FEBRUARY 21, 2020

On-device Machine Learning

Radu Dan – MobOS 2020

Contents

Core ML: great experience using machine learning

Integrate machine learning models into your app.

On-Device Training

Core ML models can be bundled into our own apps and helps us do interesting stuff, such as look for an object real-time or search an object in photos.

Advanced Neural Networks

With Core ML 3, we now have the latest models, neural networks designed to understand images, video, sound, and other rich media.

Computer Vision

You can enhance your app and add amazing features, such as face detection and recognition, image classification, object tracking, improved landmark detection, and many more.

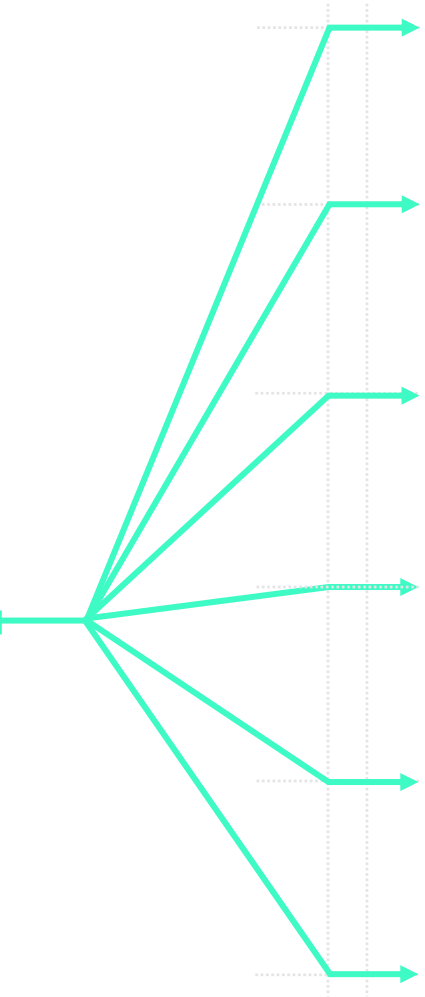
Natural Language

Perform tasks like language and script identification and tokenization. Integrate custom NLP models with Create ML.

Speech

Speech recognition for 10 languages.

Core ML provides a unified representation for all models.



