Faculdades Integradas de Taquara - Faccat Av. Oscar Martins Rangel, 4.500 Taquara, RS, CEP 95600-000

Curso de Sistemas de Informação

Nihongo Jouzu - Aplicativo Mobile para Reconhecimento de Caracteres Japoneses

Gerson Carneiro de Souza
Faculdades Integradas de Taquara - Faccat - Taquara - RS- Brasil
gersoncs@sou.faccat.br

Francisco Assis Moreira do Nascimento
Professor Orientador
Faculdades Integradas de Taquara - Faccat - Taquara - RS- Brasil
assis@faccat.br

Resumo

Este trabalho apresenta um aplicativo para celulares, que realiza o reconhecimento de caracteres japoneses com a utilização de redes neurais profundas. Visando auxiliar estudantes do idioma no nível intermediário, o software é utilizado para selecionar um pedaço de uma imagem, onde está o caractere que se deseja estudar e reconhecê-lo através de uma rede neural profunda (*deep neural network*), exibindo na tela sua pronúncia e significado. O software também cria um relatório com quantas vezes um determinado caractere precisou ser reconhecido pelo estudante, para assim auxiliar a identificar aqueles caracteres em que tem havido mais dificuldades, e possa assim focar em estudá-los.

Palavras-chave: Inteligência Artificial. Redes Neurais Profundas. Japonês.

1 Introdução

Este trabalho apresenta um software para celulares focado em auxiliar estudantes de nível intermediário do idioma japonês a identificar caracteres e palavras, que ainda não conseguem reconhecer. Através do aplicativo, o estudante

pode selecionar o caractere em uma imagem, capturar e realizar a inferência através de uma rede neural profunda. O aplicativo traz para a tela do celular o som e significado do caractere, além de armazená-lo em um diagnóstico para consultas de caracteres em que mais foi necessário o uso da inferência.

O idioma japonês é composto por três alfabetos, sendo o kanji o mais complexo deles, consistindo de aproximadamente mais de cinco mil caracteres chineses. Desses, a lista Joyo Kanji contém 2136 kanjis que são utilizados frequentemente em jornais, livros e revistas. Os kanji, diferente do alfabeto romano que é baseado em fonética, são caracteres que apresentam uma leitura a partir de ideogramas, ou seja, cada caractere representa uma idéia.

Esse trabalho visa atender a necessidade de estudantes do idioma japonês, que possuam um conhecimento intermediário do idioma. Os métodos e escolas para a aprendizagem do idioma consideram que o estudante possui um nível de conhecimento iniciante, o que pode proporcionar um rendimento menor para aqueles em nível intermediário. Assim, oferecer um sistema que se adeque ao nível de conhecimento do estudante sem dúvida facilita o aprendizado.

Hoje se tornou comum utilizar de ferramentas para auxiliar na leitura de caracteres japoneses quando o estudante possui um conhecimento intermediário ou avançado, como o KanjiTomo, uma ferramenta para identificar caracteres japoneses de imagens (KanjiTomo, 2019). Porém, a utilização de tais ferramentas pode criar um vício, que impede o estudante de aprender palavras novas. E nenhuma dessas ferramentas apresenta um relatório de palavras consultadas, que poderiam auxiliar o estudante com caracteres ou palavras que está tendo mais problemas.

Kim (2015, p13) acredita que experiência e exemplos são as melhores ferramentas para se dominar o Japonês. Pois assim, é fácil entender como cada conceito é usado em diferentes contextos. Portanto, é aconselhável praticar Japonês o máximo possível e consultar a algum manual de gramática ou dicionário quando surgirem dúvidas. A internet possui um material vasto de recursos para aprendizagem. Também é sugerido comprar livros ou histórias em quadrinhos para ajudar na prática da leitura. Portanto, ler materiais em japonês é essencial para se dominar o idioma.

O Joyo kanji (常用漢字) foi uma lista criada pelo Ministério da educação do Japão reunindo os principais kanjis utilizados no dia a dia que vem sendo atualizada desde 1941, eventualmente adicionando ou removendo novos kanjis. Possui esse nome desde 1981 e reúne hoje um total de 2136 caracteres. Ao se conhecer ao menos metade desses caracteres, já é possível ler revistas, jornais e coisas do cotidiano japonês (Suki Desu, 2019). O sistema abordado neste trabalho tem um foco nessa lista, o que proporcionará uma aprendizagem efetiva e produzirá bons resultados o mais rápido possível.

