UNIVERSIDADE FEDERAL DO ABC

MARCELO CARPINETTE GRAVE

GAMIFICAÇÃO E INTELIGÊNCIA ARTIFICIAL NA INTEGRAÇÃO DE SISTEMAS ONLINE COMO FERRAMENTA DE ENSINO

Trabalho de Graduação em Engenharia de Informação

Aluno: Marcelo Carpinette Grave

Profº Orientador: Dr. Mário Minami

Santo André – SP
2018

MARCELO CARPINETTE GRAVE

GAMIFICAÇÃO E INTELIGÊNCIA ARTIFICIAL NA INTEGRAÇÃO DE SISTEMAS ONLINE COMO FERRAMENTA DE ENSINO

Trabalho de Graduação apresentado à Universidade Federal do ABC, como requisito para obtenção do diploma no curso de graduação em Engenharia de Informação.

Orientador: Profº Dr. Mário Minami

Santo André – SP
2018

Resumo: A gamificação é um tema bastante atraente e envolvente tanto para adolescentes, jovens e adultos e, portanto, a tentativa de uso de estratégias envolvendo jogos e fins educativos é um fator motivante para o público e desafiador para os educadores. Um tema também muito atual é o uso de tecnologias ligadas à Inteligência Artificial, e *Machine Learning*, como Reconhecimento Automático de Imagens. Num mundo cada vez mais conectado, o uso de dispositivos de Tecnologia de Informação e Comunicação (TICs), como celulares, tablets e notebooks na construção de estratégias de ensino e aprendizagem, incluindo soluções online na nuvem, tornam-se cada vez mais relevantes, para não dizer necessárias. O presente trabalho procura exemplificar por meio de um jogo interativo com controle de um robô a integração de sistemas online usando conceitos de gamificação para fins educativos e princípios de inteligência artificial.

Palavras chaves: gamificação em educação, inteligência artificial, reconhecimento de imagem, TICs em educação.

Abstract: Gamification is a very attractive and engaging theme for teens, youth and adults, therefore trying to use it for educational games and goals is a motivating factor for the public and a challenge for educators. A very modern and present theme is usage of technologies related to Artificial Intelligence, Machine Learning such as Automatic Image Recognition. The more connected the world, the more relevant is the usage of information and communication technology (ICT) such as mobile phones, tablets and notebooks, even real-time cloud solutions, for building new teaching and learning strategies. The present research work exemplifies online systems integration through an interactive game using elements of artificial intelligence and gamification concepts for educational purposes.

Palavras chaves: gamification for education, artificial intelligence, image recognition, ICTs in education.

Sumário

Lista de Tabelas

Lista de Figuras

Figura 23: Credenciais do serviço de reconhecimento de imagem para serem usadas na chamada REST de teste de reconhecimento da Figura 21.

Figura 24: Setas para (a) a direita e (b) para esquerda

Figura 25: Resultados da saída do código da Figura 21 usando a interface visual do serviço com as entradas da Figura 24 respectivamente (a) e (b).

Figura 26: Exemplos de setas desenhadas a mão extraídas do vídeo gravado com qualidade do cozmo (320x240) usadas para treino

Figura 27: Interface visual de treino do serviço

Figura 28: Comparação entre imagens reconhecidas e não reconhecidas para o treino com 1000 imagens

1. Introdução

1.1. Contextualização

O processo de aprendizado do ser humano é algo intrínseco de cada ser e está condicionado, entre outras coisas, ao contato direto, variado e repetido com determinadas situações. Observando, repetindo, reproduzindo situações esse ser é então, depois de algum tempo de contato, capaz de tirar conclusões, tomar decisões, entre outras ações. Com esse aprendizado, ou treino, o ser humano é capaz de aplicar o conhecimento obtido de uma área em outra totalmente diferente sem sequer ter prévio conhecimento desta última.

Imagine a seguinte situação, são apresentadas imagens de um animal a uma pessoa que nunca viu tal animal antes, mesmo que esse fosse muito semelhante com algum outro animal. Após algumas variadas e diferentes exposições de tais imagens do animal, o humano seria capaz de identificar observando uma outra imagem não mostrada anteriormente, e ainda extrair mais informações antes não ditas previamente a ele.

Em outras palavras o ser humano é provido de inteligência natural e raciocínio lógico. Há diversos trabalhos relacionados a como enxergar essa forma natural de inteligência e como caracterizá-la. Noam Chomsky, por exemplo, afirma que a linguagem é uma propriedade inata ao cérebro humano, que influencia demasiadamente na formação da psicologia cognitiva [1], ou seja, o ser nasce com a predisposição de raciocinar e pensar .

E se uma máquina ou sistema pudesse agir ou raciocinar como um ser humano, ou ainda, e se pudessem observar, aprender e tomar decisões como seres humanos? Será que seríamos capazes de distinguir a diferença entre uma máquina e um ser humano no futuro?

O matemático e cientista da computação Alan Turing, um dos pioneiros na área de Inteligência Artificial, publicou seu trabalho intitulado por "Computing Machinery and Intelligence", considerado por muitos dos pesquisadores e cientistas como um dos pilares da área, em 1950. Ele inicia tal trabalho com a seguinte frase:

Eu proponho considerar a seguinte questão, 'Máquinas podem pensar?'.

Nesse trabalho, ele, genialmente, propõe o *Jogo da Imitação* que tem por propósito testar se máquinas ou sistemas tem a capacidade de se comportar de maneira inteligente e racional, ou seja, mais próxima de um ser humano racional [2].

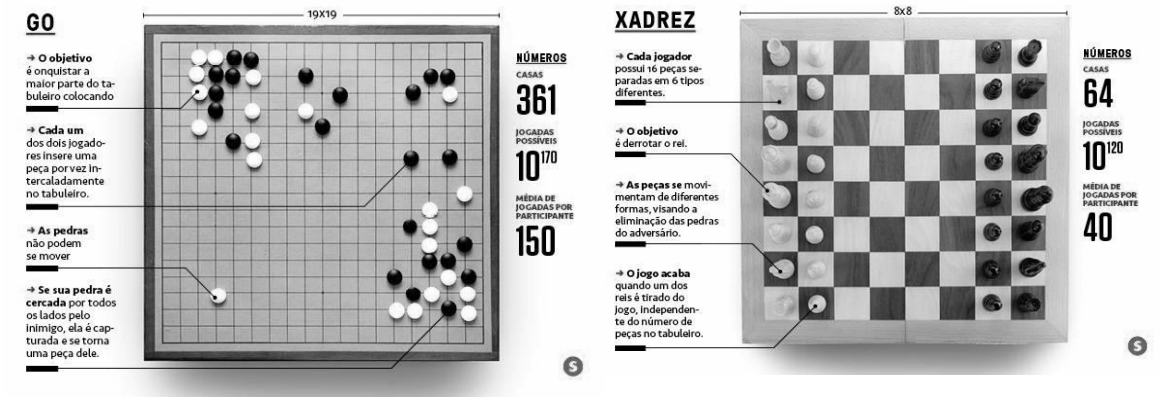
Desde a sua proposta, há mais de 68 anos, muitos sistemas e robôs tentam passar no teste de Turing. Há um prêmio chamado Loebner Prize criado por Hugh Loebner em 1990 com o intuito de avançar estudos em inteligência artificial premiando projetos que passassem pelo teste proposto por Turing de maneira mais metodológica, mas mesmo o próprio evento é muito criticado e controverso. Alguns

poucos desses sistemas ou robôs tiveram algum êxito ao longo desses anos, mas quase todos duvidosos ou com alguma ressalva [3]. Até os dias atuais, não há um consenso sobre se alguém já conseguiu passar no teste de Turing, bem como se o método proposto para avaliar o teste funcione bem de fato.

Nenhuma máquina, levando em consideração um senso mais geral, passou no teste de Turing até o atual momento, mas existem avanços tecnológicos em termos de hardware e software que estão tentando garantir que tal feito seja alcançado.

Entre os vários feitos na área de IA é possível citar o do computador Deep Blue da IBM que enfrentou o campeão mundial de Xadrez, o russo Garry Kasparov. Eles disputaram uma partida em 1996, onde Kasparov saiu vitorioso e outras cinco partidas ao longo de vários dias sendo a última em maio de 1997. O computador da IBM ganhou do considerado melhor jogador de xadrez do mundo em apenas uma partida, empatando outras duas e perdendo as restantes. Basicamente, Deep Blue tinha a capacidade de avaliar em torno de 200 milhões de posições por segundo que o possibilitou vencer o jogo na força bruta, prevendo os movimentos subsequentes de Kasparov, em outras palavras, esse é um jogo intitulado como determinístico e de informação perfeita [4]. Em teoria dos jogos, o xadrez é classificado como um jogo de informação perfeita, pois cada participante pode ver todas as peças a qualquer momento do jogo, ou seja, possui visão geral do jogo a todo momento diferentemente de jogos de cartas em que os participantes escondem sua cartas e só as mostram em momentos desejados [5].

Outro exemplo de jogo determinístico e de informação perfeita é Go que por alguns fatores explicados mais abaixo é bem mais complexo que o Xadrez. A ideia do jogo é bem simples, basta cercar uma ou mais peças do adversário e elas se tornam suas, ou seja, nenhuma peça é movida, mas apenas adicionada. Sua complexidade é muito maior que a do xadrez, como demonstrado na Figura 1 a seguir:



(a) (b)

Figura 1: Comparações de jogabilidade, dificuldade e complexidade dos jogos (a) Go e (b) Xadrez [6]

A quantia de casas do Go é muito maior que a do Xadrez, isso implica que a possibilidade de estratégias do jogo cresça significativamente, por consequência a quantia de jogadas possíveis, que é da ordem de 10^50, também. A quantia de movimentos possíveis do jogo é possivelmente maior que a quantia de átomos do

universo, logo, esse jogo não poderia ser vencido na força bruta, ou pelo menos não da forma que não projetado outros sistemas.

AlphaGo provou que isso é possível com uma combinação de Aprendizado profundo (Deep Learning - DL) e Busca em árvores avançadas (Advanced tree search), em outras palavras, usando o tabuleiro como entrada essa redes neurais criaram um grande número de camadas com milhões de conexões bioinspirada em neurônios humanos. Simplificando, o sistema foi treinado com algo em torno de 100.000 partidas de jogadores amadores e iniciantes a princípio para ajudá-lo a desenvolver sua própria tática de jogo e maneira como humanos jogam tal jogo. Após isso, o sistema disputou milhares de partidas contra ele mesmo, aprendendo com seus erros e os usando como dados de re-treino, para isso o Aprendizado por reforço (Reinforcement Learning - RL) entra na história. Duas redes neurais artificiais diferentes foram usadas: a rede de políticas ("policy network") que é responsável por selecionar a próxima jogada e a rede de valores ("value network") que é responsável por prever o ganhador do jogo naquele instante de tempo do jogo (Vide Figura 2).

