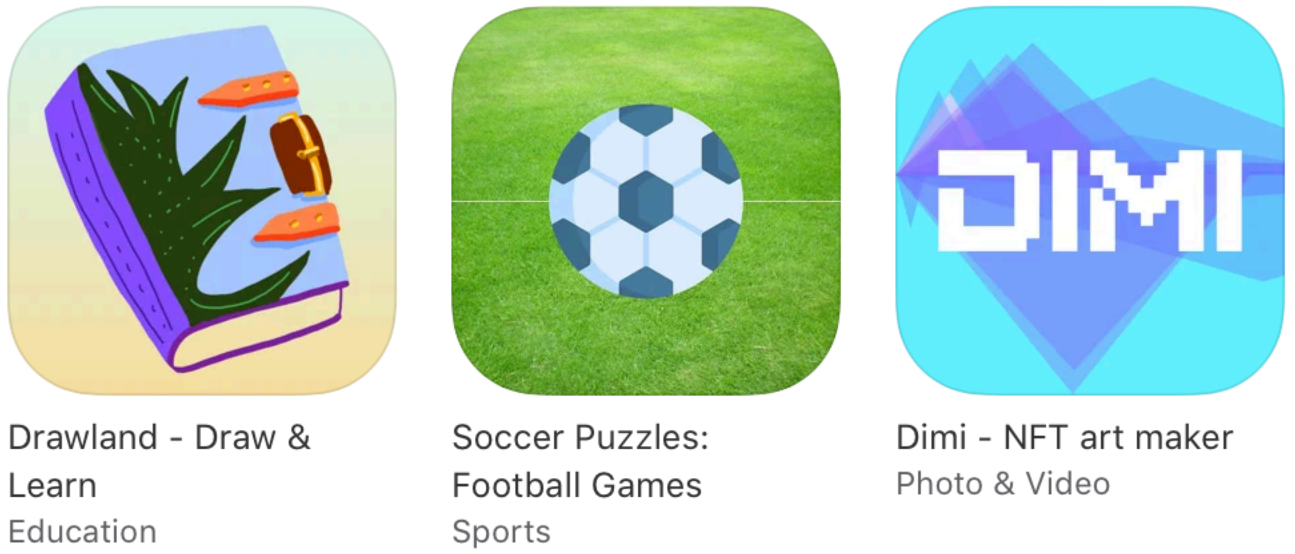
Training Dynamic ML Models on iOS 15

Martin Mitrevski @ Mobius, 25.11.2021

About Me

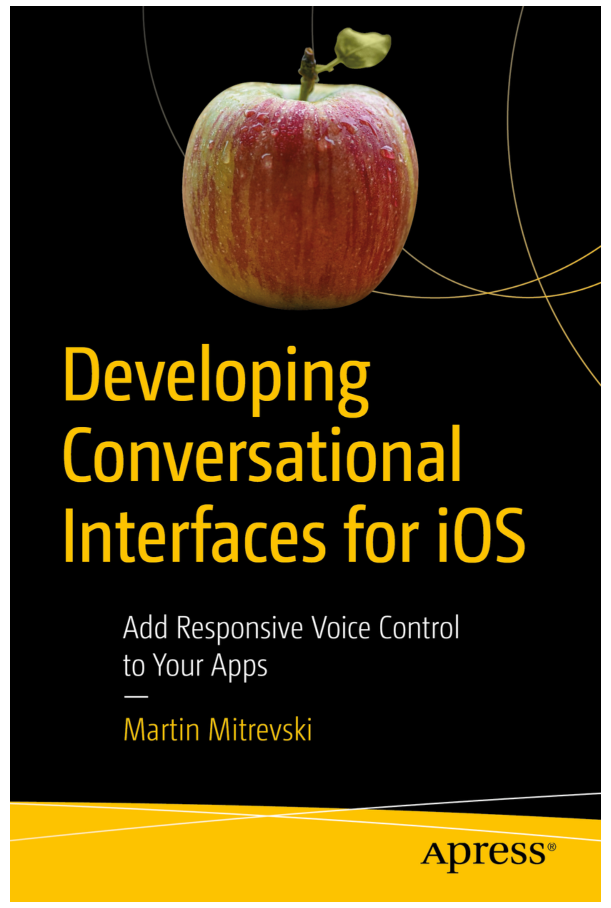
- ➤ Senior iOS Architect @ Stream
- ➤ Blogging on martinmitrevski.com
- ➤ Book author
- ➤ Personal apps
- ➤ twitter @mitrevski



Drawland - Draw & Learn
Education

Soccer Puzzles: Football Games
Sports

Dimi - NFT art maker
Photo & Video



Developing Conversational Interfaces for iOS

Add Responsive Voice Control to Your Apps

Martin Mitrevski

Apress®

User Expectations

Amazing app experience

Personalization

Simplicity

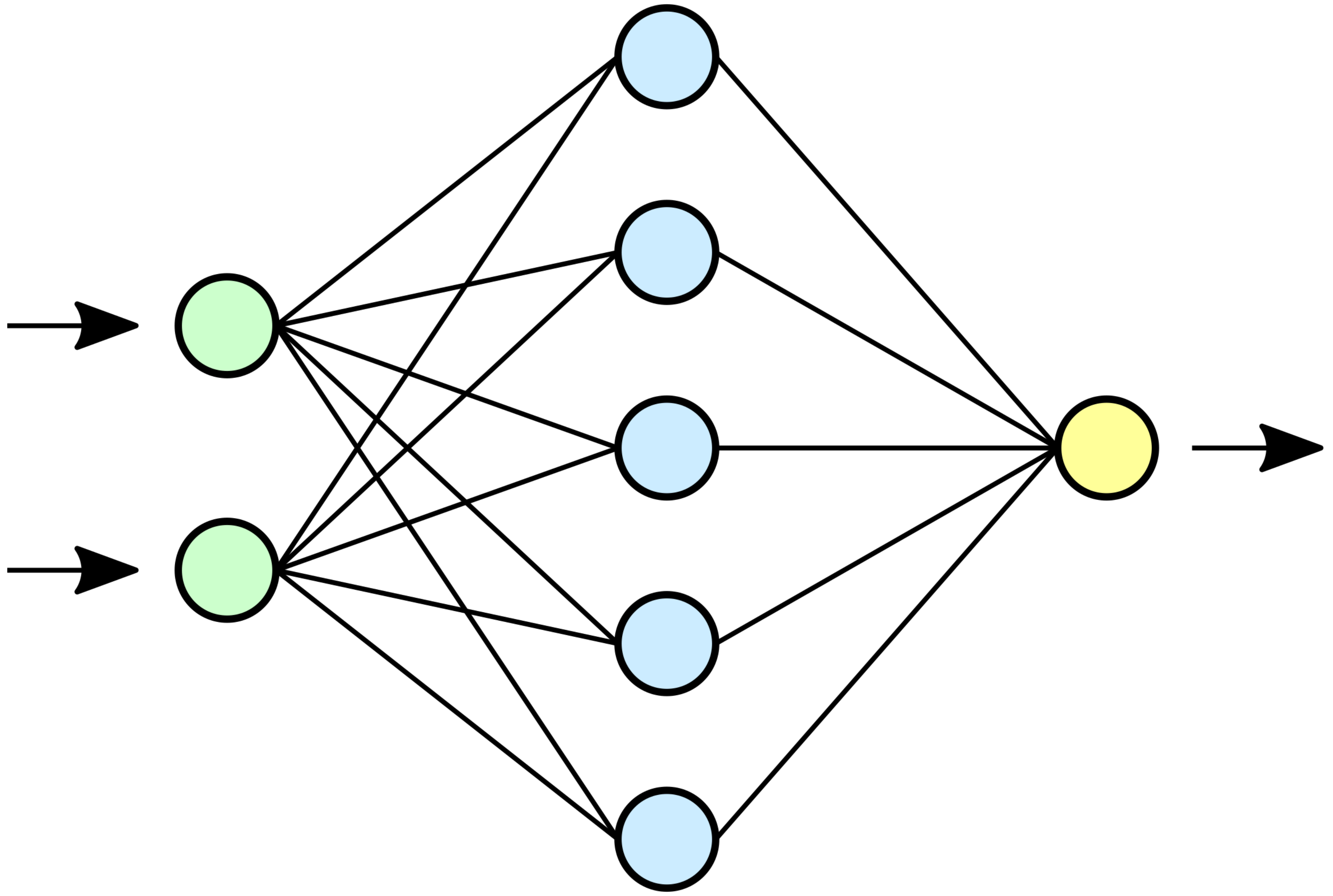
Stability

Privacy and security

Be more productive

Machine Learning

- Learn from data
- Identify patterns
- Make decisions with minimal/no user intervention



Machine Learning on the Cloud

- Handled only on the server (lighter apps)
- Easier ML model updates
- Massive training data available

- Expensive infrastructure
- Network connection required
- GDPR and other privacy regulations

- User data is sent to the cloud
- Good for companies, but not for users

Machine Learning on the Device

- No server infrastructure at all
- No network connection required
- Complete user privacy
- Personalized experience
- No GDPR or other regulations compliance

- Trickier ML model updates
- Less data available for training
- Has to be implemented on all platforms separately

ML on iOS

CoreML

Vision

CoreML tools

CreateML macOS

CreateML iOS (new)

TuriCreate

Third party:

