Prisma: Um classificador de imagens para aprendizado de idiomas

Marcos G. Guimarães Filho[1], Rafael P. Lourenço, Anderson S. do Nascimento[3]

[1,2,3] Sistemas de Informação – Universidade do Grande Rio (UNIGRANRIO)
Caixa Postal 1.160 – 25.071-202 – Duque de Caxias – RJ – Brazil

[1,2] {rafa,marcosgeovane}@unigranrio.br,
[3] anderson.nascimento@unigranrio.edu.br

Abstract. *In this paper, we propose a solution that complements language learning, focusing on factors such as lack of time, lack of opportunity and lack of motivation, with the solution being something that users can use anywhere and without an Internet connection, and also focusing on the learning of object-related information at the desired language. To do that, we explore topics like Machine Learning and platforms like TensorFlow, which allow us to retrain and utilize classification easily and TensorFlow Lite, the latter being a tool that converts TensorFlow models to versions optimized for mobile devices. Finally, we note the knowledge acquired during development and the solution's uses.*

Resumo. *Neste trabalho nos propomos a criar uma solução que complemente o aprendizado de linguagens, ao focar em fatores como a falta de tempo, falta de oportunidade e falta de motivação, sendo esta uma solução que os usuários podem usar em qualquer lugar e sem uma conexão à internet, com foco especial ao aprendizado de informações relacionadas a objetos na língua desejada. Para este fim, exploramos tópicos como Machine Learning e plataformas como o TensorFlow, que nos permitem retreinar e utilizar modelos de classificação de imagens facilmente e o TensorFlow Lite, que converte modelos do TensorFlow em versões otimizadas para dispositivos móveis. Por fim, notamos conhecimentos adquiridos e os usos da solução.*

1. Introdução

No Brasil há cada vez mais interesse no aprendizado de outros idiomas, com este interesse sendo impulsionado não só pela busca de melhores oportunidades de trabalho tanto no Brasil quanto em outros países que requerem estes idiomas, mas também pelo maior alcance e impacto social e acadêmico [FINARDE; ROJO 2015] que o mundo globalizado permite obter. Porém, por falta de tempo, oportunidades ou de motivação, um indivíduo pode se sentir desencorajado a aprender uma nova língua, ou tentar e não obter sucesso pois, segundo Pascual (1999), as atitudes do aluno em direção à língua desejada determinam, pelo menos em parte, o sucesso do aprendizado da linguagem. Há também a falta do aspecto lúdico do aprendizado tradicional, onde a presença ou não da

diversão e a manutenção do interesse durante o aprendizado dependem inteiramente das estratégias utilizadas pelo docente.

Propomos uma solução que tem por objetivo ajudar usuários interessados em aprender um novo idioma, complementando o ensino por outros meios ao permitir que os mesmos tirem fotos de objetos a sua volta e, usando a classificação de imagens de um modelo de Deep Learning, obtenham o nome daquele objeto na sua linguagem e na linguagem de sua escolha, como o Inglês, além de permitir ouvir e praticar a sua pronúncia do objeto nesse idioma; Porém visto que o usuário só irá aprender sobre uma quantidade limitada de objetos se ficar restrito a algum local específico, é vital que a solução possa ser utilizada em qualquer lugar, e é por isto que escolhemos disponibilizá-la em dispositivos móveis.

Por fim, utilizaremos o *feedback* dos próprios usuários para melhorar a detecção do modelo, as traduções e, por consequência, a eficácia da solução com o passar do tempo e explicamos como validaremos o impacto da nossa solução no conhecimento dos usuários.

2. Deep Learning

O *Deep Learning* é uma das áreas de *Machine Learning*[GULCEHRE 2015], baseada na aprendizagem de representações de dados. O *Deep Learning* descreve uma máquina que possa aprender múltiplos níveis de representação e abstração, que permite que informações coerentes sejam extraídas de imagens, sons e texto. A técnica utiliza várias camadas de unidades de processamento não-lineares para processar a informação.

