DESENVOLVIMENTO DE UM APLICATIVO iOS UTILIZANDO OS FRAMEWORKS CORE ML E VISION PARA APLICAÇÃO DE TÉCNICAS DE DETECÇÃO DE OBJETOS PREDOMINANTES PRESENTES EM IMAGENS

Alan Henrique Pégoli
alanpegoli@icloud.com
Prof. Drª. Silvia Maria Farani Costa
silvia.costa01@fatec.sp.gov.br
Fatec Carapicuiba

RESUMO: Este artigo trata do desenvolvimento de um aplicativo móvel para sistemas operacionais de dispositivos iPhone (iOS) que faz uso de dois recém-lançados frameworks da Apple Inc., citados como Core ML e Vision, para aplicar técnicas de análise de imagem de alto desempenho e de visão computacional para identificar objetos em imagens. O aplicativo contém uma interface gráfica pela qual o usuário é capaz de interagir com o sistema. A interface se compõe pela tela de câmera, onde o usuário pode enxergar o mundo real e receber respostas responsivas do sistema sobre o objeto predominante que está à frente da câmera
Palavras-chave: Visão Computacional. Aprendizado de Máquina. Aplicativo Mobile

ABSTRACT: This article is about the development of a mobile application for iPhone devices operating systems (iOS) that makes use of two recently released frameworks by Apple Inc., namely Core ML and Vision, to apply high-performance image analysis techniques to identify imaged objects. The application contains a graphical interface through which the user is able to interact with the system. The interface is composed by the camera screen, where the user can see the real world and receive responsive responses from the system about the predominant object that is in front of the camera.
Keywords: Computer Vision. Machine Learning. Mobile Application.

1 INTRODUÇÃO

1.1 APRENDIZADO DE MÁQUINA

Aprendizado de máquina é uma área da ciência da computação que surgiu do estudo do reconhecimento de padrões e de teorias de inteligência artificial e aprendizados automáticos e que consiste na implementação de *softwares* que simulam o aprendizado autônomo (HOSCH, 2009).

Samuel (1959) definiu o aprendizado de máquina como a habilidade de computadores aprenderem e/ou executarem uma tarefa para o qual não foram explicitamente programados.

Dentre as diversas abordagens e aplicações, encontra-se o uso de redes neurais artificiais e algoritmos genéticos. Um método muito divulgado é chamado de aprendizagem supervisionada, que consiste em uma rede neural que recebe interações já conhecidas de um agente externo (geralmente humano) de modo que com o passar das interações a rede se torne cada vez mais ajustada e assertiva (INTERNATIONAL CONFERENCE ON COMPUTATIONAL CREATIVITY, 2016). Por exemplo, um modelo que tenha sido treinado nos preços históricos das casas de uma região pode ser capaz de prever o preço de uma casa, dado o número de quartos e banheiros.

1.2 VISÃO COMPUTACIONAL

Visão computacional é uma área da ciência da computação que se dedica a aplicar conceitos e métodos de aprendizagem de máquina para conseguir que sistemas de softwares e hardwares, por meio de representações 2D do mundo físico, possam compreender uma cena 3D real. (DAVIES, 2015). Em suma, trata-se de uma réplica computacional do processo de visão humana (embora o potencial dessa tecnologia vá muito além), podendo incorporar outras tecnologias, como sensores capazes de ver no escuro ou através deparedes.

Dentro de visão computacional, tem-se o que é chamado de reconhecimento de imagem, que trata a respeito de análise pixel a pixel em busca de padrões que formam objetos ou símbolos. (DAVIES, 2015).

O objetivo deste trabalho é aplicar técnica de aprendizado de máquina a dispositivos iPhone por meio dos frameworks Core ML e Vision e, assim, introduzir e difundir tecnologias recém-chegadas ao mercado para fomentar o aprendizado, a reflexão e o debate sobre o desenvolvimento mobile e as tendências em torno disso no meio acadêmico.