A Figura 2a ilustra a política de lançamento ("rollout policy") e a rede de políticas ("network policy") de aprendizagem supervisionado (SL) que são treinadas para prever movimentos de melhores jogadores humanos em um conjunto de dados de posições. Uma rede de políticas ("network policy") usando aprendizagem de reforço (RL) é inicializada para a rede de políticas de SL e é então melhorada usando aprendizado do gradiente para maximizar o resultado (ou seja, ganhar mais jogos) em relação às versões anteriores da rede de políticas. Um novo conjunto de dados é gerado. Ele é composto por jogos em que o sistema disputa com ele mesmo usando a rede de políticas de RL. Finalmente, uma rede de valores ("value network") é treinada usando regressão para prever o resultado esperado (ou seja, se o jogador atual ganha) em posições do conjunto de dados citado anteriormente. A Figura 2b ilustra, por sua vez, a representação esquemática da arquitetura de rede neural usada no AlphaGo. A rede de políticas toma uma representação das posições s do tabuleiro como sua entrada, passa por muitas camadas convolucionais com os parâmetros σ (SL da rede de políticas) ou ρ (RL da rede de valores), e gera uma distribuição de probabilidade ou sobre movimentos possíveis a, serem representados por um mapa de probabilidade do tabuleiro. A rede de valores usa, de maneira semelhante, muitas camadas convolucionais com parâmetros θ, mas produz um valor escalar $v\theta(s)$ que prediz o resultado esperado na posição s

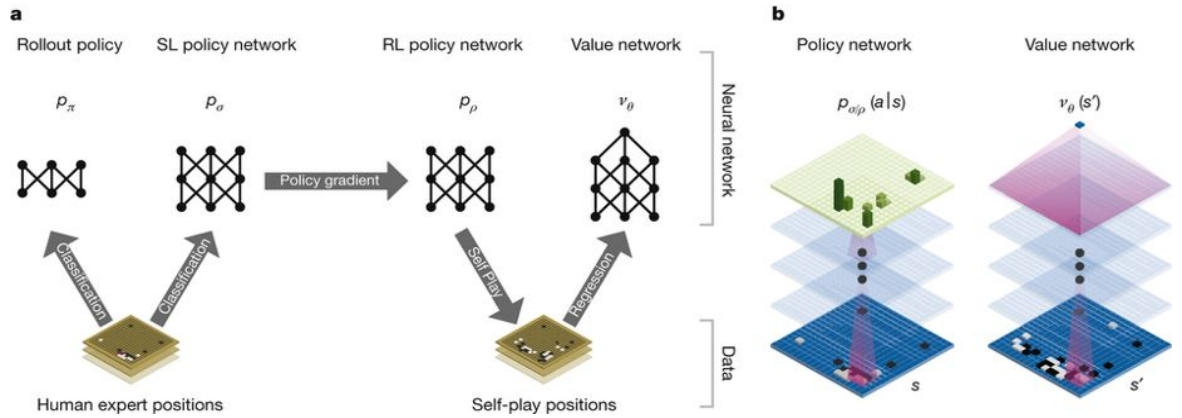


Figura 2: (a) metodologia de treino das redes neurais usadas (b) e valores preditos de saída de cada delas usadas durante o jogo [7]

A saga de AlphaGo na obtenção de sucesso e melhoria de desempenho surgiu quando derrotou um dos campeões europeus de Go, Fan Hui em outubro 2015. Para Fan Hui foi surpreendente ser derrotado por uma máquina, porém o grande feito do time AlphaGo foi derrotar Sul-Coreano Lee Sedol, o considerado melhor jogador do mundo. O sistema foi capaz de, incrível e surpreendentemente, deixar Lee boquiaberto por vezes algumas jogadas em uma das cinco partidas que jogaram. AlphaGo venceu quatro das cinco partidas (Vide Figura 3) e Lee obteve sucesso em 15 março de 2016, pois fez uma jogada bem arriscada que tinha probabilidade 0.0007% de acontecer, algo que apenas 1 entre 10000 jogadores fariam [8].

Figura 3: Resultado visual da primeira partida entre AlphaGo e Lee Sedol, onde a vitória foi equipe do DeepMind AlphaGo [9]

Deep Blue e AlphaGo foram surpreendentes derrotando os melhores jogadores de suas modalidades e o interesse em aplicar tais soluções no mundo real é grande, porém em muito casos os cenários são mais desafiadores em que jogos de informação perfeita não são capazes de cobrir.

Dado isso, um jogo que tem tomado muita atenção dos pesquisadores é Startcraft (Vide Figura 4), desenvolvido pela Blizzard Entertainment, um jogo de plataforma em tempo real que é famoso Coréia do Sul. O desafio em StarCraft está nele ser parcialmente observável, ou seja, partes do mapa ao redor de onde o jogador está podem ser observadas, nesse caso diferencia do Xadrez e do Go, pois o estado corrente não é determinado apenas pela jogada passada, mas pelas ações simultâneas de todos os jogadores. Isso implica que a tomada de decisão deve ser feita sem um conhecimento completo e profundo do que os outros jogadores estão fazendo naquele exato momento. Em geral, as estratégias se baseiam em observar e vigiar o adversário, buscando tentar antecipar seus ataques e planos.

Figura 4: StarCraft II é um jogo de ficção científica baseado em estratégias em tempo real da empresa Blizzard Entertainment

Nesse cenário, Starcraft é capaz de ilustrar um pouco da aleatoriedade do mundo real, bem pouco necessita de uso de memória, habilidade de planejar mais a longo prazo e capacidade de adaptar-se com base em novas informações. Para tal feito Blizzard e o DeepMind, empresa que criou o sistema que derrotou Lee Segol, anunciaram uma plataforma programática SC2 que todo pesquisador pudesse ter acesso. A tal plataforma SC2 de desenvolvimento de IA foi disponibilizada em agosto de 2017 [10]. Para garantir que a máquina ganhe inteligência e velocidade de ação da máquina que é significativamente mais rápido que um ser humano o sistema será limitado a 300 ações por minuto (velocidade de ação que jogadores profissionais atingem). Mesmo com toda essa tecnologia em mãos, e a derrota de Lee para AlphaGo muito jogadores de Starcraft ainda não acreditam que o novo sistema proposto possa ganhar um dia de um jogador profissional. Esse também era o sentimento das pessoas uma ano antes de AlphaGo conseguir alcançar seu ponto máximo [11].

Ainda há muita especulação sobre o que pode ser feito com IA, onde tais sistemas conseguem ser aplicados no mundo real, mas muitas aplicações visam o uso da IA na área da saúde, com foco em Seguradoras de Saúde e Hospitais. Essa área é detentora de dados exclusivos e muito interesse em pesquisa, pelo fato de

lidarem com vidas humanas e necessidade de avanço tecnológico constante, esse é um mercado bem rentável.

Entre diversos projetos de pesquisa, alguns são altamente promissores e disruptivos, entre os possíveis promissores mais estáveis e conhecidos estão Watson da IBM, InnerEye da Microsoft, DeepMind do Google, entre outros [12]. A maioria deles focada em processamento de imagem e/ou texto com o intuito de extrair informações que possam ajudar os profissionais da área da saúde na tomada de decisão e identificação mais ágil de possíveis problemas complexos como câncer.

Não está apenas em processar imagens, texto, áudio, ou qualquer outro dado o futuro de IA. O futuro também tem um pouco da ficção do filmes e séries como Exterminador do Futuro, Matrix, Blade Runner, Ex-Machina, West World entre outros, que usam IA como uma forma humana. Um caso, particular conhecido é a Sophia, robô idealizado pelo ex-funcionário da Disney David Hanson que montou uma empresa a Hanson Robotics em Hong Kong para construí-la. Ele foi projetada para se parecer mais com seres humanos comportando-se como tais, em outras palavras, para ser tão parecida com um ser humano que não seríamos capaz de distingui-la de um de nós e, sendo assim, seria o primeiro androide a passar no teste de Turing. Atualmente, ela já capaz movimentar partes da face, ainda um pouco robótica e com textura interessante, que permite-a expressar sentimentos conforme se comunica (Vide Figura 5) [13].

Figura 5: Sophia, robô humanóide, inspirada em Audrey Hepburn capaz de se comunicar, gesticular e, talvez, pensar como um ser humano.

Imagine agora, que o cérebro da Sophia pudesse ter ser, ao invés de apenas circuitos, um amontoado de células humanas compondo um cérebro conectadas por circuitos tão pequenos que seriam quase imperceptíveis, ou seja, cérebro biológico em corpo robótico interconectado com circuitos.

O professor Kevin Warwick da Universidade de Reading em Berkshire na Inglaterra estuda e testa métodos invasivos e não invasivos para uso de interfaces humano-máquina**.** O método invasivo, que é o mais novo, estuda, propõe e testa o

implante ou conexão de neurônios com estruturas nervosas prontas de animais e humanos.

Ele criou células neuronais *in vitro* que consistiu em separar pedaços do tecido cortical de roedores, os deixa crescer em câmaras especiais que fornece condições adequadas para o crescimento. Tais câmaras possuem uma base composta por microelétrodos que são responsáveis por garantir a comunicação com outros módulos e permitir que dados sejam gravados em tempo real.

Após implantados em Kevin, ele obteve sucesso em alguns testes como por exemplo controle de um braço via internet, comunicação telegráfica primitiva entre partes do sistema nervoso, mover uma cadeira de rodas com sinais neuronais e mudar a cor de joias e o comportamento de uma quantia de pequenos como resultado a sinais neuronais.

O intuito de Kevin é usar isso para fins terapêuticos, mas também para o aprimoramento humano ("*Human Enhancement*"), explica ele que o ser humano é limitado, por exemplo, com essa tecnologia capazes de melhorar o processo de memória, de pensar em muitas dimensões e se comunicar apenas com o pensamento. Outro método não invasivo propõe algo que não é muito inovador. O uso de eletrodos de eletroencefalograma externos e acoplados ao corpo é a solução proposta, porém muito falha e limitada, com uso muito mais focado em monitoramento do que estimulação, como desejado [14].

Outra abordagem ainda emergente no Brasil e de crescente e interesse é o uso de dispositivos de Tecnologia de Informação e Comunicação (TICs ou ICTs na sigla em inglês) em Educação. Sugestões de organismos preocupados com o desenvolvimento da educação a nível internacional como a UNESCO apontam um modelo de estratégia crescente (quase contínua) de uso dos TICs em educação, num ciclo de aparecimento da tecnologia, aplicação, integração na escola e transformação das práticas educativas [30], com respaldo de ganhos tanto cognitivos quanto em habilidades [31], e mesmo na formação e qualificação dos professores [32], e sendo reportadas iniciativas e incentivos em todo o Brasil inclusive vindas do governo federal brasileiro [33, 34].