O sistema foca em auxiliar usuários na leitura, criando um sistema não intrusivo para quem prefere estudar praticando a leitura do idioma e criando relatórios para que o usuário conheça melhor suas dificuldades.

Esse artigo possui a seguinte estrutura: a seção 2 apresenta os conceitos e tecnologias relacionados ao trabalho, sendo seu referencial teórico. A seção 3 apresenta o desenvolvimento do sistema, incluindo as ferramentas utilizadas, análise e metodologia que foi seguida. A seção 4 mostra o funcionamento do sistema, demonstrando testes de uso do reconhecimento e diagnóstico. A seção 5 apresenta outros sistemas, que possuem uma finalidade parecida e quais as diferenças entre o sistema desenvolvido neste trabalho com eles. Por último, a seção 6 mostra as conclusões feitas ao final do trabalho, assim como os planos de implementações futuras.

2 Conceitos e Tecnologias

Nessa seção serão apresentados os principais conceitos relacionados ao trabalho realizado, assim como as referências pesquisadas, que auxiliaram no desenvolvimento do sistema.

2.1 O Idioma Japonês

O japonês não possui somente um alfabeto, como acontece com idiomas baseados no alfabeto romano, mas três sistemas de escritas. São estes o Hiragana, Katakana e Kanji. No sistema de escrita japonesa, os três são utilizados ao mesmo

tempo, em situações diferentes. Aprender os três é essencial para se aprender a ler o idioma (RAFAEL, 2019). O sistema de escrita japonesa teve origem no chinês, que foi adaptado e transformado nos três alfabetos utilizados hoje. Mesmo após a adaptação e o tempo que se passou, ainda há muitas semelhanças entre caracteres chineses e japoneses (MITYE, 2019).

2.1.1 Hiragana

Hiragana é o caractere básico de fonética Japonês. É possível escrever qualquer palavra em japonês utilizando Hiragana. Porém, como o Japonês não usa espaços, a leitura fica difícil de decifrar quando escrevemos utilizando somente ele (KIM, 2014, p16). Hiragana consiste de 46 caracteres, cada um com sua própria fonética. Esses 46 caracteres podem ter sua fonética modificada usando-se dois diacríticos ou um pequeno círculo acima do caractere. Os caracteres básicos ainda podem ser utilizados com um tamanho menor que os outros caracteres da palavra para formar sons mais complexos, como dígrafos ou geminados (HATASA, HATASA, MAKINO, 2010, p2).

O Hiragana é usado principalmente para gramática. Palavras que são muito raras ou importadas de outros idiomas também podem usar hiragana. É muito usado por estudantes de Japonês ou crianças que ainda não possuem um bom conhecimento em kanji (KIM, 2015, p15). A Figura 1 mostra os 46 caracteres hiragana básicos assim como sua leitura.

Figura 1. Hiraganas

n	w	r	y	m	h	n	t	s	k	
ん	わ	ら	や	ま	は	な	た	さ	か	あ a
		り		み	ひ	に	ち	し	き	い i
		る	ゆ	む	ふ	ぬ	つ	す	く	う u
		れ		め	へ	ね	て	せ	け	え e
	を	ろ	よ	も	ほ	の	と	そ	こ	お o

Fonte: HATASA, HATASA, MAKINO, 2010, p3

Como visto na Figura 1, os 46 caracteres abrangem todas as fonéticas básicas do idioma japonês. O Hiragana são caracteres simples, originados de kanjis modificados e possuem um traço mais arredondado.

2.1.2 Katakana

Katakana é utilizado para representar palavras japonesas que foram trazidas de outros idiomas, principalmente do inglês (HATASA, HATASA, MAKINO, 2010, p72). Também pode ser usado para criar ênfase nas palavras. Katakana usa a mesma fonética dos hiragana e segue as mesmas regras. A única diferença entre Katakana e Hiragana, além do uso, é o formato dos símbolos (KIM, 2015, p21). A Figura 2 mostra os 46 caracteres básicos do Katakana.