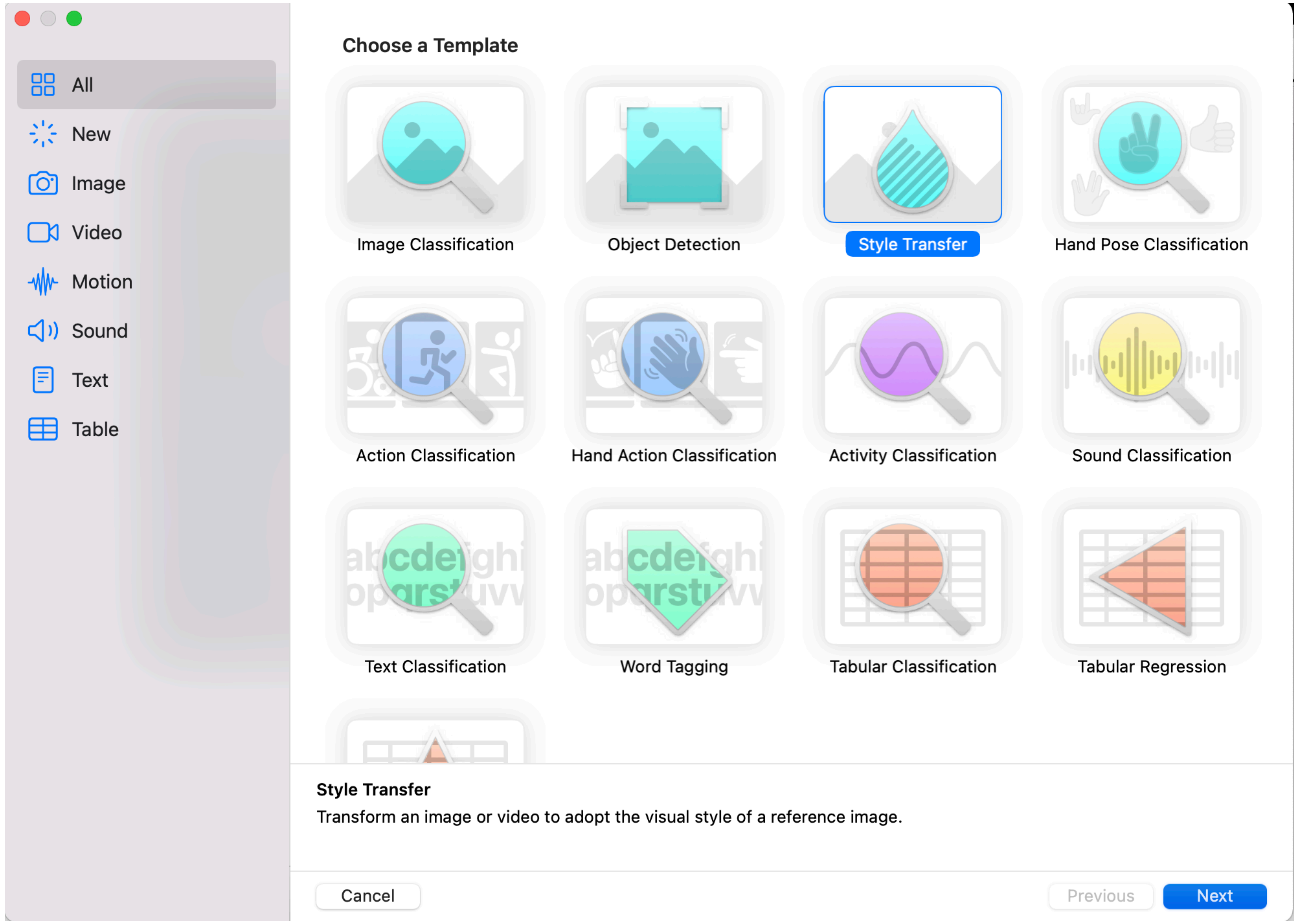
MLKit

IBM Watson

TensorFlow

…

CreateML



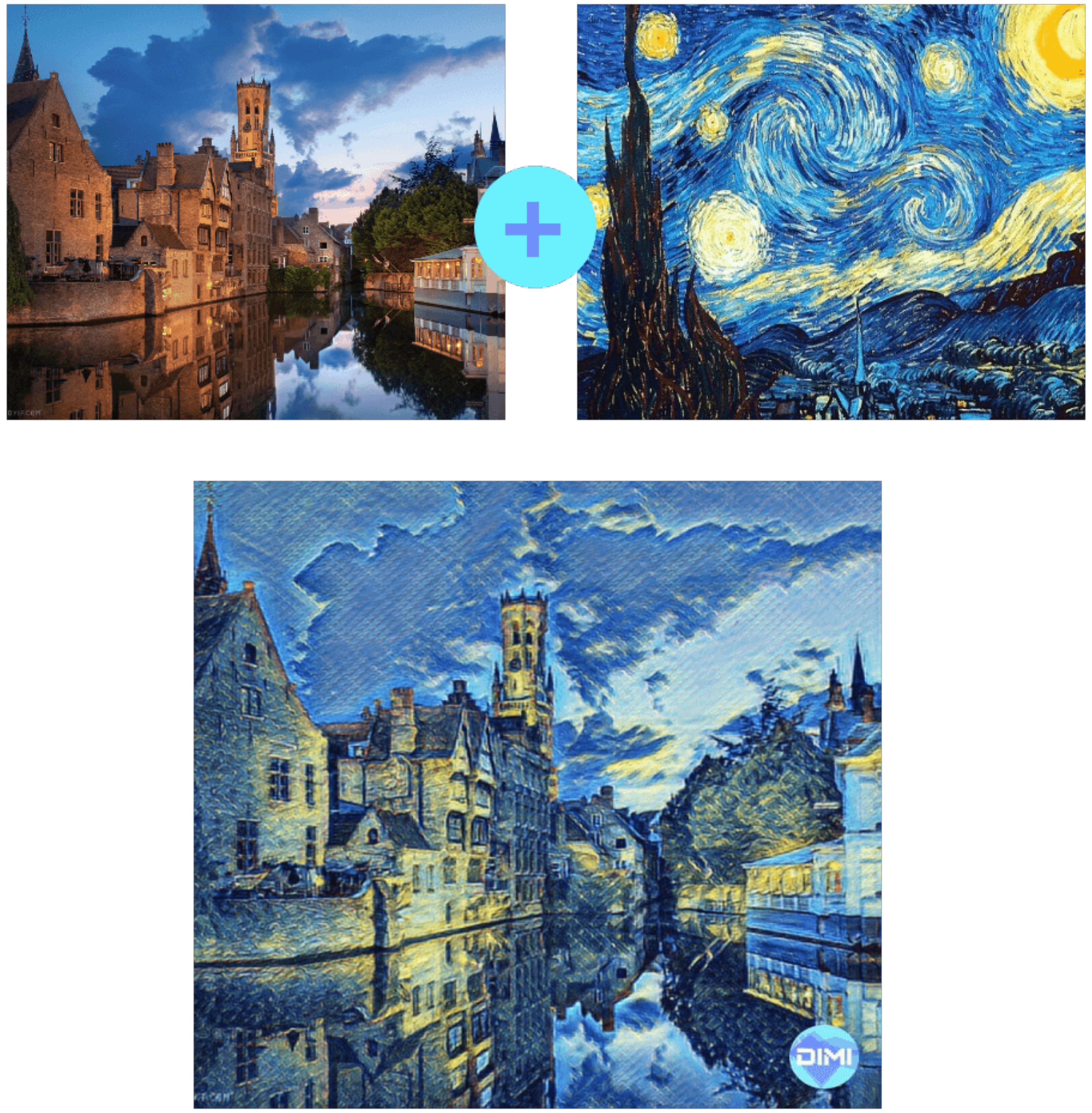
Style Transfer

Transform an image or video to adopt the visual style of a reference image.

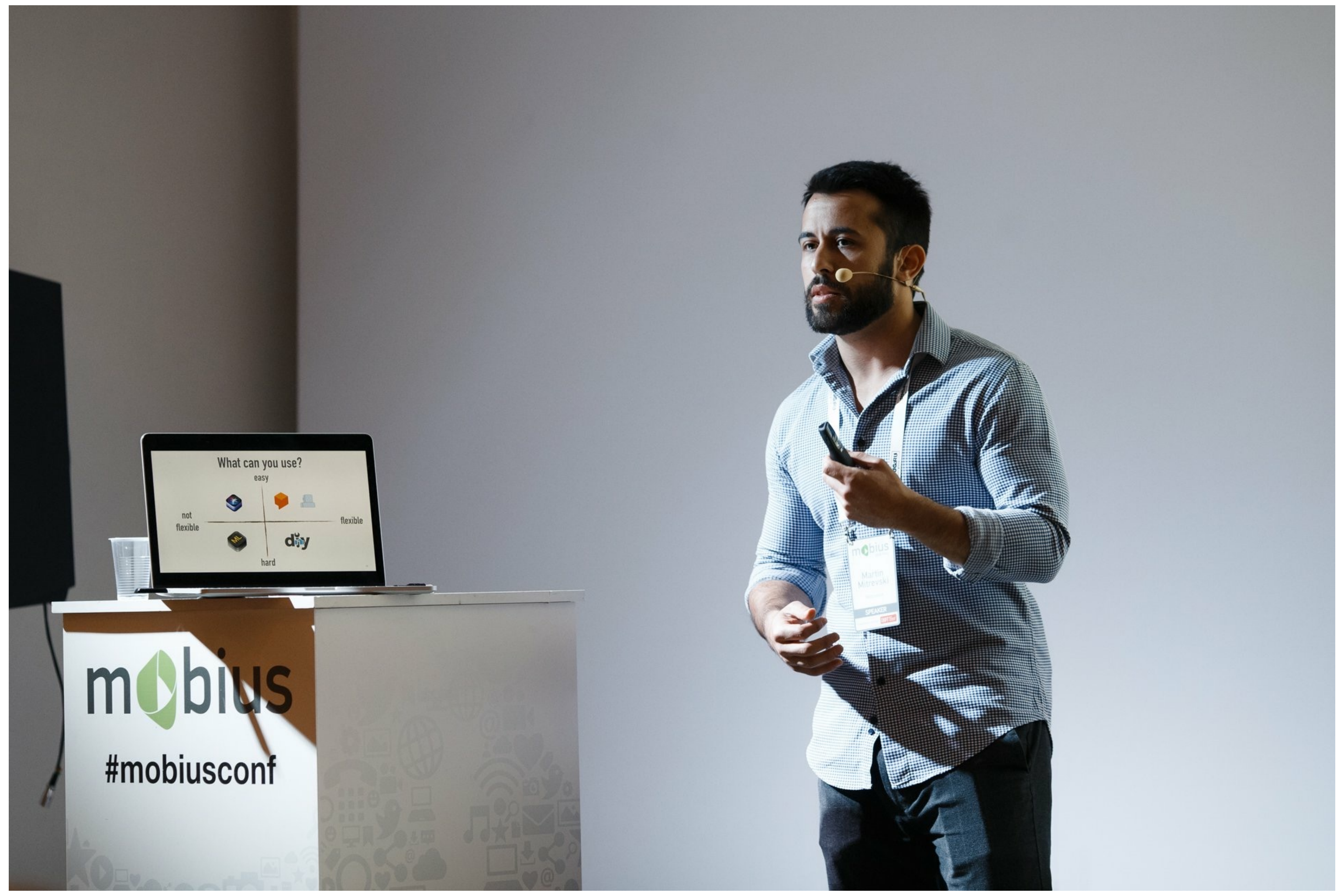
- Now available on iOS:
  - Image classification
  - Text classification
  - Hand pose classification
  - Hand action classification
  - Sound classification
  - Style transfer