1.2. Objetivos

Os objetivos deste trabalho são:

- Desenvolver um jogo interativo com informação imperfeita, para incitar a curiosidade e interesse dos estudantes na integração de sistemas online de maneira interativa usando conceitos de IA;
- Usar um robô móvel fácil de integrar com outros equipamentos;
- Comunicação com serviços rápida e simples, externos ao jogo;
- Diferentes opções para a tomada de decisão e acesso a serviços, ampliando as capacidades do robô móvel.

2. Descrição do experimento

2.1. Lista e definição das escolhas

Dentre diversas possibilidades de escolha de robôs e eletrônicos que poderiam ser conectados para construção de módulos que possibilitasse a concepção de um robô que fosse de simples conexão, fácil escalabilidade e intuitivo para o público em questão, levando em consideração um preço razoável, conexão com outros possíveis serviços externos (SDK possibilitar chamada REST) e de tamanho reduzido.

Uma possibilidade seria o Lego MindStorms, produto da lego composto por peças montáveis, sensores e controladores. O Lego MindStorms foi lançado em 1998 com uma linha de brinquedos com função didática que abordam tecnologia para crianças com um conceito de programação por bloco usando uma interface gráfica [15]. Essa opção foi uma das cogitadas, o preço do Kit é algo variável, pois depende da quantia de peças, sensores e versões. Por exemplo, o preço Kit abaixo (Vide Figura 6) varia entre R$2.000.00 e R$3.000.00.

Figura 6: Kit Lego MindStorms da Lego

Outra boa escolha seria o NAO, um robô humanóide, criado pela startup francesa Aldebaran Robotics. Foi projetado inicialmente para competir em eventos de competições de robô, como a RoboCup. É um robô projetado com diversos sensores, incluindo câmera, microfones e altofalantes. É um robô que vai mais na linha da Sophia citada anteriormente, mas com uma inteligência diferenciada (Vide Figura 7) [16]. O preço gira em torno R$30.000,00.

Figura 7: Robô NAO de 57cm da Aldebaran Robotics

O TJBot é uma outra opção interessante, é um projeto criado no laboratório de pesquisa da IBM, com intuito de servir para cunho educacional para crianças de 11 a 17 anos que tenham interesse em aprender sobre tecnologia. Esse projeto é altamente escalável, pois utiliza uma RaspBerry PI e sensores externos sem a necessidade de conexões complexas [17]. Para montagem do mesmo gasta-se algo em torno R$500,00 a R$700, pois há a necessidade de impressão do cardboard ou 3D para montagem física do mesmo. Ouve, pensa e fala através de conexões com APIs.

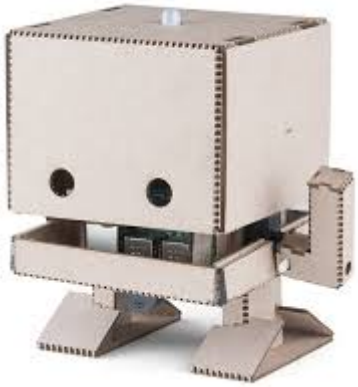


Figura 8: Visão externa do robô TJBot da IBM

E a última opção foi o Cozmo, robô projetado pela Anki, também com fins educacionais. O Cozmo (Vide Figura 9) possui um kit de desenvolvimento de software (SDK) que possibilita uso de diversas ferramentas e funcionalidades, como por exemplo, controlar suas ações, expressões e habilitar sua câmera, que são fundamentais para integração com serviços externos [18]. Seu preço varia de R$1.500,00 a R$2.000,00.

Figura 9: Robô Cozmo da Anki

Dentre todas as opções possíveis, foram descartados o NAO, por ser muito grande e fugir um pouco do propósito do jogo, apesar de ter acesso a serviços externos e o TJBot por não se locomover, apesar de ter câmera e ser bem escalável. O Lego MindStorm é uma ótima opção, porém não possibilita acesso externo fácil a outros serviços externos. Desta forma, optamos pelo Cozmo pois:

- Cobre todas as necessidades básicas para execução do projeto;
- Possui tamanho reduzido, adequado para testes com crianças e pré-adolescentes e
- Com custo adequado ao projeto.

Como o robô escolhido foi o Cozmo da Empresa Anki (Vide Figura 9) que será descrito e detalhado um pouco mais nas próximas seções, há três coisas essenciais do mesmo: conexão sem fio via com um aplicação por meio de um dispositivo eletrônico com Android ou iOS instalado, uma câmera de baixa resolução embutida e expressão de sentimentos. O Cozmo possui um kit de desenvolvimento de software (SDK) [19, 20, 21] que possibilita uso de diversas ferramentas e funcionalidades, como por exemplo, controlar suas ações, expressões e habilitar sua câmera, que são fundamentais para integração com serviços e para a demonstração jogo que será descrito a seguir.

2.2. Dinâmica e objetivos do jogo

Os objetivos do jogo são:

- Engajar os alunos e incentivá-los a jogar e por meio disso aprender sobre programação, integração de sistemas e Inteligência Artificial;
- Dado um ponto inicial de partida pré-determinado, inserir obstáculos para que o robô chegue ao ponto final;
- Expressar emoções a cada ação tomada com finalidade de compreender sobre chamada de função e modularização;
- Instruir e demonstrar de maneira ilustrativa chamada REST;
- Uso de serviço de reconhecimento imagem em Inteligência Artificial

A dinâmica do jogo é a seguinte:

I. O robô inicia o jogo em uma possível posição do tabuleiro e começa a caminhar até achar uma seta que indicará a nova direção.
II. Encontrando uma seta o robô envia um foto da seta para um reconhecedor de imagem, pré-treinado que tentará entender se é uma seta e para que lado está apontando.
III. Caso a confiança do reconhecimento da imagem do robô seja baixa ele expressará um sentimento de tristeza e caso seja alta ele expressará um sentimento de alegria.
IV. Quando a seta for reconhecida com alta confiança o robô deverá seguir o que está indicado na seta, caso contrário algum caminho pré-configurado deverá ser indicado para que ele prossiga.
V. Independente do reconhecimento, o robô manterá seu curso correto até o final, girando 90 graus para o lado que a seta deveria indicar e continuará andando até que encontre outra seta e assim por diante até o fim do tabuleiro.
VI. Um sentimento baseado em uma pontuação geral, baseada nas confianças do reconhecimento de cada seta, será o indicativo do sentimento feliz ou triste ao final do jogo.

2.3. Conexão de Hardware

Para atingir os objetivos propostos no item anterior é necessário seguir alguns procedimentos que serão descritos nas seções abaixo.

2.3.1. Materiais

Os materiais básicos necessários para garantir o controle remoto do Cozmo e programação básica dos comandos para execução do jogo são:

- Um robô Cozmo com cabo de energia e base de recarga;
- Um dispositivo eletrônico com Android ou iOS e seu cabo USB
- Modo desenvolvedor do dispositivo habilitado;
- Um computador com sistema operacional (Windows, MacOS ou Linux, de preferência o Ubuntu);
- Python 3.5.X ou mais novo instalado no computador

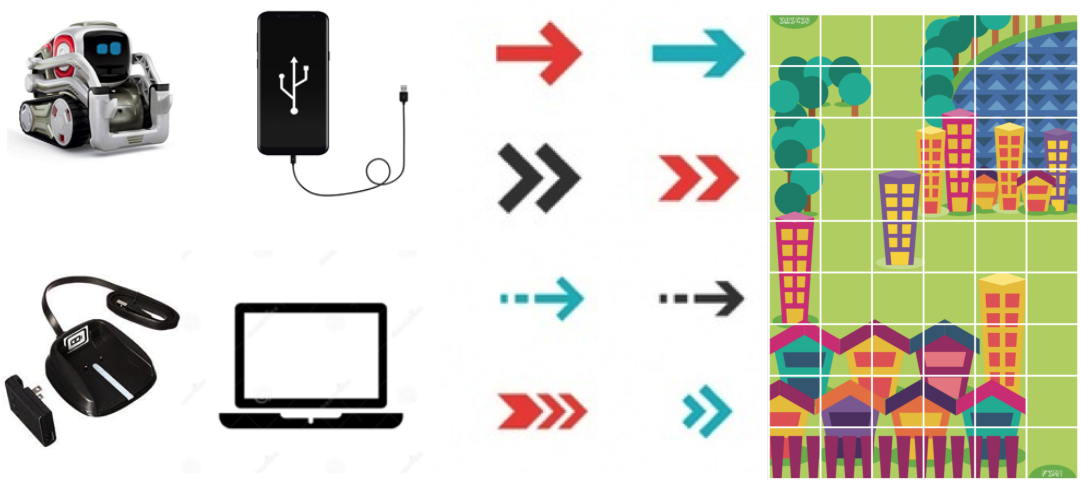


Figura 10: Itens básicos

A metodologia utilizada para conectar os subsistemas ao Cozmo e controlá-lo remotamente para tomar decisões, chamar serviços externos, entre outras funcionalidades é a que se segue, uma visualização do processo pode ser vista no Figura 11 abaixo [20]:

I. Baixar o aplicativo móvel Cozmo (plataformas suportadas: iOS e Android);
II. Habilitar a função modo desenvolvedor do dispositivo;
III. Conectar o dispositivo via cabo a um computador para controle remoto via terminal do computador;
IV. Conectar Smartphone e Cozmo via conexão sem fio;
V. Habilitar o modo escravo via Cozmo no Smartphone;
VI. Conectar o computador a internet

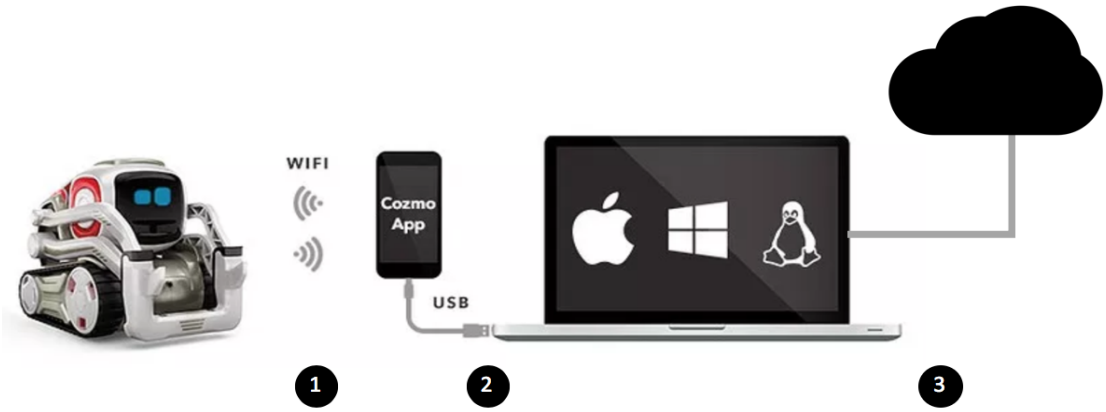


Figure 11: Conexão entre Cozmo e Dispositivo via Wi-Fi (1), Dispositivo e Computador via cabo USB (2) e computador conectado a internet (3)

2.4. Conexão de Hardware

Nesta seção serão explicadas, demonstradas e exemplificadas as etapas descritas na Figura 11 e algumas outras que não foram explicitadas:

2.4.1. Conectando o Cozmo ao Smartphone

O Cozmo foi projetado para funcionar de maneira simples e para que isso ocorra uma aplicação é usada para configurá-lo e controlá-lo (Vide passo 1 da Figura 11).

