



AI on the Edge

 Core ML

Arthur Emídio - arthur@loopkey.com.br
Desenvolvedor de Software @ LoopKey

Por que levar AI para a ponta?

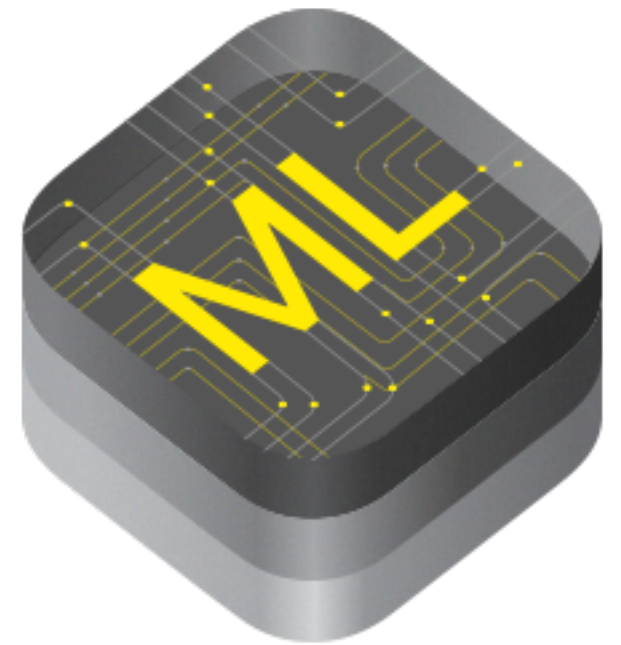
- Grande parte do processamento de aplicações de inteligência artificial rodam na nuvem.
 - IBM Watson;
 - Google Cloud Machine Learning Engine;
 - Microsoft Azure Cognitive Services;
 - Amazon Rekognition / Polly / Lex.
- Problemas principais:
 - Latência ou indisponibilidade;
 - Privacidade;

Como levar AI para dispositivos móveis?

- Avanços em hardware:
 - Apple A11 Bionic e sua Neural Engine;
- Desenvolvimento de *frameworks* de ML que suportam dispositivos móveis:
 - Caffe2;
 - TensorFlow Mobile / Lite;
 - **Core ML.**

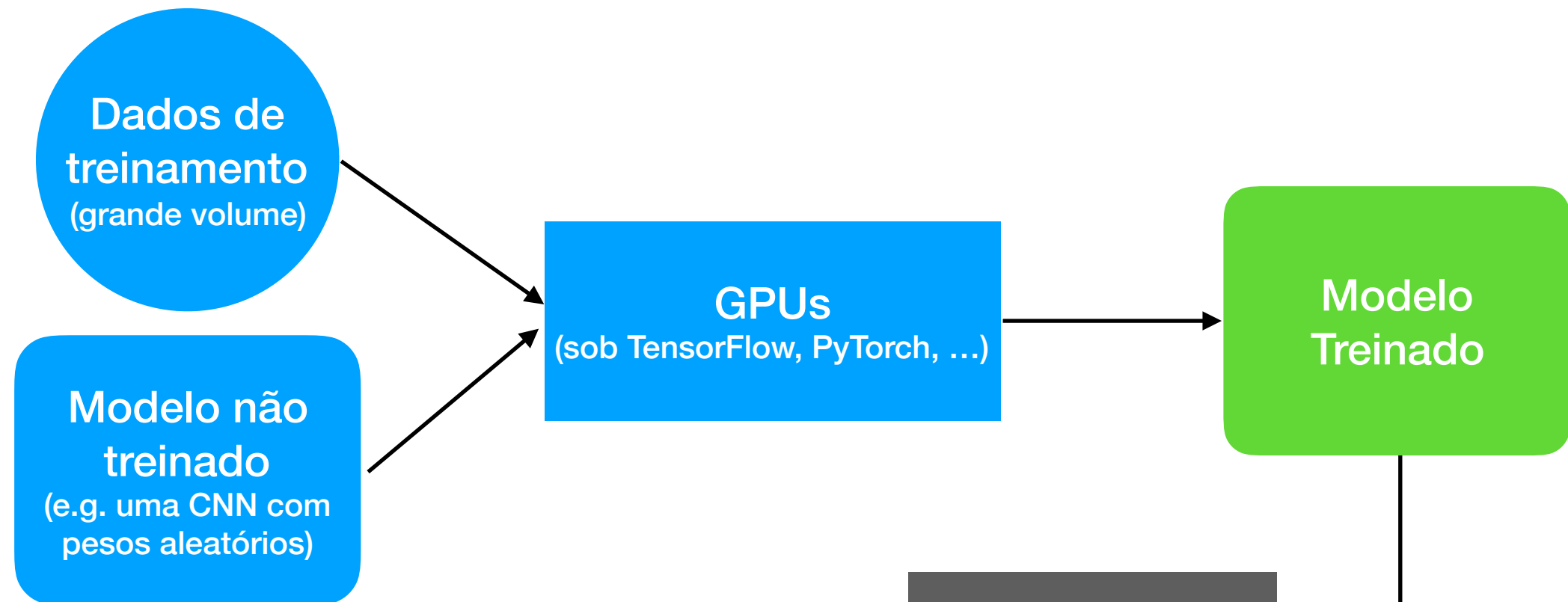
🍏 Core ML - O que é?