Vision
ANALYZING IMAGES

Natural Language
PROCESSING TEXT

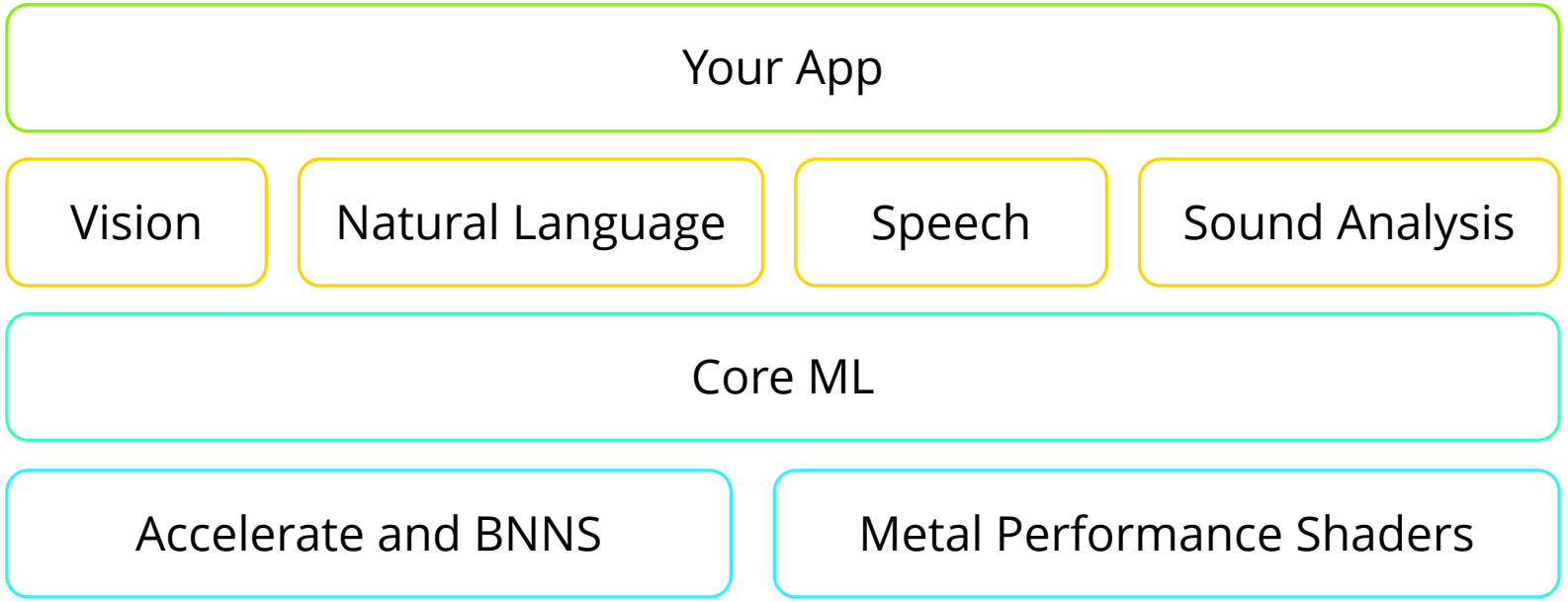
Speech
CONVERTING AUDIO TO TEXT

Sound Analysis
IDENTIFYING SOUNDS IN AUDIO

Accelerate and BNNS
NEURAL NETWORKS

Metal Performance Shaders
OPTIMIZE GRAPHICS

Core ML Layers



Create ML

Training models

Create ML is a simple app, used to create and train ML models, preparing them, making them as ready-to-go to use with Core ML.