Os dispositivos que funcionam estavam rodando no mínimo: iOS 9, Android 5 (Lollipop) ou Fire OS 5. A lista completa pode ser verificada no site oficial da Anki:

Entre os dispositivos não suportados estão:

- Huawei Lite ou outros dispositivos que possuam x86, x86-64, Intel, AMD, or Atom;

Apesar do dispositivo não estar na lista, talvez por ser mais novo, funcionou perfeitamente. Outro ponto importante é um computador com conexão à internet. Neste trabalho foram usados o Smartphone da ASUS Modelo Zenfone 4 Z01KD e macbook Pro com macOS High Sierra.

O Smartphone Zenfone é usado como um hub para o Cozmo através do aplicativo móvel (Vide Figura 12 abaixo) que deve ser instalado no seu celular.

Figura 12: Aplicativo do Cozmo na Google Play Store

Todas as funcionalidades já habilitadas pelo Cozmo, projetadas pela Anki, podem ser usadas através desse aplicativo. Essa conexão é feita através dos seguintes passos, demonstrados na Figura 13, abaixo:

Figura 13: Passos visuais para ligar e obter informações para conectar o Cozmo.

Os passos indicados na Figura 13 acima são os seguintes:

A. Ligar a base do Cozmo na tomada;
B. Colocá-lo na base, e ele acenderá um LED Verde na parte superior;
C. Nome da rede WI-FI, que usará para conexão, aparecerá no visor do Cozmo
D. Levante as garras do Cozmo uma vez e abaixe-a
E. A senha de conexão WI-FI se mostrada no visor.

Após isso, basta ligar seu Smartphone ou Tablet, conectar a rede indicada no passo 4 da Figura 13.

- Desbloqueie seu celular, abra as opções de rede, conecte a rede acima identificada;
- Abra o aplicativo e estará apto a controlar e interagir com o Cozmo.

Os passos anteriores completos temos a primeiro passo completo (Vide Figura 11). Com essa conexão feita é possível, jogar alguns jogos e desfrutar de funcionalidades embarcadas no próprio robô, porém o interesse é ter mais liberdade para usar e interagir com o robô, permitindo-o acessar serviços externos a ele inclusive via internet, como por exemplo, acessar serviços de inteligência artificial que estão localizados em um servidor remoto.

2.4.2. Controlando o Cozmo via acesso remoto

Para aproveitar as funcionalidades disponibilizadas pelo SDK da Anki para o robô Cozmo, é necessário usar o mesmo dispositivo, no caso deste trabalho um Smartphone, em que foi instalado o aplicativo móvel anteriormente descrito.

O Smartphone será conectado ao computador por meio de um cabo USB, como demonstrado no Passo 2 da Figura 11 e esse computador deverá estar conectado a internet, como no Passo 3.

Há partes partes importantes para assegurar o controle remoto do Cozmo:

- Habilitar o modo desenvolvedor no smartphone;
- Instalar ferramentas no computador;
- Inicializar o modo acesso remoto no APP e testar alguns exemplos;

2.4.3. Habilitando o modo desenvolvedor e Instalando Android Device Bridge

Habilitar o modo desenvolvedor, como demonstrado abaixo, no Smartphone possibilita desfrutar de opções antes escondidas. É recomendado apenas para pessoas que desenvolvem e testam o software. No caso do Cozmo, essa opção é necessária para que possamos habilitar o acesso remoto a ele, e rodar exemplos usando as ferramentas de desenvolvimento disponibilizadas pela Anki (A instalação e demonstração serão feitas próximas subseções).

Para habilitar o modo desenvolvedor basta seguir os seguintes passos:

Configurações > Sobre o dispositivo/telefone > Informações do Software > Clicar sete vezes no número de versão ou compilação

Com esse módulo habilitado, agora basta ATIVAR a "Depuração USB", para permitir de fato o acesso remoto, no caminho descrito abaixo:

Configurações > Sistema > Opções do desenvolvedor > Depuração USB

E para finalizar, é preciso instalar um módulo do SDK do Android (Vide Figura 15), o Android Debug Bridge [22], que serve para gerenciar o dispositivo Android, em outras palavras, sem esse módulo o acesso remoto não funcionaria.

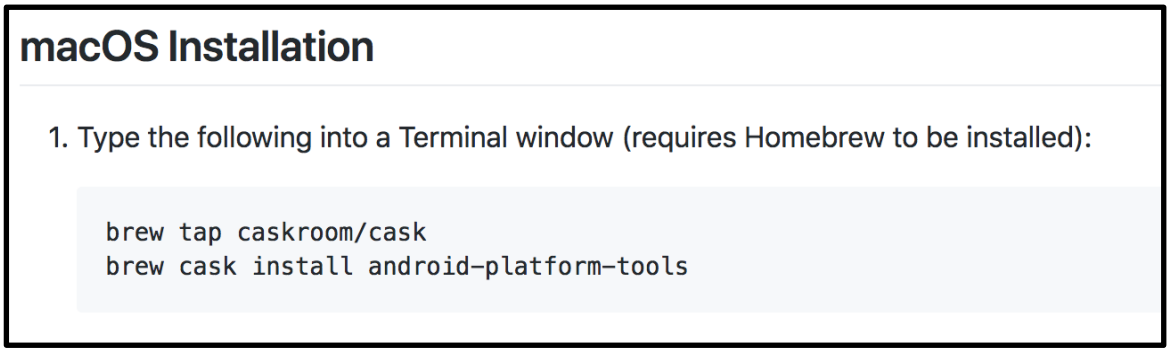


Figura 15: Passos para instalação via terminal do macOS [22]

Uma vez instalado basta conectar o celular no usb, aceitar a opção de confiar neste computador e seguir o passo da Figura 16 abaixo:

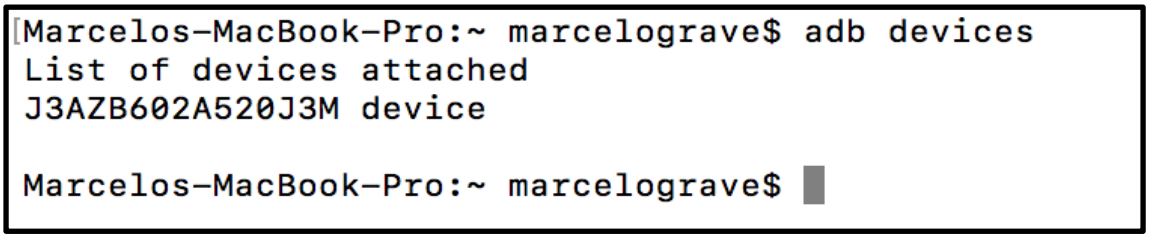


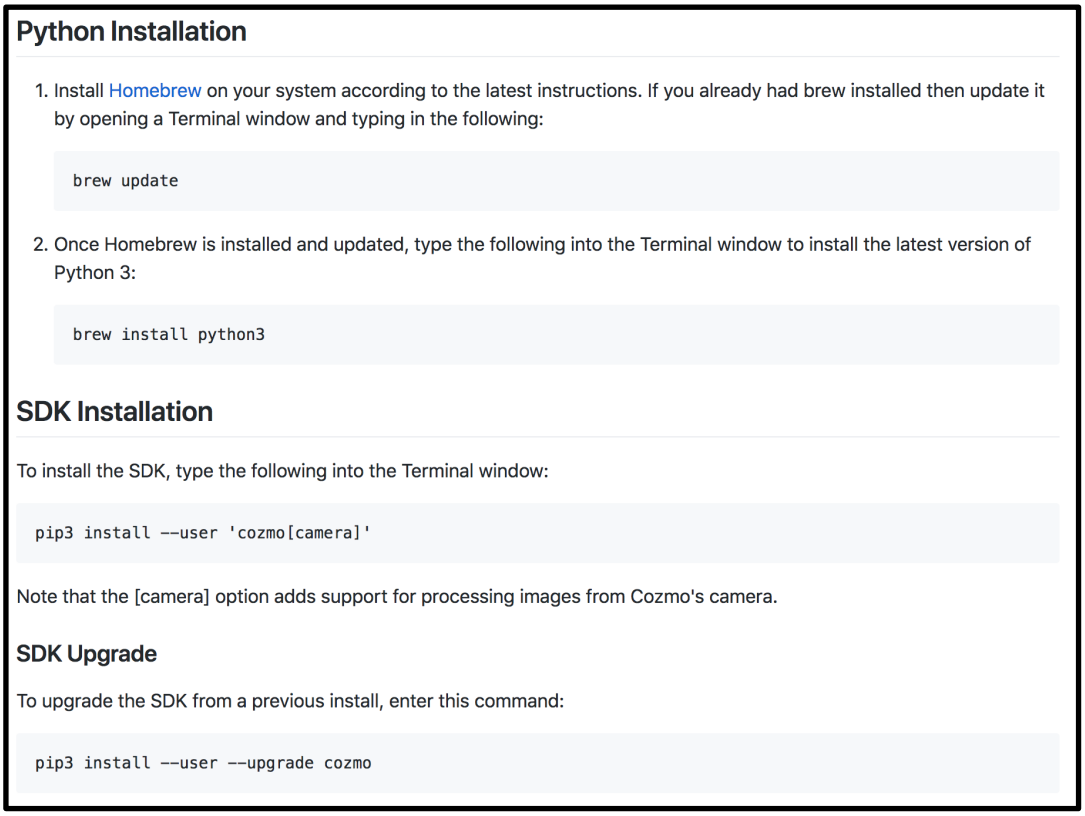
Figura 16: Testando e listando dispositivos Android via Android Debug Bridge [22]

Caso o que processo tenha ocorrido de maneira correta, algum dispositivo Android será listado com um código alfanumérico associado e flag *"device"*, caso contrário a flag que aparece é *"unauthorized"*. Para dispositivos com iOS instalado não há a necessidade de instalação de um módulo parecido.