- Framework de aprendizado de máquina para executar modelos offline em dispositivos da Apple, incluindo iPhone, iPad, Apple Watch, Apple TV e Mac.
- Exemplos de aplicações: detecção facial, reconhecimento de imagens, análise de sentimento, transferência de estilo, tradução...
- Benefícios: simplicidade de implementação, performance, redução de custos, disponibilidade.

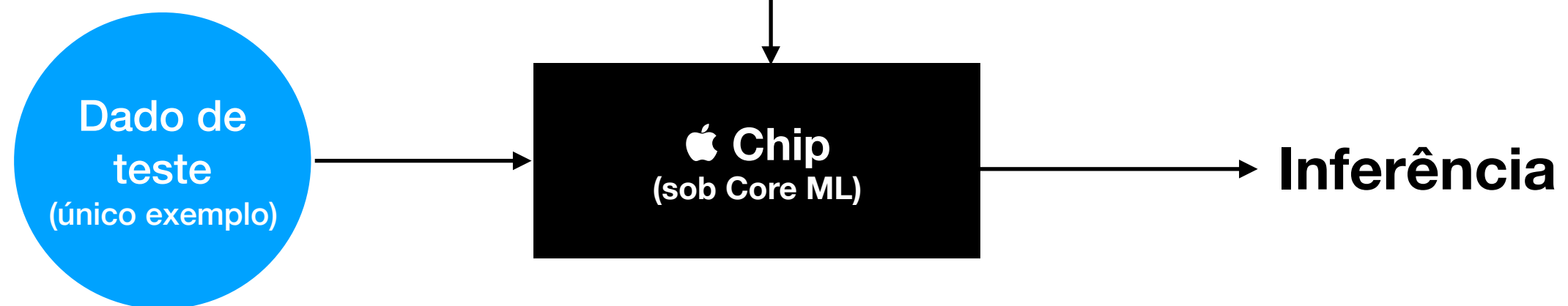


Como funciona?