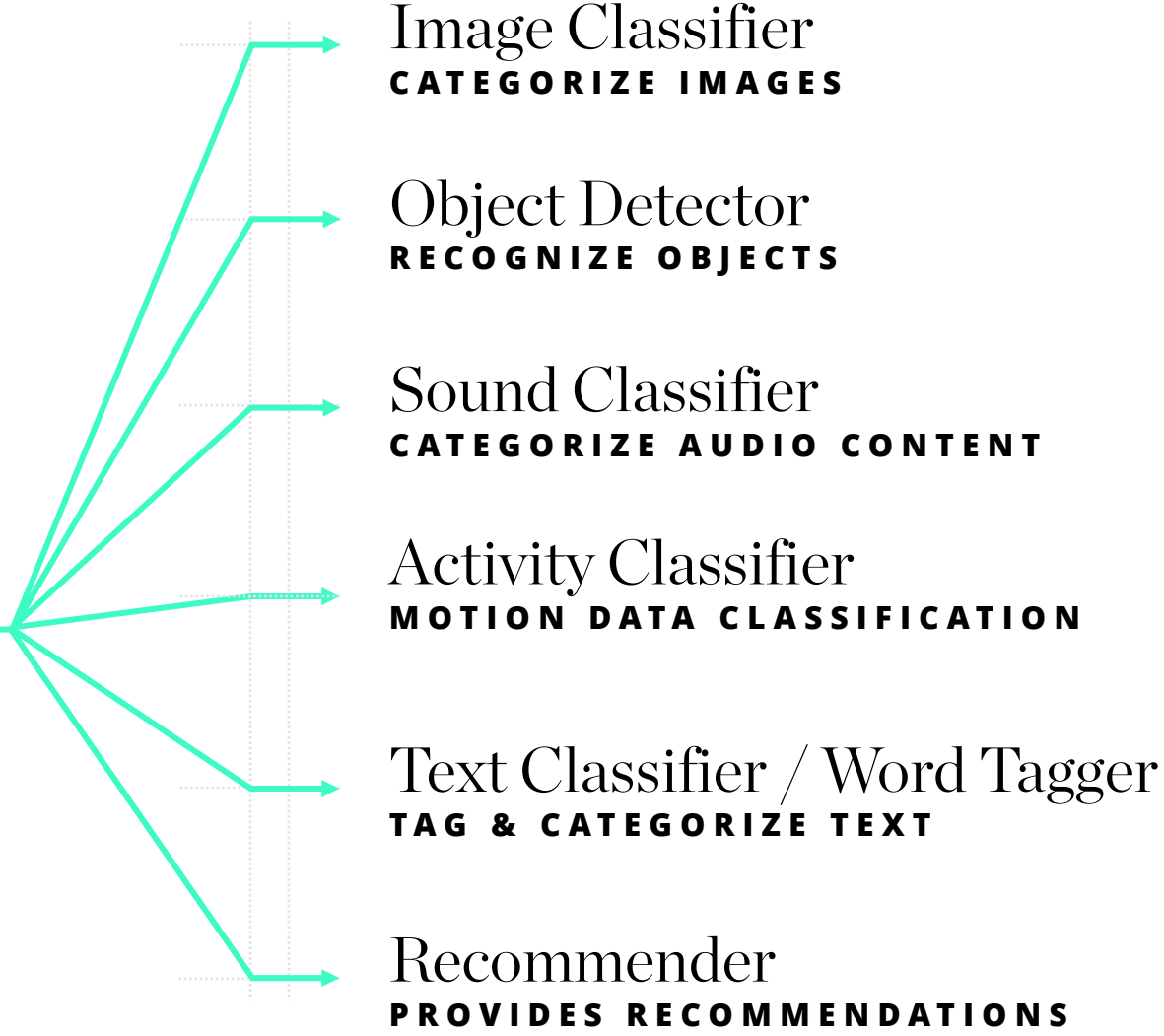
Classifiers

Supports a lot of classifiers: image, object detector, sound, activity, text, word, tabular, recommender.

Multi-model training

You can train multiple models using different datasets, all at the same time.

Create ML is a fun app that makes ML models training very easy.



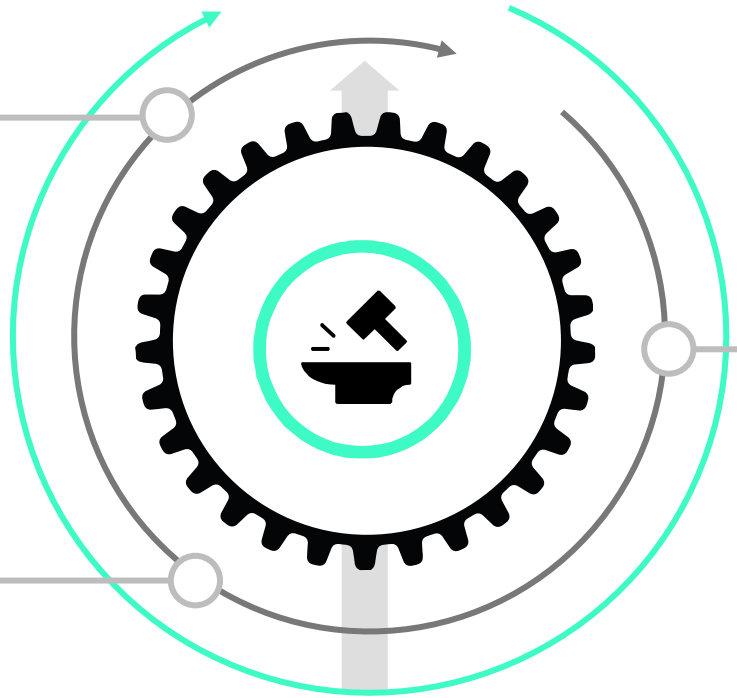
Create ML

Image from: https://developer.apple.com/documentation/createml

Core ML vs Create ML

Core ML

Create ML - Apple's way for model training (other tools: Turi Create)



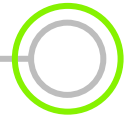
Create ML and Core ML were designed to work together

