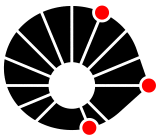

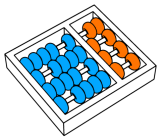

Aplicativo iOS para Identificação de Letras Desenhadas na Tela do Celular Utilizando Aprendizado de Máquina

B. D. Sepulveda *H. Pedrini*

Relatório Técnico - IC-PFG-21-01
Projeto Final de Graduação
2021 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Aplicativo iOS para Identificação de Letras Desenhadas na Tela do Celular Utilizando Aprendizado de Máquina

Bruno Dias Sepulveda[1], Hélio Pedrini[2]

[1] Instituto de Computação, Universidade Estadual de Campinas, 13083-852 Campinas, SP.

[2] Instituto de Computação, Universidade Estadual de Campinas, 13083-852 Campinas, SP.

Resumo. Este relatório descreve as principais atividades relacionadas ao desenvolvimento de um aplicativo iOS para identificação de letras desenhadas na tela de um celular. Para isso, tecnologias modernas foram empregadas, como uma rede neural para a identificação das letras e uma interface de usuário implementada com os ambientes SwiftUI e PencilKit. Uma potencial aplicação do sistema é o auxílio a crianças no processo de alfabetização por meio de um celular, um dispositivo cada vez mais comum no cotidiano das pessoas.

1. Introdução

Para superar o problema mundial de saúde que enfrentamos desde 2020, devido à pandemia da COVID-19, muitas das atividades que eram realizadas presencialmente hoje precisam ser realizadas de forma remota devido à necessidade de um distanciamento social. Uma das áreas que sofreram um enorme impacto por essa mudança foi a educação.

Mesmo com um professor ao nosso lado, aprender a ler e escrever não é uma tarefa simples. Com o distanciamento social, o acompanhamento no ensino das crianças por parte dos professores ficou ainda mais difícil, sendo possível fazê-lo apenas por meio da Internet.

Nesse momento, as tecnologias de ensino à distância vêm sendo muito importantes para a continuidade dos estudos e novas ferramentas são desenvolvidas e aprimoradas para nos adaptarmos a essa nova realidade.

Devido a isso, celulares e computadores estão cada vez mais presentes em nossas vidas. Segundo pesquisa realizada pelo IBGE em 2019 [10], o celular é o equipamento mais usado pelo brasileiro para o acesso à Internet.

O objetivo deste projeto é desenvolver uma solução que auxilie crianças no aprendizado da escrita com uma verificação automatizada das respostas e uma interface simples e de fácil uso, utilizando tecnologias modernas, como aprendizado de máquina e aplicativos móveis.

2. Desenvolvimento

Esta seção descreve as principais etapas do desenvolvimento do projeto. O modelo de aprendizado de máquina foi desenvolvido em uma máquina com processador i5 9600K, 16 GB de memória e placa gráfica Nvidia Geforce 1060 6 GB com sistema operacional Windows 10 Pro. A interface de usuário foi desenvolvida em um MacBook Pro com 16 GB de memória e sistema operacional macOS Big Sur 11.4.

2.1 Ferramentas utilizadas

Para desenvolvimento da interface foi utilizada a linguagem de programação Swift com auxílio de alguns frameworks fornecidos pela Apple. Para o desenvolvimento do modelo de aprendizado de máquina foi utilizada a linguagem de programação Python com auxílio de algumas bibliotecas. A seguir temos uma lista com os *frameworks* e bibliotecas utilizadas:

- SwiftUI: utilizado para desenvolvimento de interfaces de usuário [1].
- PencilKit: provê um ambiente de desenho para capturar o toque do usuário e transformá-lo em uma imagem a ser apresentada na tela [2].
- CoreML: faz a integração de modelos de aprendizado de máquina em aplicativos móveis [3].
- AVFoundation: utilizado para trabalhar com mídia audiovisual em sistemas operacionais desenvolvidos pela Apple (iOS, macOS, watchOS, tvOS) [4].
- Keras: biblioteca de rede neural para desenvolvimento de modelos de aprendizado de máquina profundo utilizando a linguagem de programação Python.
- Coremltools: converte modelos de aprendizado de máquina desenvolvidos em Python para o formato utilizado pelo *framework* CoreML.
- OpenCV: utilizado para leitura das imagens do conjunto de dados.

2.2 Aprendizado de Máquina

Esta seção descreve as principais etapas do desenvolvimento do modelo de aprendizado de máquina.

2.2.1 Conjunto de dados

O conjunto de dados utilizado foi o "*Handwritten Math Symbols Dataset*" [7], que contém 375974 imagens em escala de cinza de tamanho 45 x 45 divididas em 82 classes, entre elas letras, números e símbolos matemáticos.

Para o propósito deste projeto, as classes que não representam letras do alfabeto romano foram removidas do conjunto de dados. Assim, restaram 122552 imagens divididas em 26 classes, sendo elas as 26 letras do alfabeto, tanto maiúsculas quanto minúsculas.

