CENTRO UNIVERSITÁRIO SAGRADO CORAÇÃO

RODOLPHO BRUNO DOS SANTOS NOGUEIRA

TECNOLOGIAS DE INTELIGÊNCIA ARTIFICIAL NATIVAS DO ANDROID E IOS

BAURU
2020

RODOLPHO BRUNO DOS SANTOS NOGUEIRA

TECNOLOGIAS DE INTELIGÊNCIA ARTIFICIAL NATIVAS DO ANDROID E IOS

Trabalho de Conclusão de Curso de especialização apresentado como parte dos requisitos para obtenção do título de Especialista em Engenharia de Software - Centro Universitário Sagrado Coração.

Orientador: Prof. Dr. Elvio Gilberto da Silva.

Coorientação: Prof. Esp. Thiago Macedo Silvestre.

BAURU

2020

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD

Nogueira, Rodolpho Bruno Dos Santos

N778t

Tecnologias de inteligência artificial nativas do Android e IOS / Rodolpho Bruno Dos Santos Nogueira. -- 2020.

19f. : il.

Orientador: Prof. Dr. Elvio Gilberto da Silva

Coorientador: Prof. Esp. Thiago Macedo Silvestre

Monografia (Especialização em Engenharia de Software) - Centro Universitário Sagrado Coração - UNISAGRADO - Bauru - SP

1. Machine Learning. 2. Inteligência Artificial. 3. Desenvolvimento Nativo. I. Silva, Elvio Gilberto da. II. Silvestre, Thiago Macedo. III. Título.

Elaborado por Lidyane Silva Lima - CRB-8/9602

RODOLPHO BRUNO DOS SANTOS NOGUEIRA

TECNOLOGIAS DE INTELIGÊNCIA ARTIFICIAL NATIVAS NO ANDROID E IOS

Trabalho de Conclusão de Curso de especialização apresentado como parte dos requisitos para obtenção do título de Especialista em Engenharia de Software - Centro Universitário Sagrado Coração.

Aprovado em: _____/_____/_____.

Banca examinadora:

_____
Prof. Dr. Elvio Gilberto da Silva
Centro Universitário Sagrado Coração

_____
Prof. Esp. Thiago Macedo Silvestre
Centro Universitário Sagrado Coração

_____
Prof. Esp. Paulo Vinícius Ferreira
Centro Universitário Sagrado Coração

_____
Prof. M.e Enedy Allan Rodrigues Cordeiro
Centro Universitário Sagrado Coração

SUMÁRIO

TECNOLOGIAS DE INTELIGÊNCIA ARTIFICIAL NATIVAS NO ANDROID E IOS

RODOLPHO BRUNO DOS SANTOS NOGUEIRA[1]

[1]Elvio Gilberto da Silva. [1]Thiago Macedo Silvestre. [1]Paulo Vinícius Ferreira. [1]Enedy Allan Rodrigues Cordeiro.

[1]Centro Universitário Sagrado Coração – UNISAGRADO – egilberto@uol.com.br; thiagomsilvestre@gmail.com; pvf.ferreira@gmail.com; enedy_allan@yahoo.com.br;

RESUMO

As tecnologias mais populares existentes no mercado referente a inteligência artificial utilizam as linguagens Python e R acompanhadas de um conjunto de bibliotecas. Essas linguagens são concebidas de forma diferente em comparação a arquitetura dos dispositivos móveis, sendo necessário que os aplicativos móveis que necessitem utilizar algum recurso de *machine learning*, passaria a necessitar de alguma API externa, e caso o dispositivo perca definitivamente a conectividade com a internet o aplicativo não funcionaria corretamente. É importante que estas plataformas móveis contenham em seu ecossistema a tecnologia necessária para o processamento e uso do *machine learning* (aprendizado de máquina) e inteligência artificial, suprindo o dentro das possibilidades o consumo de recursos externos.

Palavras-chave: *Machine Learning*. Inteligência Artificial. Desenvolvimento Nativo.

ABSTRACT

The most popular technologies on the market regarding artificial intelligence use the Python and R languages accompanied by a set of libraries. These languages are conceived differently in comparison to the architecture of mobile devices, making it necessary that mobile applications that need to use some machine learning resource, would need some external API, and if the device permanently loses connectivity to the internet o application would not work properly. It is important that these mobile platforms contain in their ecosystem the necessary technology for the processing and use of machine learning (artificial learning) and artificial intelligence, supplying the consumption of external resources as much as possible.

Keywords**:** Machine Learning. Artificial Intelligence. Native Development.