Create ML - quickly creates models

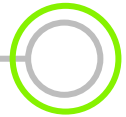
Create ML

Core ML vs Create ML

On-device training has nothing to do with Create ML



With Create ML - gather / collect a lot of user data and build a model



Core ML on-device personalization is regularly used to use minimum data

Demo

Image Classifier

On-device Training

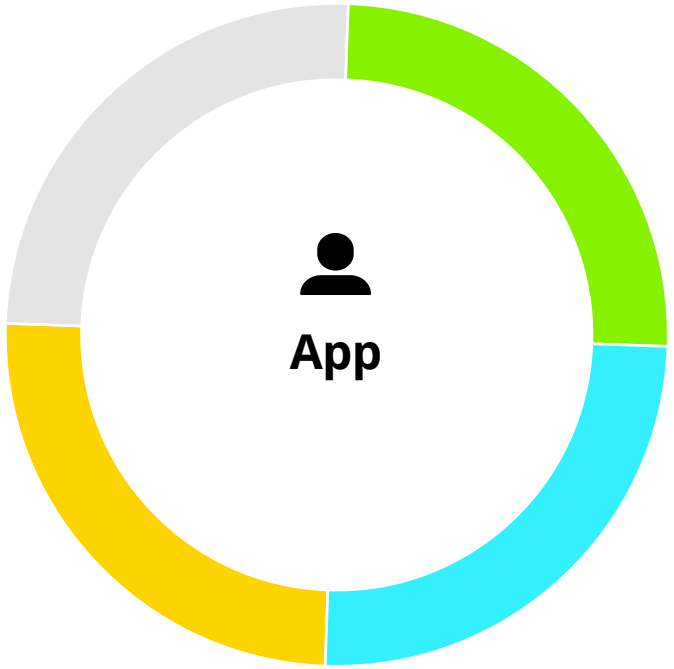
Every machine model at first doesn't know anything

Adapt the model to each user's specific usage

Provide a model which is trained before, and understands data in general terms

Training from zero: do it when the data is unique to each user

On-device Training



Create models specific to user needs

No expensive servers required

Data is protected, it lives on device

It’s fast, and provides accurate results

On-device Training

Finding the right amount of data that is enough to improve the model

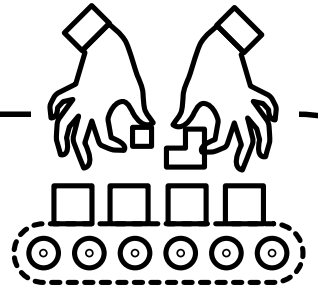
Trained model: do it when the data has similar patterns

Distributed Learning: Looks at each model of all your app users

Distributed Learning: The results are aggregated and combined into a new model

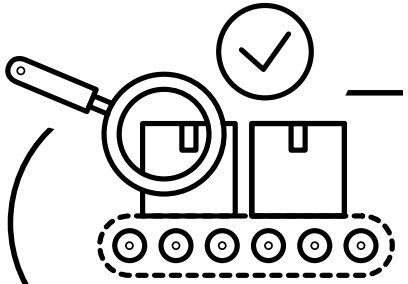
How to do it

BASE MODEL



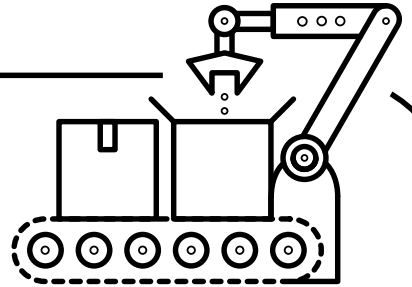
- Create a base model
- Configured to allow updating

IMPORT IN APP



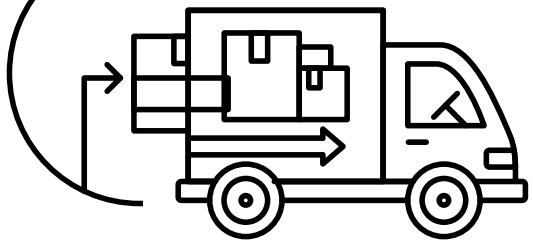
- Model doesn't know about your specific data
- Neural network & k-NN classifier