Style Transfer

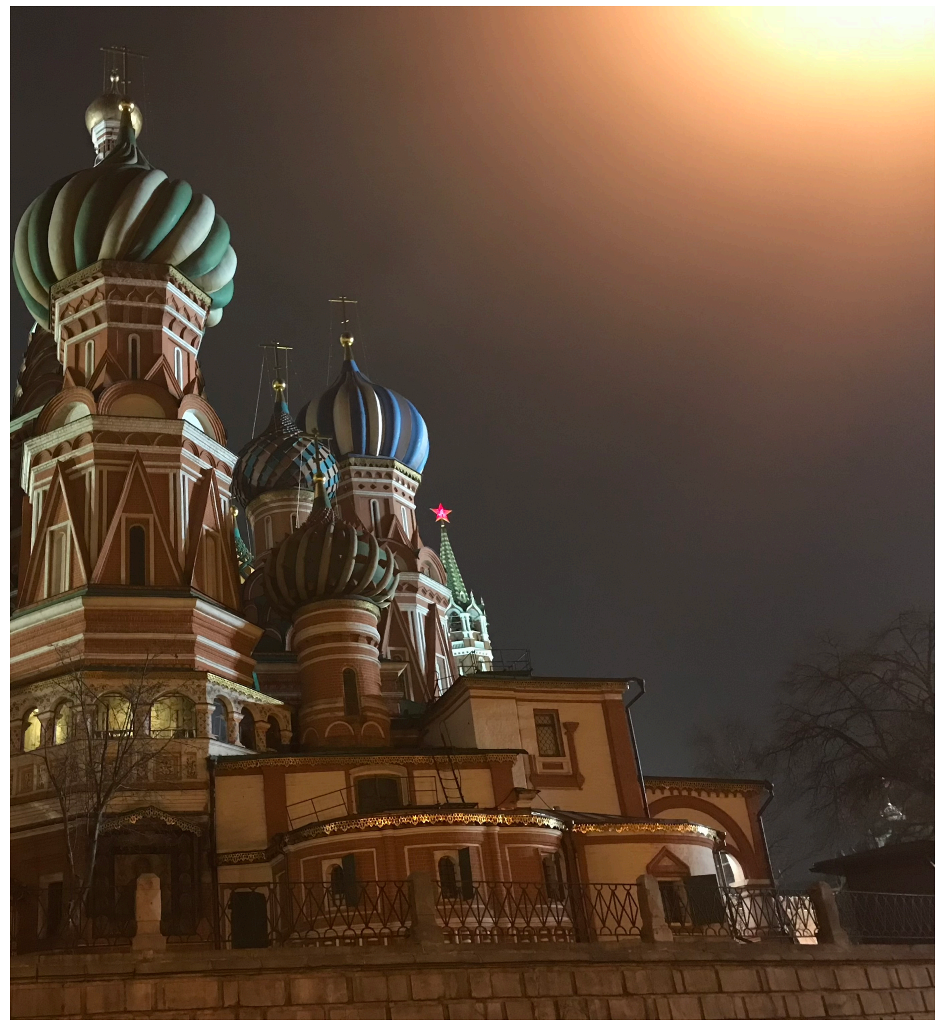
- Applying style from one image to another
- Use of deep neural networks
- Possible use-cases
  - Image filtering app
  - Creation of artificial artwork from photos (e.g. NFTs)



