



ARTHUR FERREIRA POMPILIO
TOMÉ DA COSTA LIMA
RAPHAEL AUGUSTO DA SILVA LINS

**Projeto de Inteligência Artificial - Distinção Entre
Paisagens Naturais e Urbanas**

Recife
2023

Descrição do problema

O problema consiste em reconhecer paisagens e dizer se a paisagem em questão é urbana ou se contém elementos urbanos em um projeto com uso de inteligência artificial. As características de uma paisagem são distintas em cada caso por conter elementos específicos, em paisagens urbanas, foi considerado qualquer elemento urbano, então uma casa no campo deve ser reconhecida como uma paisagem urbana, por exemplo.

Por meio de pesquisa no Google, algumas palavras-chave foram utilizadas para encontrar imagens relevantes para a base de dados, como ambientes internos, lugares característicos e diversas paisagens naturais distintas, incluindo fotografias diurnas e noturnas. Foram utilizadas 50 imagens para cada classe.

Classes

- Natural

As paisagens naturais são compostas por uma variedade de características que podem ser observadas e categorizadas de várias maneiras diferentes de acordo com o seu relevo, geografia, fauna, flora, clima e hidrografia. Porém, no projeto, buscamos identificar os elementos comuns a maioria das paisagens naturais em contraste àquelas construídas ou alteradas pela mão humana.

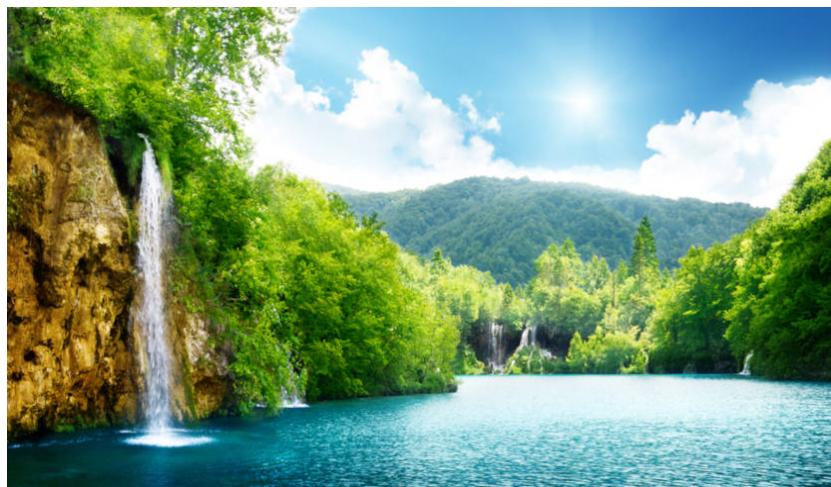


Figura 1: Paisagem de um lago com cachoeiras em uma floresta

- Artificiais

As paisagens artificiais, caracterizadas pela ação humana sobre elas, também têm também grande variação dentro da mesma classe. Elas não se limitam apenas aos asfaltos das grandes cidades, mas também a fábricas, petrolíferas em alto mar, fazendas e quadras esportivas.



Figura 2: Fotografia da Avenida Governador Agamenon Magalhães, Recife

Dificuldades

- Seleção de imagens relevantes para o treinamento do modelo:

Como só haviam 100 imagens de exemplo para o treinamento, deveriam ser feitas algumas escolhas de qual seria o foco, então buscamos uma ou duas imagens de cada tipo para as classes, ou seja, se tentamos treinar o programa para reconhecer mares e oceanos, utilizamos duas imagens deste tipo.

- Semelhança entre algumas paisagens artificiais com naturais

Algumas paisagens particulares, apesar de claramente artificiais ao olho humano, acabam por “enganar” o modelo treinado, gerando classificações falsas, porém com alta confiança. Por exemplo, campos de futebol, parques e campos de basquete, devido à presença de elementos aparentemente naturais tais como predominância da cor verde e a ausência de construções, são todos consistentemente classificados como naturais, com confianças de 80% a 95%.

- Dificuldade na integração com AppInventor

Inicialmente, o objetivo era fazer uma aplicação no *App Inventor* que permitisse ao usuário tirar uma foto de um cenário qualquer usando a câmera de seu celular e que, assim, permitiria ao modelo classificar a imagem em natural ou artificial. Porém, o suporte para integração do aplicativo com o modelo do ML4K não está mais recebendo suporte por parte da equipe do *App Inventor*, haja vista as múltiplas issues abertas no repositório <https://github.com/kylecorry31/ML4K-AI-Extension/issues>. Devido a essa gama de problemas com a plataforma do MIT, decidimos pivotar a proposta do projeto e passar a desenvolvê-lo no Scratch, uma plataforma de ensino de código com integração robusta com o ML4K.