1 INTRODUÇÃO

Segundo Pointer (2018) dentre as tecnologias mais populares existentes no mercado referente a inteligência artificial estão linguagens como Python e R, acompanhadas de um conjunto de bibliotecas primárias para o processamento de *machine learning, deep learning,* mineração de dados, ou seja, estão na vanguarda da pesquisa e construção de inteligência artificial. Essas linguagens são concebidas de forma diferente em comparação a arquitetura dos dispositivos móveis, e caso os aplicativos móveis que necessitem utilizar algum recurso de *machine learning* (aprendizado de máquina), necessitam de alguma API externa para esse processamento podendo ser utilizados serviços como o Firebase. Mas essa forma de uso pode se tornar inadequada para aplicativos de tempo real e que necessitem baixa latência, ou seja, uma conexão de internet problemática pode comprometer o modelo e consequentemente o funcionamento da aplicação (FIREBASE, 2019).

É importante que estas plataformas móveis contenham em seu ecossistema a tecnologia necessária para o processamento e uso do *machine learning* e inteligência artificial de forma nativa e este estudo vem fazer um levantamento das soluções criadas por Apple e Google, pesquisando as tecnologias e ferramentas disponíveis para a implementação de *machine learning* e inteligência artificial para as referidas plataformas, como parte do estudo serão implementados exemplos para cada plataforma de forma nativa, tecnologias como TensorFlow Lite (TENSORFLOW LITE EXEMPLOS, 2019) e o projeto UpdatableDrawingClassifier (APPLE DEVELOPERS CORE ML, 2019) disponibilizados por Android e Apple respectivamente e serão descritos os resultados obtidos do funcionamento desses aplicativos.

2 PRINCIPAIS PLATAFORMAS MOBILE

Para este estudo foram utilizadas as plataformas Android e IOS que segundo a Statcounter Globalstats (2020) em um levantamento entre janeiro de 2019 a janeiro de 2020 possuem 74,3% e 24,76% respectivamente do mercado mundial de smartphones.

> “[...] Quando se desenvolve uma aplicação móvel, é importante considerar o sistema operacional do dispositivo móvel. O sistema operacional afeta a linguagem, ferramentas e tecnologias que você utiliza para desenvolver a sua aplicação móvel, bem como sua capacidade de dar suporte a manutenção da aplicação [...]”, (VALENTINO; SCHNEIDER; ROBBIE; 2005; p. 53, tradução nossa).

Conforme Iverson e Eirman (2003) o desenvolvimento para plataformas móveis é de certa forma semelhante ao desenvolvimento para outras plataformas mais tradicionais, mas forma de execução das aplicações é diferente. Dessa forma é importante obter o conhecimento sobre cada plataforma.

2.1 ANDROID

Segundo a Developers Android (2019), o Android é uma pilha de software com base em Linux de código aberto, criada para diversos dispositivos. Seus componentes são distribuídos da seguinte forma (DEVELOPERS ANDROID, 2019):

a) Aplicativos de Fábrica: e-mail, SMS, calendário, navegador entre outros;
b) Java Api: São API's implementadas na linguagem Java para criar aplicativos Android simplificando a reutilização de componentes e serviços modulares e principais;
c) Bibliotecas Nativas C e C++: Bibliotecas nativas implementadas nessas linguagens para complementar serviços principais do sistema como o ART e HAL;
d) Android Runtime: Cada aplicativo executa o próprio processo com uma instância própria. O ART executa várias máquinas virtuais em dispositivos de baixa memória em arquivos DEX que otimiza o consumo de memória;
e) *Hardware Abstraction Layer* (HAL): A HAL são bibliotecas que implementam uma *interface* para manipular determinados componentes de *hardware*, como o módulo de câmera ou bluetooth;
f) Linux Kernel: Utiliza recursos de segurança do Linux e permite que os fabricantes dos dispositivos desenvolvem *drivers* de *hardware* padronizados.

2.2 IOS

Segundo a Apple INC (2019) o IOS *(Iphone Operational System)* também conhecido como MAC OS X é o sistema proprietário dos *Iphones* tendo a versão atual o iOS 13 onde será possivel utilizar tecnologias para criar experiências de realidade aumentada com o *ARKit* 3, *Reality Composer* e *RealityKit*. O *Core ML* 3 com o aplicativo *Create ML*, fornecem experiências personalizadas de aplicativos de maneira rápida e fácil com o aprendizado de máquina no dispositivo. Nessa versão são disponibilizadas melhorias mais recentes nos *Siri Shortcuts*, APIs de câmera, modo escuro e outras tecnologias. A Apple detalha em camadas com as principais tecnologias utilizadas no sistema operacional (APPLE INC, 2015):

a) Cocoa: São tecnologias para criar a interface de usuário de um aplicativo, responder a eventos e gerenciar o comportamento do aplicativo;
b) Media: Abrange tecnologias especializadas para reprodução, gravação e edição de mídia audiovisual e para renderização e animação de gráficos 2D e 3D;
c) *Core Services*: contém muitos serviços e tecnologias fundamentais, comunicação de rede de baixo nível à manipulação de strings e formatação de dados;
d) *Core OS*: Define *interfaces* de programação relacionadas ao *hardware* e à rede, incluindo interfaces para executar tarefas de computação de alto desempenho;
e) *Kernel* e *Device*: Consiste no ambiente do *Kernel Mach, drivers* de dispositivo, funções da biblioteca BSD (*libSystem*) e outros componentes de baixo nível.