TRAIN MODEL



- Takes pictures
- Uses local photos

UPDATE MODEL



- MLFeatureProvider
- MLBatchProvider
- MLUpdateTask

Limitations

Support only for k-NN classifiers and some Neural Networks

No support for linear regression or decision trees

Limited support for Neural Networks: only convolution and fully-connected layers

Limitations

Loss functions: Categorical Cross Entropy and MSE

Optimizers: Plain SGD and Adam

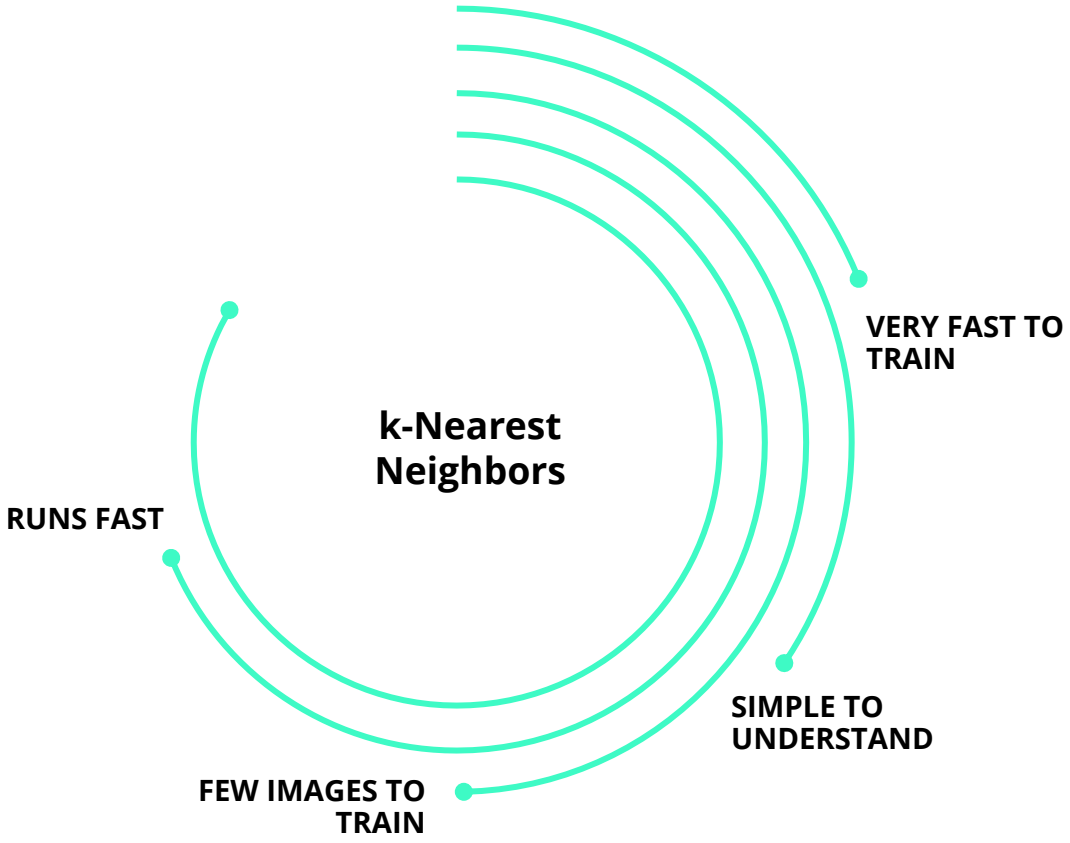
Core ML is not yet a replacement for Tensor Flow and PyTorch

Defining our base model

Create a customizable classifier using k-Nearest Neighbors.

Given a **feature vector**, finds the closest *k* training examples.