2.4.4. Instalando e configurando ferramentas no computador

Efetuadas o passo acima, estamos aptos a controlar o Cozmo remotamente e interagir com o mesmo usando as funcionalidades básicas já pré-programadas no Cozmo, porém o intuito é ter um pouco mais de liberdade nas tarefas e tomadas de decisão exercidas sobre ele. Para que seja possível, por exemplo, fazê-lo mover-se ou simplesmente efetuar alguma ação desejada que exige usar algo do kit de desenvolvimento dele é necessário ter o SDK instalado [19, 20, 21].

Para facilitar a instalação do SDK e testar alguns códigos básicos é ideal ser familiar com a interface linha comando (terminal) do computador nesse caso macbook com macOS High Sierra, a linguagem Python versão 3.5 ou mais atual. Dado isso basta seguir os passos da Figura 17 abaixo:



Python Installation

1. Install Homebrew on your system according to the latest instructions. If you already had brew installed then update it by opening a Terminal window and typing in the following:

```
brew update
```

2. Once Homebrew is installed and updated, type the following into the Terminal window to install the latest version of Python 3:

```
brew install python3
```

SDK Installation

To install the SDK, type the following into the Terminal window:

```
pip3 install --user 'cozmo[camera]'
```

Note that the [camera] option adds support for processing images from Cozmo's camera.

SDK Upgrade

To upgrade the SDK from a previous install, enter this command:

```
pip3 install --user --upgrade cozmo
```

Figura 17: Como instalar o Python 3.X e instalar e atualizar o SDK do Cozmo [21]

Para facilitar também o desenvolvimento do código recomenda-se usar um IDE (Ambiente de desenvolvimento integrado), de acordo com seu gosto e/ou sua familiaridade. Neste trabalho optou-se pelo VSCode [23], uma vez que o mesmo possui uma gama de pacotes para auxílio no desenvolvimento e possui inclusive um terminal na própria interface, como por exemplo, o pylint [24] que verifique bugs e qualidade de códigos.

2.4.5. Acessando remotamente via aplicativo e testando códigos simples

Estando com a montagem do aparato do experimento, como segue na Figura 11, e as configurações anteriores concluídas, basta estar no menu principal do aplicativo do Cozmo no celular e selecionar Configurações ("Settings") no canto superior direito, e deslizar para a direita até encontrar "Enable SDK", como demonstrado na FIgura 18 abaixo:

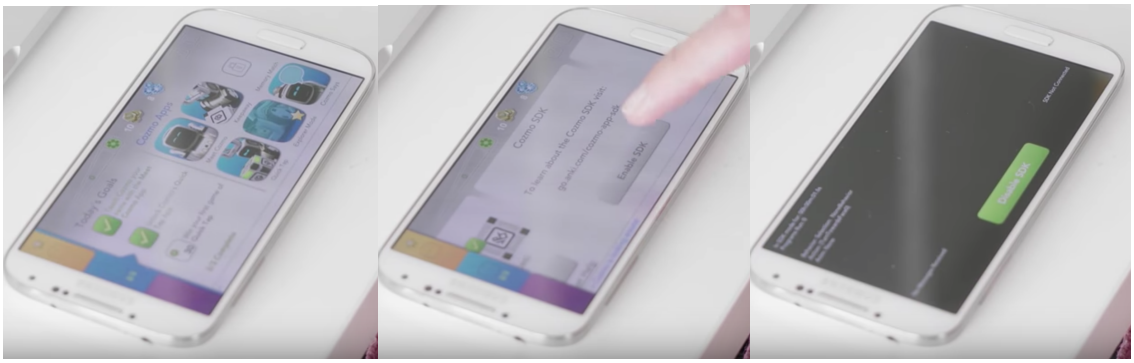


Figura 18: Iniciando o acesso remoto ao Cozmo [25]

Após isso é possível seguir o tutorial de uso e primeiros passos de familiarização com o Cozmo usando alguns códigos sugeridos e propostos pela própria Anki (Vide Figura 19) para primeiros contatos com Cozmo controlado remotamente [26].

```
'''Hello World

Make Cozmo say 'Hello World' in this simple Cozmo SDK example program.
'''

import cozmo


def cozmo_program(robot: cozmo.robot.Robot):
    robot.say_text("Hello World").wait_for_completed()


cozmo.run_program(cozmo_program)
```

Figura 19: Primeiro exemplo de código - "Hello World".

Algo que pode ajudar, caso algum problema fora do comum ocorra são os fóruns de discussão do robô e o troubleshoot básico [27].

3. Solução de reconhecimento de imagem na nuvem

Uma das partes propostas pelo jogo foi utilizar reconhecimento de imagem, como ferramenta de explicação sobre Inteligência Artificial, logo a ideia foi utilizar um dos serviços de reconhecimento de imagem da IBM, o *IBM Watson Visual Recognition*. Inicialmente, bastava criar uma conta na IBM Cloud e uma instância do serviço de Reconhecimento de imagem (Vide Figura 20):

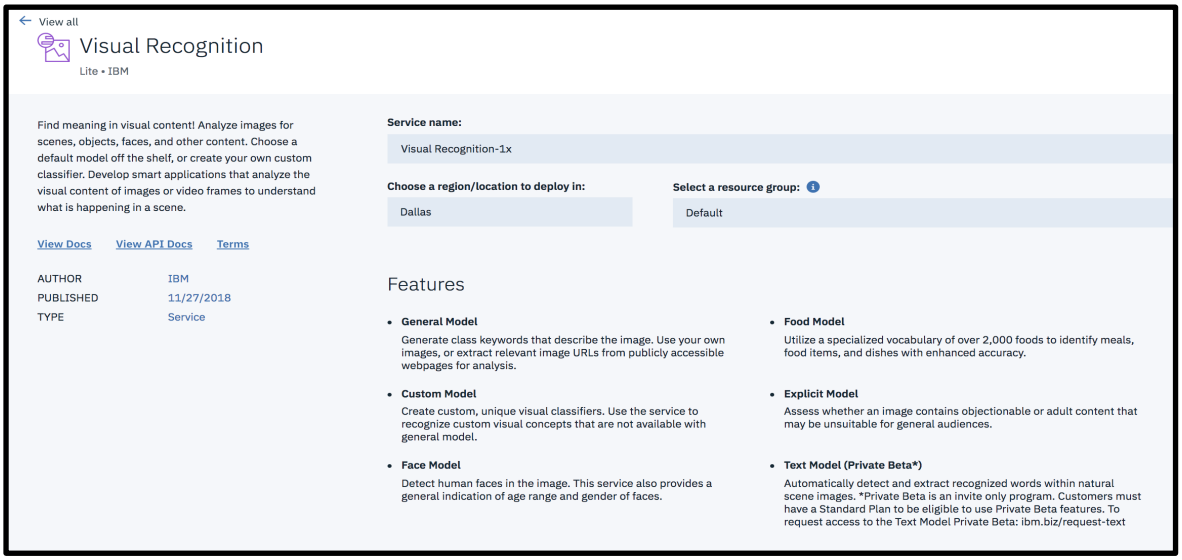


Figura 20: Serviço de reconhecimento de imagem

Há algumas linguagens e disponíveis para uso entre elas: cURL, Java, Javascript, Python e Ruby, logo serviço atende nossas necessidades de maneira bem simples em termos de REST API.

Uma vez que, a linguagem do robô Cozmo para acesso remoto usada é Python versão 3.X, a melhor opção a ser adotada é no mínimo a mesma linguagem utilizada pelo robô, ou seja, em termos de integração é a melhor opção, pois o tipo de linguagem, ambiente e compiladores são os mesmos.

Foi possível testar a qualidade do serviço para a proposta do jogo em que o serviço seria usado para reconhecer setas para direita e esquerda e virar de acordo com o que foi reconhecido. Para isso foi usado o código em Python (Vide Figura 21), além seta de teste (Vide Figura 24) e obtidos os seguintes resultados (Vide Figura 25):

```python
import json
from watson_developer_cloud import VisualRecognitionV3

visual_recognition = VisualRecognitionV3(
    '2018-03-19',
    iam_apikey='API_KEY')

with open('./seta_esquerda.jpeg', 'rb') as images_file:
    classes = visual_recognition.classify(
        images_file,
        threshold='0.6').get_result()
    print(json.dumps(classes, indent=2))
```

Figura 21: Código da chamada REST para classificar uma imagem, usando uma confiança mínima (threshold de 60%)

Dentre os possíveis classificadores existentes abaixo demonstrados na Figura 22. Os que mais adequam-se com a necessidade do jogo é o genérico descrito por *"General"* e o customizado *"Custom"*. A classe possui milhares de classes diferentes sobre assuntos dos mais variados. As outras classes podem ser definidas por:

- *"Faces"* para identificar rostos em uma foto;
- *"Food"* para identificar se é ou não comida;
- *"Explicit"* para identificar se é ou não uma imagem inapropriada;
- *"Text"* para identificar texto em uma imagem

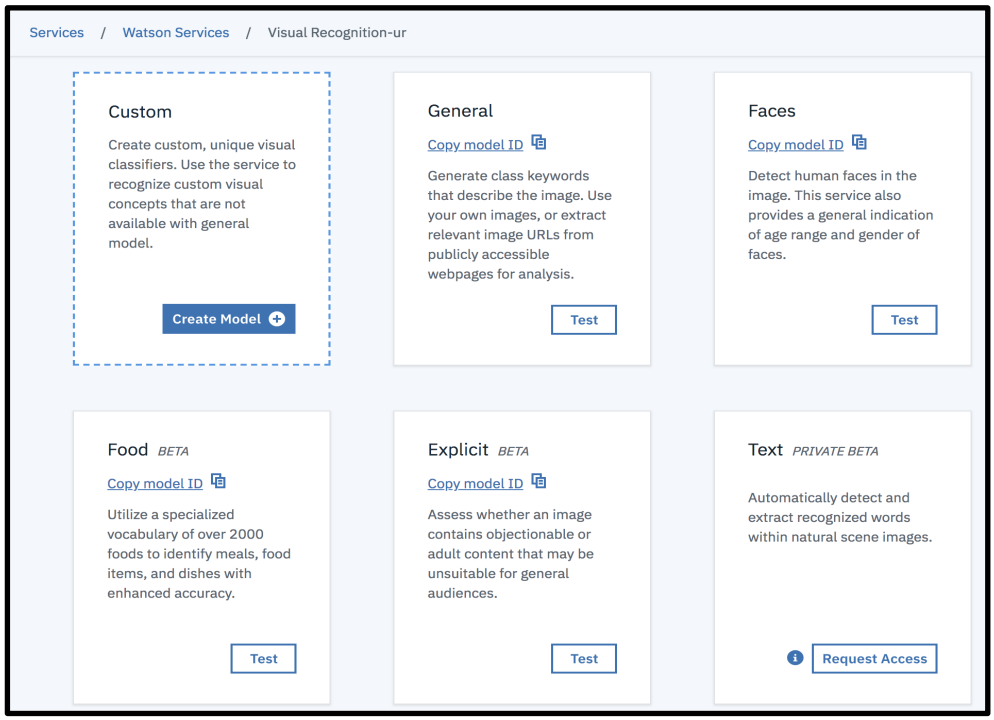


Figura 22: Classificadores treinado e disponíveis no serviço de reconhecimento de imagem.