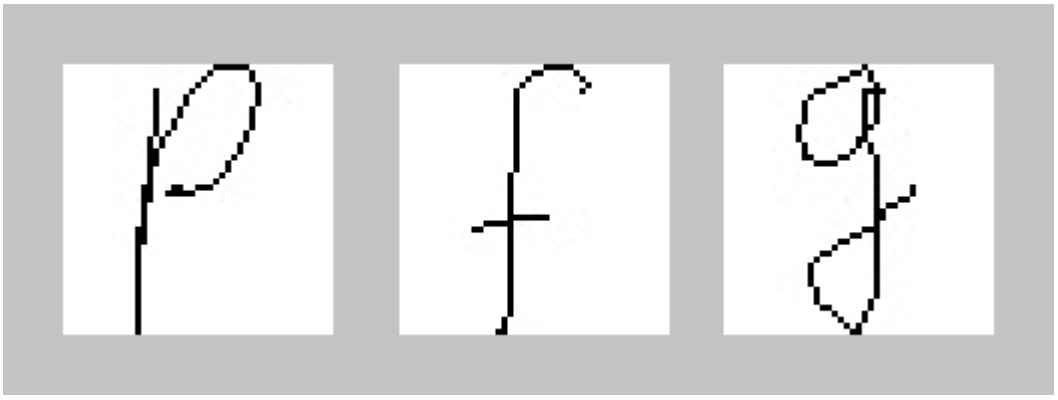

Figura 1 – Exemplos de amostras do conjunto de dados

2.2.2 Modelo

Para o desenvolvimento de um modelo de aprendizado de máquina para classificação das imagens foi utilizada a biblioteca Keras [5]. Os dados foram lidos de uma pasta no sistema e adicionados a um vetor utilizando a biblioteca OpenCV.

Para formar o conjunto de testes, 15% dos dados foram separados utilizando a função "train_test_split" da biblioteca Scikit-Learn com Stratify para que tal conjunto tivesse representantes de todas as classes. O conjunto de dados restante foi dividido em

80% para treinamento e 20% para validação. A intensidade dos pixels das imagens foi dividida por 255 como pré-processamento para que todos os valores de entrada estivessem entre 0 e 1.

A arquitetura da rede neural foi baseada em uma rede utilizada para classificação de dígitos numéricos escritos à mão (banco de dados MNIST), em que ela obteve uma acurácia de 99,17% [8]. A mudança feita na rede para o conjunto de dados deste projeto ocorreu nas camadas de entrada e de saída, pois o MNIST possui imagens de tamanho 28 x 28 e apenas 10 classes. Portanto, a rede recebe como entrada uma imagem em escala de cinza de tamanho 45 x 45 e tem como saída um vetor de tamanho 26, onde cada elemento do vetor representa a chance de a imagem ser de determinada classe de acordo com a predição do modelo. A arquitetura da rede pode ser vista na Figura 2.

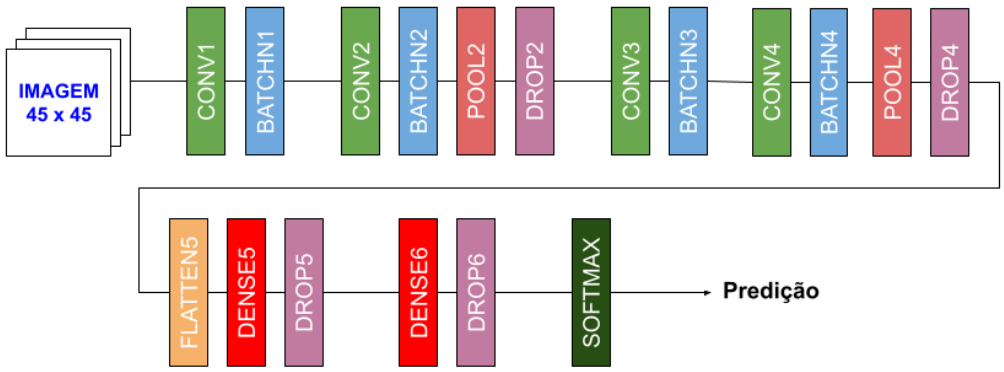

Figura 2 – Arquitetura da rede neural convolucional utilizada no projeto.

Após realizar o treinamento da rede, observou-se que algumas classes possuíam uma acurácia baixa devido ao baixo número de amostras para tais classes, como por exemplo a letra O, W e L. Para resolver esse problema, foi aplicada uma aumentação de dados para as classes com menos de 3000 amostras. O número de imagens sem aumentação e com aumentação podem ser vistos na Tabela 1.

Com o novo número de amostras e um conjunto de dados mais balanceado obteve-se uma acurácia de 99,79% no conjunto de validação. Sendo esse resultado muito satisfatório para o objetivo do projeto, realizou-se a verificação da acurácia no conjunto de testes para esse modelo, sendo essa de 99,84%.