2 METODOLOGIA

Para a execução deste trabalho foi usado um MacBook Air de 13'' do começo de 2015, com processador de 1,6GHz Intel Core i5, memória de 8 GB 1600 MHz DDR3, gráficos Intel HD Graphics 6000 1536 MB e disco SSD de 128GB, rodando o macOS High Sierra Versão 10.13.1; e um iPhone 8 Plus de capacidade de 64 GB, tela Retina HD, processador A11 Bionic com arquitetura de 64 bits, processador neural e coprocessador de movimento M11 integrado, câmera de 12 MP com lentes grande-ocular (de abertura: f/1.8) e teleobjetiva (de abertura: f/2.8), zoom óptico e zoom digital até 10x, rodando o iOS 11.1.2.

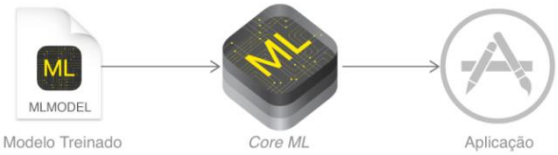
Como ferramenta de trabalho foi usado o ambiente de desenvolvimento fornecido pela Apple Inc.para desenvolvimento de aplicações para o seu universo, o Xcode, em sua versão 9.1. A linguagem trabalhada foi Swift, em sua versão 4.0. Também foram usados dois novos frameworks apresentados este ano pela Apple Inc., juntamente com a apresentação da nova versão da linguagem, que são a base para trabalhar com aprendizado de máquina intrinsicamente no iOS, e que tornam a aplicação de aprendizado de máquina em aplicativos que implementam visão computacional menoscomplexa: o Core ML e o Vision.

2.1 CORE ML

O Core ML é um framework recém-lançado pela Apple Inc. que foi desenvolvido para transformar a aplicação de aprendizado de máquina em dispositivos móveis. Ele permite que modelos treinados sejam integrados e trabalhem intrinsicamente nos aplicativos, conforme Figura 1, o que promove a privacidade do usuário final, a responsividade e disponibilidade das funcionalidades do aplicativo, mesmo offline, reduzindo custos de transferência de dados (para o usuário final) e anula custos com

servidores dedicados (para o desenvolvedor), uma vez que desloca o processamento direto para o dispositivo móvel. (APPLE INC., 2017).

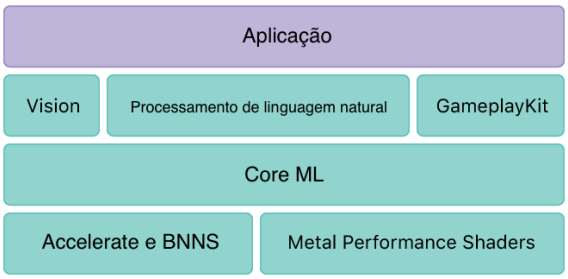
Figura 1 – Framework Core ML



Fonte: Reprodução adaptada de Apple Inc.

O Core ML é a base para outros frameworks e funcionalidades direcionados ao aprendizado de máquina dentro do ecossistema do iOS. Ele suporta o frameworkVision para análise de imagens, o Foundation para processamento de linguagem natural (por exemplo, a classe NSLinguisticTagger) e o GameplayKit para avaliar árvores de decisão aprendidas. Por sua vez, o Core ML se constrói em cima de outros frameworks ainda mais primitivos e de baixo nível do iOS, como Accelerate e BNNS, bem como o Metal Performance Shaders, conforme demonstrado na Figura 2.

Figura 2 – Camadas do Core ML



Fonte: Reprodução adaptada de Apple Inc.

O Core ML suporta uma variedade de modelos de aprendizado de máquinas, incluindo redes neurais, conjuntos de árvores, modelos lineares generalizados, dentre outros. O Core ML requer um formato específico de modelo (arquivo com extensão.mlmodel). A Apple Inc. fornece alguns desses modelos populares de código aberto que já estão no formato do modelo Core ML. No entanto, modelos e dados de treinamento que não estão no formato do modelo Core ML podem ser convertidos através de ferramentas do próprio framework (APPLE INC., 2017).