Figura 2. Katakanas

ン ん	ワ わ	ラ ら	ヤ や	マ ま	ハ は	ナ な	タ た	サ さ	カ か	ア あ
		リ り		ミ み	ヒ ひ	ニ に	チ ち	シ し	キ き	イ い
		ル る	ユ ゆ	ム む	フ ふ	ヌ ぬ	ツ つ	ス す	ク く	ウ う
		レ れ		メ め	ヘ へ	ネ ね	テ て	セ せ	ケ け	エ え
		ロ ろ	ヨ よ	モ も	ホ ほ	ノ の	ト と	ソ そ	コ こ	オ お

Fonte: HATASA, HATASA, MAKINO, 2010, p72

Como mostrado na Figura 2, para cada hiragana, há uma katakana equivalente com a mesma fonética. O traço do katakana é mais angular e geométrico que o hiragana.

2.1.3 Kanji

O terceiro sistema de escrita Japonês é o Kanji. Enquanto Hiragana e Katakana são criados modificando-se caracteres chineses, os kanjis são exatamente dos mesmos caracteres usados no idioma Chinês. Foram importados da China junto com seus significados, mas a forma de lê-los foi adaptada às palavras japonesas. Por esse motivo, o kanji em japonês costuma ter duas ou mais formas de leitura. A primeira, chamada de leitura on, é a leitura original do caractere Chinês, enquanto a leitura kun é leitura japonesa do caractere. Não é claro o número exato de quantos kanjis existem, mas se estima que são mais de 40000. Aproximadamente 3000 são usados regularmente em japonês. Kanjis são escritos de acordo com uma determinada ordem de linhas. A regra geral é escrever de cima para baixo e da esquerda para direita, e linhas horizontais são escritas antes de linhas verticais (HATASA, HATASA, MAKINO, 2010, p160). Diferente dos hiraganas e katakanas, os kanjis não representam sons, mas ideias e conceitos. Também podem ter mais de um significado. É um ideograma, ou seja, uma “imagem pictográfica” daquilo que representa (MITYE, 2019).

Em japonês quase todos os substantivos e o radical dos verbos são escritos utilizando kanjis. Advérbios frequentemente são escritos utilizando kanjis também. Portanto, é necessário saber kanjis para ler a maioria das palavras japonesas (KIM, 2015, p26). O kanji também ajuda na leitura, considerando que não há espaços na escrita japonesa e ainda ajuda a diferenciar entre palavras com um mesmo som, fenômeno que ocorre frequentemente no idioma, considerando o número limitado de sons distintos que ele possui (KIM, 2015, p15). A Figura 3 mostra o kanji para a palavra “estudar”, assim como todas as suas possíveis leituras e ordem para desenhar as linhas.

Figura 3. Kanji Para a Palavra “Estudar”

Fonte: HATASA, HATASA, MAKINO, 2010, p160

A Figura 3 mostra um dos milhares de kanjis existentes no idioma. Ele representa uma ideia (estudar), mas também possui sons, como “mana”, “gaku” e “ga”, representados com katakanas na imagem. Geralmente tendo mais traços, kanjis costumam ser mais complexos que hiragana e katakana, embora existam também kanjis mais simples.

2.2 Aprendizagem do Idioma Japonês

Hiragana é o primeiro alfabeto japonês a ser aprendido, pois será a base para toda a aprendizagem em Japonês. Não há livros ou material de estudo para japonês que não considere que o estudante saiba hiragana. É, portanto, o primeiro passo para se aprender japonês. Hiragana pode ser aprendido em poucos dias, no máximo semanas (KOICHI, 2014).

Sendo que Katakana utiliza as mesmas regras e sons de Hiragana, é aconselhável aprendê-lo logo após dominar Hiragana. Como é mais raro de ser utilizado, se torna mais difícil de memorizar. Outro desafio aparece no fato que as palavras estrangeiras representadas pelo Katakana precisam ser acomodadas no número limitado de sons que podem ser representados em japonês. O resultado final pode ser tão diferente, que mesmo um falante de Inglês pode não reconhecer palavras que são derivadas do Inglês. Portanto, o aconselhável é esquecer a palavra original em Inglês e tratar a escrita em Katakana como uma nova palavra (KIM, 2015, p21).