Por fim, como o *framework* CoreML[4], usado para integrar modelos de aprendizado de máquina em aplicativos iOS, utiliza modelos no formato MLModel, fez-se a conversão do modelo desenvolvido utilizando a biblioteca Coremltools disponível em Python.

Tabela 1 – Número de amostras por classe no conjunto sem e com aumentação de dados

Sem aumentação		Com aumentação	
Classe	Número de imagens	Classe	Número de imagens
A	12367	A	12367
B	8651	B	8651
C	5802	C	5802
D	4852	D	7540
E	3003	E	3003
F	3712	F	3712
G	1692	G	3384
H	1464	H	2928
I	5140	I	5140
J	1536	J	3070
K	3074	K	3074
L	1017	L	3051
M	2476	M	4952
N	10862	N	10862
O	449	O	7136
P	2680	P	2680
Q	1230	Q	2460
R	2671	R	2671
S	1413	S	2826
T	3274	T	3274
U	1269	U	3807
V	1558	V	3116
W	556	W	5004
X	26594	X	26594
Y	9340	Y	9340
Z	5870	Z	5870
Total	122552	Total	152314

2.3 Interface de Usuário

O desenvolvimento da interface de usuário foi dividido em etapas. Primeiramente, fez-se uma pesquisa sobre quais ferramentas poderiam ser utilizadas no desenvolvimento de aplicativos iOS, na criação de uma área em que o usuário pudesse desenhar e na integração de modelos de aprendizado de máquina com aplicativos. Entre as opções disponíveis, escolhemos como base do projeto os ambientes SwiftUI, PencilKit e CoreML.

2.3.1 Primeira iteração

O objetivo da primeira iteração foi criar um protótipo para testar a integração entre a interface desenvolvida utilizando SwiftUI e PencilKit com o modelo de aprendizado de máquina e também verificar se o modelo teria uma boa acurácia utilizando as imagens geradas no canvas do PencilKit como entrada da rede neural.

A ideia inicial era de que o usuário receberia uma palavra e teria que desenhá-la letra por letra. Ao clicar em “Send” a imagem desenhada no canvas é adicionada a um fundo branco quadrado e redimensionada para 45 x 45, de acordo com a entrada da rede neural. Então, a imagem é enviada para a rede, que retorna como resultado a letra com maior probabilidade. A letra resultante seria adicionada ao campo de texto de resposta e o usuário teria que desenhar a letra seguinte, porém nessa iteração ela é apenas impressa no terminal.

A imagem do protótipo pode ser vista na Figura 3 e o fluxo para predição da letra desenhada pode ser visto na Figura 4.

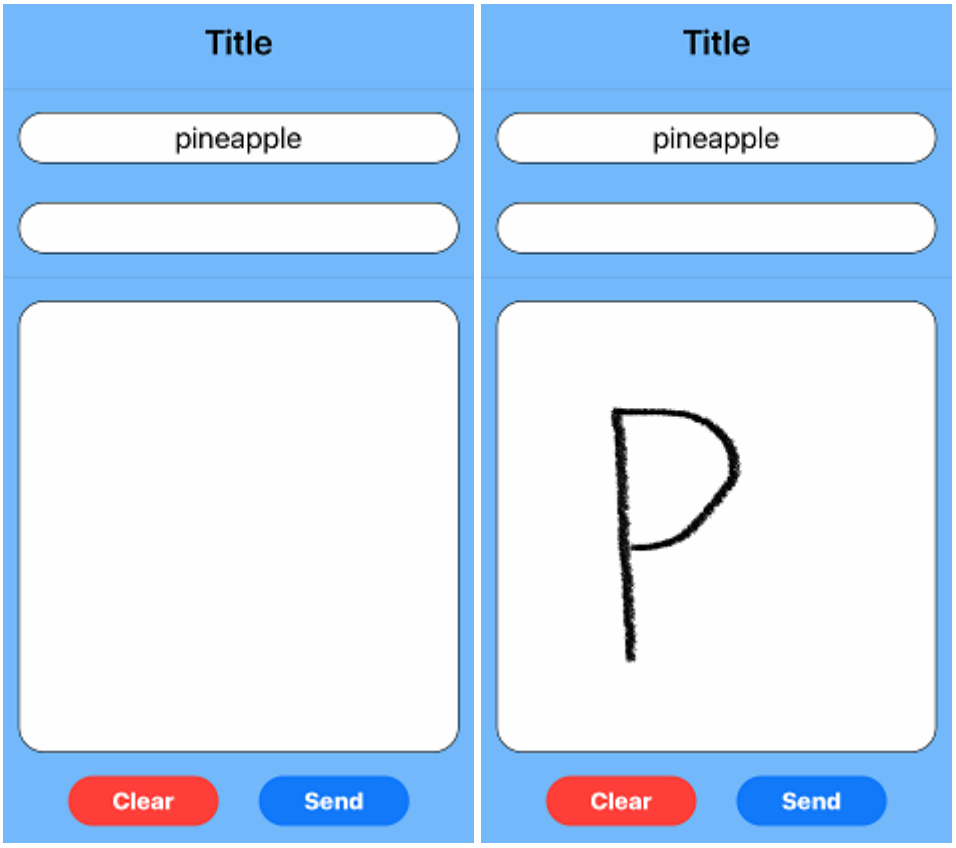

Figura 3 – Protótipo do aplicativo iOS – sem e com desenho do usuário.