3 REFERENCIAL TEÓRICO

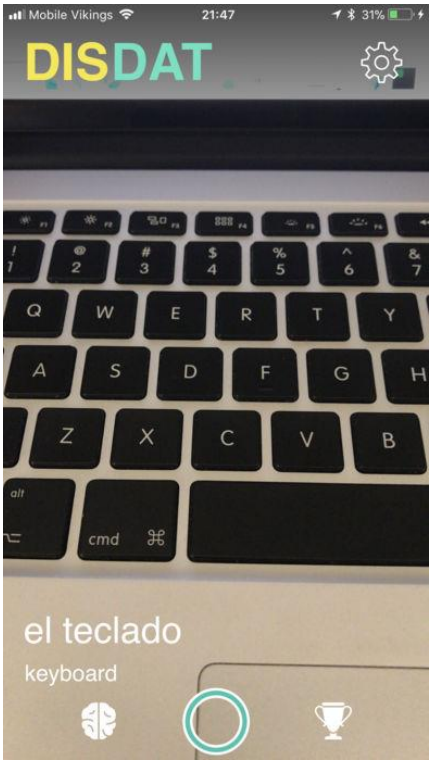
Foram encontrados alguns aplicativos já fazendo uso dos frameworks recém-lançados pela Apple Inc. logo apósa abertura oficial da nova AppStore, loja de e-commerce de aplicativos para dispositivos Apple em geral, que ocorreu no dia 19 de setembro de 2017. Ressaltam-se aqui dois deles, que inclusive usam o mesmo modelo Core ML que o usado neste trabalho.

3.1 DISDAT

DISDAT é um aplicativo educativo (Fig. 3), desenvolvido pela Balloon Inc., que faz uso do Core ML, do Vision e do NLP (Natural LanguageProcessing, em tradução livre, Processamento de Linguagem Natural), para ensinar seus usuários novas línguas por meio da observação do mundo (BALLOON INC., 2017).

Segundo seus criadores, Balloon Inc. (2017), a ideia é reproduzir o processo pelo qual uma criança aprende os objetos do mundo e “gamificar” isso num aplicativo. O usuário deve apontar a câmera do dispositivo para o objeto esperar pelo retorno, que é devido ao Core ML, imediato.

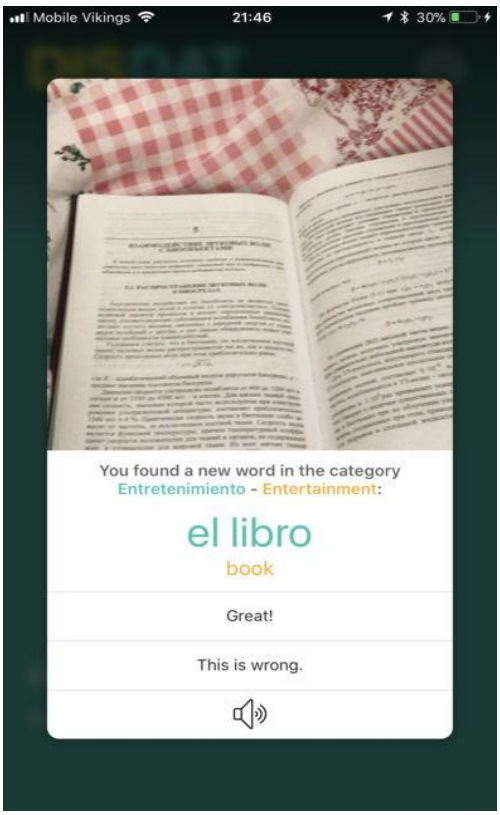
Figura 3 – Captura da tela principal do aplicativo DISDAT



Fonte: Reprodução de Balloon Inc.

São 120 palavras divididas em 22 categorias, em 6 línguas (inglês, espanhol, alemão, francês, italiano e russo). A medida em que o usuário encontra os objetos recém-classificados, o aplicativo exibe um cartão, como um alerta, com a escrita e a pronúncia do objeto na língua desejada (Fig. 4), e também conta o progresso do usuário naquele idioma conforme o usuário for encontrando outras novas palavras.

Figura 4 – Captura do cartão do aplicativo DISDAT



Fonte: Reprodução de Balloon Inc.