O último sistema de escrita que se deve aprender quando aprendendo japonês é o Kanji, sendo o mais complexo. O que torna esquecer um kanji tão natural é a falta de padrões da memória visual. O cérebro humano está acostumado a reconhecer as coisas por aparência de algo parecido que já visualizou anteriormente, por exemplo, se reconhece flores novas porque já foi vista outra flor anteriormente. Essa memória está ausente quando se começa a aprender Kanji. A maior aproximação é o conhecimento de outros alfabetos e sistemas numéricos conhecidos (HEISIG, 2001, p1).

A diferença é que são poucos os símbolos relacionados com o som, enquanto há milhares de kanjis e nenhuma consistência fonética. Mesmo assim,

métodos tradicionais de aprendizagem tem sido o mesmo para aprender outros alfabetos: desenhar as formas uma por uma repetidamente. O modo mais eficiente seria relacionar os caracteres a algo, ao invés de seus sons, e abandonar a memória visual que é usada para se aprender o alfabeto ocidental (HEISIG, 2001, p1).

2.3 Auxílio ao Aprendizado do Idioma Japonês

Koishi (2014) sugere um método para se aprender a fixar hiraganas que envolve imagens mnemônicas para auxiliar a memorização dos caracteres. Também sugere não aprender a escrever, caso o objetivo seja aprender a ler, visto que com a tecnologia, escrever a mão é cada vez menos utilizado, podendo ser aprendido posteriormente.

Sendo parecido com Hiragana, as técnicas para aprender Katakana são semelhantes. Kim (2015, p23) lembra que os sons de Katakana são exatamente os mesmos de Hiragana. Há casos especiais, como o caractere wo (ヲ) que é utilizado para gramática, e raramente será usado em Katakana. Ele ainda lembra de prestar atenção nas diferenças entre os caracteres shi (シ), n (ン), tsu (ツ) e so (ソ) por serem muito similares. Como japonês não usa espaço, o sinal 「・」 pode ser utilizado para representar espaços em palavras compostas estrangeiras.

Heisig (2001, p2-3) defende o uso da memória imaginativa para aprender o significado dos kanjis. Memória imaginativa é a capacidade de lembrar imagens criadas dentro da mente, sem o uso de estímulos visuais. Para isso, sugere descobrir um número limitado de elementos básicos dentro dos caracteres, associar uma imagem a cada um deles, e associar imagens à fusão desses elementos. Sendo assim, é possível lembrar desses caracteres utilizando quase que inteiramente a memória imaginativa.

Ainda de acordo com Kim (2015, p12), a melhor forma de aprender Japonês é não tentar traduzir da linguagem para seu idioma, mas aprender como funciona o idioma em sua forma gramatical e construção de frases. Aprender frases em japonês de uma forma que faça sentido em Japonês. Ou seja, ao invés de aprender frases úteis, construir uma sólida fundação em gramática antes mesmo de dominar o idioma. O desafio com esse método é que conceitos gramaticais são os mais

difíceis. Portanto, o estudante estará aprendendo a parte mais difícil do idioma antes das outras.

A possibilidade de se utilizar uma ferramenta de software que auxilie o estudante neste processo de aprendizagem pode permitir ainda mais eficiência e eficácia no estudo do idioma japonês.

2.4 Inteligência Artificial

A Inteligência Artificial é a habilidade de um computador, ou um robô controlado por um computador, realizar tarefas associadas a seres inteligentes. O termo é aplicado aos projetos de desenvolver sistemas que englobam processos característicos de seres humanos, como raciocinar, descobrir, generalizar ou aprender com experiências passadas (COPELAND, B. J., 2020).

Os estudos de Inteligência Artificial podem ser resumidos em quatro estratégias que se relacionam a processos de pensamento e raciocínio e comportamento. São eles “pensando como um humano”, “agindo como seres humanos”, “pensando racionalmente” e “agindo racionalmente”. As duas primeiras medem o nível de fidelidade desses sistemas comparado ao desempenho do ser humano, enquanto as duas últimas medem um conceito ideal de inteligência, chamado “racionalidade”. O sistema age de forma racional ao fazer “a coisa certa” (NORVIG, RUSSEL, 2013).