1. Etapa de Treinamento

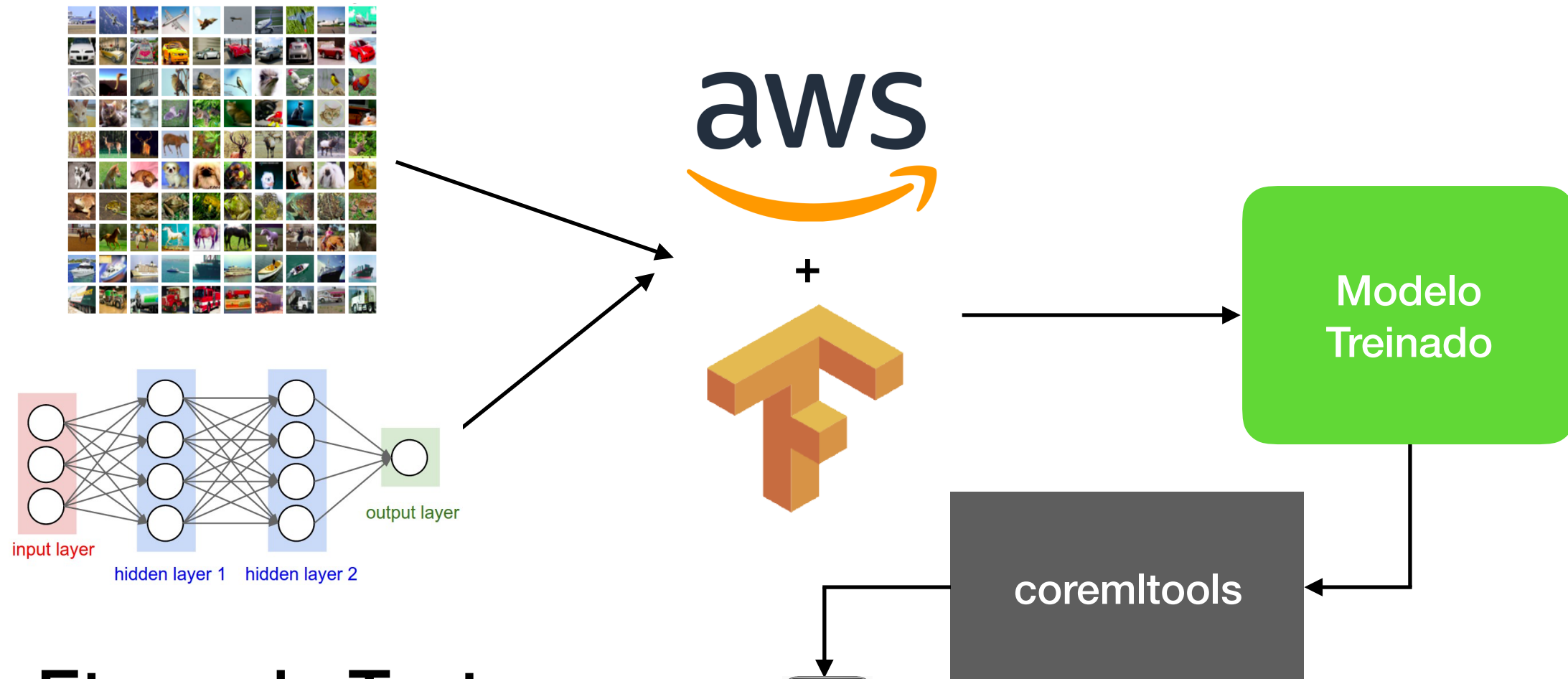


2. Etapa de Teste



Como funciona?