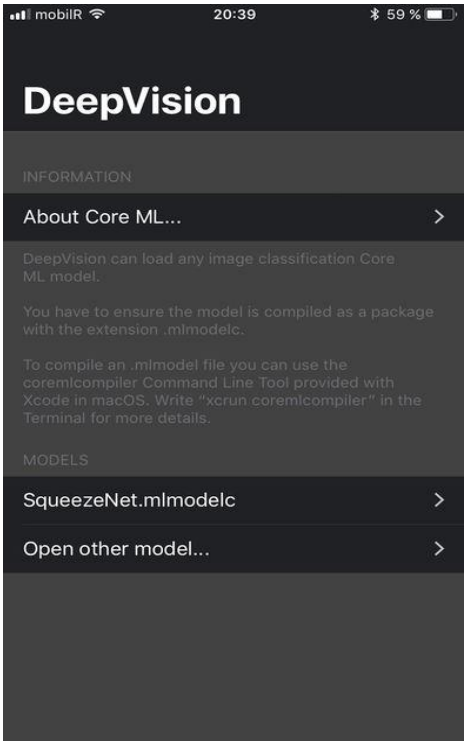
3.2 *DEEPVISION*

DeepVision é um aplicativo comercial, desenvolvido por Pedro Jose Pereira Vieito, tanto para iPhone como para Mac. Ele fornece uma plataforma para testes de modelos Core ML de classificação de imagem (VIEITO, 2017).

O aplicativo funciona basicamente da mesma forma, usando Core ML e Vision, com a ressalva de que seu modelo Core ML não é fixo. Ele tem um modelo predefinido, que é o SqueezeNet, mas seu grande diferencial é que também permite que usuários incluam seus próprios modelos Core ML de classificação de imagem para testarem (Fig. 5). Assim, ele é capaz de executar o modelo em tempo real na câmera de vídeo do dispositivo, seja um iPhone ou um Mac, e trazer os resultados do

modelo, bem como suas informações descritivas, como criadores, licença e descrição.

Figura 5 – Captura da tela do DeepVision para iPhone



Fonte: Reprodução de Apple Inc.

4 RESULTADOS E DISCUSSÃO

4.1 IMPLEMENTANDO O CORE ML

Neste trabalho foi usado um modelo fornecido pela Apple Inc., o Inception v3, que foi originalmente convertido de um modelo de classificação de imagem treinado para Keras, uma biblioteca de rede neural de código aberto escrita em Python. Este modelo detecta os objetos dominantes presentes em uma imagem de um conjunto de 1000categorias, como árvores, animais, alimentos, veículos, pessoas, dentre outros. A margem de erro da publicação original é de 5,6%.

Na Figura 6 é demonstrado uma captura da tela de informações no Xcode (IDE para desenvolvimento iOS) sobre o modelo Core ML adicionado ao projeto da aplicação, incluindo o tipo de modelo, sua entrada e saídas esperadas. A entrada para o modelo é uma imagem colorida de tamanho 299p X 299p. A produção do modelo são a categoria do objeto dominante na imagem e a probabilidade da categoria.

