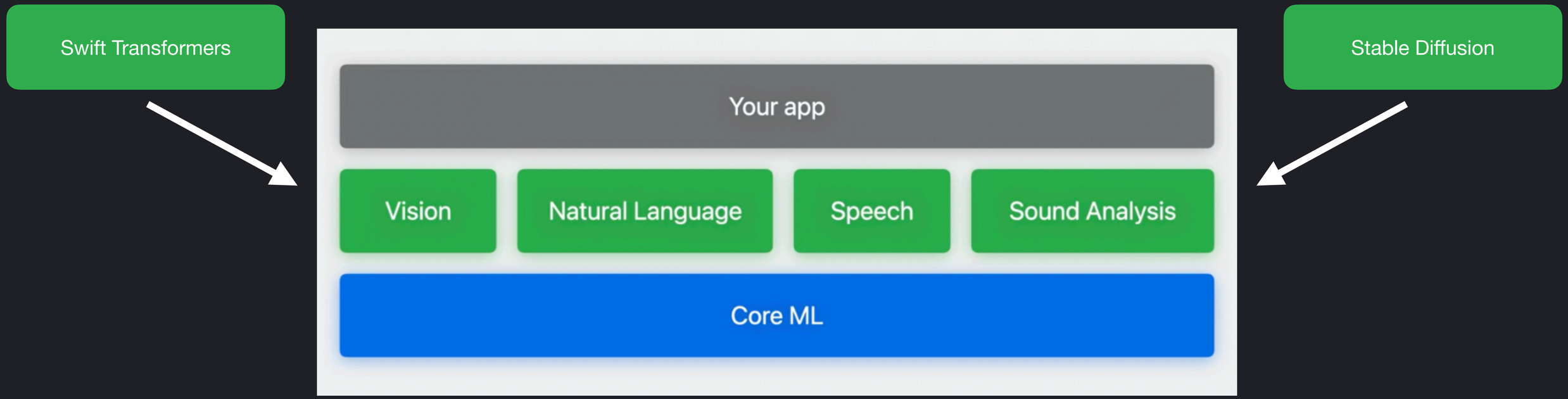
is on-device

“On-device execution is going through a period of extraordinary interest triggered by the popularity of models such as Stable Diffusion and Large Language Models with chat interfaces.”