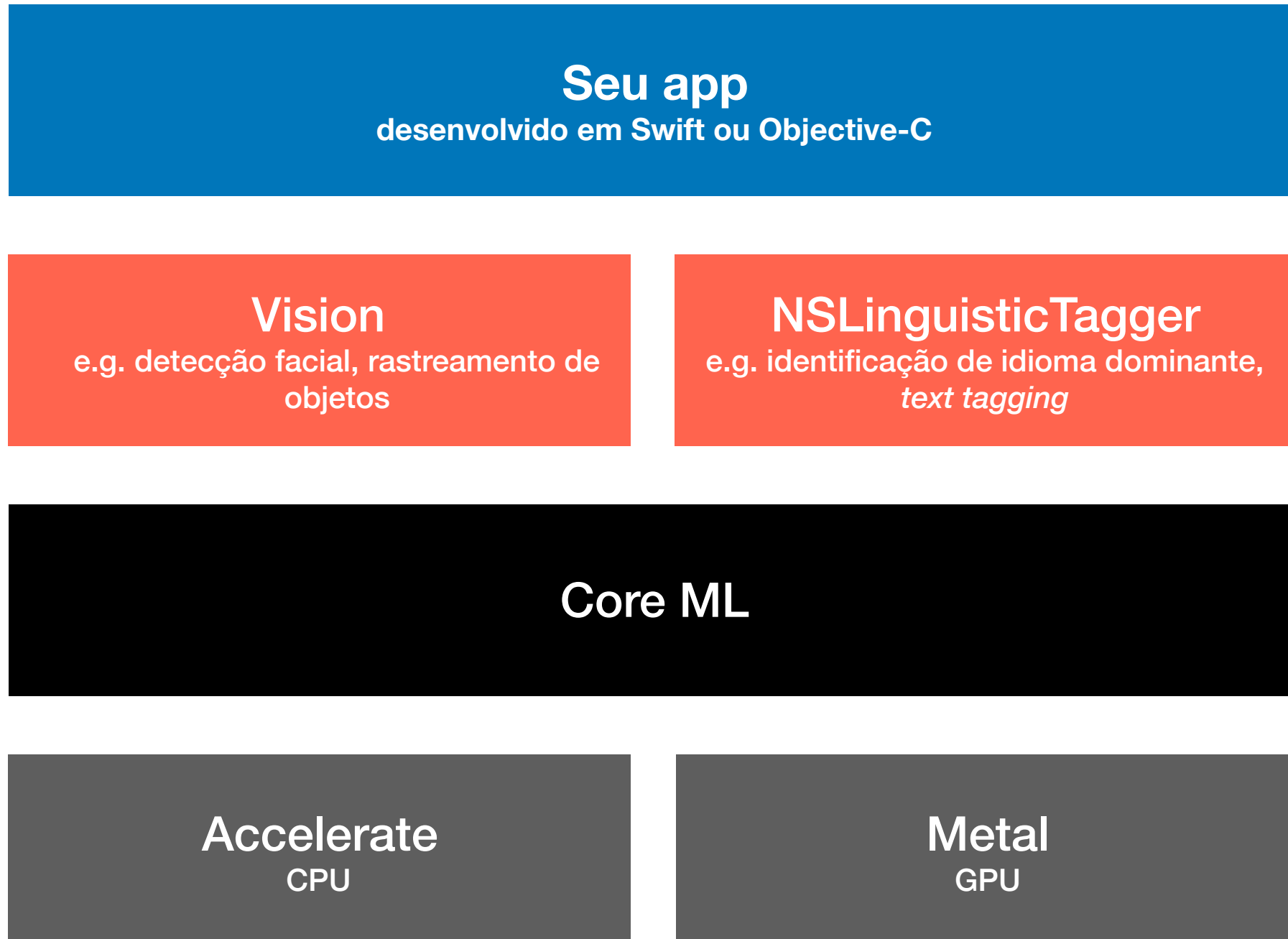
1. Etapa de Treinamento



2. Etapa de Teste



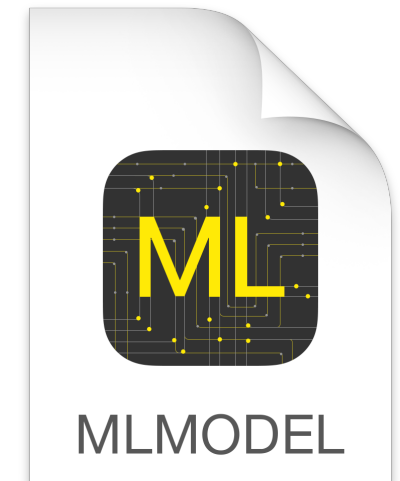
Frameworks disponíveis



Tipos de Modelos Suportados

- Redes Neurais Artificiais: Feedforward, CNNs e RNNs:
 - Caffe;
 - Keras com TensorFlow.
- Árvores: random forests e árvores de decisão:
 - scikit-learn;
 - XGBoost.
- SVM: regressão escalar e classificação multi-classe:
 - scikit-learn;
 - LIBSVM.
- Modelos lineares: regressão linear e regressão logística:
 - scikit-learn.

Arquivo de modelo do Core ML



- Extensão: **.mlmodel**
- Como obter modelos no formato Core ML:
 - Página de desenvolvimento da Apple: modelos de reconhecimento de imagem;
 - Awesome Core ML Models (repositório do GitHub): modelos de reconhecimento de imagem, *style transfer*, análise de texto, e mais.
 - Convertendo modelos de outros *frameworks*.

Convertendo modelos

- A conversão é realizada através do pacote Python `coremltools`;

```
pip install coremltools
```

- Apple fornece conversão nativa para os frameworks: Caffe, Keras (com TensorFlow), scikit-learn, XGBoost, LIBSVM.
- Conversores desenvolvidos pela comunidade: Torch7, MXNet.

Convertendo modelos

Exemplo de conversão:

```
import coremltools
coreml_model = coremltools.converters.caffe.convert('my_caffe_model.caffemodel',
                                                    image_input_names = 'image',
                                                    class_labels = ['cat', 'dog'])
coreml_model.save('my_model.mlmodel')
```

Mais informações sobre conversão na documentação oficial do coremltools: <https://apple.github.io/coremltools/>


Importando um modelo

- Para importar um modelo do Core ML para um projeto, basta arrastar o arquivo .mlmodel sobre o Xcode.