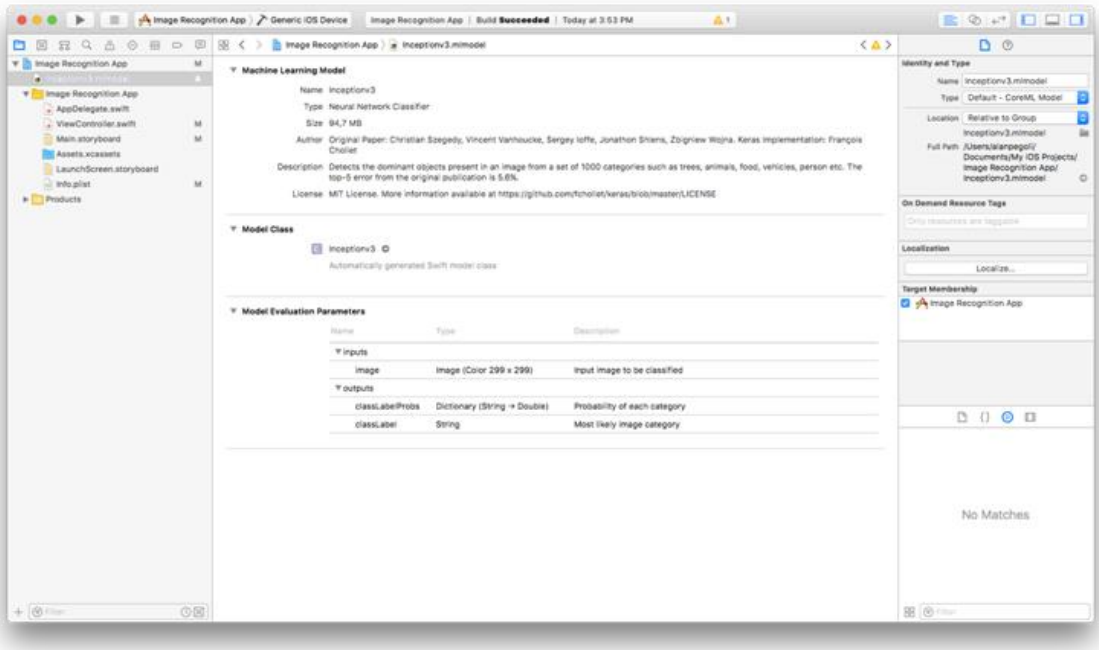
O Xcode usa essas informações sobre as entradas e saídas do modelo para gerar automaticamente uma interface programática personalizada para o modelo, que é usada para interagir com o próprio modelo no código. Para o modelo “Inceptionv3.mlmodel”, o Xcode cria classes para representar o próprio modelo (Inceptionv3), as entradas do modelo (Inceptionv3Input) e as saídas (Inceptionv3Output). O Snippet 1 traz um exemplo de instanciação de um objeto da classe Inceptionv3:

Snippet 1 – Instanciando um objeto da classe Inceptionv3

```
varmodel:Inceptionv3!
model = Inceptionv3()
```

A classe Inceptionv3 tem um método de predição gerado (prediction(image: CVPixelBuffer)) que é usado para prever a categoria e sua probabilidade do objeto dominante numacerta imagem, que é usada como valor de entrada do modelo. O resultado desse método é uma instância Inceptionv3Output, no código chama-se o resultado prediction, conforme Snippet 2:

Figura 6 – Captura de tela das informações do modelo Core ML



Fonte: Elaboração própria

Snippet 2 – Recuperando o resultado do método de predição

```
guard let prediction = try?
model.prediction(image: pixelBuffer!)
else {
      return
```

Para acessar a informação contida no resultado que traz a categoria do objeto dominante na imagem, usa-se a propriedade classLabel da classe Inceptionv3Output, conforme Snippet 3:

Snippet 3 – Utilizando o resultado do método de predição

```
self.classifierLabel.text= "I think
this is a \(prediction.classLabel)."
```

A predição gerada pode resultar um erro. O tipo mais comum de erro encontrado ao trabalhar com o Core ML ocorre quando o tipo de dado de entrada passado para o método não corresponde ao tipo de entrada que o modelo espera − no caso, uma imagem no formato errado poderia resultar num erro.

Quaisquer tipos de desajustes são capturados em tempo de compilação, e o aplicativo gera um erro fatal se algo der errado.

O Xcode compila o modelo Core ML em um recurso para otimizar a execução no dispositivo. Essa representação otimizada do modelo está incluída no pacote do aplicativo e é usada para fazer predições enquanto o aplicativo está sendo executado no dispositivo.

4.2 IMPLEMENTANDO O VISION

Neste trabalho o Vision foi utilizado para realizar todo o tratamento de imagens, do modo natural e humanamente familiar para a forma correta de entrada do modelo Core ML.

O fluxo de trabalho padrão do Vision é criar um modelo, fazer uma ou mais requisições e, em seguida, criar e executar um

completionhandler da requisição, ou, em tradução livre,um manipulador de conclusão da requisição, que serve justamente para trabalhar com serviços assíncronos no Swift.