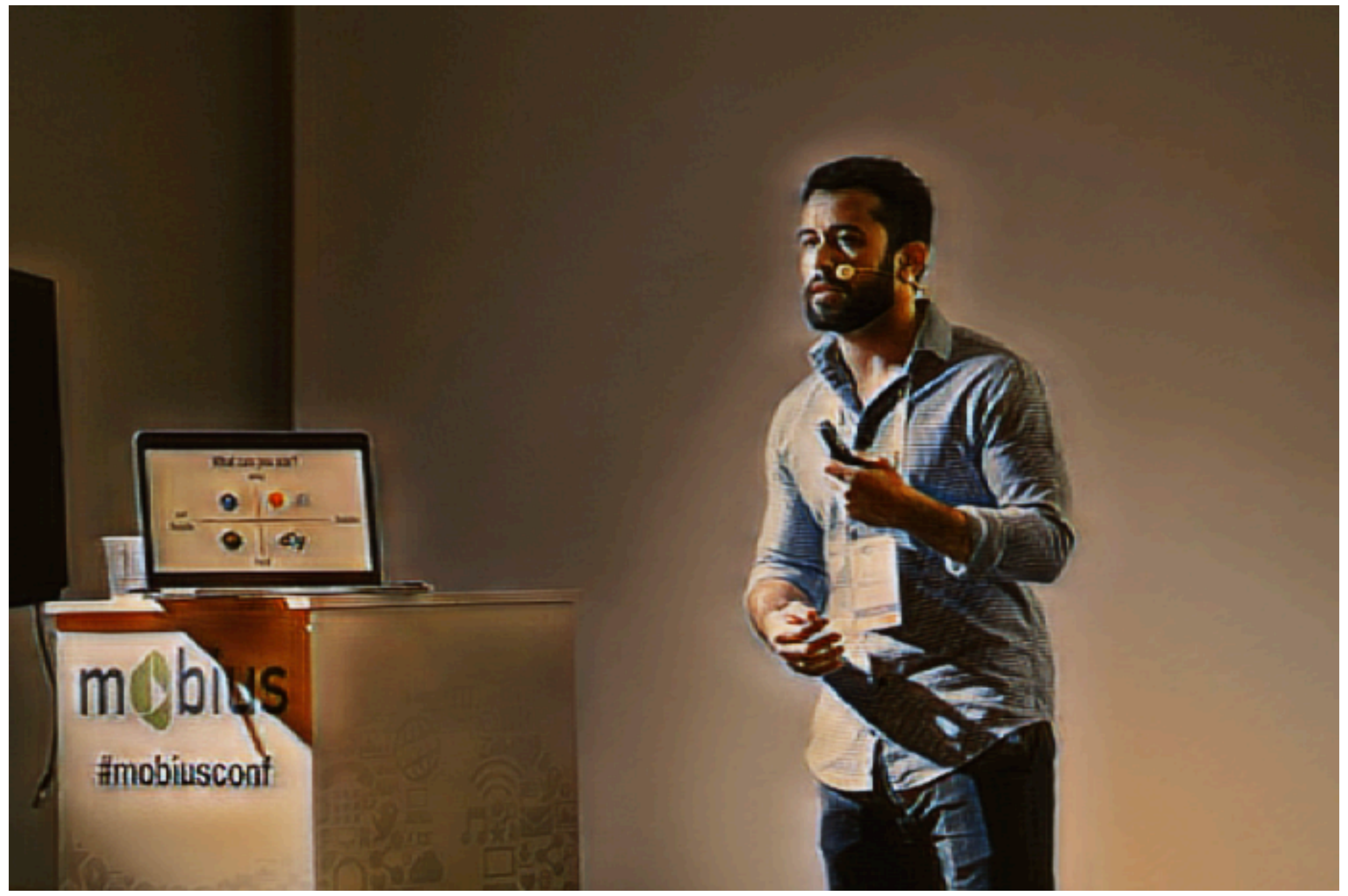
Style Transfer Showcase



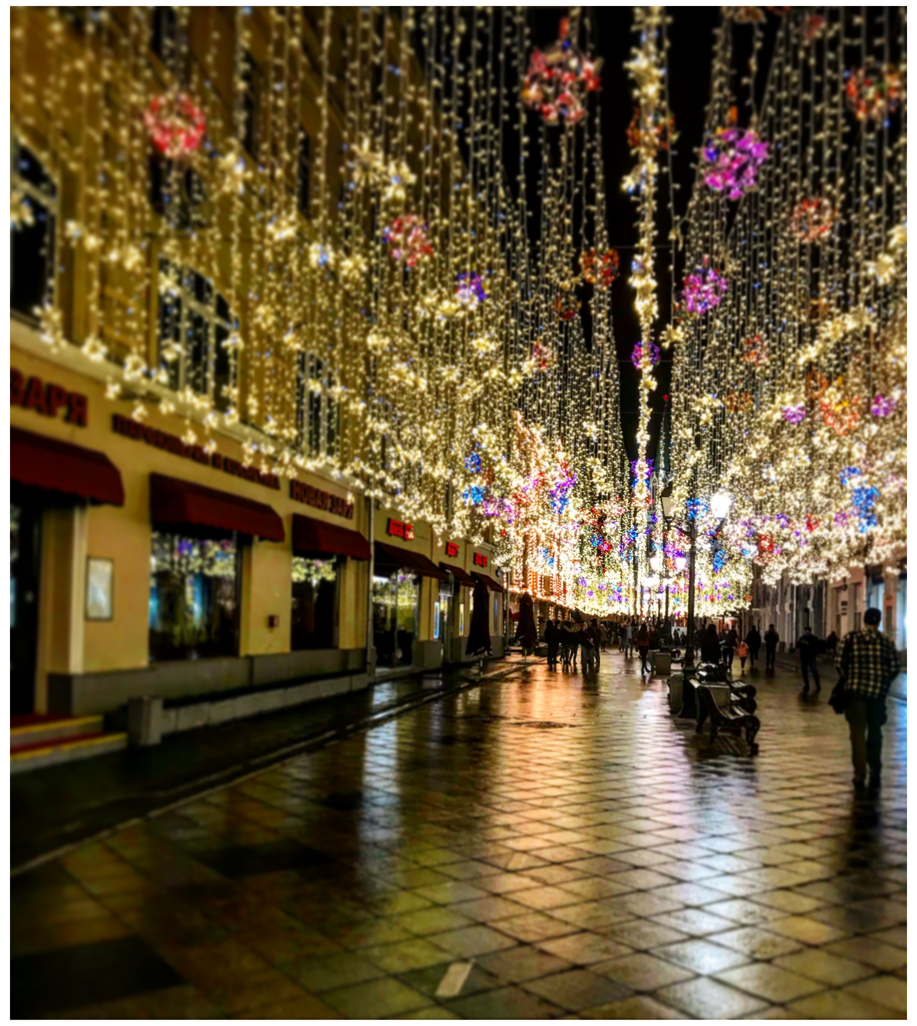
+



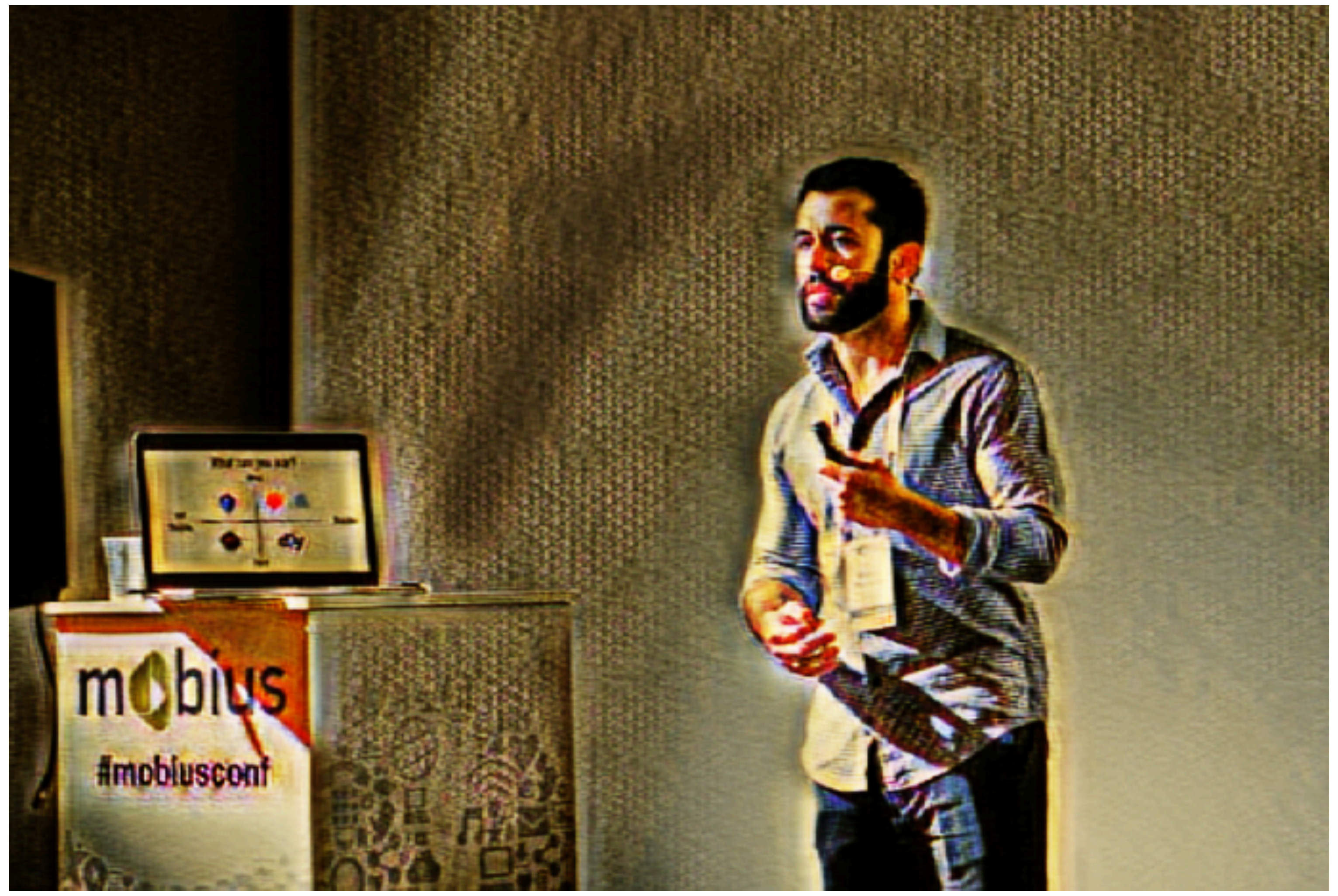
=



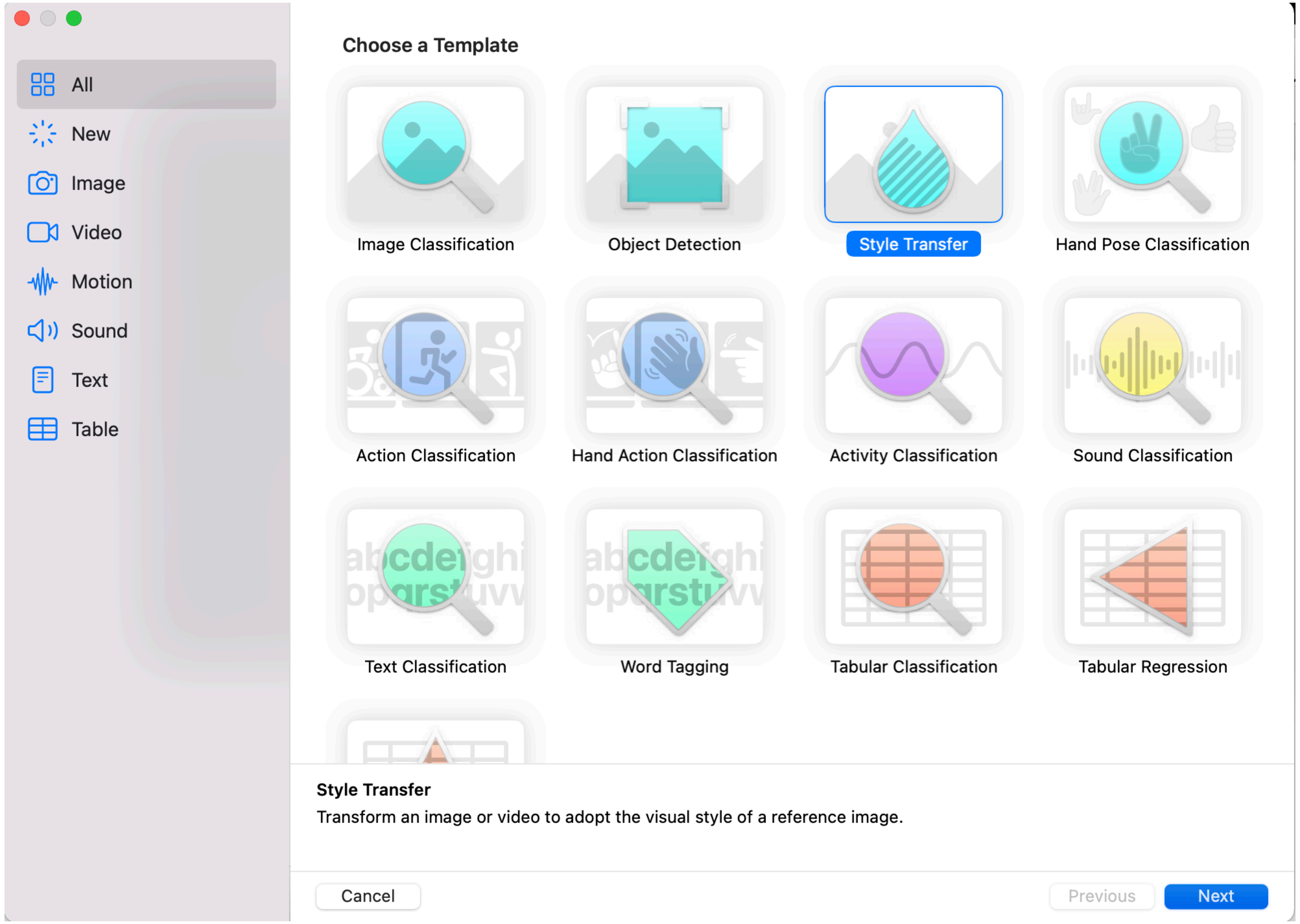
+



=



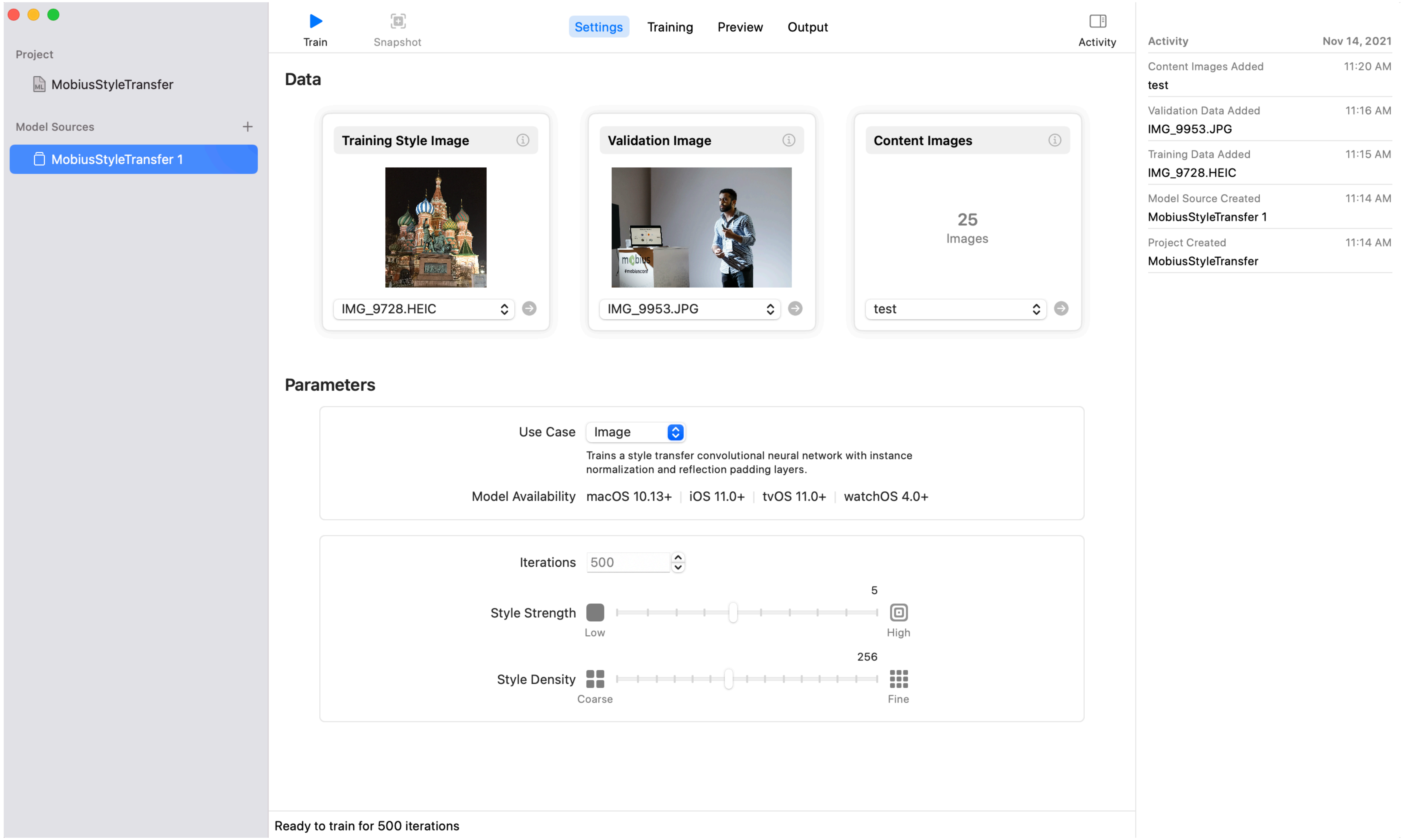
Creating Filters With CreateML App (Demo)



Style Transfer

Transform an image or video to adopt the visual style of a reference image.