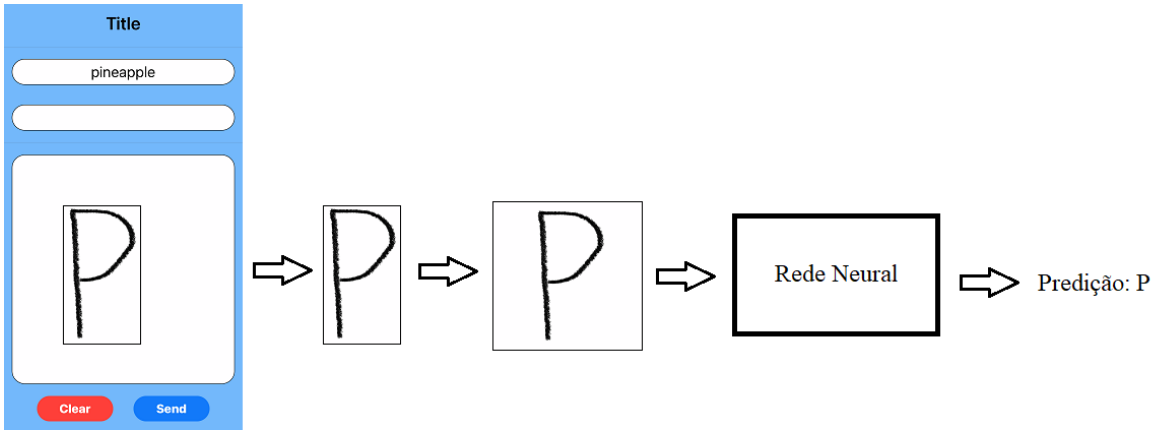

Figura 4 – Sequência para predição da letra desenhada.

2.3.2 Segunda iteração

Na segunda iteração, o objetivo foi trazer uma melhoria visual e adicionar botões para novas funcionalidades, porém as ações dos botões não foram implementadas nesse momento. Entre essas funcionalidades temos:

- esconder/mostrar a palavra que deve ser escrita
- apagar a última letra que o usuário enviou
- mostrar uma imagem que representa a palavra que deve ser escrita
- emitir um áudio de uma pessoa falando a palavra que deve ser escrita
- mostrar a letra predita pela rede neural sem que o usuário tenha que adicioná-la ao campo de resposta
- um botão com informações sobre o aplicativo
- ir para a próxima palavra a ser escrita

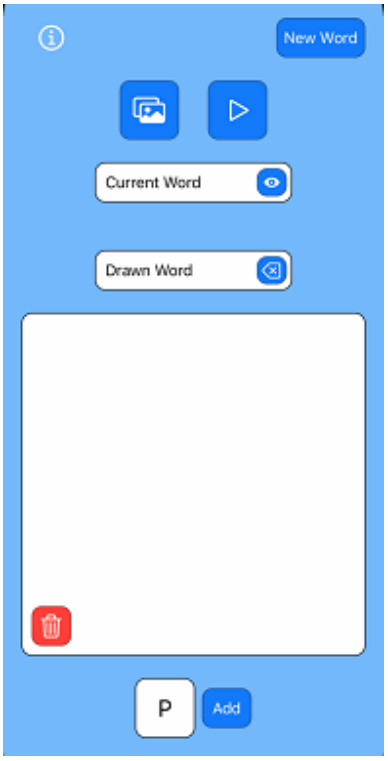

Figura 5 – Interface implementada na segunda iteração.

2.3.3 Terceira iteração

Na terceira e última iteração, foi feita uma melhora no visual, trazendo botões maiores e uma cor de fundo mais contrastante com o restante dos elementos. As funcionalidades citadas na iteração anterior foram implementadas.

Mais imagens podem ser vistas na seção de resultados.