Metodologia usada para o Treinamento

Como cada classe é bastante ampla, foi preciso pensar em como capturar a essência de cada uma, assim como escolher imagens específicas muito distintas entre si, o que é um desafio, pois não queremos acabar fazendo *overfitting* e nem *underfitting*, então encontrar um equilíbrio entre esses eventos é uma dificuldade. Para isso, decidimos utilizar de uma a três imagens de cada categoria, dentro de cada classe, com base em quão frequentemente essas paisagens são referenciadas. Por exemplo, para “ensinar” o programa a reconhecer o mar, utilizamos duas imagens, uma por cima do mar, e outra por dentro, para capturar a essência de cada ambiente, e não utilizamos imagens demais por sabermos que é uma paisagem facilmente reconhecível pela quantidade de tons de azul e pelos ruídos característicos das ondas do mar.

O mesmo foi pensado para florestas, desertos, cavernas, entre outros, e para paisagens urbanas utilizamos diversas cidades ao redor do mundo, e também lugares que imaginamos que o programa teria dificuldade em reconhecer, para que ele consiga reconhecer bem uma ampla gama de paisagens.

Acurácia no Conjunto de Testes

O programa tem uma precisão boa em diversas categorias, mas existem algumas falhas consistentes que nós não conseguimos evitar, um exemplo disso é o caso de campos de futebol ou parques. O campo de futebol, por ser basicamente só grama com alguns detalhes pequenos, não é classificado como uma paisagem urbana, apesar de ser um artefato criado por humanos.



Figura 3: Campo de futebol

Para parques, acontece algo parecido, por ter uma grande quantidade de árvores, grama, e poucos detalhes urbanos.

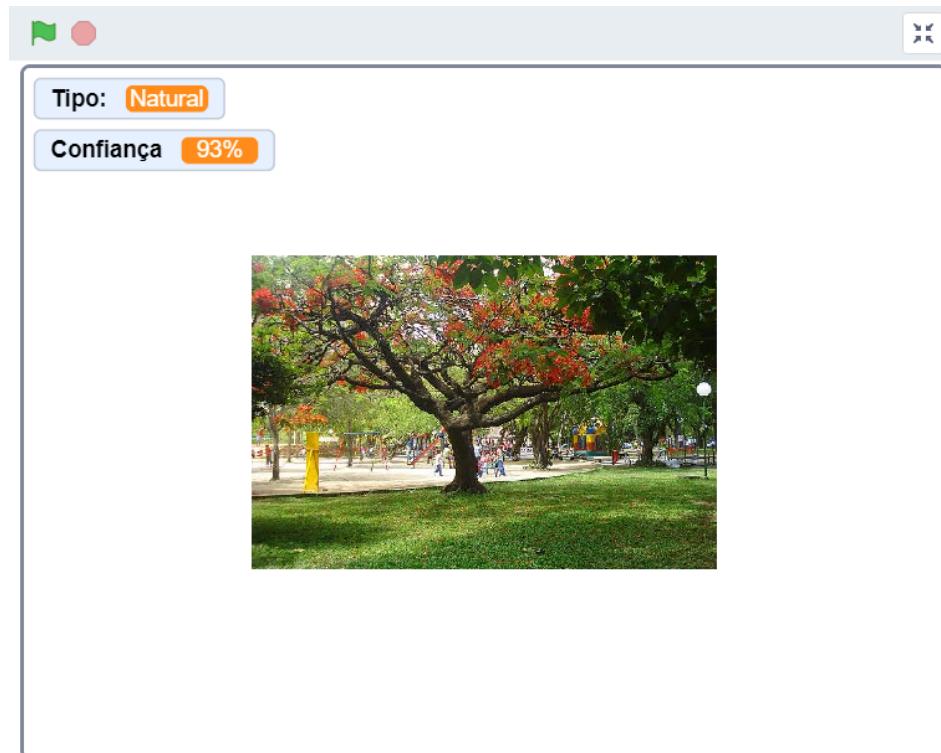


Figura 4: Parque da Jaqueira

De resto, pelo que fomos capazes de testar, o programa é bom em paisagens genuinamente urbanas, com prédios e casas, apesar de ser fraco em paisagens com muita grama e árvores. E também é capaz de reconhecer com facilidade