Setting Up the Training



Project

MobiusStyleTransfer

Model Sources +

MobiusStyleTransfer 1

Data

Training Style Image

IMG_9728.HEIC

Validation Image

IMG_9953.JPG

Content Images

25 Images

test

Parameters

Use Case: Image

Trains a style transfer convolutional neural network with instance normalization and reflection padding layers.

Model Availability: macOS 10.13+ | iOS 11.0+ | tvOS 11.0+ | watchOS 4.0+

Iterations: 500

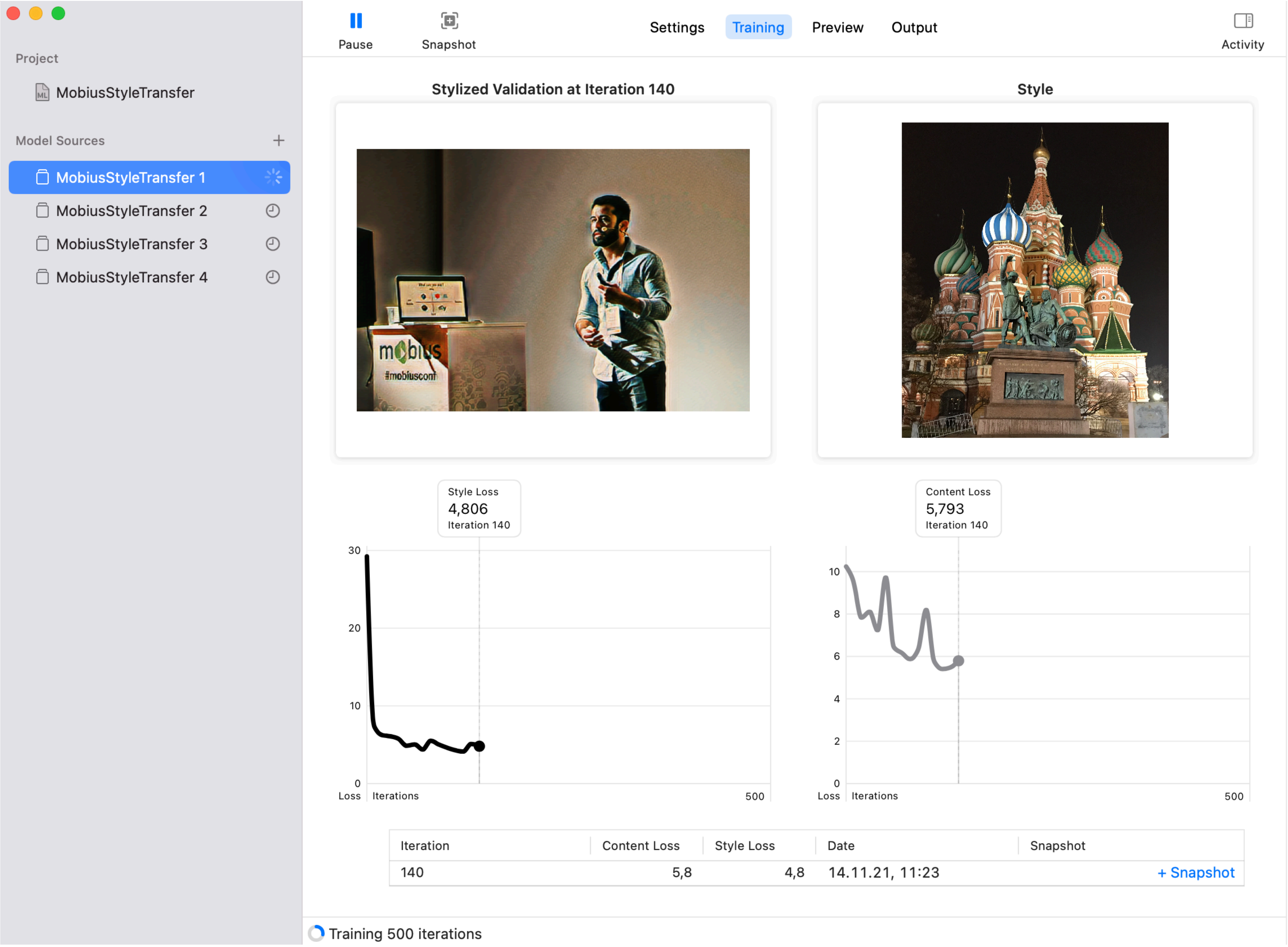
Style Strength: 5 (Low – High)

Style Density: 256 (Coarse – Fine)

Ready to train for 500 iterations

Activity	Nov 14, 2021
Content Images Added test	11:20 AM
Validation Data Added IMG_9953.JPG	11:16 AM
Training Data Added IMG_9728.HEIC	11:15 AM
Model Source Created MobiusStyleTransfer 1	11:14 AM
Project Created MobiusStyleTransfer	11:14 AM

Training Progress and Evaluation



Stylized Validation at Iteration 140

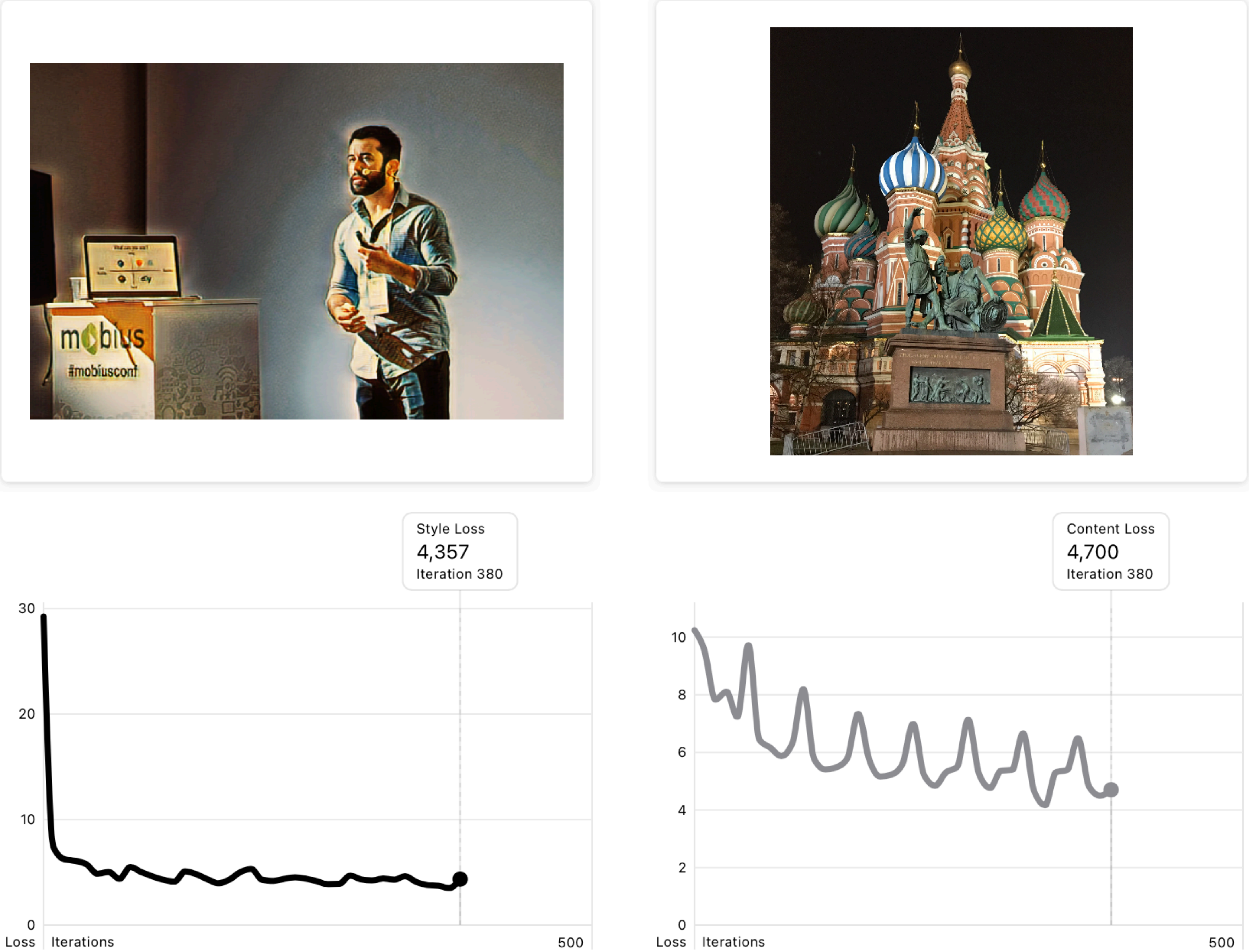
Style

Iteration	Content Loss	Style Loss	Date	Snapshot
140	5,8	4,8	14.11.21, 11:23	+ Snapshot