3 MACHINE LEARNING

O *Machine Learning* ou Aprendizado de Máquina consiste em:

> "[...] Aprendizado de Máquina – AM – é uma subárea de pesquisa muito importante em inteligência Artificial – IA – pois a capacidade de aprender é essencial para um comportamento inteligente. AM estuda métodos computacionais para adquirir novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente. [...]", (BATISTA, 2013, p. 11).

De acordo com Monard e Baranaukas (2003), a indução é a forma lógica na qual é possível obter conclusões genéricas sobre um conjunto particular de exemplos, sendo o aprendizado indutivo realizado a partir de raciocínio sobre exemplos fornecidos de um processo externo ao sistema de aprendizado. O aprendizado indutivo é dividido em supervisionado onde é oferecido para o algoritmo de aprendizado ou indutor. Um conjunto de treinamento para os quais da classe associada é conhecido e não supervisionado onde o indutor analisa os exemplos fornecidos e tenta determinar se alguns podem ser agrupados de alguma forma sendo necessário verificar o que cada agrupamento significa no contexto o problema analisado.

Segundo Haykins (2008, p 35, tradução nossa) para o aprendizado supervisionado o professor é quem detém todo o conhecimento do meio ambiente, sendo representado por um conjunto de entrada e saída neural em um vetor de treinamento. Dessa forma o professor fornece a rede neural a resposta desejada para o vetor de treinamento, representada pela ação "ótima" a ser realizada pela rede neural. Seus parâmetros são reajustados influenciados pela combinação do vetor de treinamento e do sinal de erro, que é definido como a diferença entre a resposta desejada e a resposta real da rede. O ajuste é realizado iterativamente com o objetivo que a rede neural possa emular o professor, ou seja, absorver e assimilar conhecimento e com o passar do tempo poder interagir com o ambiente de forma autônoma e coesa.

Para o aprendizado não supervisionado não existe um professor para tutelar o processo de aprendizado pela rede neural. Existem duas subcategorias conforme Haykins (2008, p. 36, tradução nossa):

a) Aprendizado por reforço: O aprendizado em um mapeamento de entrada de saída é realizado por meio da interação contínua com o ambiente dinâmico (HAYKINS, 2008, p. 37, tradução nossa). A cada iteração com o ambiente é retornado um sinal de retorno definido como reforço ou recompensa indicando a qualidade da ação escolhida. Conforme são gerados os retornos o agente começa a aprender quais as ações que possuem as melhores recompensas e começa a mapear as situações as ações para maximizar as recompensas com o passar do tempo (MARTINS, 2007, p. 553).
b) Aprendizagem não supervisionada: Definida também como auto organizada, não há professor ou crítico externo para supervisionar o processo de aprendizagem. Pode se usar uma rede neural de duas camadas, a de entrada que recebe os dados disponíveis e a competitiva que consiste em neurônios que competem entre si de acordo com o uma regra de aprendizado pela oportunidade de processar os dados. Da forma

> trivial, a rede opera com uma estratégia “o vencedor leva tudo”. Assim o neurônio com maior contribuição total vence a competição e liga e todos os outros são desligados (HAYKINS, 2008, p. 37, tradução nossa).

3.1 MACHINE LEARNING NO ANDROID

Conforme a Android Developers ML (2019), para os aplicativos Android o aprendizado de máquina é a técnica de programação que concede ao aplicativo a capacidade de aprender e melhorar automaticamente a partir de experiências sem ser explicitamente programado para essa finalidade. Indicado para aplicativos que utilizam dados não estruturados, como imagens e texto, ou problemas com uma grande quantidade de parâmetros, como a previsão de um time vencedor de uma partida.

O Android é compatível com uma variedade de ferramentas e métodos de aprendizado de máquina. O programador se concentrará na inferência (que será explanado adiante), implantação, criação e treinamento de modelos de *machine learning* (ANDROID DEVELOPERS ML, 2019).

a) Projeto: Identificar as metas do produto e aproveitar os padrões de projeto de aprendizado de máquina definidos.
b) Construção e treinamento: opção de criar o aprendizado de máquina ou escolher modelos pré-treinados no Google.
c) Inferência: Analisando dados usando um modelo treinado em execução no dispositivo Android ou na nuvem.
d) Desenvolvimento: Instalar e atualizar modelos de aprendizado de máquina para o aplicativo.

A plataforma disponibiliza alguns guias como *People + AI Guidebook* que mostra as melhores práticas recomendadas para criação de produtos centrados para o ser humano e *The Material Design for Machine Learning* que contém uma diretriz de padrões e design para recursos de aprendizado de máquina como detecção de objetos e leitura de código de barras (ANDROID DEVELOPERS ML, 2019).