Inteligência Artificial pode resolver problemas através de um computador. Há dois paradigmas, ou modelos, básicos para a solução de problemas. Um deles é o de situação-espaço, onde se tem uma situação inicial, um grupo de situações possíveis e um grupo de ações possíveis, assim como uma especificação de para qual situação cada ação poderá levar e uma situação desejável, ou objetivo. O problema ainda pode ter uma situação específica, que deve ser evitada. A solução do problema é, então, qualquer caminho de ações que leva ao objetivo. O outro paradigma é o de inferência no sistema. Há várias maneiras de realizar essa inferência do sistema envolvendo fórmulas ou cálculos matemáticos. A máquina fazendo a inferência precisa ter uma linguagem para representar sua informação e precisa determinar o melhor sistema para realizar a inferência. A máquina realizando

a inferência precisa conseguir ou tentar detectar que a evidência introduzida para inferência é auto-contraditório e propor soluções melhores (JACKSON, 1985).

Uma técnica utilizada para fazer inferência com uma máquina é a de redes neurais artificiais. As redes neurais artificiais na computação são baseadas no funcionamento do cérebro humano. Desde 1943, a ciência tem evoluído para modelos mais detalhados de neurônios ou outros sistemas do cérebro. Os pesquisadores de IA e estatísticos demonstraram interesse nas propriedades mais abstratas de redes neurais, como a capacidade de realizar computação distribuída, tolerar entradas ruidosas e aprender. Apesar de haver outros tipos de modelos semelhantes, redes neurais ainda são uma das formas de aprendizagem mais populares e eficazes de aprendizagem de sistema (NORVIG, RUSSEL, 2013). Um algoritmo raso aprende diretamente dos exemplos de treinamento. A maioria dos algoritmos é raso, a exceção principal sendo redes neurais que possuem mais de uma camada entre entrada e saída. Essas redes neurais são também chamadas de Redes Neurais Profundas. Em aprendizagem profunda (deep learning), a maioria dos parâmetros do modelo não são aprendidos dos exemplos de treinamento, mas das camadas posteriores (BURKOV, 2019).

Redes neurais são compostas de nós conectados por ligações direcionadas. Esses nós são os “neurônios” da rede. Cada ligação entre nó tem um peso associado, que determina a força e sinal de conexão. O neurônio de destino do nó calcula uma soma ponderada de suas entradas e aplica uma função de ativação a essa soma para obter uma saída. Um dos modos de conectar esses neurônios para formar uma rede é com alimentação para a frente, tendo conexões somente em uma direção. Essa rede normalmente é disposta em camadas, de forma que cada unidade recebe entradas somente a partir de unidades da camada anterior. Quando todas as entradas da rede estão conectadas diretamente à saída, essa é uma rede de camada única, ou rede perceptron. Porém, é possível adicionar camadas ocultas para melhorar o resultado final (NORVIG, RUSSEL, 2013).

2.5 Aprendizagem Profunda (*Deep Learning*)

Aprendizagem profunda, ou deep learning, é um tipo de aprendizagem de máquina (machine learning) que permite que sistemas de computadores aprendam com dados e experiência. Deep learning é um tipo específico de machine learning que possui poder e flexibilidade ao aprender a representar o mundo como um grupo hierárquico de conceitos, cada conceito definindo relações com conceitos mais simples. Apesar de deep learning e redes neurais serem inspiradas no cérebro biológico, o objetivo não é ser uma representação realista dele. O conceito neural de deep learning é motivado por duas idéias principais. A primeira é que o cérebro apresenta uma prova de que ações inteligentes são possíveis, e um caminho para reproduzir inteligência é fazer engenharia reversa dos princípios computacionais de um cérebro, duplicando sua funcionalidade. Outra idéia é que seria interessante entender o cérebro e os princípios da inteligência humana, então aprendizagem de máquina que simulam esses sistemas são úteis, além da habilidade de resolver problemas computacionais (BENGIO, COURVILLE, GOODFELLOW, 2016).

O conceito moderno de deep learning vai além da perspectiva neurocientífica. Se trata de um princípio mais geral sobre vários níveis de decomposição, que podem ser aplicados em técnicas de aprendizagem de máquina e não são necessariamente inspirados em neurônios (BENGIO, COURVILLE, GOODFELLOW, 2016).