Para poder usar o serviço foi necessário obter <API_KEY> acessando diretamente do serviço de reconhecimento de imagem por meio da interface do painel de controle, como mostrado no Figura 23:

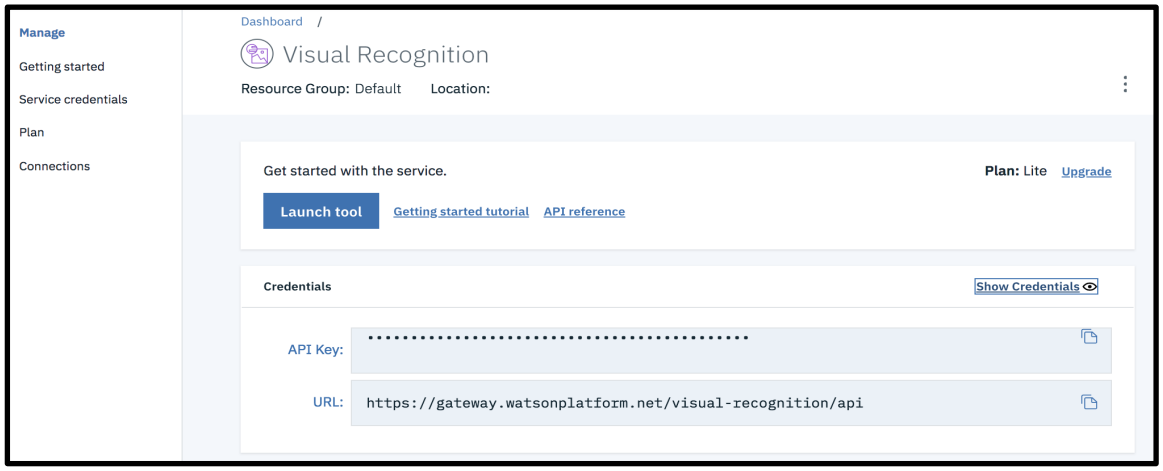
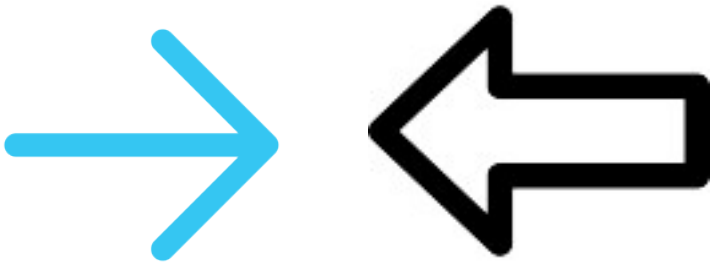


Figura 23: Credenciais do serviço de reconhecimento de imagem para serem usadas na chamada REST de teste de reconhecimento da Figura 21.

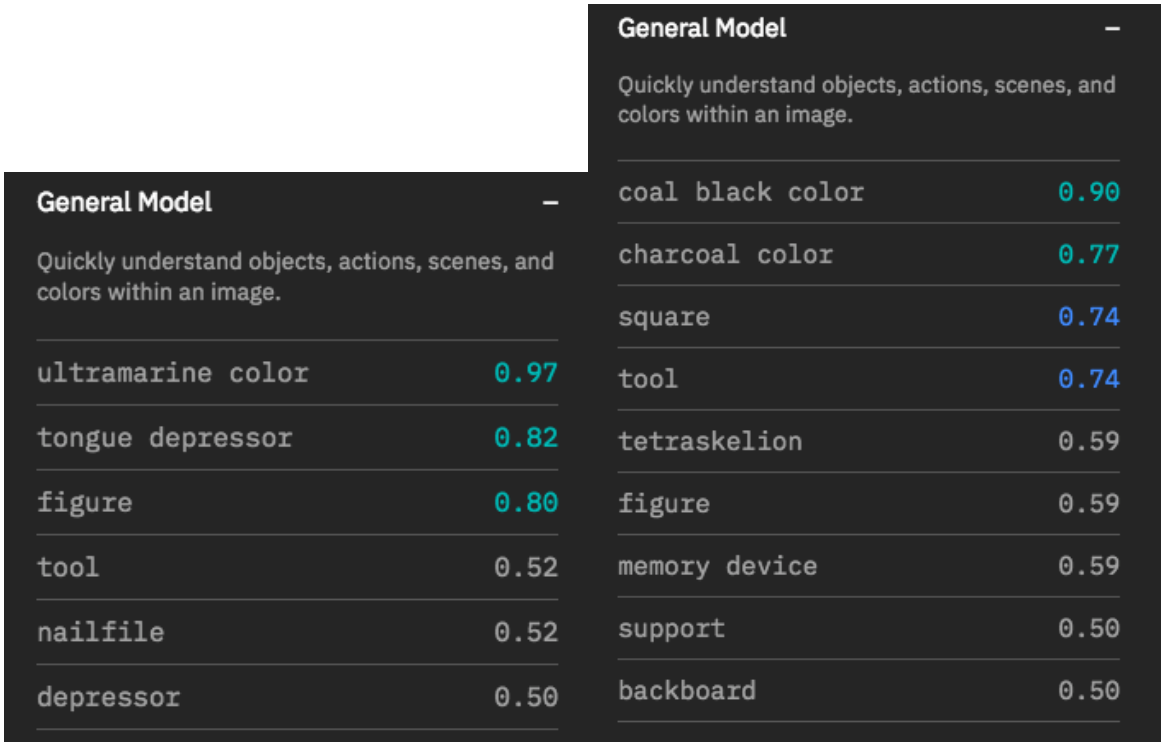
Os testes foram efetuados usando algumas setas, como por exemplo, as duas setas a seguir na Figura 24 (a) e (b):



(a) (b)

Figura 24: Setas para (a) a direita e (b) para esquerda

Os resultados da Figura 25 abaixo foram obtidos por meio das chamadas usando o código da Figura 21 e os dados de entrada os exemplos de setas da Figura 24:



(a) (b)

Figura 25: Resultados da saída do código da Figura 21 usando a interface visual do serviço com as entradas da Figura 24 respectivamente (a) e (b).

Usando o modelo de reconhecimento de imagem genérico foram testados alguns casos, incluindo os casos acima, e da mesma maneira mostrada acima o uso dessa classe genérica não era a melhor opção no caso deste trabalho. Algo como um modelo customizado para essa situação seria uma melhor opção.

De acordo com a documentação oficial do serviço de reconhecimento de imagem [28] para customizar o serviço deve-se:

- Treinar o serviço com no mínimo 10 e máximo 10.000 imagens para classe positiva ou no máximo 10MB por imagem com resolução mínima de 32x32;
- Cada arquivo deverá ser compactado para facilitar seu manuseio;
- Exemplo negativos também só podem ter no mínimo 10 imagens;
- O upload total para treino da rede só pode ser no máximo 256MB.

Antes de treinar o modelo é preciso levar em consideração que as imagens a serem usadas não podem seguir o modelo da Figura 24, uma vez que o Cozmo possui uma câmera com resolução muito baixa 320x240, logo as imagens a serem usadas deveriam seguir o mesmo padrão (Vide Figura 26).

Outro ponto, as crianças deveriam desenhar as setas que seriam usadas no jogo, logo no treino no serviço customizado deveriam ser usadas setas que representasse bem a aplicação, logo um montante de setas, algo em torno 500 setas.

Como a quantia de setas a serem desenhadas a mão era muito grande, 10 pessoas desenharam em torno de 5 setas cada uma. Ainda assim, a quantia não era muito grande. O seguinte processo foi usado para aumento do banco de setas:

I. Selecionar a imagem da seta;
II. Gravar um vídeo da seta afastando e aproximando o vídeo;
III. Salvar o vídeo no computador;
IV. Usar o programa FFMPEG para cortar o vídeo em 25 imagens;
V. Reduzir o tamanho e qualidade da imagem para representar a resolução usada no Cozmo.

O processo de afastar e aproximar o vídeo para obter as imagens desejadas foi relativamente eficiente, foi efetuada um curadoria rápida para garantir que as imagens estavam de acordo. Essa processo foi muito mais rápido que desenhar as mais de 100 setas desenhadas a mão. Obviamente esse foi o método encontrado para aumentar a quantia de dados de treino, mas não era o único, talvez tenha sido o mais rápido dada a necessidade de apresentação do projeto.

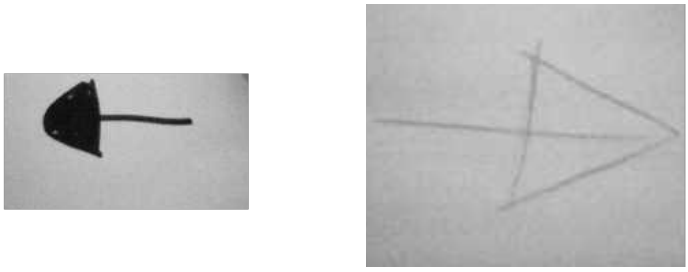


Figura 26: Exemplos de setas desenhadas a mão extraídas do vídeo gravado com qualidade do cozmo (320x240) usadas para treino

Para o treino do serviço foram selecionadas aleatoriamente em torno de 150 imagens de teste e a restantes para treino o que contabiliza 90% de dados treino e 10% de dados de teste.

Para treinar esses modelos basta seguir os passos indicados gerais abaixo que estão detalhados na documentação do serviço [28, 29]. Passos II e III podem ser feitos e visualizados na Figura 27.