Training 500 iterations

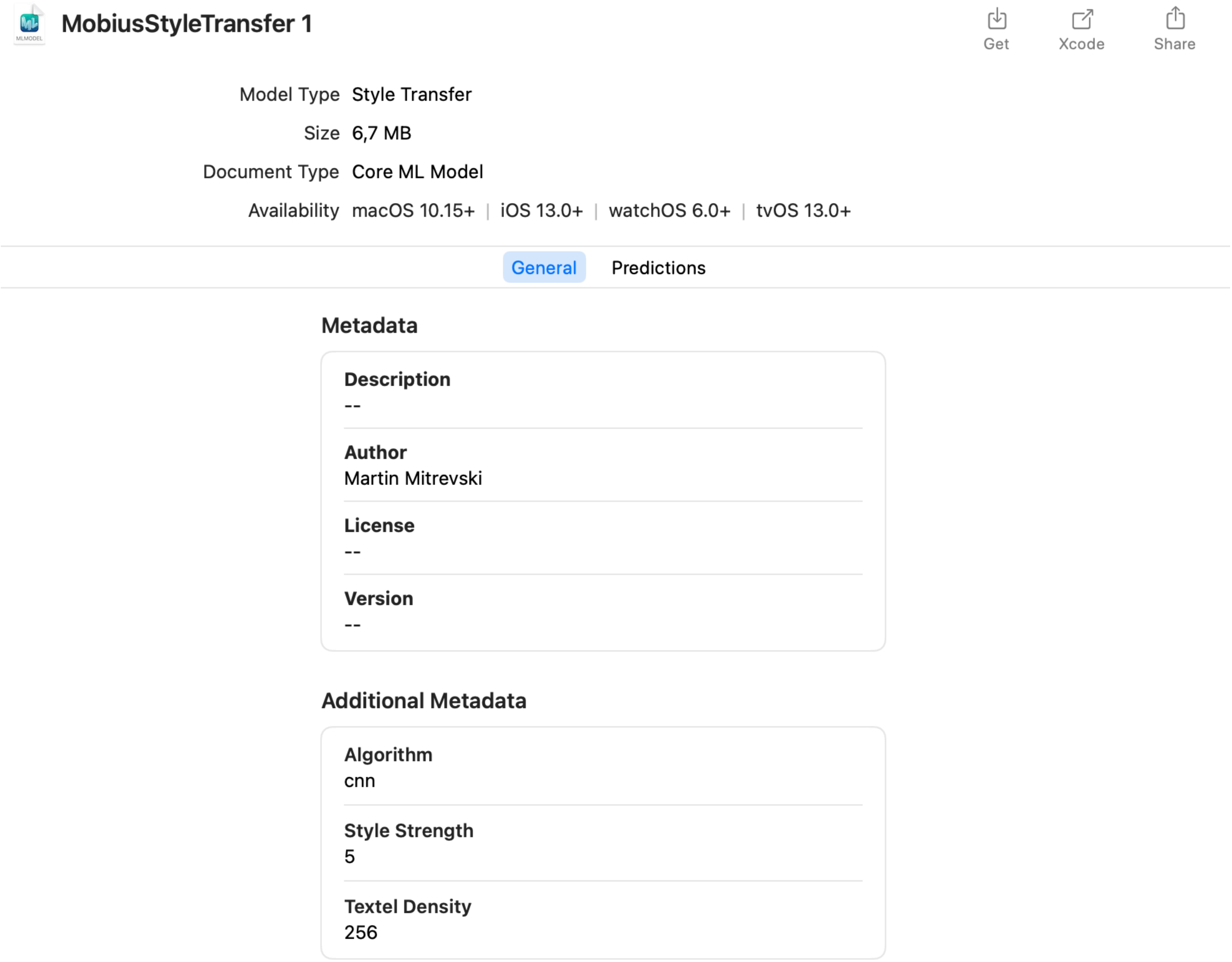
Stylized Validation at Iteration 380

Style



Iteration	Content Loss	Style Loss	Date	Snapshot
380	4,7	4,4	14.11.21, 11:26	+ Snapshot

Exporting the Model



MobiusStyleTransfer 1

Model Type	Style Transfer
Size	6,7 MB
Document Type	Core ML Model
Availability	macOS 10.15+ | iOS 13.0+ | watchOS 6.0+ | tvOS 13.0+

Metadata

Description
--

Author
Martin Mitrevski

License
--

Version
--

Additional Metadata

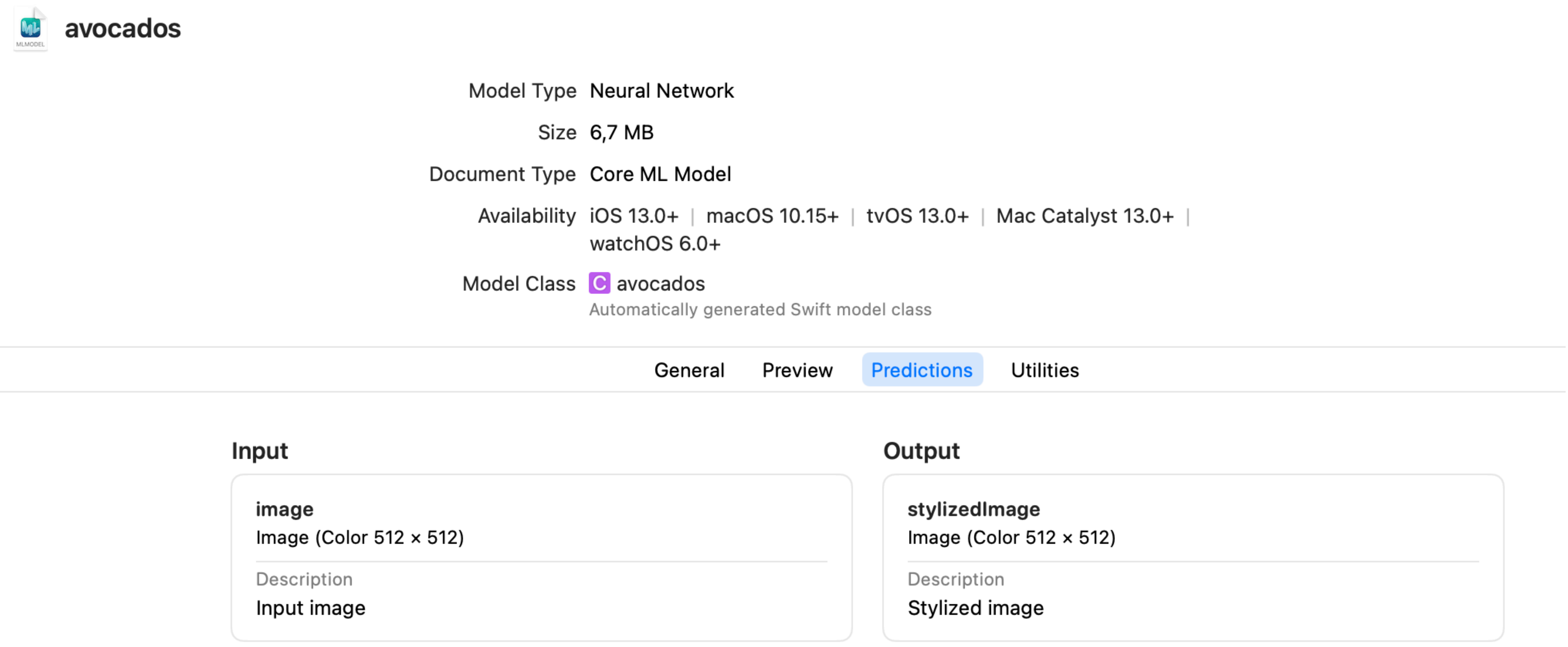
Algorithm
cnn

Style Strength
5

Textel Density
256

Integrating in an iOS App

- Drag & drop generated CoreML model
- Xcode generates the interface for interacting with the model
- Prepare the input for the model
- Handle the appropriate output



avocados

Model Type Neural Network

Size 6,7 MB

Document Type Core ML Model

Availability iOS 13.0+ | macOS 10.15+ | tvOS 13.0+ | Mac Catalyst 13.0+ | watchOS 6.0+

Model Class C avocados
Automatically generated Swift model class

Input

image
Image (Color 512 × 512)
Description
Input image

Output

stylizedImage
Image (Color 512 × 512)
Description
Stylized image

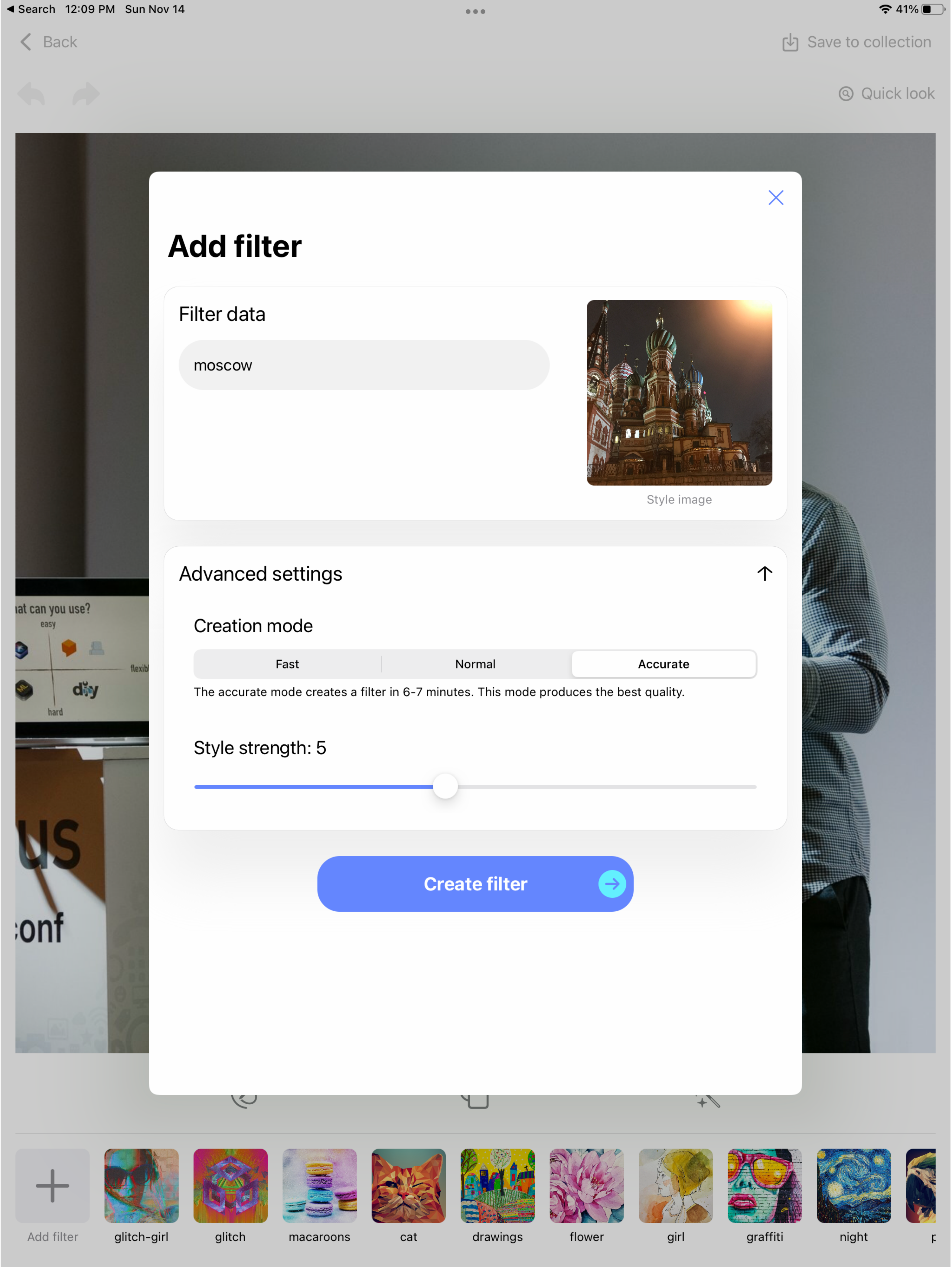
CreateML on macOS

- Model creation caveats
  - Training was not done on the device
  - Required internet connection to update model
  - Personalization was not on-the-go