3. Ferramentas Utilizadas

3.1. Python

Para a criação dos modelos de treino será utilizado o Python, por ser uma das principais linguagens de programação para *Machine Learning* e *Deep Learning* possuindo várias as bibliotecas/frameworks para tornar a implementação do projeto possível. Esta linguagem tem a capacidade de criar abstrações de alto nível para estes frameworks que permite que tudo seja tratado como um objeto e que não haja preocupação com o gerenciamento de memória e as peculiaridades da programação com CUDA[1], sem deixar de obter seus benefícios.

3.2. TensorFlow e TensorFlow Lite

Também será utilizado o TensorFlow, que é uma biblioteca open-source para computação numérica e machine learning em grande escala. Ele é utilizado para treinamento e criação de redes neurais para detectar e decifrar padrões e correlações e

[1] CUDA, sigla para Compute Unified Device Architecture, é uma tecnologia possibilita o uso de computação paralela, permitindo usar a GPU para realizar os treinos do modelo de rede neural de forma mais rápida em hardware compatível.

foi desenvolvido pela divisão Google Brain[2]. Utilizaremos também as bibliotecas do TensorFlow Lite para converter o modelo gerado pelo TensorFlow em uma versão que sacrifica um pouco da precisão das classificações, mas é otimizado para celulares, possuindo um menor tempo de processamento e ocupando menos espaço.

3.3. MobileNet

O modelo pré-treinado utilizado foi o MobileNet[3] pois ele foi feito especificamente para dispositivos mobile, e foi optimizado para ser pequeno e mais eficiente em dispositivos com menos recursos ao custo de um pouco da sua precisão na identificação de objetos. Retreinaremos apenas as últimas camadas deste modelo para adicionar novas classificações (objetos); A Figura 1 mostra uma simplificação da estrutura do modelo que transforma representações de baixo nível, como pixels em representações de alto nível, como categorias:

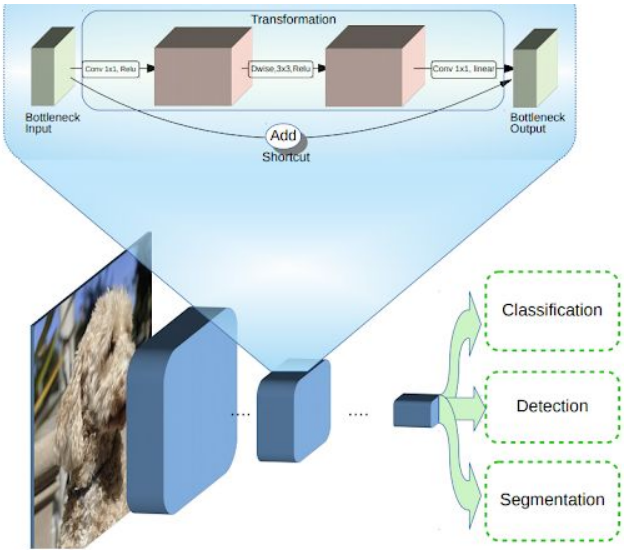

Figura 1: Simplificação da estrutura do modelo MobileNet. A primeira camada representa os pixels e *metadata* da imagem, enquanto a segunda camada representa as transformações feitas que tem como resultado a última camada, que já pode ser utilizada para classificação.

3.4. Outras Ferramentas

1. Git (Bitbucket) para versionamento do código-fonte do cliente e do servidor.
2. PostgreSQL para armazenamento das imagens e sugestões enviadas pelos usuários.

[2] https://ai.google/research/teams/brain

[3] MobileNet é uma família de redes neurais de visão de máquina projetados tendo dispositivos móveis em mente, suportando funcionalidades como classificação e detecção.

6. Desenvolvimento da Solução

6.1. Cliente

Para desenvolvermos o aplicativo, usamos a IDE do Visual Studio Code[4] em prol dos seus benefícios à agilidade do desenvolvimento.