No Snippet 4, tem-se o exemplo de um desempacotamento de modelo Core ML para um modelo Vision. Uma vez que o desempacotamento pode retornar um erro, usa-se o comando try:

Snippet 4 – Desempacotando o modelo Core ML para um modelo Vision

```
let model = try! VNCoreMLModel(for:
Inceptionv3().model)
```

VNCoreMLRequest é uma solicitação de análise de imagem que usa um modelo Core ML para processar. Seu completionhandler recebe os objetos de solicitação e erro. O resultado dessa solicitação no caso deste trabalho é um VNClassificationObservation, o que o Vision retorna quando o modelo Core ML é um classificador, ao invés de um preditor ou um processador de imagem. E o Inception v3 é um classificador porque ele prediz apenas um recurso: a categoria do objeto predominante na imagem.

A VNClassificationObservation tem duas propriedades: identifier (uma String que classifica a categoria do objeto) e confidence (um número decimal entre 0 e 1, é a probabilidade da classificação estar correta). Neste trabalho, considerou-se válido apenas aqueles resultados com confidence acima de 50%.

Foi usado um modelo treinado para detectar os objetos dominantes presentes em uma imagem de um conjunto de 1000 categorias, como árvores, animais, alimentos, veículos, pessoas, dentre outros. O próximo subcapítulo trata a respeito deste modelo.

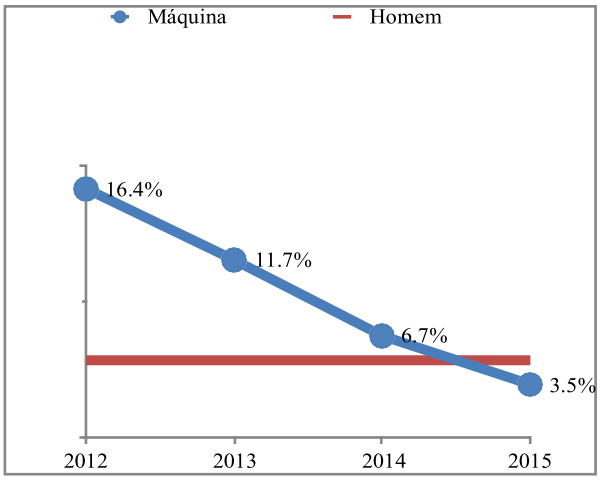
4.3 INCEPTION V3

Neste trabalho fez-se uso da rede neural Inception v3, pois ela está classificada como uma rede neural convolucional, que é um tipo muito usado para reconhecimento de imagens, uma vez que seus neurônios individuais são organizados de modo a responder regiões de sobreposição no campo visual (SZEGEDY, 2015).

No desafio de reconhecimento de imagem da IMAGEnet de 2014, o GoogleInc. se apresentou com uma abordagem de rede neural convolucional para reconhecimento de objetos em imagens, que teve taxa de erro de 6,6%, quase metade da taxa de 11,7% do ano anterior. No entanto, um humano foi capaz de cumprir o mesmo desafio com uma taxa de erro de apenas 5,1%.

Já em 2015, a Microsoft Inc. anunciou que havia conseguido bater o recorde humano com uma taxa de erro de apenas 4,94%. Alguns meses depois, novamente no desafio da IMAGEnet, a Microsoft Inc. quebrou seu próprio recorde com uma taxa de erro de 3,5% (Fig. 7). Foi então que a visão computacional se tornou novamente popular e o tópico central em diversas discussões sobre inteligência artificial,aprendizado de máquina e deeplearning. (RUSSAKOVSKY et al., 2014).

Figura 7 – Teste IMAGEnet



Fonte: IMAGEnetInc.

Este trabalho colocou à disposição informações e conhecimentos acerca do que tem sido encarado como a próxima tendência em termos de tecnologia para desenvolvimento de aplicações para dispositivos iPhone. Com os frameworksCore ML e Vision, bem como NLP, é possível afirmar que a Apple Inc. está colocando uma pedra fundamental para as inúmeras aplicações que regerão os próximos anos de desenvolvimento em sua plataforma. (DAVIES et al., 2016).

Grande ponto a favor dessa nova tecnologia é que ela foi projetada para ser executada em dispositivos existentes, o que significa que os usuários finais não terão que atualizar para um hardware mais caro e especializado para aproveitar os benefícios do aprendizado de máquinas em seus dispositivos. Isso dá uma abertura ainda maior para a sua adoção.