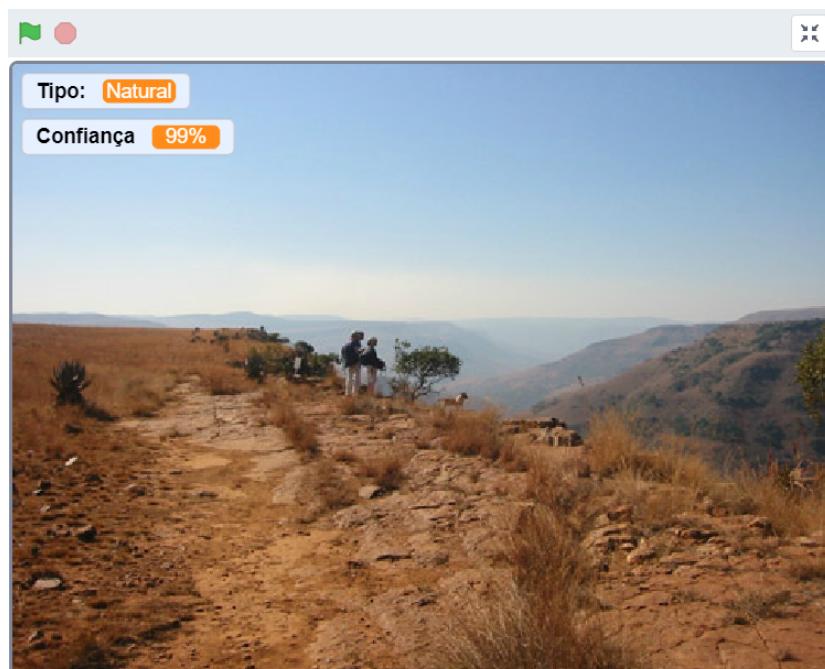


Figura 5: Grand Canyon

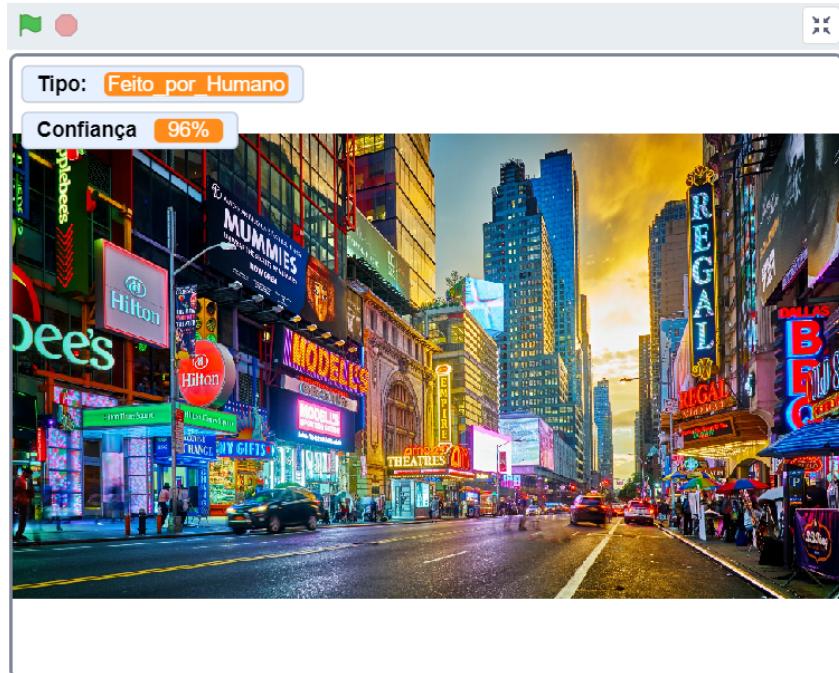


Figura 6: Nova York

Implementação do projeto

Na plataforma Scratch, foi elaborado um código simples que utiliza a integração da plataforma com o modelo treinado no ML4K.