O propósito do cliente é permitir que usuários tirem fotos de objetos, e, com a ajuda do classificador instalado, obter o nome do objeto em questão, que então pode ser traduzido para qualquer linguagem, gerando conhecimento. Caso o objeto não seja identificado, o cliente permitirá que o usuário ajude no treinamento do modelo ao enviar a imagem e o nome sugerido do objeto em questão, ou na tradução ao enviar a tradução correta de um nome de objeto.

O cliente também permitirá que o usuário ouça e pratique a pronúncia dos nomes de objetos utilizando a solução de reconhecimento de voz do próprio dispositivo.

6.1.1. Desenvolvimento usando React Native

O React Native é uma plataforma que tem como objetivo gerar código nativo a partir de uma única base de código escrita em Javascript/Typescript[5]. Diferente de outras plataformas como o Ionic, que são focados na estilização do aplicativo e só permitem o uso de funcionalidades de mais baixo nível, como a câmera, por meio de componentes específicos, o React Native obtém esse objetivo ao criar duas dimensões: A dimensão Javascript e a dimensão Nativa.

A dimensão Javascript é onde a parte do código comum a todas as plataformas se encontra, e deve ser onde a maior parte do código está, contendo tanto a lógica de negócio quanto a especificação da interface está, enquanto a dimensão Nativa contém as implementações dos componentes de interface e implementações de componentes de mais baixo nível. As duas dimensões se comunicam por meio de um barramento criado pela biblioteca, simplesmente chamada de "ponte". Por meio desta, a dimensão Javascript envia comandos para o lado nativo, usando mensagens como por exemplo "construa uma caixa de texto de 50 *pixels* de altura e 100 *pixels* de largura nessas coordenadas", e "tire uma foto", enquanto a dimensão Nativa envia os resultados dessas ações de volta por meio de *callbacks*, que são eventos que disparam funções no lado Javascript. Outro ponto interessante é que como código é usado para gerar a interface, é possível criar métodos que retornam certos componentes da tela, como exemplificado na Figura 2 abaixo.

---

[4] O Visual Studio Code é uma IDE com uma interface leve, simples e a rápida, possuindo uma vasta gama de extensões e boa integração com qualquer fornecedor de versionamento usando GIT.

[5] https://facebook.github.io/react-native

```
public state:FilmesState = {filmes: []};

render() {
    return (
        <View>
            {
                this.state.filmes.forEach(filme => {
                    this.criarAnuncioFilme(filme.id, filme.titulo, filme.descricao);
                })
            };
        </View>
    );
}

criarAnuncioFilme(id:number, titulo:string, descricao:string){
    return <Text key={id}>{titulo}, {descricao}</Text>
}
```

Figura 2: Exemplo de criação de uma tela simples, sem estilização, onde filmes seriam exibidos com título e descrição.

Como mostrado na Figura 2, propriedades não estáticas, como filmes, não são guardadas como simples propriedades à nível de classe como em outros paradigmas de programação. No React Native, tudo o que não é estático precisa estar dentro de uma propriedade denominada *State* que, conforme o nome sugere, mantém o estado da aplicação e é usada para atualizar a tela quando o método herdado da classe React.Component, setState, é executado. Quando isto acontece, o método render() pode ser usado ou não(Figura 3) para atualizar a interface usando os novos valores do state.

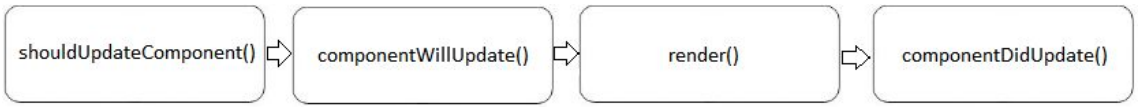

Figura 3: Ilustração do ciclo de vida da interface, indo da esquerda para a direita, quando o método setState(state) é utilizado. (BACHIR-CHERIF, 2019)
Imagem alterada, removendo o passo de rerrenderização por questões de simplicidade.