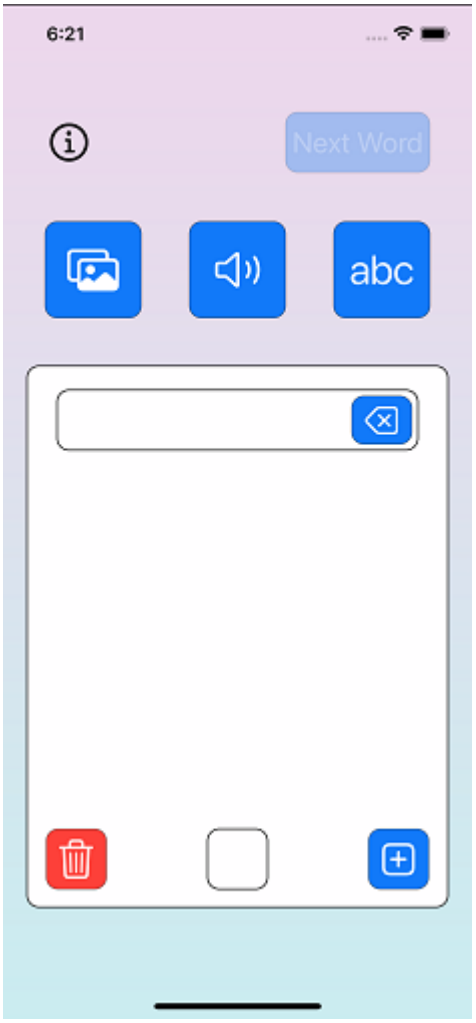

Figura 6 – Interface implementada na terceira iteração.

3. Resultados

Esta seção apresenta as funcionalidades que foram implementadas. Além dessas apresentadas também temos a funcionalidade de reprodução de áudio.

Informação sobre o aplicativo

Ao clicar no botão, um modal é exibido para o usuário, explicando brevemente sobre o objetivo do aplicativo e como utilizá-lo.

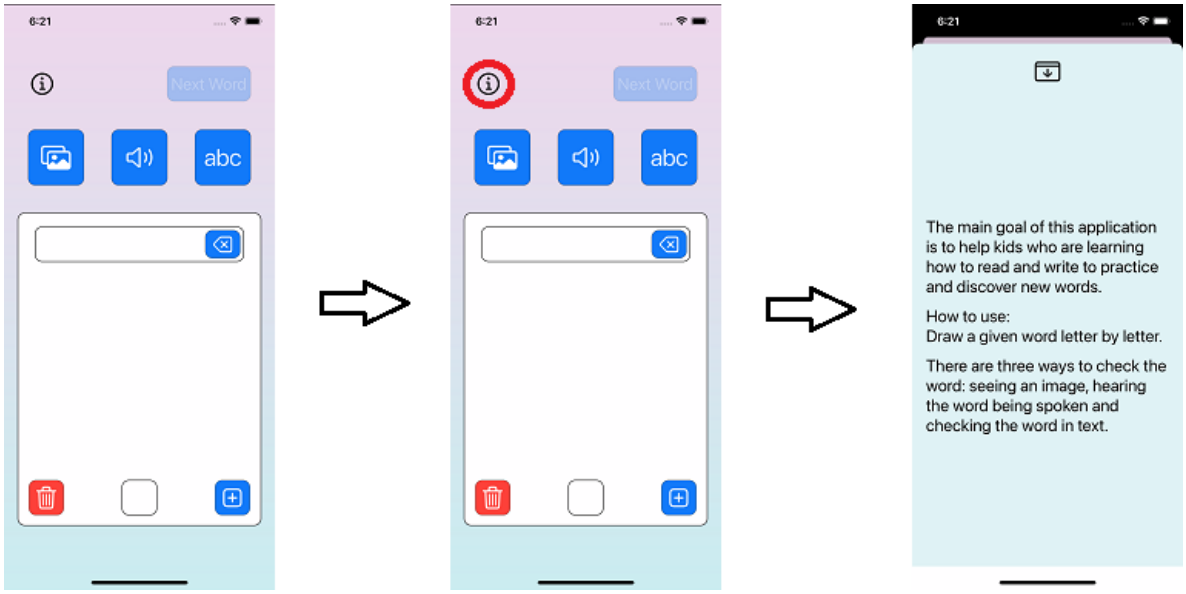

Figura 7 – Funcionalidade de mostrar informações sobre o aplicativo.

Exibir imagem

Ao clicar no botão, um modal é exibido com uma imagem informando ao usuário qual é a palavra que ele deve escrever.