2.5.1 Conceitos principais

Aprendizagem profunda é o treinamento de redes neurais com mais de duas camadas além da camada de entrada e saída. Implementações modernas de redes neurais permitem um treinamento eficiente de redes neurais. Na prática, muitos problemas podem ser resolvidos com a rede neural tendo duas ou três camadas entre a entrada e a saída. As camadas que não são nem entrada e nem saída são frequentemente chamadas de *hidden layers* (camadas invisíveis) (BURKOV, 2019).

Mesmo para seres humanos, pode se tornar difícil reconhecer caracteres escritos. O processo se torna ainda mais complexo quando se tenta fazer um computador reconhecer padrões de escritas, considerando exceções e casos especiais. Redes neurais podem utilizar um número grande de dígitos escritos à

mão, conhecidos como “exemplos de treinamento”, e implementar um sistema que aprende a reconhecer dígitos escritos à mão a partir desses exemplos. Assim, uma rede neural artificial se baseia nesses exemplos para criar regras de inferência, que ajudam na identificação dos caracteres similares (NIELSEN, 2015).

O MNIST é um banco de dados criado pelo NIST (*National Institute of Standards and Technology*) dos EUA, com milhares de imagens com caracteres escritos à mão. O uso de redes neurais artificiais para aprendizagem de caracteres a partir dos exemplos contidos no banco de dados MNIST chegou a resultados com taxas de erro de 1,6%, ou seja, uma taxa de acerto de 98,4% (LeCun, Bottou, Bengio, Haffner, 1998).

2.5.2 Redes Neurais Convolucionais

Uma rede neural convolucional (CNN) é um tipo especial de rede neural usada para reduzir o número de parâmetros em uma rede neural profunda com muitas unidades sem muita perda na qualidade do modelo. CNNs obtém os melhores resultados comparado com outros tipos de redes neurais no processamento de imagens ou textos. CNNs foram inventadas com o objetivo de processar imagens (BURKOV, 2019).

Imagens possuem muitas propriedades estatísticas não variáveis em alterações. Uma foto de um gato ainda é uma foto de um gato mesmo quando movido um pixel para a direita, por exemplo. Redes convolucionais consideram essa propriedade ao compartilhar parâmetros em vários locais na imagem. Uma camada invisível com os mesmos pesos é computada em locais diferentes na entrada. Isso significa que se pode reconhecer o gato com o mesmo detector, não importando se o gato aparece na coluna i ou na coluna i+1 na imagem. Compartilhamento de parâmetros permite que as CNN baixem consideravelmente o número de modelos únicos de parâmetros e aumentem o número de neurônios sem precisar aumentar o volume de dados (BENGIO, COURVILLE, GOODFELLOW, 2016).

3 Desenvolvimento do Sistema

Nessa seção, será mostrado como o sistema foi desenvolvido. A metodologia utilizada, como foi realizada a análise, assim como as tecnologias empregadas.

3.1 Metodologia

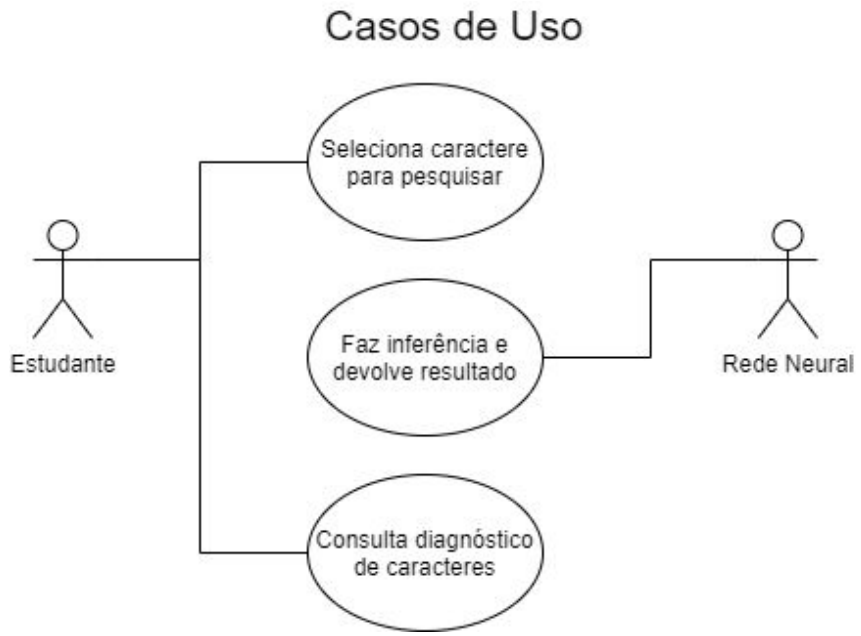
A metodologia utilizada no desenvolvimento do projeto foi a Solo Scrum, um processo iterativo e incremental unindo as boas práticas delineadas pelo Personal Software Process (PSP) e pelo Scrum. Essa metodologia tem como objetivo suprir a ausência de um processo para desenvolvedores solo (FABRI, GONÇALVES, L´ERARIO, PAGOTTO, 2016).