6.1.2. Reconhecimento das Imagens

Para a classificação das imagens, utilizamos o TensorFlow Lite, que é implementado tanto no iOS quanto no Android pela biblioteca do React Native React-Native-TensorFlow-Lite. O processo é bastante simples, sendo necessário somente repassar o local onde o modelo treinado se encontra e o tamanho, com largura e altura iguais, conforme pode ser visto na Figura 4. Após uma foto ser tirada pelo usuário, utilizamos o classificador instanciado para identificar o que há na foto, que então retorna uma lista de probabilidades e uma lista de resultados, ordenados pela maior probabilidade.

```
try {
    this.classifier = new TFLiteImageRecognition({
        model: 'mobilenet_v2_1.0_224.tflite',
        labels: 'labels.txt'
    })

} catch (err) {
    console.log('Error[TFLite]:', err);
}
```

Figura 4: Instanciação do classificador.

6.2. Desenvolvimento do Servidor

Desenvolvemos o servidor usando o Visual Studio Code, pelos mesmos motivos, usando Javascript junto do Node.JS. O servidor ficará encarregado de receber imagens de sugestões de nome, tradução, detecção correta ou detecção incorreta dos usuários via REST e assim conseguirá dados para a melhoria do modelo de classificação de objetos.

6.2.1. Node.JS

O Node.JS é um *software* feito com conectividade e paralelização de processamento em mente que executa código escrito em Javascript para permitir que o mesmo tenha acesso a funcionalidades do sistema operacional como leitura de arquivos e utilização de protocolos de rede, como o protocolo HTTP, para agir como um cliente ou servidor. O utilizaremos para executar o software que irá receber e tratar o *feedback* dos usuários a fim de melhorar o modelo de classificação de imagens e as traduções realizadas.

6.2.2. Recebendo Imagens dos Usuários

Ao receber uma requisição do tipo *post*, o servidor armazenará a imagem enviada, a categoria da sugestão, que pode ser a adição de nome, solicitação de alteração de tradução ou avaliação de detecção, e dados como o nome ou alteração sugeridos, ou "correto" ou "incorreto" com o nome real do objeto na imagem para avaliações de detecção. Essas requisições ficarão armazenadas como "Não avaliadas" até avaliação manual e serão utilizadas para o treinamento de um determinado objeto no modelo; Caso não sejam verídicas, serão descartadas ou utilizadas para treinar a detecção de outro objeto ao alterar o nome relacionado à imagem.

6.2.3. Retreinamento do Modelo

Para que fosse possível retreinar o modelo foi utilizado um script[6] em Python disponibilizado pelo TensorFlow Hub. No script é necessário passar o modelo do MobileNet a ser utilizado e o caminho onde estão as imagens para treino. Também utilizamos alguns parâmetros no script para pré-processar os dados do modelo para obter maior precisão, sendo eles:

[6] https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py

--flip_left_right - Aplica efeito de "espelho" aleatoriamente em 50% das imagens durante o treinamento. Ao especificar esse parâmetro, preparamos o modelo para imagens tiradas de outros ângulos/direções.
--random_crops - Corta algumas imagens de forma randômica durante o treinamento.
--random_scale - Escala algumas imagens de forma randômica durante o treinamento.
--random_brightness - Controla o brilho das imagens de forma randômica durante o treinamento.

Após experimentação e verificação da precisão do modelo em cima da base de teste, decidimos utilizar o valor de 10 nesses parâmetros para pré-processar 10% dos dados de treinamento de modo a cobrir casos de imagens de má qualidade/de orientação diferente, porém sem prejudicar a detecção de imagens em condições normais; Com isto o comando utilizado para o retreinamento do modelo ficou como pode ser visto abaixo:

```
python scripts/retrain.py --image_dir tf_files/classes --how_many_training_steps 1000
--output_graph tf_files/graph.pb
--output_labels tf_files/labels.txt
--tfhub_module
https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/2
--random_crops 10 --random_scale 10 --random_brightness 10 --flip_left_right
```

Após o processo de retreino, é gerado um modelo em formato de protobuf (.pb), que é um modelo de definição de estruturas de dados genérico desenvolvido pela Google, que então pode ser usado para gerar classes de domínio em várias linguagens, dentre elas C#, Python, Java, e Javascript. No nosso caso, o protobuf possui as definições do gráfico e "pesos" do modelo retreinado.