Para se utilizar modelos pré-treinados é recomendada a utilização dos Recursos do TensorFlow (2019).

3.1.1 Tensor Flow e TensorFlow Lite

É uma plataforma de código aberto para aprendizado em larga escala, que possui ferramentas, bibliotecas e recursos da comunidade de desenvolvedores e pesquisadores (TENSORFLOW, 2019). O TensorFlow Lite (2019) executa os modelos do TensorFlow em dispositivos móveis, incorporados e IoT, permitindo a inferência de aprendizado de máquina no dispositivo com baixa latência e um tamanho binário pequeno.

3.1.2 API Neural Networks (NNAPI)

É uma API do Android C desenvolvida para executar operações com uso intenso de recursos computacionais para aprendizado de máquina em dispositivos Android, sendo criada para oferecer uma camada básica para a funcionalidade de

machine learning de alto nível, como TensorFlow Lite e Caffe2 que compilam e treinam redes neurais (API NEURAL NETWORKS, 2019).

3.1.3 Firebase

É um contêiner de aplicativos para Android, IOS e projetos Web que oferece recursos como banco de dados, configurações e notificações entre apps multiplataforma (FIREBASE, 2019).

3.1.4 KIT de ML Firebase

Reúne em único SDK as Tecnologias API Cloud Vision, Tensor Flow Lite e a API Neural Networks para Android nos dispositivos móveis que leva o *machine learning* do Google para aplicativos Android e iOS (ML KIT FIREBASE, 2019). É disponibilizando modelos pré-treinados para uso nativo.

3.1.5 Inferência

Segundo a Android Developers ML (2019), inferência é o processo de usar um modelo de aprendizado de máquina já treinado para executar uma tarefa específica. O desenvolvedor pode escolher onde a inferência será executada, no dispositivo ou em um serviço de nuvem acessado remotamente.

Segundo a Android Developers ML (2019) existem alguns fatores a serem considerados pelo desenvolvedor na escolha de qual inferência será utilizada, conforme podemos verificar abaixo:

a) Latência:
   i. Nativo: Uma latência menor melhora a experiência em tempo real;
   ii. Nuvem: Comunicação assíncrona e largura de banda podem afetar a latência;
b) Recursos:
   i. Nativo: Os recursos específicos do dispositivo, como capacidade de processamento e armazenamento podem limitar o desempenho;
   ii. Nuvem: Recursos baseados em nuvem são mais potentes e o armazenamento mais amplo;
c) Off-line/On-line:
   i. Nativo: A capacidade de operar off-line é uma vantagem para a execução com infraestrutura de rede deficiente ou inexistente;
   ii. Nuvem: É necessário conexão de rede;
d) Custo:
   i. Nativo: Uso da bateria, tempo de download do modelo para usuários finais;
   ii. Nuvem: Largura de banda da transferência de dados para os usuários finais, custo de computação para os desenvolvedores;
e) Privacidade:
   i. Nativo: Os dados do usuário nunca saem do aparelho;
   ii. Nuvem: Os dados podem deixar o dispositivo. Outras precauções podem ser necessárias.

3.2 MACHINE LEARNING NO IOS

A Apple Inc. (2019) também possibilita o desenvolvimento de aprendizado para os seus chips A-Series e a *Neural Engine* que é dedicada ao processamento de redes neurais. Para isso são disponibilizados ambiente e ferramentas para o desenvolvimento de *machine learning* em seu ecossistema.

3.2.1 Core ML

Segundo a Apple Developers Core ML (2019), o *Core ML* é responsável por integrar os modelos de aprendizado para o aplicativo. O aplicativo usa as APIs do *Core ML* e os dados do usuário para fazer previsões e treinar ou ajustar modelos.

Modelo é o resultado da aplicação de algoritmo de aprendizado de máquina a um conjunto de dados de treinamento. Os modelos podem realizar uma ampla variedade de tarefas que seriam difíceis ou impraticáveis para se escrever em códigos. Pode se treinar um modelo para categorizar fotos ou detectar objetos específicos em uma foto (APPLE DEVELOPERS CORE ML, 2019).

Os modelos podem ser criados e treinados pelo aplicativo *Create ML* já deixando no formato de modelo do *Core ML*. Pode-se usar outras bibliotecas de *machine learning* e em seguida converter o modelo no formato Core ML. Quando um modelo está no dispositivo do usuário o Core ML pode ser usado para treina-lo novamente e ajustá-lo no dispositivo com os dados desse usuário (APPLE DEVELOPERS CORE ML, 2019).

É possível a otimização de desempenho no dispositivo aproveitando a CPU, GPU e o *Neural Engine* minimizando o consumo de memória e energia. A execução direta de um modelo no dispositivo do usuário elimina a necessidade de uma conexão de rede e mantém os dados do usuário privados.