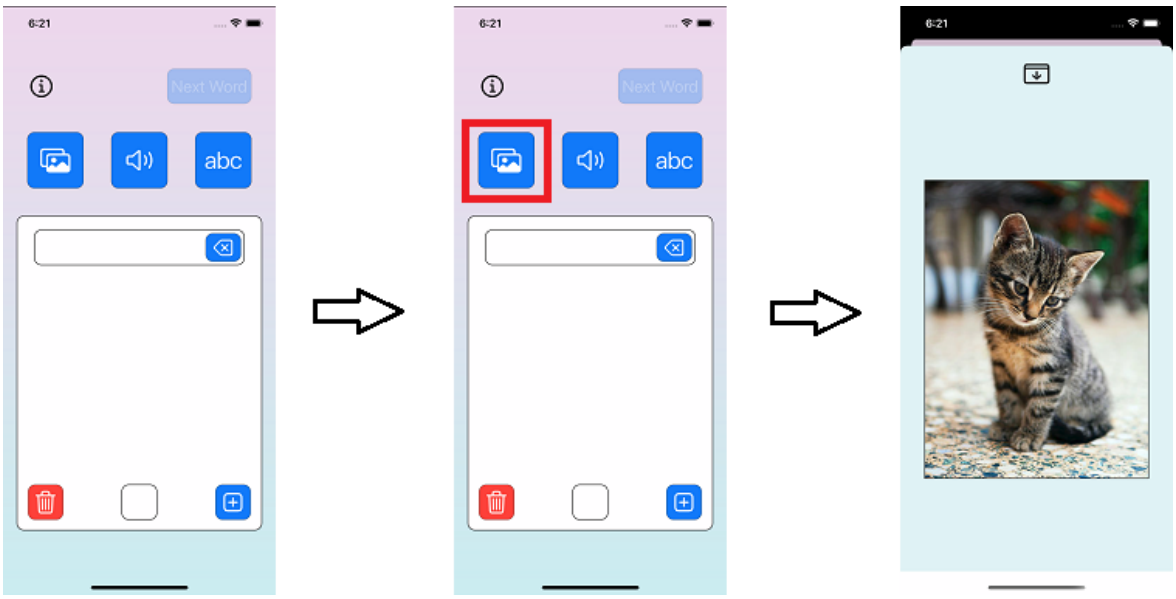

Figura 8 – Funcionalidade de exibir imagem de exemplo [9].

Exibir palavra

Ao clicar no botão, um modal é exibido informando diretamente ao usuário qual é a palavra que ele deve escrever.

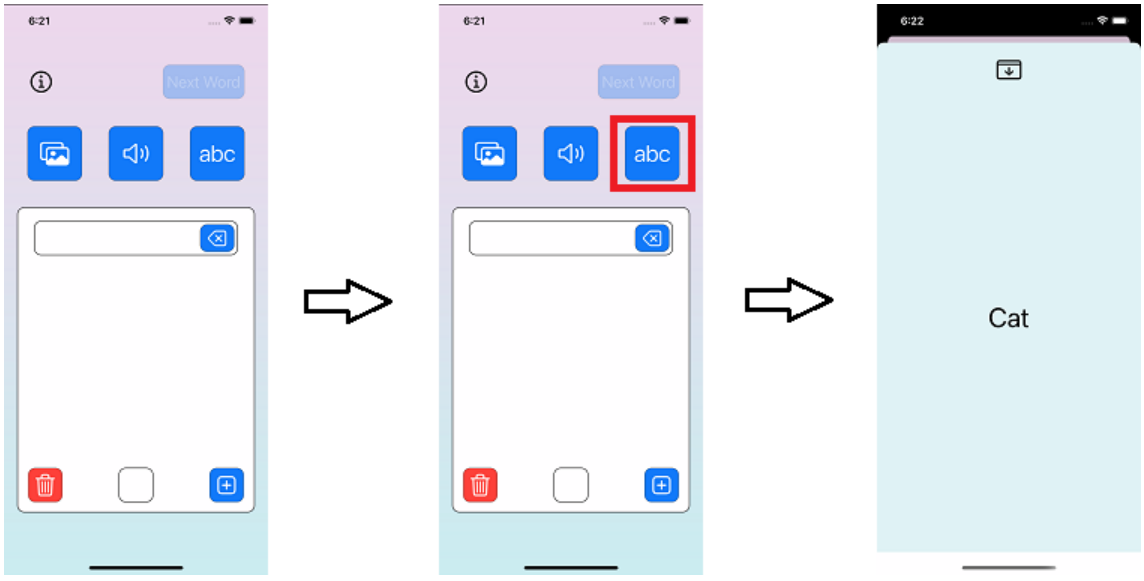

Figura 9 – Funcionalidade de exibir palavra a ser escrita.

Desenhar no canvas e adicionar letra

O usuário pode desenhar a letra no canvas e, ao clicar no botão, adicionar a letra para formar a palavra.