Então realizamos pré-processamentos no modelo para trocar um pouco da precisão por maior desempenho usando o formato TFLite utilizando o script tflite_convert, incluso na instalação do TensorFlow:

```
tflite_convert
--output_file=/tf_files/mobilenet_v2_1.0_224.tflite
--inference_type=FLOAT --input_arrays=input --mean_values=128
--output_arrays=MobilenetV2/Predictions/Reshape_1 --std_dev_values=127
--input_shapes=1,224,224,3 --graph_def_file=/tf_files/graph.pb
```

7. Plano de Testes da Solução

Pretendemos testar o impacto da solução ao aplicar dois testes, um de associação de nomes a imagens e um de pronúncia à uma parcela da população, com idades entre 10 e 40 anos, pois a capacidade aprendizado de novas linguagens é melhor a partir de 10 anos e muito baixo aos 40 anos de idade [Hartshorne, Tenenbaum e Pinker 2018], guardar os resultados, porém não mostrar as respostas corretas, e então oferecer e incentivar o uso do aplicativo para então reaplicar os testes após um tempo determinado, com o objetivo de obter tanto o nível de engajamento do aplicativo com o usuário quanto a sua eficácia no processo de aprendizado da nova linguagem.

8. Conclusão

Com o aplicativo desenvolvido, esperamos que se torne mais fácil o processo de aprendizado de outras linguagens e que o conhecimento de informação da língua desejada sobre os objetos escaneados complemente o ensino de outras partes do aprendizado de uma língua, como a gramática e a pronúncia, desde que o usuário encontre variados objetos durante a sua rotina ou durante viagens ao exterior.

Além disto, para a criação da solução, foi treinado um modelo de reconhecimento de objetos do TensorFlow. Este modelo e os dados usados para treiná-lo ficarão disponíveis publicamente conforme os mesmos forem melhorados e amadurecidos e poderão ser utilizados como ponto de partida de outros projetos que desejam utilizar a classificação de objetos.

Referências Bibliográficas

FINARDE, Kyria Rebeca; ROJO, Ramón Andrés Ortiz. GLOBALIZATION, INTERNATIONALIZATION AND EDUCATION: WHAT IS THE CONNECTION?. International E-Journal of Advances in Education, [*S. l.*], ano 2015, v. 1, n. 1, p. 20-20, 1 jan. 2015. Disponível em: http://ijaedu.ocerintjournals.org/tr/download/article-file/89347. Acesso em: 18 out. 2019.

SANDLER, Mark; HOWARD, Andrew. MobileNetV2. *In*: SANDLER, Mark; HOWARD, Andrew. MobileNetV2: The Next Generation of On-Device Computer Vision Networks. 1. ed. Google AI Blog: Google, 25 jan. 2017. Disponível em: https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html. Acesso em: 18 out. 2019.

FRATCHET, Marvin. Understanding the React Native bridge concept. [S. l.], 22 jun. 2019. Disponível em: https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8. Acesso em: 12 mar. 2018.

BACHIR-CHERIF, Salah Eddine. Understanding React Native Component Lifecycle Api. 18 ago. 2019. 1 figura. Disponível em: https://blog.usejournal.com/understanding-react-native-component-lifecycle-api-d78e06870c6d. Acesso em: 19 maio 2018.

GULCEHRE, Caglar. Welcome to Deep Learning. [S. l.], 1 dez. 2015. Disponível em: http://deeplearning.net/. Acesso em: 13 out. 2018.

HARTSHORNE, Joshua K.; TENENBAUM, Joshua B.; PINKER, Steven. A critical period for second language acquisition: Evidence from 2/3 million English speakers. Cognition, [*S. l.*], ano 2018, v. 177, n. 1, p. 263-277, 1 ago. 2018. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6559801/. Acesso em: 18 out. 2019.