O *Core ML* é a base para estruturas e funcionalidades específicas (APPLE DEVELOPERS CORE ML, 2019):

a) Vision: Realiza a detecção de ponto de referência de face e rosto, detecção de texto, código de barras, registro de imagens e rastreamento de recursos em geral (APPLE DEVELOPERS VISION, 2019);
b) Natural *Language*: Executa tarefas como identificação do idioma, lematização, marcação de partes de fala e reconhecimento de entidade nomeada (APPLE DEVELOPERS NATURAL LANGUAGE, 2019);
c) *Speech*: Reconhece palavras faladas em áudio gravado ou ao vivo (APPLE DEVELOPERS SPEECH, 2019);
d) *Sound Analysis*: Analisa o áudio e o reconhece como um tipo específico, como risos e aplausos (APPLE DEVELOPERS SOUND ANALYSIS, 2019);
e) *Accelerate*: Fornece computação de alto desempenho e economia de energia na CPU, aproveitando a capacidade de processamento vetorial (APPLE DEVELOPERS ACCELERATE, 2019);
f) BNNS: É uma coleção de funções que são utilizadas para implementar e executar redes neurais, usando dados de treinamento obtidos anteriormente (APPLE DEVELOPERS BNNS, 2019);
g) *Metal Performance Shaders*: É uma coleção de sombreadores gráficos e de computação altamente customizados (APPLE DEVELOPERS METAL PERFORMANCE SHADERS, 2019).

3.2.2 Create ML

O Create ML reúne ferramentas para criar e treinar modelos personalizados de aprendizado de máquina no próprio computador Mac. É possível criar modelos para executar tarefas como reconhecer imagens, extrair significado do texto ou encontrar relações entre valores numéricos (APPLE DEVELOPERS CREATE ML, 2019).

O modelo é treinado para reconhecer padrões, mostrado amostras representativas, podem, por exemplo treinar modelos para reconhecer cães mostrando muitas imagens de cães diferentes. Quando o modelo estiver com bom desempenho, estará pronto para ser integrado ao aplicativo (APPLE DEVELOPERS CREATE ML, 2019).

Pode ser aproveitada também a infraestrutura de aprendizado de máquina incorporada aos produtos *Apple* como *Photos* e Siri, e significa que classificação da imagem e os modelos de linguagem natural levam menos tempo para serem treinados (APPLE DEVELOPERS CREATE ML, 2019).

4 METODOLOGIA

Foram implementados dois aplicativos nativos que utilizam modelos disponibilizados pelas respectivas fabricantes. Os códigos para as plataformas Android e IOS (Apple INC) foram implementados nos *frameworks* proprietários Android Studio e XCode respectivamente

5 RESULTADOS E DISCUSSÕES

Nos itens abaixo serão demonstradas as implementações dos aplicativos realizadas para cada ambiente.

5.1 IMPLEMENTAÇÃO PARA ANDROID

A seguir são listadas as opções de inferência disponíveis para desenvolvimento nativo nos aplicativos Android (ANDROID DEVELOPERS MACHINE LEARNING, 2019):

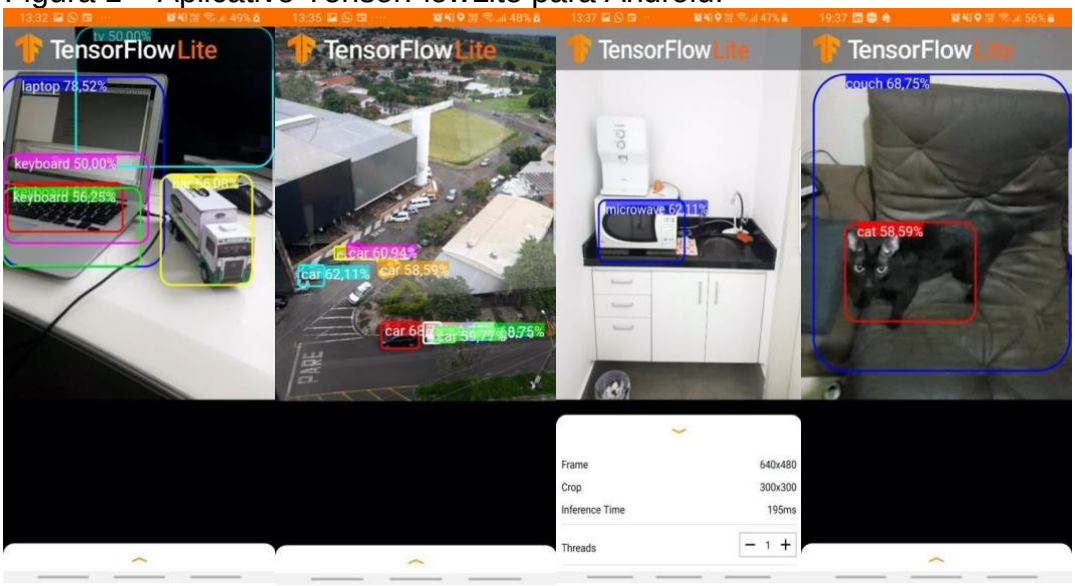
a) Kit de ML Firebase:
    i. Marcação de Imagens, Reconhecimento de texto (OCR), Reconhecimento Facial (inclusive contorno do rosto), Leitura de Código de Barras, Identificação de Idioma, Respostas Inteligentes, Tradução;
b) Google Cloud:
    i. Pode-se treinar um modelo de visão personalizado no Google Cloud e executar o resultado no Android com o AutoML Vision Edge;