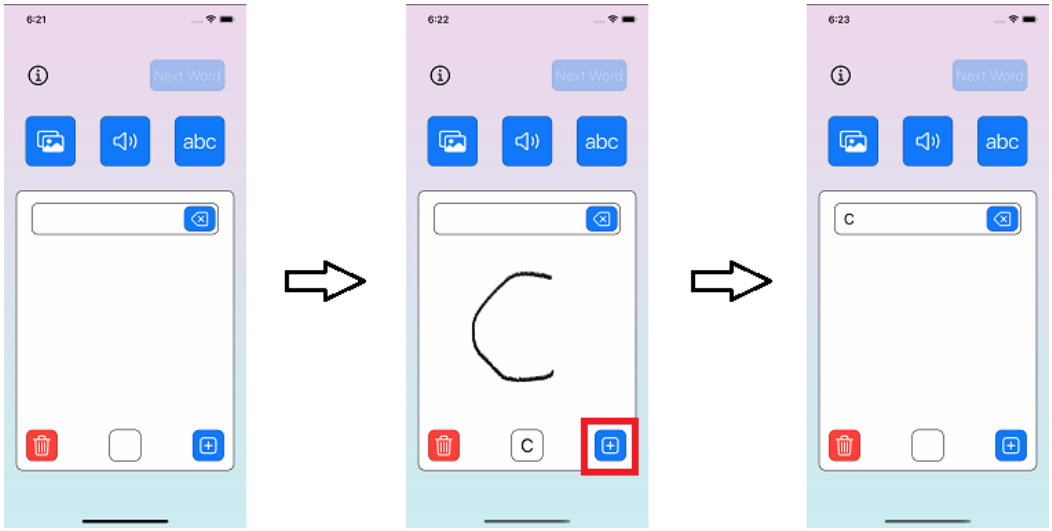

Figura 10 – Funcionalidade de desenhar no canvas e adicionar letra no campo de resposta.

Atualização da prévia a cada traço

A cada traço do usuário no canvas, a predição da letra desenhada é atualizada.

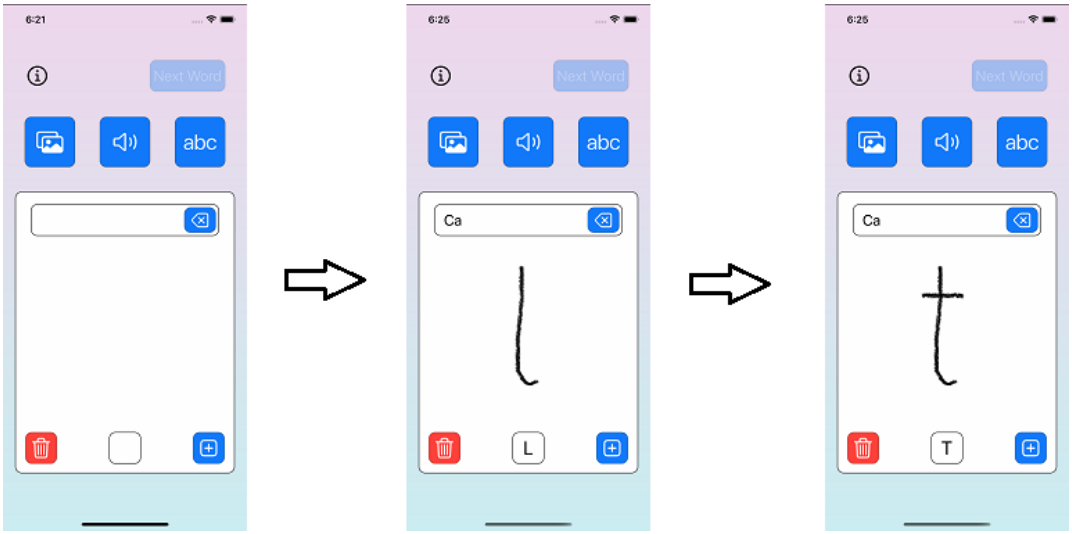

Figura 11 – Funcionalidade de atualizar a predição da rede a cada traço do usuário.

Indicação ao completar palavra corretamente

Ao completar uma palavra corretamente, o usuário recebe uma indicação no campo de texto de resposta e o botão para ir para a próxima palavra é habilitado.

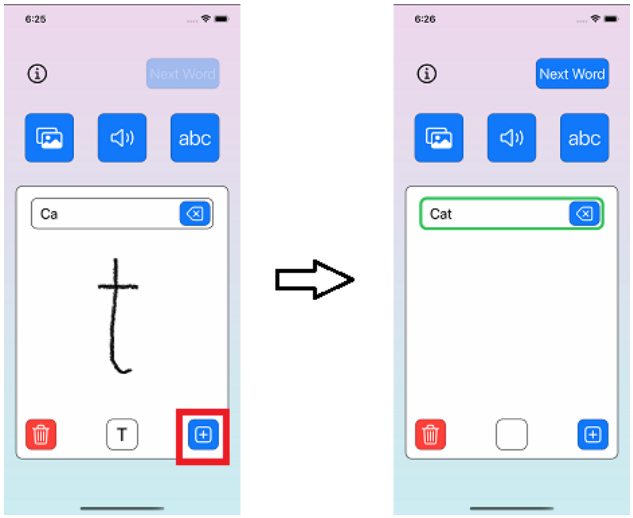

Figura 12 – Funcionalidade de mostrar ao usuário que a palavra foi completada.

Limpar canvas

Ao clicar no botão, o usuário apaga o que desenhou no canvas.