https://huggingface.co/blog/fast-diffusers-coreml

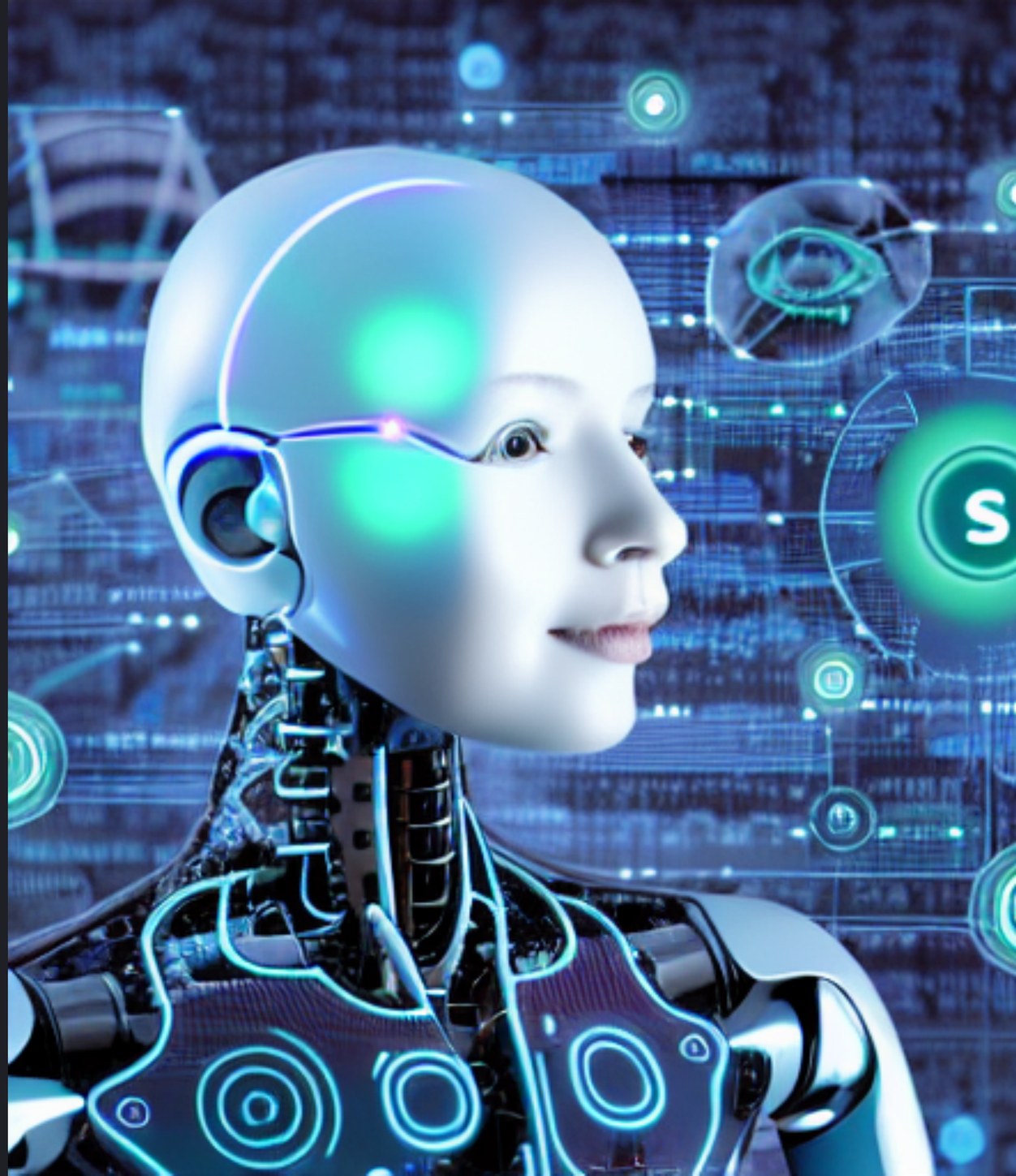
The Future of AI in iOS

is on-device



The Future of AI in iOS

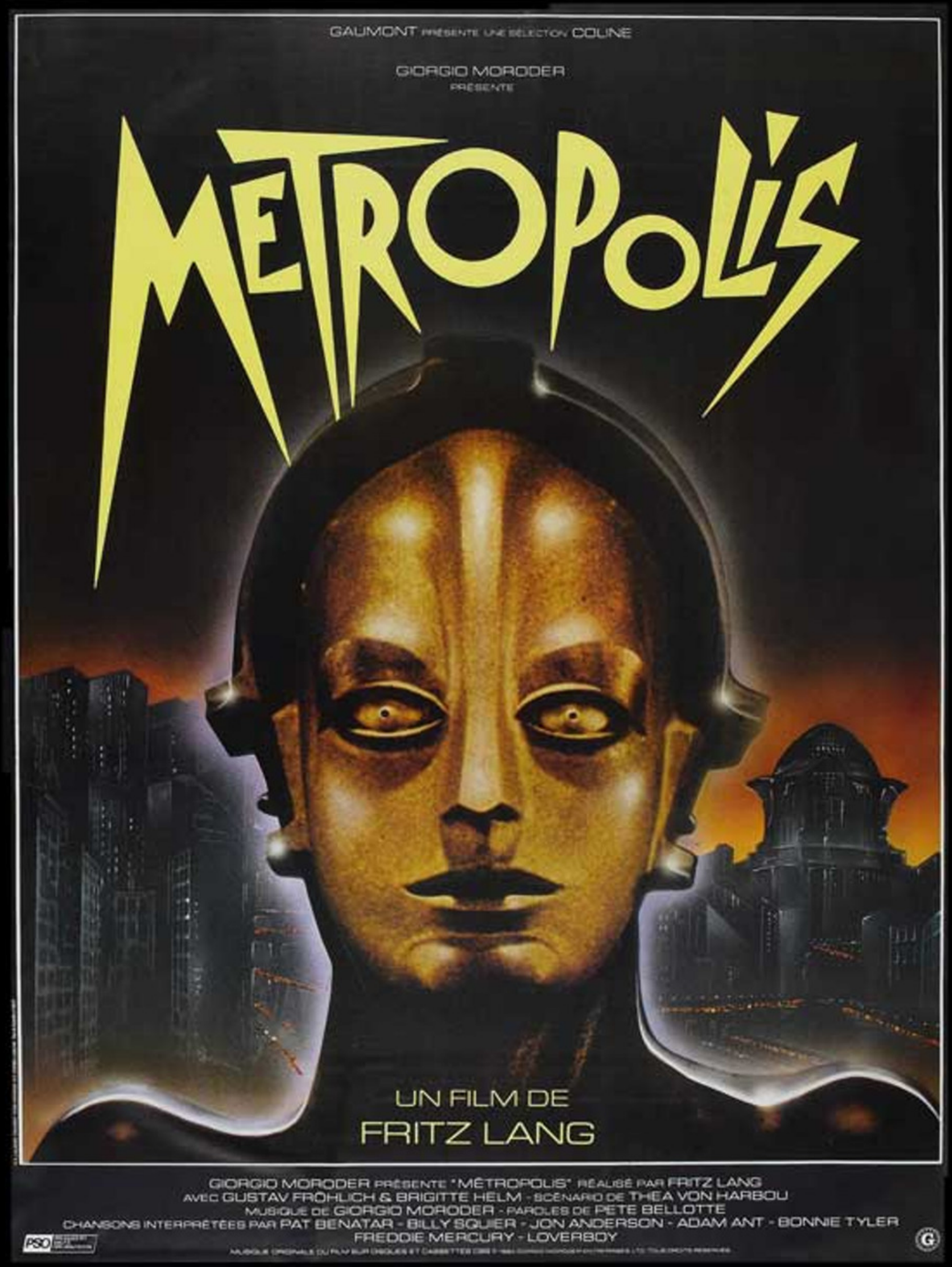
is open source



prompt: “The Future of Artificial Intelligence is Open Source”

Resources and Repos

- https://github.com/apple/ml-stable-diffusion
- https://github.com/huggingface/swift-coreml-diffusers
- https://developer.apple.com/videos/play/wwdc2023-10047
  Use Core ML Tools for machine learning model compression
- https://developer.apple.com/videos/play/wwdc2023-10049
  Improve Core ML integration with async prediction
- https://machinelearning.apple.com/research/stable-diffusion-coreml-apple-silicon
  Stable Diffusion with Core ML on Apple Silicon
- https://machinelearning.apple.com/research/neural-engine-transformers
  Deploying Transformers on the Apple Neural Engine
- https://huggingface.co/docs/diffusers/tutorials/basic_training
  Train a Diffusion Model
- https://huggingface.co/blog/fast-diffusers-coreml
  Faster Stable Diffusion with Core ML on iPhone, iPad, and Mac
- https://huggingface.co/blog/swift-coreml-llm
  Releasing Swift Transformers: Run On-Device LLMs in Apple Devices



https://www.youtube.com/watch?v=yFzHH9EL9x0