3.1.1 Análise

O objetivo do trabalho foi criar um aplicativo de celular, que pudesse ser usado para orientar estudantes intermediários do idioma japonês nos seus estudos. É considerado como estudante intermediário, estudantes que já possuem noção de gramática e conhecem ao menos Hiragana e Katakana. Sendo assim, o foco do projeto foi desenvolver uma aplicação para auxiliar o estudante a reconhecer sobre quais kanjis e palavras ele possui mais dificuldades. Para isso, foram definidos quais os requisitos necessários para melhor se atender esse objetivo, satisfazendo as necessidades do usuário.

A decisão foi criar um aplicativo onde o usuário possa selecionar uma imagem, que passará por uma inferência, para reconhecer o caractere e esse caractere será então armazenado em um banco de dados, realizando-se a contagem de quantas vezes aquele caractere precisou ser consultado. A Figura 3 mostra os casos de usos identificados para o aplicativo.

Figura 4. Casos de Uso



Fonte: Autor (2020)

Como mostrado na Figura 4, a rede neural, que faz a inferência, é representada como um ator, assim como o estudante, o usuário do aplicativo. O estudante pode então selecionar um caractere em uma imagem para pesquisar. Esse caractere será enviado para a rede neural, que fará a inferência e mostrará o resultado na tela. Esse resultado pode ser salvo pelo estudante para depois consultar em sua lista pessoal de diagnóstico, com os caracteres vistos e sua frequência.

A partir da atividade de levantamento de requisitos, foi elaborada uma lista com os requisitos funcionais principais, que o sistema deverá atender, que é mostrada na Tabela 1 a seguir.

Tabela 1. Requisitos do Sistema

Reconhecer mais de 2000 caracteres japoneses
Ser possível selecionar caractere para reconhecer
Identificar caractere selecionado
Mostrar informações do caractere reconhecido na tela
Consultar diagnóstico sobre caracteres adicionados
Persistência de dados offline para reconhecimento e diagnóstico

Fonte: Autor (2020)

A Tabela 1 mostra os requisitos que precisavam ser atendidos para desenvolver o sistema. O sistema deve reconhecer caracteres japoneses, no mínimo abrangendo todo o Jōyō kanji, tratando assim de quase todas as palavras mais usadas no idioma. Deve ser possível selecionar em uma imagem o caractere que se deseja reconhecer e enviar esse caractere para reconhecimento. As informações do caractere devem então aparecer na tela para o usuário validar o reconhecimento. Também é preciso ter um relatório, que mostre quais caracteres mais foram necessários se fazer o reconhecimento. O reconhecimento e diagnóstico dos caracteres deve funcionar offline, para que o usuário não dependa de uma conexão com a internet para os estudos.

3.1.2 Projeto

A partir da análise dos requisitos levantados, foram tomadas as seguintes decisões de projeto. O sistema deveria possuir uma rede neural para reconhecer 3036 caracteres japoneses, reconhecendo portanto os caracteres mais utilizados diariamente no idioma. Deveria ser possível selecionar uma parte de uma imagem, onde estivesse o caractere que se desejaria identificar, realizar essa identificação e mostrar as informações sobre o caractere na tela. Depois, deveria ser possível montar uma tela com uma lista com caracteres que mais precisaram ser identificados, montando assim um diagnóstico para o estudante identificar quais caracteres possui mais dificuldade em reconhecer. Também era importante manter persistência de dados offline, para que essa tabela pudesse ser consultada a qualquer momento.

Foi planejado o desenvolvimento do projeto através de um Kanban elaborado na plataforma online Trello. O Trello é uma ferramenta online gratuita para gerenciamento de projetos, que é organizada em quadros onde são inseridas uma lista de tarefas que precisam ser concluídas (LOUBAK, A. A. 2019). O desenvolvimento foi planejado para ser concluído em cinco sprints, listados na Tabela 2 a seguir.