▼ Machine Learning Model

Name	SqueezeNet
Type	Neural Network Classifier
Size	5 MB
Author	Forrest N. Iandola and Song Han and Matthew W. Moskewicz and Khalid Ashraf and William J.
Description	Detects the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, parameters https://github.com/DeepScale/SqueezeNet
License	BSD License. More information available at https://github.com/DeepScale/SqueezeNet/blob/master/LICENSE

▼ Model Class

 SqueezeNet ➔
Automatically generated Swift model class

▼ Model Evaluation Parameters

Name	Type	Description
▼ inputs		
image	Image (Color 227 x 227)	Input image to be classified
▼ outputs		
classLabelProbs	Dictionary (String → Double)	Probability of each category
classLabel	String	Most likely image category

Importando um modelo

- Para importar um modelo do Core ML para um projeto, basta arrastar o arquivo .mlmodel sobre o Xcode.

▼ Machine Learning Model

Name SqueezeNet
Type Neural Network Classifier
Size 5 MB
Author Forrest N. Iandola and Song Han and Matthew W. Moskewicz and Khalid Ashraf and William J.
Description Detects the dominant objects present in an image from a set of 1000 categories such as trees, parameters <https://github.com/DeepScale/SqueezeNet>
License BSD License. More information available at <https://github.com/DeepScale/SqueezeNet/blob/master/LICENSE>

▼ Model Class

 SqueezeNet ➔
Automatically generated Swift model class



▼ Model Evaluation Parameters

Name	Type	Description
▼ inputs		
image	Image (Color 227 x 227)	Input image to be classified
▼ outputs		
classLabelProbs	Dictionary (String → Double)	Probability of each category
classLabel	String	Most likely image category



Utilizando a classe do modelo

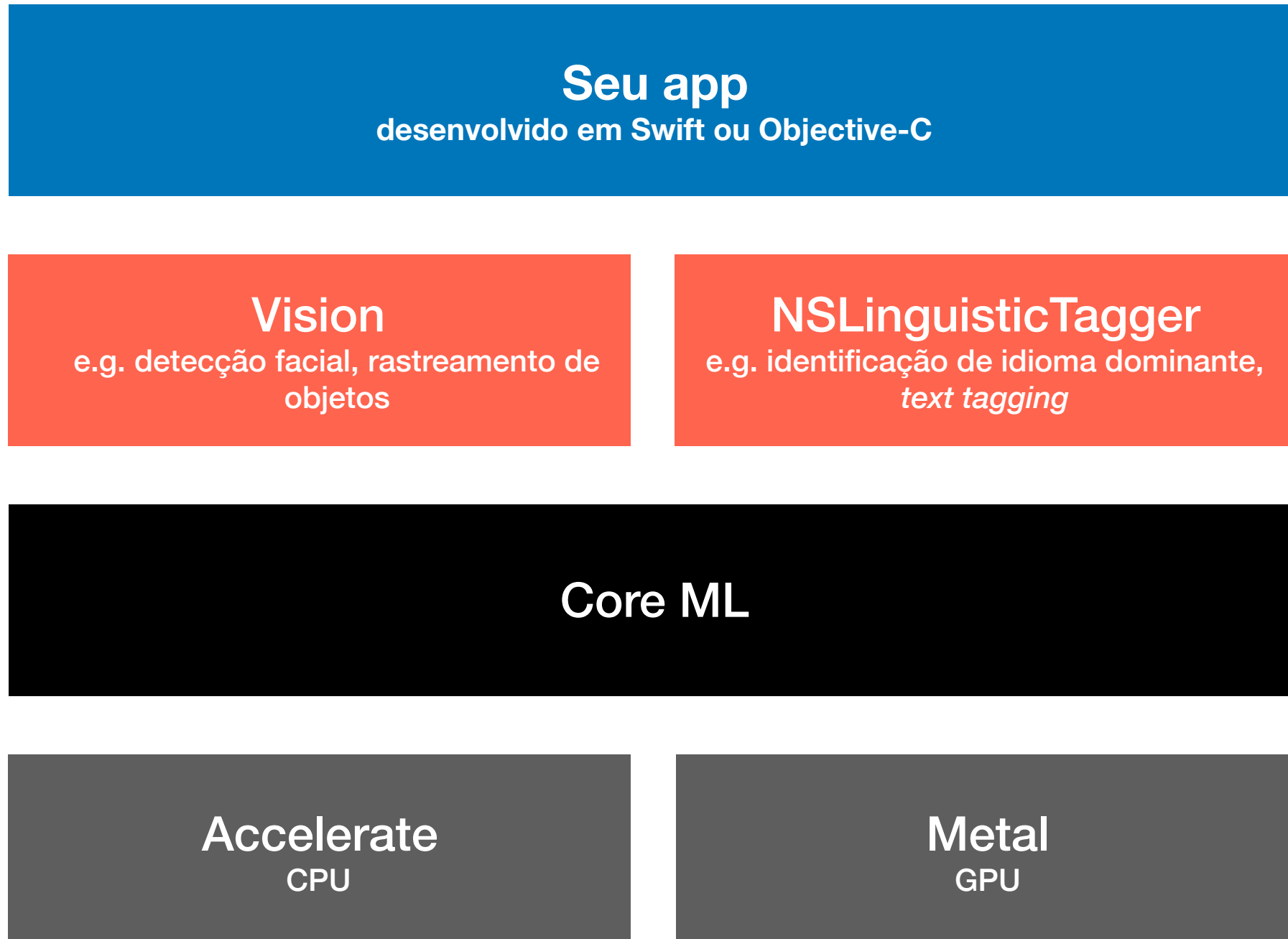
```
func classify(inputImage: CVPixelBuffer) throws
{
    let prediction = try SqueezeNet().prediction(image: inputImage)
    let predictionLabel = prediction.classLabel
    let predictionProb = prediction.classLabelProbs[predictionLabel]!
    print("Classifico essa imagem como \(predictionLabel) com
          \(predictionProb * 100.0)% de confiança.")
}
```