O *software* desenvolvido em conjunto com o trabalho trouxe uma breve nuance da capacidade dos frameworks desenvolvidos e recém-lançados pela Apple Inc., e ajuda a tanger as possibilidades de aplicações que podem vir a implementar aprendizados de máquina de uma forma mais segura, barata, offline e muito menos complexa em termos de desenvolvimento.

O modelo de rede neural convolucional utilizado neste trabalho apresenta uma taxa de erro top-5 de 5.06% no conjunto de teste da IMAGEnet. (SZEGEDY, 2015). Há, porém, relato de que com um conjunto de três resíduos e um Inception-v4, é possível alcançar 3.08% de erro top-5 neste mesmo conjunto de tese. (INTERNATIONAL CONFERENCE ON COMPUTATIONAL CREATIVITY, 2016).

5 CONSIDERAÇÕES FINAIS

A implementação do Core ML se mostrou relativamente simples e eficaz. Ela permitiu que o foco no desenvolvimento estivesse todo voltado para o manejo dos dados e das informações no “pré e pós- processamento”. A implementação do Vision,por sua vez, serviu para atuar justamente nesse primeiro quesito, onde haveria bastante complexidade no tratamento de imagens.

Com o uso de ambas as plataformas foi possível desenvolver um aplicativo totalmente funcional capaz de detectar objetos predominantes presentes à frente da câmera e imprimir na tela informações sobre esses objetos.

REFERÊNCIAS

APPLE INC. (Cupertino) (Org.). Core ML Apple Developer Documentation: Integrate machine learning models into your app. 2017. Disponível em: <https://

developer.apple.com/documentation/coreml>. Acesso em: 30 set. 2017.

APPLE INC. (Cupertino) (Org.). Vision Apple Developer Documentation: Apply high-performance image analysis and computer vision techniques to identify faces, detect features, and classify scenes in images and video. 2017. Disponível em: <https:// developer.apple.com/documentation/vision>. Acessoem: 30 set. 2017.

BALLOON INC. (Org.). DISDAT: iOS app to learn a language using machine learning. 2017. Disponível em: <https:// disdat.ai>. Acessoem: 30 set. 2017.

DAVIES, Roy. Machine Vision: Theory, Algorithms, Practicalities. 3. ed. Amsterdã: Elsevier, 2015. 934 p.

DAVIES, Sam; RAMES, Jeff; TURTON, Rich. iOS 10 by Tutorials: Learning the new iOS APIs with Swift 3. Virgínia: RazewareLlc, 2017. 324 p.

HOSCH, William L..Machine Learning. Chicago: Encyclopædia Britannica, Inc., 2009. Disponívelem: <https://global.britannica.com/ technology/machine-learning>. Acessoem: 30 set. 2017.

INTERNATIONAL CONFERENCE ON COMPUTATIONAL CREATIVITY, 7., 2016, Paris. Proceedings Of The Seventh International Conference On Computational Creativity. Paris: Sony CSL, 2016.403 p.Disponível em: <http:// www.computationalcreativity.net/iccc2016/wp-content/uploads/2016/08/Proceedings_ICCC16.pdf>. Acessoem: 30 set. 2017.

RUSSAKOVSKY, Olga et al. ImageNet Large Scale Visual Recognition Challenge.3. ed. Carolina do Norte: Ijcv, 2014. 43 p. Disponível em: <https://arxiv.org/pdf/ 1409.0575.pdf>. Acessoem: 30 set. 2017.

SAMUEL, Arthur Lee. Some Studies in Machine Learning Using the Game of Checkers. Ibm Journal Of Research And Development. EUA, p. 535-554. mar. 1959. Disponívelem: <https://www.cs.virginia.edu/ ~evans/greatworks/samuel1959.pdf>. Acessoem: 30 set. 2017.

SZEGEDY, Christian. Rethinking the Inception Architecture for ComputerVision. 2015. 10 f. Cornell University, Ithaca, 2015. Disponível em: <https://arxiv.org/abs/ 1512.00567?context=cs>. Acesso em: 30 set. 2017.

VIEITO, Pedro José Pereira. PVIEITO. Disponível em: <https://pvieito.com>. Acesso em: 30 set. 2017.