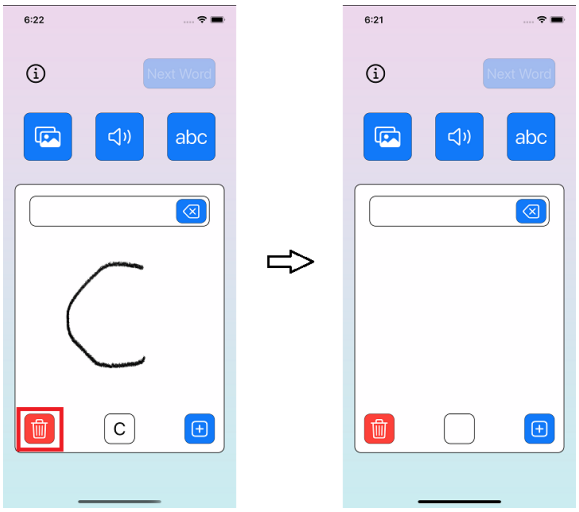

Figura 13 – Funcionalidade de limpar o canvas.

Apagar letra no campo de texto de resposta

Ao clicar no botão, a última letra adicionada é removida do campo de texto de resposta.

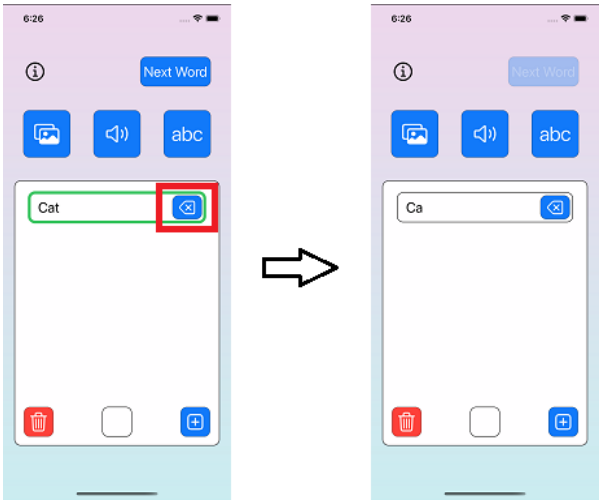

Figura 14 – Funcionalidade de apagar uma letra no campo de resposta.

4. Conclusões e Trabalhos Futuros

Com o avanço tecnológico dos celulares, temos aparelhos com cada vez mais capacidade de processamento de dados e mais acessíveis pela população, assim sendo possível aplicar o aprendizado de máquina como uma forma de facilitar nossas vidas.

Neste trabalho, desenvolvemos uma solução para um problema relevante da sociedade utilizando ferramentas modernas para desenvolvimento de sistemas computacionais, como redes neurais e aplicativos móveis.

Foi possível fazer o treinamento de uma rede neural convolucional de forma a obter uma acurácia adequada para alcançarmos o objetivo do projeto, sendo essa acurácia de 99,84% no conjunto de testes utilizado. Algumas razões que podem explicar esta alta acurácia são a baixa variabilidade nos formatos das letras presentes na base de dados e a homogeneidade do fundo da imagem.

Verificamos que a qualidade de um conjunto de dados é fundamental para o desenvolvimento de um modelo de aprendizado de máquina. Também pudemos verificar que aparelhos celulares modernos possuem um bom desempenho quando utilizamos uma rede neural para realizar a análise de imagens digitais.

Como sugestões para trabalhos futuros, pretendemos aumentar a base de dados para permitir uma maior variabilidade no formato das letras e realizar testes com usuários, uma vez que o propósito original do trabalho era a construção de uma ferramenta para auxílio à alfabetização de crianças.

Referências

[1] “SwiftUI,” https://developer.apple.com/documentation/swiftui/, (Acesso em 16/06/2021).

[2] “AVFoundation,” https://developer.apple.com/documentation/avfoundation, (Acesso em 16/06/2021).

[3] “PencilKit,” https://developer.apple.com/documentation/pencilkit, (Acesso em 16/06/2021).

[4] “CoreML,” https://developer.apple.com/documentation/coreml, (Acesso em 16/06/2021).

[5] “Keras,” https://keras.io/about/, (Acesso em 16/06/2021).

[6] “OpenCV,” https://opencv.org/, (Acesso em 21/06/2021).

[7] “Handwritten Math Symbols Dataset”, https://www.kaggle.com/xainano/handwrittenmathsymbols, (Acesso em 21/06/2021).

[8] “Welcome to Deep Learning (CNN 99%)” https://www.kaggle.com/toregil/welcome-to-deep-learning-cnn-99/notebook, (Acesso em 23/06/2021).

[9] “Cat Photo by Edgar on Unsplash” https://unsplash.com/photos/nKC772R_qog, (Acesso em 29/06/2021).

[10] “Uso de Internet, Televisão e Celular no Brasil”, https://educa.ibge.gov.br/criancas/brasil/2697-ie-ibge-educa/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html, (Acesso em 05/07/2021).