- Basta instanciar um objeto da classe gerada e chamar o método `prediction()` com os dados de entrada;
- As saídas podem ser obtidas a partir do objeto retornado por `prediction()`.

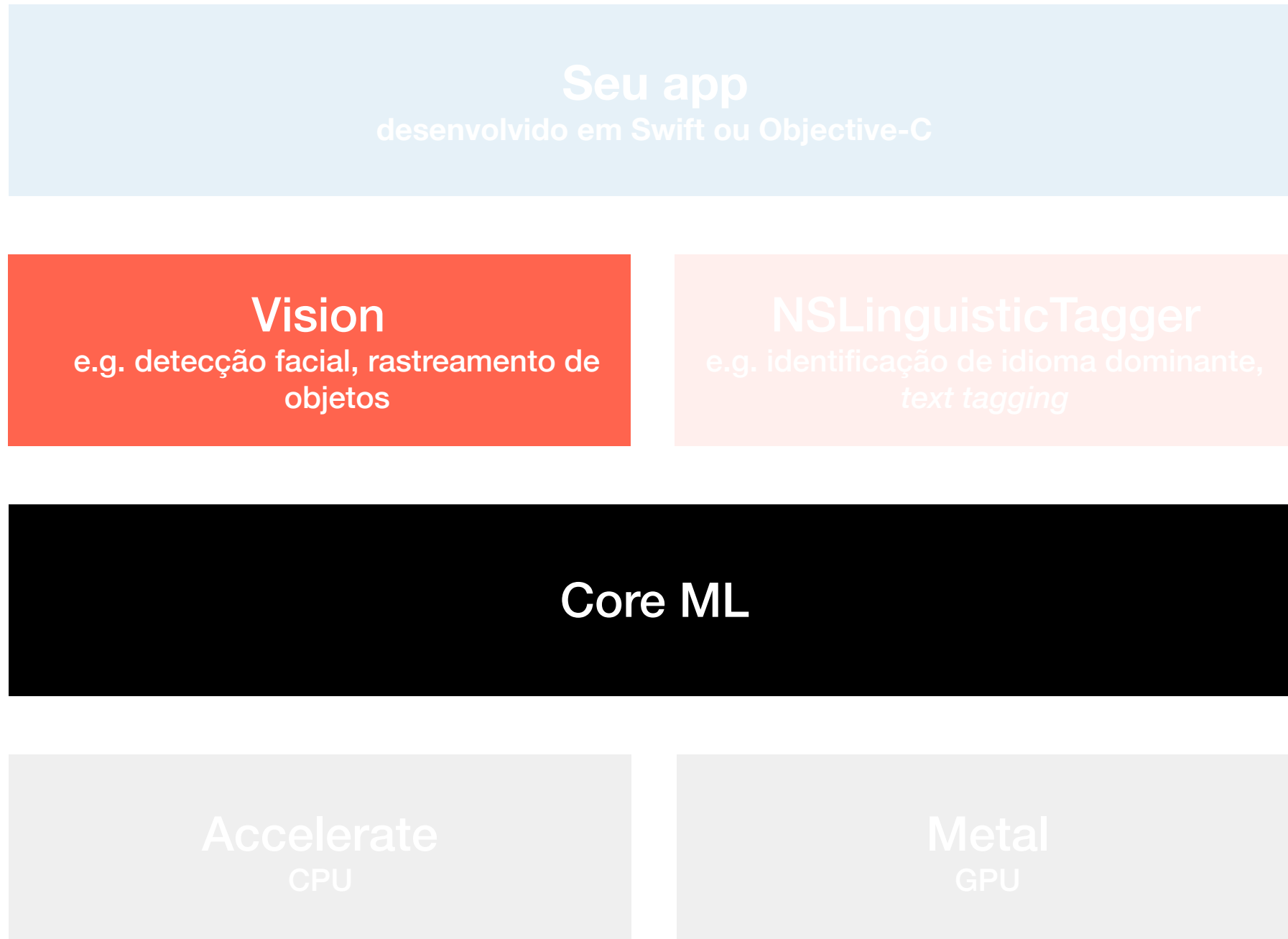
Demo

Análise de Sentimentos

Relembrando...



Frameworks para aplicações de Visão Computacional



Vision

- Framework de visão computacional da Apple utilizado para:
 - Identificação de faces e de características da face;
 - Detecção de códigos de barra;
 - Distorção de imagens;
 - Detecção de textos em imagens;
 - Rastreamento de objetos em vídeo;
 - **Fornecer entrada para modelos Core ML.**
- Disponível em: iOS 11.0, macOS 10.13 e tvOS 11.0;
- Para funcionalidades mais avançadas: OpenCV.

Vision: API

- Todas as operações são baseadas em **requests**. Exemplos:
 - `VNDetectFaceRectanglesRequest`;
 - `VNDetectBarcodesRequest`.
- Requests são executados através de **handlers** do tipo `VNImageRequestHandler`;
- Requests são instanciados com um bloco de conclusão do tipo `(VNRequest, Error?) -> Void`
 - Através do objeto `VNRequest` é possível obter os resultados através da propriedade **results**: **[Any]**?
 - `results` deve ser convertido para array de alguma subclasse de `VNObservation` (e.g. `VNFaceObservation`).

Demo

Detecção de Faces

Vision + Core ML