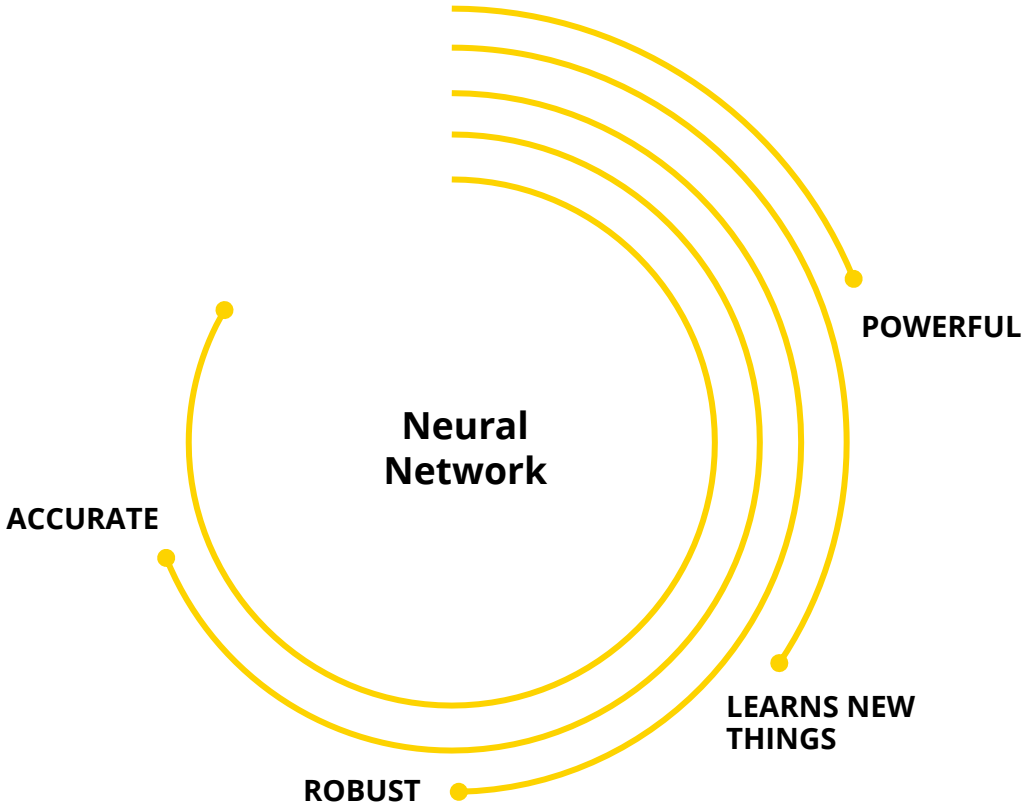
Offers a great advantage: it can add new data / classes, instead of neural networks which come with a fixed number of classes.



Defining our base model

Having a large set of data, you show the model one by one the training examples.

1. **Forward pass** – model makes predictions
2. The loss function computes how wrong are those predictions
3. Using back propagation, the optimizer slightly adjusts the learned parameters (**backward pass**)
4. Next time, the model's predictions will be slightly better



Demo

On-device personalization

Core ML 3 – expanded

MLFeatureValue

Each input has exactly one MLFeatureValue.

Each output has exactly one MLFeatureValue.

Is a wrapper for the data.

MLFeatureDescription

Contains the name of the feature and the type of data.

It can also contain constraints, specifying the limitations of the model's input and output features.

MLImageConstraints

The width, height, and pixel format constraints of an image feature.

An image is a collection of pixels represented by CVPixelBuffer.

It defines the feature's limitations in the images within MLFeatureValue.

MLFeatureProvider

A protocol that represents a collection of values for an input or output.

Mlmodels implements this protocol.

MLDictionaryFeatureProvider is a wrapper that holds data as Dictionary.

MLBatchProvider

A protocol that contains multiple feature providers.

Multiple MLBatchProvider objects can form a MLArrayBatchProviders.

MLUpdateTask

Updates the ML model on a user's device.