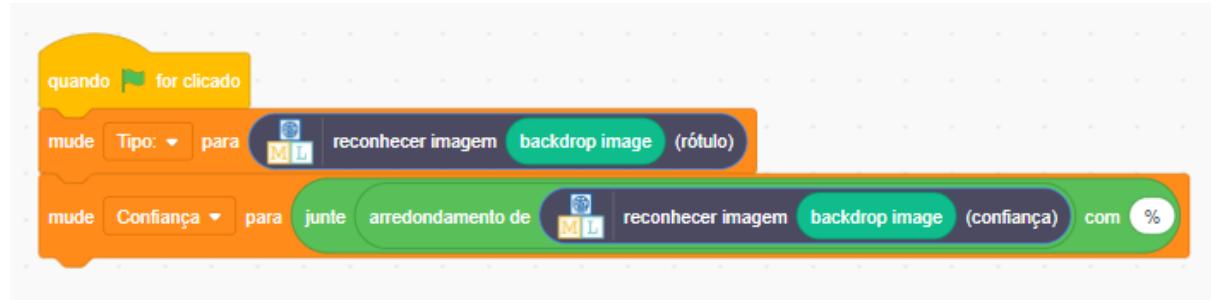


Figura 7: Código no Scratch

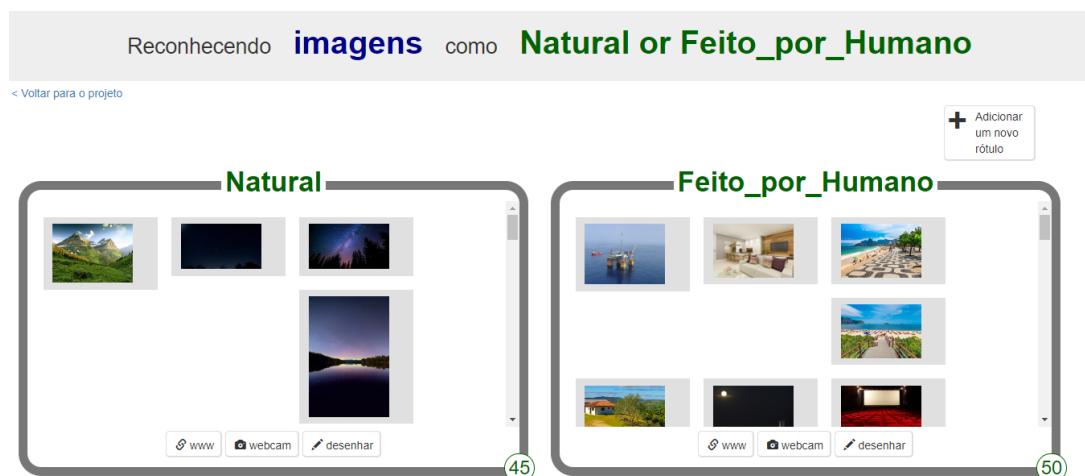


Figura 8: Algumas imagens usadas para treinar o modelo no ML4K

O programa toma a imagem de background como input, retornando, de acordo com o modelo, a classificação (“Natural” ou “Feito_por_Humano”) e sua confiança na classificação dada.

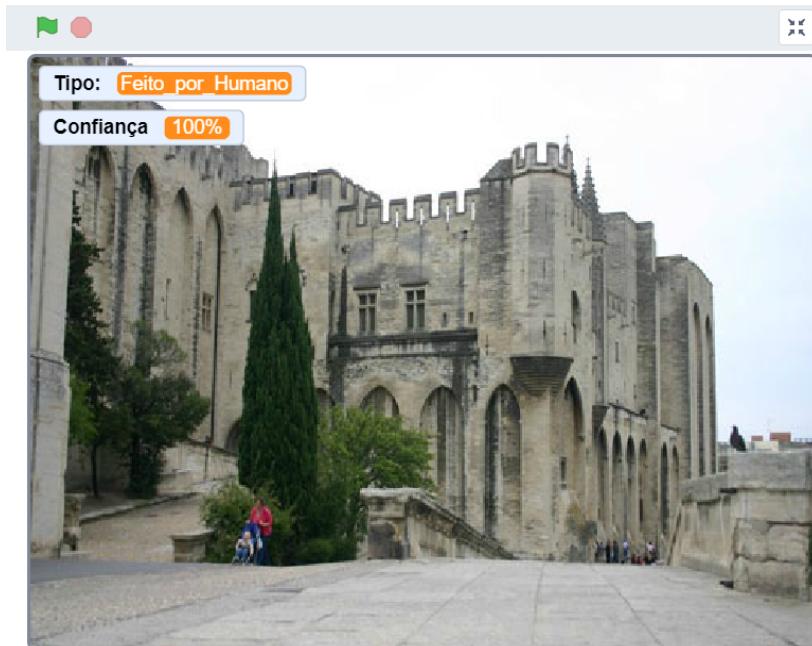


Figura 9: Castelo do banco de imagens do Scratch

Documentação e versionamento

Para acompanhar o desenvolvimento do projeto, e compartilhar em tempo real todas as alterações, o projeto foi upado e mantido no github pelo repositório público [projetoIC](#). Nele, é possível ver versões distintas do projeto, assim como tentativas anteriores de utilizar o site *App Inventor* e tentativas de um projeto completamente diferente de reconhecimento de texto. Dessa forma, nenhuma etapa foi desperdiçada e qualquer erro feito poderia ter sido facilmente desfeito.