- Capaz de realizar as transformações necessárias na imagem de entrada do modelo;
- **Request:** `VNCoreMLRequest`
- **Observation:**
 - `VNClassificationObservation` para modelos de classificação;
 - `VNPixelBufferObservation` para modelos que produzem imagem como saída;
 - `VNCoreMLFeatureValueObservation` para outros casos.

Vision + Core ML: Observations

- **VNClassificationObservation**
 - **(identifier: String)**: rótulo dado como saída pelo modelo classificador;
 - **(confidence: Float)**: confiança acerca da classificação, em uma escala [0.0, 1.0];
- **VNPixelBufferObservation**
 - **(pixelBuffer: CVPixelBuffer)**: imagem de saída.
- **VNCoreMLFeatureValueObservation**
 - **(featureValue: MLFeatureValue)**: valor genérico dado como saída. Pode guardar valor de diversos tipos, como: Int64, Double, String, MLMultiArray.

Demo

Classificador de Objetos

Conclusões

- Sugestões para Android:
 - TensorFlow Mobile / Lite;
 - Caffe2;
 - OpenCV.
- Recomendações:
 - Repositório do Webinar
 - [Core ML Basics \(WWDC 2017\)](#)
 - [Core ML in Depth \(WWDC 2017\)](#)
 - [Core ML + Vision \(WWDC 2017\)](#)
 - [Relatório técnico de como a Apple implementa reconhecimento facial](#)
 - [Tutorial de Core ML + Vision](#)
 - [Tutorial de Core ML + NLP](#)





Obrigado! 😊

Perguntas? Sugestões?

Arthur Emídio - arthur@loopkey.com.br
Desenvolvedor de Software @ LoopKey