Starts a task given the model URL, the training data, some optional configuration (run on GPU or CPU).

```
Button(action: {
    let trainer = Trainer(car: .porsche)
    trainer.updateModel {
        self.showingPorscheAlert = true
    }
}) {
    Text("🏎")
        .padding()
        .clipShape(Circle())
        .font(.largeTitle)
        .background(Color.white)
        .cornerRadius(40)
}.alert(isPresented: $showingPorscheAlert) {
    Alert(title: Text("Training complete"), message: Text("Your porsche 🏎 classifier has been
        updated. "), dismissButton: .default(Text("OK")))
}
```

```swift
struct Trainer {

    private let modelHandler: MLModelHandler
    private let car: Car

    private var featureBatchProvider: MLBatchProvider {
        var featureProviders: [MLFeatureProvider] = []
        let imageSet = ImageSetFactory.makeImageSet(for: car)

        imageSet.images.forEach { image in
            let inputValue = modelHandler.featureValue(usingImage: image)
            let outputValue = MLFeatureValue(string: car.rawValue)

            let dataPointFeatures: [String: MLFeatureValue] = [modelHandler.inputName: inputValue,
                                                               modelHandler.outputName: outputValue]

            if let provider = try? MLDictionaryFeatureProvider(dictionary: dataPointFeatures) {
                featureProviders.append(provider)
            }
        }

        return MLArrayBatchProvider(array: featureProviders)
    }
}
```

```swift
extension MLModelHandler {
    var modelDescription: MLModelDescription { model.modelDescription }
    var inputName: String { "image" }
    var outputName: String { "label" }

    func featureValue(usingImage image: UIImage) -> MLFeatureValue {
        let imageInputDescription = modelDescription.inputDescriptionsByName[inputName]!
        let constraint = imageInputDescription.imageConstraint!
        let imageFeatureValue = try? MLFeatureValue(cgImage: image.cgImage!, constraint: constraint)

        return imageFeatureValue!
    }
}
```

```swift
struct Trainer {

    func updateModel(completion: (() -> Void)? = nil) {
        DispatchQueue.global(qos: .userInitiated).async {
            ModelUpdater.updateWith(trainingData: self.featureBatchProvider) {
                completion?()
            }
        }
    }

}
```

```swift
extension MLModelHandler {
    static func updateModel(at url: URL,
                            with trainingData: MLBatchProvider,
                            completionHandler: @escaping (MLUpdateContext) -> Void) {

        // Create an Update Task.
        guard let updateTask = try? MLUpdateTask(forModelAt: url,
                                                 trainingData: trainingData,
                                                 configuration: nil,
                                                 completionHandler: completionHandler)
            else {
                print("Could't create an MLUpdateTask.")
                return
        }

        updateTask.resume()
    }
}
```

```
static func updateWith(trainingData: MLBatchProvider,
                       completionHandler: @escaping () -> Void) {
    let usingUpdatedModel = updatedCarClassifier != nil
    let currentModelURL = usingUpdatedModel ? updatedModelURL : defaultModelURL

    /// The closure an MLUpdateTask calls when it finishes updating the model.
    func updateModelCompletionHandler(updateContext: MLUpdateContext) {
        // Save the updated model to the file system.
        saveUpdatedModel(updateContext)

        // Begin using the saved updated model.
        loadUpdatedModel()

        // Inform the calling View Controller when the update is complete
        DispatchQueue.main.async { completionHandler() }
    }

    ToyCarsUpdateableModel.updateModel(at: currentModelURL,
                                       with: trainingData,
                                       completionHandler: updateModelCompletionHandler)
}
```

Prediction

Name	Type	Description
Inputs		
image	Image (Color 227 x 227)	Input image
Outputs		
label	String	Predicted label
labelProbability	Dictionary (String → Double)	Probabilities for each possible label

Update

Name	Type	Description
Inputs		
image	Image (Color 227 x 227)	Example image
label	String	True label

Parameters

Name	Type	Default	Description
Model			
kNNClassifier			
numberOfNeighbors	Int64	3	Number of neighbors to use for prediction

Wrapping it up

ON-DEVICE PERSONALIZATION

Make extremely powerful apps, tailored to what users want.

Reduce the costs of expensive servers / cloud infrastructure by using their devices

Protect their data – the user's data never leaves the phone

Highlights

- Neural Networks with 100-layer types
- Maximum performance and efficiency
- Models created with ease
- Fast learning
- Runs fast
- Accurate results

Thank you!

 @radude89

 radude89

 radude89.com