I. Abra o serviço, crie um novo modelo e nomeio
II. Fazer upload dos arquivos ".zip" com as imagens de cada classe
III. Arraste para a classe criada e treine o serviço

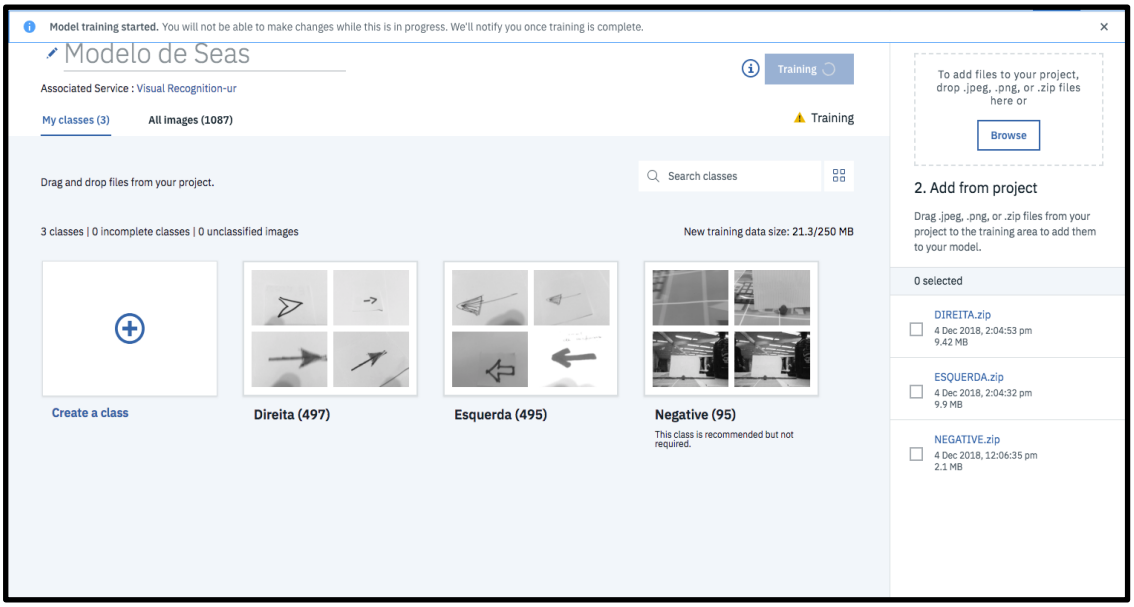


Figura 27: Interface visual de treino do serviço

Foram criadas as classes "Direita", "Esquerda" e foi usada classe "negative" para imagens que representavam classes que não deveriam ser classificadas (Vide Figura 27). Para a classe "Negative" foram usados três vídeos onde foram extraídas em torno de 120 imagens.

Foram usados dois métodos comparativos de treino, um com 100 imagens de treino e outro com 1000 imagens de treino e as mesmas 150 imagens de teste, os resultados estão expressos nas duas tabelas abaixo:

Tabela 1: Resultados do reconhecimento com confiabilidade de 0,70 e 100 imagens de treino por categoria

100 Imagens	Direita	Esquerda
Direita	63,2%	36,8%
Esquerda	10,8%	89,2%

Tabela 2: Resultados do reconhecimento com confiabilidade de 0,70 e 1000 imagens de treino por categoria

1000 Imagens	Direita	Esquerda
Direita	66,41%	33,59%
Esquerda	14,49%	85,51%

Como se pode notar não houve muita diferença em relação a as taxas de reconhecimento com treinamento de 100 e de 1000 imagens, alguns dos motivos que poderia listar:

- A amostragem aleatória pode ter deixado fora imagens que são necessárias para identificar determinados testes, como se pode notar na Figura 28 abaixo, onde foram usados testes que não foram usados no treino
- O algoritmo pode não estar otimizado para tal tarefa, considerando que o modelo genérico não reconheceu poucas das imagens como setas ou flecha.
- O método utilizado para aumentar a quantia de imagens de treino por meio de imagens extraídas do vídeo, pode não ser a melhor opção, mas no caso foi a mais rápida pelo menos reconhecer de fato as setas.

Figura 28: Comparação entre imagens reconhecidas e não reconhecidas para o treino com 1000 imagens

Mesmo com uma taxa de acerto baixa os modelos foram usados. Uma estratégia montada para contornar esse problema, foi a de criar um cenário onde o Cozmo sempre andasse na direção desejada e apenas demonstrasse algum sentimento de dúvida quando o que era pra feito tinha uma confiança baixa.

4. Resultados e discussão

Como pode-se notar acima dado que a taxa de acerto do reconhecedor de imagem usada em questão ser baixa, talvez pela quantia de setas ou pela maneira que as que o set de treino de foi aumentado usando o vídeo. Ainda assim, a apresentação foi bem relevante e os estudantes ficaram bem empolgados e interessados por aprender mais sobre o assunto. Para ilustrar algumas das coisas que foram mostradas.

1. Raciocínio lógico e divisão dos problemas em pequenas partes.

Como parte integrante da interação era entender o problema e separá-los em pequenos pedaços o próprio código foi dividido dessa maneira. Foi explicado aos alunos como era a dinâmica do jogo e o que o Cozmo deveria fazer:

- Andar, verificar seta e girar para o lado correto.

def getWatsonAnswer(file_name):
def walk(robot, blocks):
def getDirection(robot):

Por exemplo na função "*walk*" o aluno foi induzido a compreender que dependendo da quantia de blocos que fosse inserido o robô andaria aquela determinada quantia de blocos. Além da quantia de casas, deveria ser informado também o robô que tomaria tal ação, ilustrando em linhas de código posteriores.

2. Explicação de chamada REST e JSON

Os parâmetros das funções foram explicados, como identificadores e feita uma analogia com caixa de correio, como muitos deles não sabiam o que é correio, foram usados trocadores de mensagem e então surgiu a ideia de exemplificar chamada de função e chamada REST como troca de mensagens com restrições

Basicamente, as turmas foram dividas em duas partes e um jogo de perguntas de respostas pré-determinado foi proposto. Alunos do grupo 1 tinham frases em português e alunos do grupo 2, frases em outras línguas.

Com o menor número de perguntas, eles tinham que descobrir qual era a língua e se os pares de frases conferiam, exemplo de pares de frases:

1. Oi - Hi
2. Tchau - Goodbye
3. Estou bem - I am fine

Desta maneira, chamadas de funções e exemplos de chamadas REST foram explicados.

JSON foi explicado e exemplificado como: selos, tipos de cartas, peso. Em geral, pelos atributos de uma carta, porém os estudantes não tinham a menor ideia do que era aquilo obviamente. O aprendizado foi um pouco mais direto quando usada a ideia de trocadores de mensagem com atributos: foto, número de telefone, código de cidade, por aí.

3. Explicação de integração de sistema e Inteligência artificial

Como dito anteriormente o Cozmo e o jogo foram projetados com intuito de ensinar integração de sistemas, então a exemplicação de como o Cozmo trabalhava foi feita da seguinte maneira:

- Cozmo andava até o ponto desejado
- Levantava a cabeça
- Tirava a foto que era salva localmente, para que os estudantes entendessem
- A foto era enviada para o serviço que reconhecia e mostrava na tela o que foi reconhecido (não havia uma interface visual no momento), logo a ideia foi mostrar usando um JSON Editor
- Cozmo expressava alegria quando o reconhecimento estava acima de um threshold e tristeza quando estava abaixo.

Em termos de didática os estudantes tiveram que enumerar e colocar em sequência os blocos que seriam executados:

direction = getDirection(robot)
action_robot(robot, direction, TURNING)
turn(robot, TURNING)
walk(robot, BLOCK)

E por fim, quando o robô chegava ao fim havia uma função que era chamada apenas para demonstrar o sentimento de alegria ao chegar ao final do curso.

Desta maneira, foi possível explicar como os sistemas estavam integrados, como a chamada REST foi efetuada de maneira ilustrativa e como o sistema que reconhecia a imagem funcionava.

O treino do sistema de reconhecimento de imagem foi exemplicado mostrando algo que os estudantes desconheciam, como por exemplo, frases mais complexas em língua e explicando um pouco de que esses sistemas tendem a serem parecidos como seres humanos atuam.

5. Conclusões

A conclusão é relativamente subjetiva dado que não houve um questionário ou metodologia para aquisição de feedback no momento dos testes. Alguns dos resultados desse trabalho estão descritos abaixo. Ainda assim, o engajamento dos alunos foi bom, todos ficaram muito empolgados e interessados com a proposta e o desempenho visual do robô.

Os pontos alcançados em termos práticos foram:

- Auxílio no entendimento de lógica de programação usando estrutura de blocos e criação de funções. Por exemplo, uma função que recebe um número e anda a quantia de casas relativas àquele número, cmo foi demonstrada acima.

- Uso de raciocínio lógico para colocar a funções em ordem e determinar o percurso correto que o Cozmo deveria fazer, bem como para que lado deveria virar, isso ilustra solução passo a passo na estratégia de solução de problemas do tipo "bottom-up"

- Demonstrando um código concatenado (verificar Anexo I) em uma só função e dividindo o código em funções que fizesse sentido, como andar, girar e reconhecer imagem.

Os pontos a serem melhorados:

- A solução de reconhecimento de imagem padrão não atendia o propósito, logo o serviço de reconhecimento foi customizado para diferentes quantias de imagens 100 e 1000 e obteve as taxas de reconhecimento expressas nas Tabelas 1 e 2. Era esperado que a taxa para ambas setas fossem próximas, mas foram observadas taxas diferentes.
- Outro ponto importante a ser investigado é a classe "negative" que foi treinada com imagens de vídeos do suposto local de apresentação da demo. Talvez o uso do openCV no treinamento de modelo específicos para tais imagens, pudesse ajudar e influenciar positivamente na melhora da taxa de acerto para esse trabalho especificamente.

O presente trabalho juntamente a outros similares foi importante para diversos ações e trabalhos subsequentes. Interesse de escolas de Ensino Fundamental e Médio no conhecimento e aplicação de soluções de inteligência artificial como parte dos cursos extracurriculares dos alunos.

Algumas das ações foram:

- Aulas testes sobre Inteligência artificial para 48 alunos do ensino Médio durante o ano de 2017;

- Disciplina de Inteligência Artificial para 42 alunos do ensino Médio divididos em duas turmas no ano de 2018;
- E haverá essa disciplina ministrada durante dois semestres para o ensino médio no ano de 2019;
- Envolvimento em semana de tecnologias, eventos de tecnologias e outras demonstrações como foi no *"Preparadão"* [35]

6. *Referências*

[1] Chomsky, Noam, *Aspects of the Theory of Syntax*, Cambridge, Massachusetts: MIT Press, 1965

[2] A. M. TURING; I. *Computing Machinery and Intelligence*, Mind, Volume LIX, Issue 236,1 October 1950, pp 433–460

[3] L. Floridi, M. Turilli, M. Taddeo, Turing's Imitation Game: Still an Impossible Challenge for All Machines and Some Judges – An Evaluation of the 2008 Loebner Contest. Minds and Machines, February 2009, Volume 19, Issue 1, pp 145–150

[4] M. MCPHEE , K.C. BAKER, C. SIEMASZKO; *Deep Blue, IBM's supercomputer, defeats chess champion Garry Kasparov in 1997*, Daily News, New York, Estados Unidos, 2015. Disponível em: <https://www.nydailynews.com/news/world/kasparov-deep-blues-losingchess-champ-rooke-article-1.762264>. Acesso em: 26 novembro 2018.

[5] Gautam Narula, Machine Learning in Gaming – Building AIs to Conquer Virtual Worlds, January, 2017. Disponível em: <https://www.techemergence.com/machine-learning-in-gaming-building-ais-to-conquer-virtual-worlds/> Acesso em: 1 dezembro 2018.