CreateML on iOS

- CreateML on iOS
  - Training is done on the device dynamically
  - No internet connection
  - Personalization on-the-go

Creating the Model in an iOS App (Demo)



Add filter

Filter data

moscow

Style image

Advanced settings

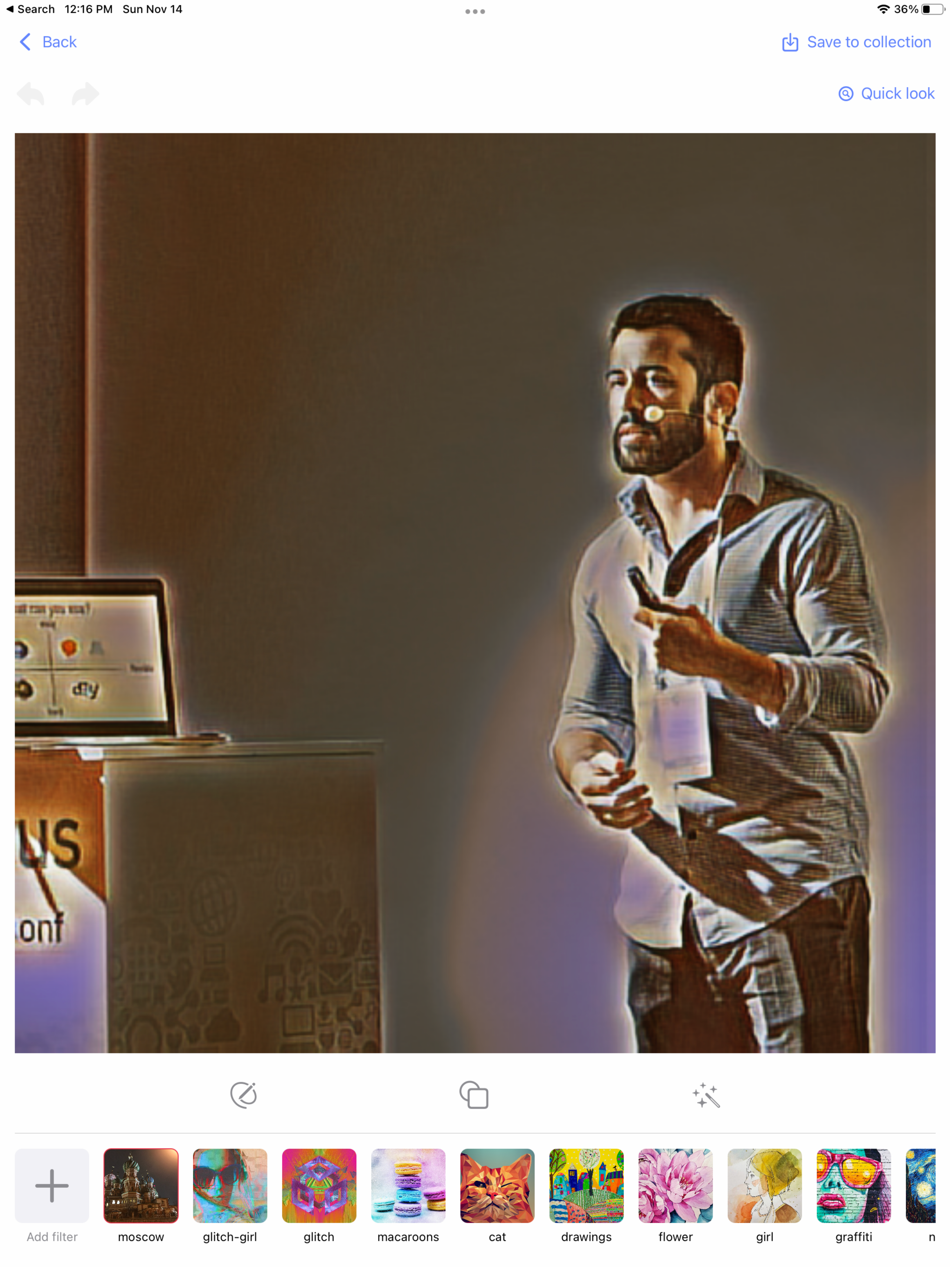
Creation mode

Fast | Normal | [✓ Accurate]

The accurate mode creates a filter in 6-7 minutes. This mode produces the best quality.

Style strength: 5

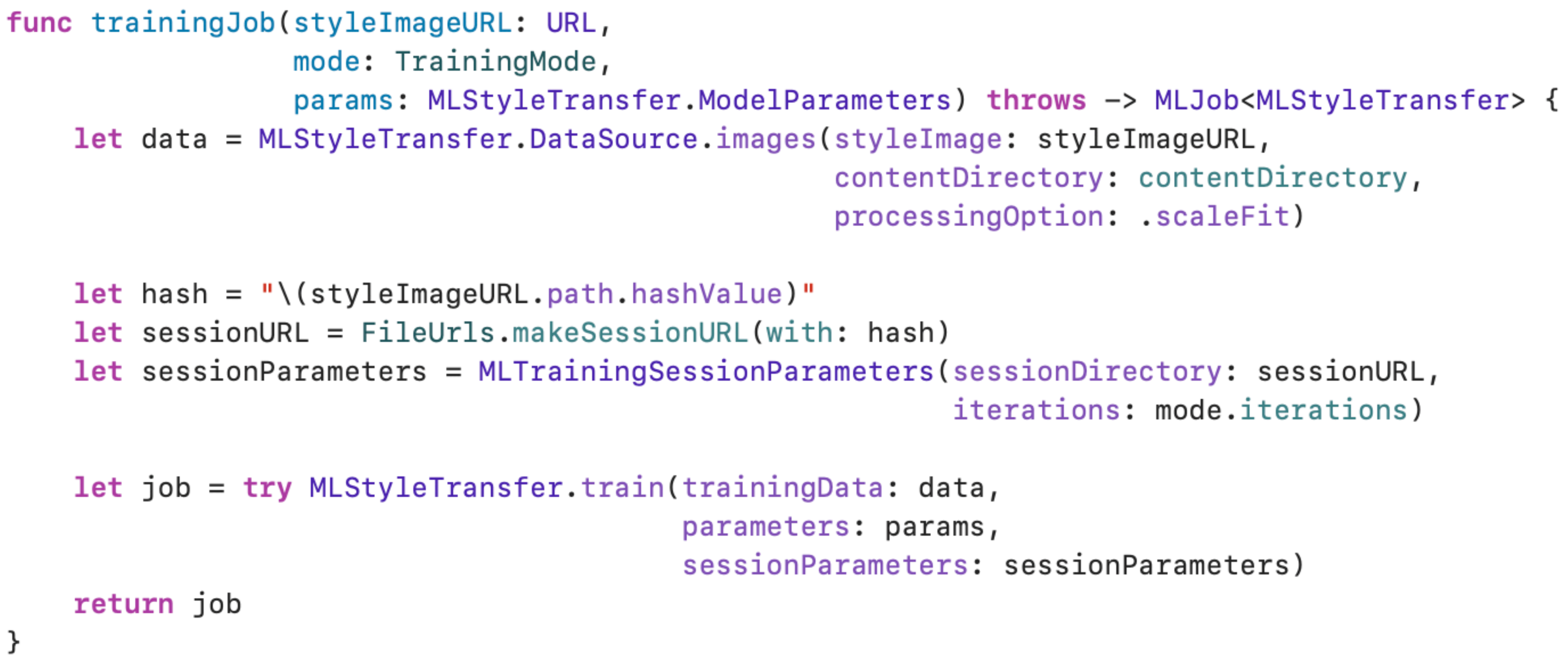
Create filter



Steps

- Take the user's photo and the style image
- Setup local content directory with images
- Setup parameters for training
- Create a training job (MLJob)
- Listen for updates (Combine publisher)
- Save the created model on the device