c) TensorFlow Lite:
    i. Executa os modelos do TensorFlow no dispositivo móvel. É possível reutilizar ou reciclar um modelo existente.

Para a implementação nativa do *machine learning* no Android foi utilizado TensorFlow Lite que disponibiliza uma demonstração para a detecção de objetos continuamente em tempo real com caixas delimitadoras pela câmera traseira (TENSORFLOW LITE EXEMPLOS, 2019).

Utiliza modelos SDD MobileNet e treinado no conjunto de dados detecção, segmentação e legenda de objetos em larga escala denominado COCO Data set (2020) conforme a figura 1.

Figura 1 – Aplicativo TensorFlowLite para Android.



Fonte: Elaborada pelo próprio autor.

Conforme a figura 1, é possivel visualizar que o TensorFlow Lite conseguiu identificar objetos, veículos e animais tais como, Laptop e com o teclado, veículos, micro-ondas, sofá e o gato.

Segundo Android Developers ML (2019) TensorFlow Lite pode ficar hospedado remotamente no servidor *Firebase*. O modelo pode ser atualizado sem a necessidade de lançar uma nova versão e atualização do aplicativo, caso haja indisponibilidade desse servidor o modelo nativo pode ser usado.

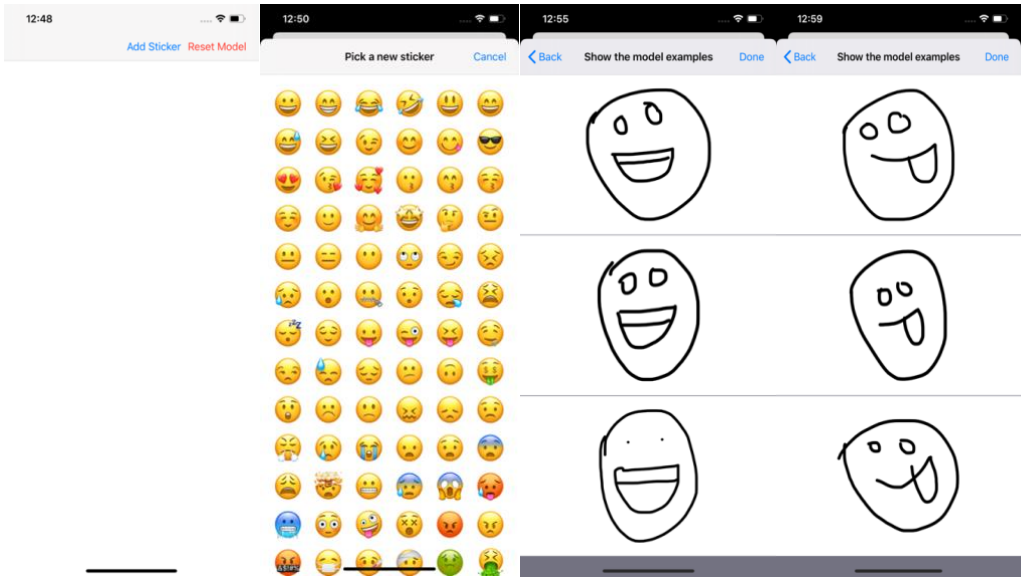
5.2 IMPLEMENTAÇÃO PARA IOS

Foi implementado no XCode que é a ferramenta de desenvolvimento de aplicações nativas Apple, o projeto UpdatableDrawingClassifier disponibilizado também pela Apple que contém um modelo classificador de desenhos que aprende a reconhecer desenhos com base no modelo K-Nearest Neghbors (KNN). Segundo Bastista (2003, p. 97), consiste em apenas armazenar os dados do treinamento, quando um novo exemplo é apresentado, um novo conjunto de exemplos similares é recuperado do conjunto de treinamento e utilizado na classificação.

O projeto utiliza *emojis* como rótulo no qual o classificador deverá reconhecer o desenho conforme a entrada manuscrita do usuário. Selecionado o *emoji* o usuário treina o modelo três vezes fazendo o desenho como exemplo correspondente ao *emoji* selecionado. Esse treinamento pode ser feito com mais de um *emoji*, dando mais opções de desenhos que o aplicativo poderá reconhecer. Feito isso o usuário poderá inserir a entrada para que seja feito o reconhecimento

dos desenhos conforme os exemplos inseridos para o modelo, conforme a figura 2 e figura 3.