[6] Felipe Germano; *Computador vence humano em Go, jogo mais complexo que xadrez*, Superinteressante, Brasil, 2016. Disponível em: <https://super.abril.com.br/tecnologia/computador-vence-humano-em-go-jogo-mais-complexo-que-xadrez/>. Acesso em: 28 novembro 2018.

[7] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L.,Vandendriessche, G., Schrittwieser, J., Antonoglou, I.,Panneershelvam, V., Lanctot, M.,Et Al. *Mastering the game of Go with deep neural networks and tree search.* Nature 529, 7587(2016), 484–489.

[8] AlphaGo. Gary Krieg, Kevin Proudfoot, Josh Rosen. 2017. Documentário. Estados Unidos.

[9] The Associated Press; *AlphaGo vs. Lee Sedol: Google's computer program wins round one, Macleans, Canadá, Março, 2016*. Disponível em: <https://www.macleans.ca/economy/business/alphago-vs-lee-sedol-googles-computer-wins-round-one/>. Acesso em: 29 novembro 2018.

[10] Oriol Vinyals, Stephen Gaffney, Timo Ewalds, *DeepMind and Blizzard open StarCraft II as an AI research environment,* DeepMind, Agosto, 2017. Disponível em: <https://deepmind.com/blog/deepmind-and-blizzard-open-starcraft-ii-ai-research-environment/>. Acesso em: 29 novembro 2018.

[11] *Predicting eye disease with Moorfields Eye Hospital*, DeepMind, Inglaterra, Novembro, 2018. Disponível em: <https://deepmind.com/blog/predicting-eye-disease-moorfields/>. Acesso em: 30 novembro 2018.

[12] Daniel Faggella, *Machine Learning Healthcare Applications* – 2018 and Beyond, Emerj, Novembro, 2018. Disponível em: <https://www.techemergence.com/machine-learning-healthcare-applications/>. Acesso em: 30 novembro 2018.

[13] Michael Greshko, *Meet Sophia, the Robot That Looks Almost Human*, National Geographic, Maio, 2018. Disponível em: <https://www.nationalgeographic.com/photography/proof/2018/05/sophia-robot-artificial-intelligence-science/>. Acesso em: 29 novembro 2018.

[14] Kevin Warwick, *The Future of Artificial Intelligence and Cybernetics*, OpenMind. Disponível em: <https://www.bbvaopenmind.com/en/articles/the-future-of-artificial-intelligence-and-cybernetics/>. Acesso em: 29 novembro 2018.

[15] *31313 MINDSTORMS EV3*, Mindstorms EV3, Lego. Disponível em: <https://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313>. Acesso em: 1 dezembro 2018.

[16] NAO - Technical overview, Aldebaran documentation. Disponível em: <http://doc.aldebaran.com/2-1/family/robots/index_robots.html>. Acesso em: 1 dezembro 2018.

[17] IBM TJBot, IBM Watson Maker Kits. Disponível em: <https://ibmtjbot.github.io/>. Acesso em: 1 dezembro 2018.

[18] *What Makes the Cozmo SDK Unique?*, Anki Developer. Disponível em: <https://developer.anki.com/>. Acesso em: 1 dezembro 2018.

[19] Kaiser, *Getting Started with the SDK*, Anki Developer. Disponível em: <https://developer.anki.com/blog/learn/tutorial/getting-started-with-the-cozmo-sdk/>. Acesso em: 1 dezembro 2018.

[20] *Getting Started with Cozmo*, Cozmo SDK Documentation. Disponível em: <http://cozmosdk.anki.com/docs/getstarted.html#starting-up-the-sdk>. Acesso em: 1 dezembro 2018.

[21] *Installation - macOS / OS X,* Cozmo Python SDK Github. Disponível em: <https://github.com/anki/cozmo-python-sdk/blob/master/docs/source/install-macos.rst#id1>. Acesso em: 3 dezembro 2018.

[22] Android Debug Bridge, Cozmo Python SDK Github. Disponível em: <https://github.com/anki/cozmo-python-sdk/blob/master/docs/source/adb.rst>. Acesso em: 3 dezembro 2018.

[23] Visual Studio Coding, Editing. Redefined, Visual Studio Code. Disponível em: <https://code.visualstudio.com/> Acesso em: 1 dezembro 2018.

[24] *Linting Python in Visual Studio Code*, Visual Studio Code. Disponível em: <https://code.visualstudio.com/docs/python/linting> Acesso em: 3 dezembro 2018.

[25] Cozmo SDK — Installation for Android / Windows, Anki Developer, Youtube Video. Disponível em: <https://www.youtube.com/watch?v=9TJeK_AEFYo>. Acesso em: 3 dezembro 2018.

[26] *Downloads Cozmo SDK Documentation*. Cozmo SDK. Última atualização em 30 de Outubro de 2018. Disponível em: <http://cozmosdk.anki.com/docs/downloads.html#sdk-examples> Acesso em: 1 dezembro 2018.

[27] *Initial Setup Cozmo SDK Documentation*, Cozmo SDK, Anki Development. Disponível em: <http://cozmosdk.anki.com/docs/initial.html#trouble> Acesso em: 1 dezembro 2018.

[28] Visual Recognition, IBM Cloud. API DOCS. Disponível em: <https://console.bluemix.net/apidocs/visual-recognition> Acesso em: 4 dezembro 2018.

[29] Release Notes, IBM Cloud. Disponível em: <https://console.bluemix.net/docs/services/visual-recognition/release-notes.html> Acesso em: 4 dezembro 2018.

[30] KHVILON, E. and PATRU, M. (Eds), Information and Communication Technology in Education - A Curriculum for Schools and Programme of Teacher Development, UNESCO, 2002.

[31] HAMIDI, F. et all, Information Technology in Education, Procedia Computer Science 3, pp. 369-373, 2011.

[32] MOURSUND, D., Information and Communications Technology in Education: A Personal Perspective, Contemporary Issues In Technology And Teacher Education, 2016.

[33] TIC na Educação do Brasil, Representação da Unesco no Brasil. Disponível em: <http://www.unesco.org/new/pt/brasilia/communication-and-information/access-to-knowledge/ict-in-education/>. Acesso em 04 de dezembro de 2018 às 16:32h.

[34]. Na rede pública, tecnologia atende 24 milhões de brasileiros, Ministério da Educação. Disponível em: <http://portal.mec.gov.br/component/tags/tag/33994>. Acesso em 04 de dezembro de 2018 às 16:32h.

[35] Preparadão. Santander Universidades Apresenta Preparadão Universia. Disponível em: <https://preparadao.universia.com.br/>. Acesso em 04 de dezembro de 2018.

Apêndice I: Código completo utilizado no experimento

```
import cozmo
import time
import json
from cozmo.util import degrees, distance_mm, speed_mmps
from watson_developer_cloud import VisualRecognitionV3
from os.path import join, dirname

cozmo_vr = VisualRecognitionV3(
    '2018-11-18', api_key='<API_KEY>')
classifier_id_vr = '<Customized_classifier>'
threshold = '0.3'

block_distance = 70  # distance in milimeter
right_angle = -90
left_angle = 90
head_angle = 5

right_class = "right"
left_class = "left"
end = "end"


def turn(robot, direction):
    if direction == right_class:
        robot.turn_in_place(degrees(right_angle)).wait_for_completed()
    elif direction == left_class:
        robot.turn_in_place(degrees(left_angle)).wait_for_completed()
    elif direction == end:
        robot.play_anim_trigger(
            cozmo.anim.Triggers.KnockOverSuccess).wait_for_completed()

    else:
        robot.play_anim_trigger(
            cozmo.anim.Triggers.MajorFail).wait_for_completed()
        robot.drive_straight(distance_mm(
            10), speed_mmps(block_distance)).wait_for_completed()


def action_robot(robot, detected, right):
    if detected == right:
        robot.play_anim_trigger(
            cozmo.anim.Triggers.MeetCozmoFirstEnrollmentCelebration).wait_for_completed()
        robot.drive_straight(distance_mm(
            -12), speed_mmps(block_distance)).wait_for_completed()
    else:
        robot.play_anim_trigger(
            cozmo.anim.Triggers.MajorFail).wait_for_completed()
        robot.drive_straight(distance_mm(
            10), speed_mmps(block_distance)).wait_for_completed()


def walk(robot, blocks):
    robot.drive_straight(distance_mm(block_distance * blocks),
                         speed_mmps(block_distance)).wait_for_completed()
```

```python
def move_head_up(robot):
    robot.move_lift(-3)
    robot.set_head_angle(degrees(head_angle)).wait_for_completed()


def savePic(robot):
    # Get image from camera and try to save it
    print("Esperando a imagem chegar aqui ... ")
    robot.camera.image_stream_enabled = True
    event = robot.world.wait_for(
        cozmo.camera.EvtNewRawCameraImage, timeout=15)
    file_name = '~/Projetos/Cozmo/Game/' + 'foto' + '.jpg'
    event.image.save(file_name, 'JPEG')
    print("Imagem salva!")

    return file_name


def getWatsonAnswer(file_name):
    class_answer = None

    # From file image get the answer from Watson
    with open(file_name, 'rb') as image_file:
        print('Reconhecendo Imagem ...')
        # print(json.dumps(cozmo_vr.list_classifiers(), indent=2))
        # print(json.dumps(cozmo_vr.get_classifier(classifier_id_vr), indent=2))

        classify_result = cozmo_vr.classify(
            images_file=image_file, classifier_ids=[classifier_id_vr], threshold=threshold)

        print(json.dumps(classify_result, indent=2))

        # Gettting the class and high score
        if classify_result['images'][0]['classifiers']:
            classes = classify_result['images'][0]['classifiers'][0]['classes']

            class_answer = ''
            score = 0.2
            for i in classes:
                if float(i['score']) >= score:
                    print(i)
                    score = float(i['score'])
                    class_answer = i['class']

            print(json.dumps(classes, indent=2))
        else:
            print('No class found!')

    return class_answer
```

```
def getDirection(robot):
    turn = None

    print("Moving head up...")
    move_head_up(robot)

    try:
        file_name = savePic(robot)
        turn = getWatsonAnswer(file_name)

    except Exception as e_except:
        print("Error: " + str(e_except))

    return turn


def cozmo_program(robot: cozmo.robot.Robot):
    walking = [2, 2, 1, 3, 2, 1, 2, 4]
    turning = ['left', 'right', 'right',  'left',
               'left', 'left',  'right',  'right']

# Initial State
    if len(walking) == 0:
        # Game's loop
        end = False
        while(not end):
            walk(robot, 1)
            direction = getDirection(robot)
    else:
        for i in range(len(walking)):
            print(turning[i])
            direction = getDirection(robot)
            action_robot(robot, direction, turning[i])
            turn(robot, turning[i])
            print(walking[i])
            walk(robot, walking[i])
        turn(robot, 'end')

cozmo.run_program(cozmo_program)
```