Creating Training Job

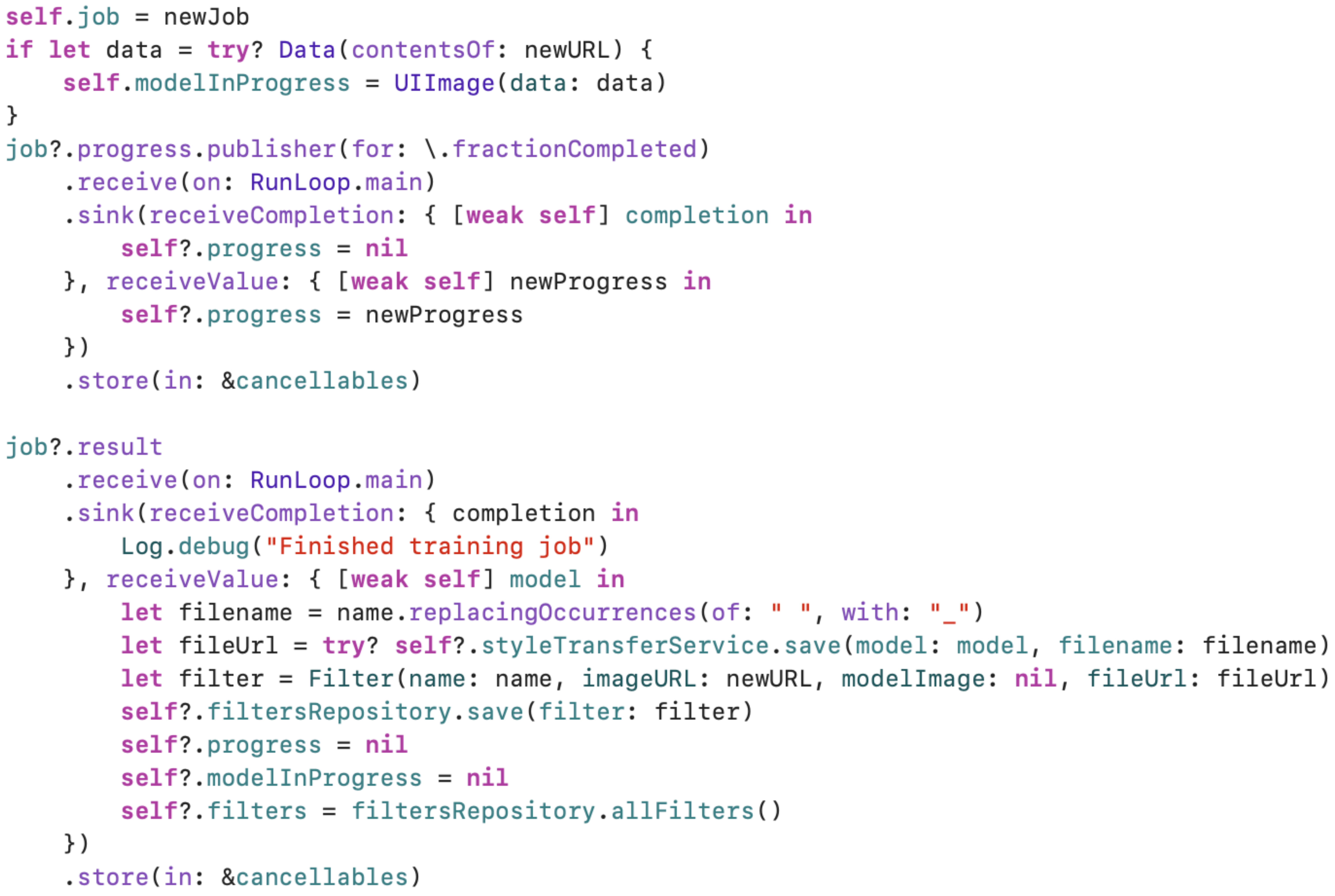


```swift
func trainingJob(styleImageURL: URL,
                 mode: TrainingMode,
                 params: MLStyleTransfer.ModelParameters) throws -> MLJob<MLStyleTransfer> {
    let data = MLStyleTransfer.DataSource.images(styleImage: styleImageURL,
                                                 contentDirectory: contentDirectory,
                                                 processingOption: .scaleFit)

    let hash = "\(styleImageURL.path.hashValue)"
    let sessionURL = FileUrls.makeSessionURL(with: hash)
    let sessionParameters = MLTrainingSessionParameters(sessionDirectory: sessionURL,
                                                        iterations: mode.iterations)

    let job = try MLStyleTransfer.train(trainingData: data,
                                        parameters: params,
                                        sessionParameters: sessionParameters)
    return job
}
```

Listening for Updates



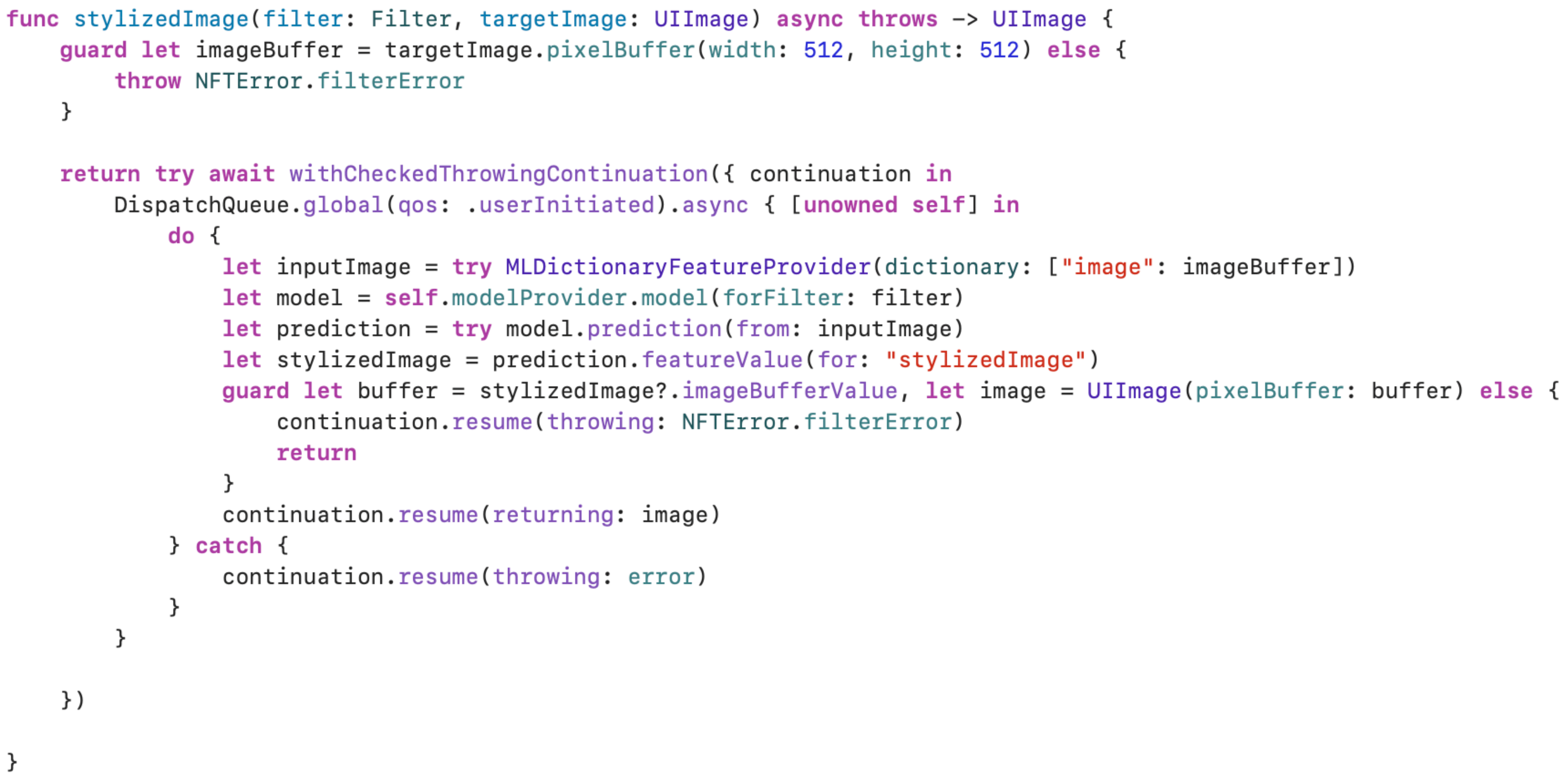
```swift
self.job = newJob
if let data = try? Data(contentsOf: newURL) {
    self.modelInProgress = UIImage(data: data)
}
job?.progress.publisher(for: \.fractionCompleted)
    .receive(on: RunLoop.main)
    .sink(receiveCompletion: { [weak self] completion in
        self?.progress = nil
    }, receiveValue: { [weak self] newProgress in
        self?.progress = newProgress
    })
    .store(in: &cancellables)

job?.result
    .receive(on: RunLoop.main)
    .sink(receiveCompletion: { completion in
        Log.debug("Finished training job")
    }, receiveValue: { [weak self] model in
        let filename = name.replacingOccurrences(of: " ", with: "_")
        let fileUrl = try? self?.styleTransferService.save(model: model, filename: filename)
        let filter = Filter(name: name, imageURL: newURL, modelImage: nil, fileUrl: fileUrl)
        self?.filtersRepository.save(filter: filter)
        self?.progress = nil
        self?.modelInProgress = nil
        self?.filters = filtersRepository.allFilters()
    })
    .store(in: &cancellables)
```

Training Notes

- Training time depends on number of iterations
- Less iterations bring lower filter quality
- Training is slower on device, compared to the Mac
- Consider using background tasks for longer filtering tasks
- Performing style transfer with trained model is very fast

Performing Style Transfer



```swift
func stylizedImage(filter: Filter, targetImage: UIImage) async throws -> UIImage {
    guard let imageBuffer = targetImage.pixelBuffer(width: 512, height: 512) else {
        throw NFTError.filterError
    }

    return try await withCheckedThrowingContinuation({ continuation in
        DispatchQueue.global(qos: .userInitiated).async { [unowned self] in
            do {
                let inputImage = try MLDictionaryFeatureProvider(dictionary: ["image": imageBuffer])
                let model = self.modelProvider.model(forFilter: filter)
                let prediction = try model.prediction(from: inputImage)
                let stylizedImage = prediction.featureValue(for: "stylizedImage")
                guard let buffer = stylizedImage?.imageBufferValue, let image = UIImage(pixelBuffer: buffer) else {
                    continuation.resume(throwing: NFTError.filterError)
                    return
                }
                continuation.resume(returning: image)
            } catch {
                continuation.resume(throwing: error)
            }
        }

    })

}
```

Making Predictions

- Prediction method is synchronous
- Blocks the UI if run on the main thread
- In the example, the new Swift concurrency is used (async/await)
- Other options like Combine Futures, completion handlers, etc. are also possible

Usage

```
func stylizeSelectedImage() {
    guard let selectedFilter = selectedFilter else {
        return
    }

    Task {
        do {
            self.loading = true
            self.targetImage = try await self.styleTransferService.stylizedImage(filter: selectedFilter,
                                                                                 targetImage: image)
            self.loading = false
        } catch {
            Log.debug(error.localizedDescription)
            errorOccurred = true
            self.loading = false
        }
    }
}
```

Takeaways - the Good Parts

- Straightforward implementation
- No advanced prior ML knowledge required
- No server infrastructures
- Personalized user experience

Takeaways - the Not So Good Parts

- App size grows with each new model creation
- Training takes time, good background concept is needed
- No support on other platforms such as Android
- No insights to user data (for improvements)
- CreateML is not available on the iOS simulator
  - harder development

Using in Production

- Around 5% using the custom filter feature
  - Harder for regular users
  - Premium feature
- Most users using predefined filters created with CreateML
- Focus on experiences with no/little user effort

Other Use-Cases

- Recommendations
  - E.g. music, based on previous selections
  - Classifiers and regressors
  - More details at https://martinmitrevski.com/2021/07/11/ml-recommendation-app-with-create-ml-on-ios-15/
- Sound classification
  - E.g. recognize voices for meeting notes, music learning app or authentication
- Hand pose classification
  - accessibility, games

Relevant Links

- WWDC session: https://developer.apple.com/videos/play/wwdc2021/10037/
- Dimi app: https://apps.apple.com/us/app/dimi-nft-art-maker/id1585569333#?platform=iphone
- Blog: https://martinmitrevski.com/2021/07/11/ml-recommendation-app-with-create-ml-on-ios-15/

Thank You!

- Questions?