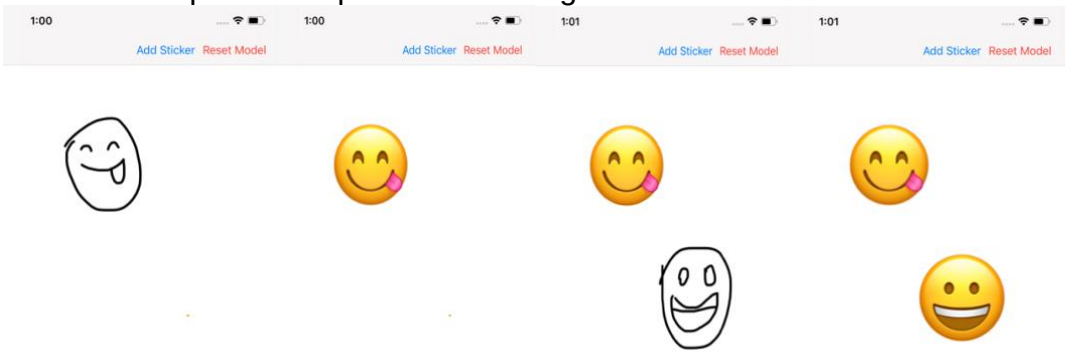
Figura 2 – Aplicativo UpdatableDrawingClassifier para IOS nas telas de rótulos e treinamento.



Fonte: Elaborada pelo próprio autor.

Conforme a figura 2, o usuário seleciona os rótulos e faz o treinamento do modelo conforme o *emoji* selecionado

Figura 3 – Tela de entrada dos desenhos e reconhecimento de imagem do aplicativo UpdatableDrawingClassifier.



Fonte: Elaborada pelo próprio autor.

Conforme a figura 3, após inseridos os exemplos o usuário realiza a entrada manuscrita dos desenhos e classificador faz o reconhecimento e converte na imagem que ele interpreta ser correspondente.

6 CONSIDERAÇÕES FINAIS

Este estudo veio mostrar que o Android e IOS em suas versões mais atuais possuem os recursos próprios para o processamento de *machine learning* e inteligência artificial de forma nativa, onde os resultados das inferências realizadas nos testes realizados foi satisfatório mas a acurácia na identificação dos objetos pode melhorar a medida que o modelo vai ficando mais inteligente e atualizado. Verificou-se também que as tecnologias nativas podem ser auxiliadas com APIs externas de forma assíncrona como o Firebase disponível para ambas as plataformas, podendo manter os modelos de *machine learning* atualizados, sem a necessidade de atualizar a aplicação inteira. Para o futuro pode-se esperar que estes recursos de inteligência artificial fiquem mais acessíveis e refinados mediante a rápida evolução tecnológica. Para o Android por ser uma tecnologia de código aberto mantida pelo Google, empresas ou comunidade de desenvolvedores poderão criar suas próprias tecnologias, recursos e produtos de inteligência artificial para a plataforma em paralelo ao mantenedor, abrindo novos caminhos para novos estudos. No caso do IOS por ser um sistema operacional de código fechado e proprietário, o caminho pode ser um pouco mais limitado pois tudo depende da fabricante, sendo ela apenas quem disponibiliza as ferramentas e recursos para o desenvolvimento na plataforma, mas a Apple por sempre estar na vanguarda da tecnologia pode se esperar novidades e criações serem estudadas, como por exemplo as tecnologias lançadas recentemente para o processador A13 *bionic* que possui área processamento destinada exclusivamente para processar as atividades de inteligência artificial e *machine learning* denominada *Neural Engine*, podendo ser objeto de estudos futuros. Um outro possível caso de estudo futuro seria uma possível integração das plataformas móveis com as soluções de *machine learning* da Amazon e de outras empresas de tecnologia criar novas soluções ou aprimorar as já existentes nesta área, verificando como é promovida a integração com as plataformas, possiblidade do uso nativo e a possibilidade de utilização de *serverless* para o utilização em alta demanda destes recursos.

REFERÊNCIAS

ANDROID DEVELOPERS. **Arquitetura da Plataforma.** 2019. Disponível em: https://developer.android.com/guide/platform. Acesso em: 20 jan. 2020.

ANDROID DEVELOPERS ML. **Crie apps mais inteligentes com aprendizado de máquina**. 2019. Disponível em: https://developer.android.com/ml. Acesso em: 30 jan. 2020.

API NEURAL NETWORKS. **API Neural Networks**. 2019. Disponível em: https://developer.android.com/ndk/guides/neuralnetworks/. Acesso em: 06 fev. 2020.

APPLE DEVELOPERS ACCELERATE. **Accelerate**. 2019. Disponível em: https://developer.apple.com/documentation/accelerate. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS BNNS. **BNNS.** 2019. Disponível em: https://developer.apple.com/documentation/accelerate/bnns. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS CORE ML. **Core ML**. 2019. Disponível em https://developer.apple.com/documentation/coreml. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS CREATE ML. **Create ML**. 2019. Disponível em: https://developer.apple.com/documentation/createml. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS NATURAL LANGUAGE. **Natural Language**. 2019. Disponível em: https://developer.apple.com/documentation/naturallanguage. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS PERFORMACE SHADERS. **Metal Performance Shaders**. 2019. Disponível em: https://developer.apple.com/documentation/metalperformanceshaders. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS SPEECH. **Speech**. 2019. Disponível em: https://developer.apple.com/documentation/speech. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS SOUND ANALYSIS. **SoundAnalysis**. 2019. Disponível em: https://developer.apple.com/documentation/soundanalysis. Acesso em: 10 fev. 2020.

APPLE DEVELOPERS VISION. **Vision**. 2019. Disponível em: https://developer.apple.com/documentation/vision. Acesso em: 10 fev. 2020.

APPLE INC. **About Developing for Mac**. 2015. Disponível em: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/OSX_Technology_Overview/About/About.html#//apple_ref/doc/uid/TP40001067-CH204-TPXREF101. Acesso em: 22 jan. 2020.

APPLE INC. **Get Ready For IOS 13**. 2019. Disponível em https://developer.apple.com/ios/. Acesso em: 20 jan. 2020.

BATISTA, G.E.A.P.A, **Pré-processamento de dados em aprendizado de máquina Supervisionado**. 2003. Tese (Doutorado) – Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, São Carlos. Disponível em: https://teses.usp.br/teses/disponiveis/55/55134/tde-06102003-160219/publico/TeseDoutorado.pdf. Acesso em: 23 jan. 2020.

COCO DATASET. **Common Objects in Context**. 2020. Disponível em: http://cocodataset.org/#home. Acesso em: 12 fev. 2020.

FIREBASE. **Perguntas frequentes sobre o firebase.** 2019. Disponível em: https://firebase.google.com/support/faq/. Acesso em: 06 mar. 2020.

HAYKIN, S. **Neural Networks and Learning Machines.** 3rd Edition. Prentice Hall. 2008.

IVERSON, J. EIRMAN, M. **Lerning Mobile APP Delevopment: A hands-n Guide to Building Apps with IOS and Android.** 1ª Ed. Nova York: Ed. Editora Addison-Wesley, 2013.

MARTINS, Weber et al. **Tutoriais Inteligentes Baseados em Aprendizado por Reforço: Concepção, Implementação e Avaliação Empírica**. Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE**)**, [S.l.], p. 550-559, nov. 2007. ISSN 2316-6533. Disponível em: https://www.br-ie.org/pub/index.php/sbie/article/view/604. Acesso em: 30 jan. 2020.

ML KIT FIREBASE. **ML Kit para Firebase**. 2019. Disponível em: https://firebase.google.com/docs/ml-kit. Acesso em: 06 fev. 2020.

MONARD, Maria Carolina, BARANAUSKAS, José Augusto. **Conceitos Sobre Aprendizado de Máquina. Sistemas Inteligentes Fundamentos e Aplicações**. 2003. 1 ed. Barueri-SP: Manole Ltda. p. 89 -114. ISBN 85-204-168.

POINTER. I. **Conheça as 5 Melhores linguagens de programação para inteligência artificial – Computerworld EUA.** 2018. Disponível em https://computerworld.com.br/2018/07/04/conheca-5-melhores-linguagens-de-programacao-para-inteligencia-artificial/. Acesso em: 13 ago. 2020.

STATSCOUNTER Globalstats. **Mobile Operating System Market Share Woldwide – January 2020**. 2020. Disponível em: https://gs.statcounter.com/os-market-share/mobile/worldwide. Acesso em: 30 jan. 2020.

TENSORFLOW. **End-to-end open source machine learning platform**. 2020. Disponível em: https://www.tensorflow.org/. Acesso em: 03 fev. 2020.

TENSORFLOW LITE. **TensorFlow Lite Guide**. 2020 Disponível em: https://www.tensorflow.org/lite/guide. Acesso em: 06 fev. 2020.

TENSORFLOW LITE EXEMPLOS. **TensorFlow Lite Object Detection Android Demo**. 2019. Disponível em: https://github.com/tensorflow/examples/blob/master/lite/examples/object_detection/android/README.md. Acesso em: 12 fev. 2020.

VALENTINO, L.; SCHNEIDER, H.; ROBBIE, S.; **Aplicações Móveis. Arquitetura, projeto e desenvolvimento**. PEARSON Makron Boooks, 2005.