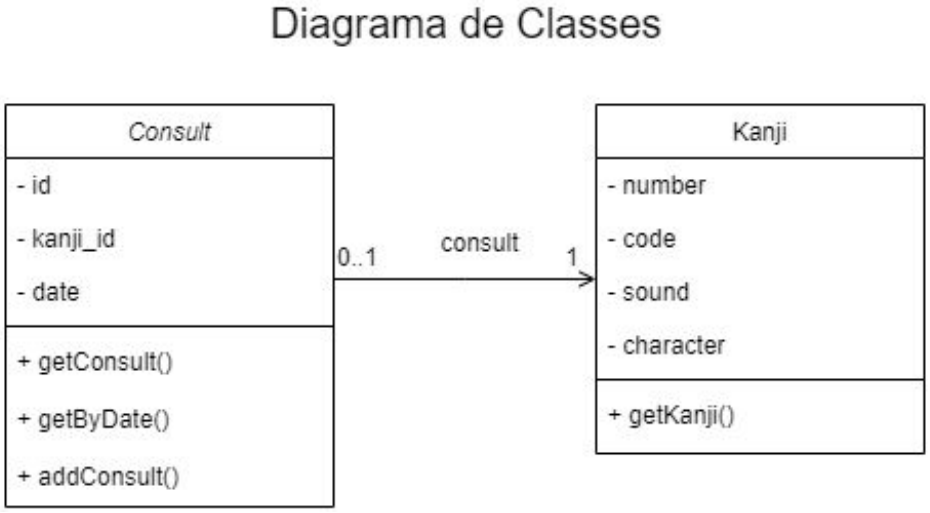
Tabela 2. Sprints do Projeto

Sprint	**Tarefas**
Primeira Sprint	Treinamento da rede neural. Preparar o dataset. Tratamento das imagens.
Segunda Sprint	Conclusão de treinamento com inferência e validação.
Terceira Sprint	Começar desenvolvimento da interface. Selecionar imagem, cortar e trabalhar.
Quarta Sprint	Carregar rede neural na aplicação. Validar inferência.
Quinta Sprint	Criar banco de dados SQLite. Montar tela de diagnóstico.

Fonte: Autor (2020)

Conforme mostrado na Tabela 2, na primeira sprint, foi determinada a realização das tarefas, que teriam ligações com o treinamento da rede neural, como a preparação do dataset e o próprio treinamento. Na segunda sprint, foi feita a conclusão deste treinamento com a validação do resultado. Na terceira sprint, foi onde se iniciou o desenvolvimento da interface para o aplicativo e como selecionar, recortar e tratar imagem. Na quarta sprint, a rede neural previamente desenvolvida foi adicionada ao aplicativo final, validando seu funcionamento. Na quinta sprint, foram realizadas as tarefas relacionadas a criar o banco de dados, que o aplicativo iria utilizar e que seria usado para montar a tela de diagnóstico para o usuário. Para auxiliar na criação do banco de dados e das funcionalidades, foi modelado um diagrama de classes, mostrado na Figura 5.

Figura 5. Diagrama de Classes



Fonte: Autor (2020)

A Figura 5 mostra o Diagrama de Classes, com as duas classes principais do sistema. A classe Kanji é relacionada com as informações e ações relacionadas à interação com a rede neural. Em seus atributos, são salvas informações do kanji, sendo “number” o número da classe dele dentro da rede neural e também seu atributo identificador; “code”, o código JIS que o identifica; “sound”, as possíveis leituras dele; e “character”, o próprio símbolo que representa o kanji. Como a rede neural é imutável, essa classe também tem atributos imutáveis, sendo modificada somente se uma nova rede foi treinada. Sua única funcionalidade é buscar informações do kanji através do número da classe que a rede neural irá devolver.

A classe Consult é relacionada ao diagnóstico salvo pelo usuário. O atributo “id” é um identificador daquela linha de diagnóstico, “kanji_id” é uma referência ao id do kanji na classe Kanji, “date” é a data e hora que o caractere foi consultado. Suas ações são “getConsult” que trará uma lista com todos os diagnósticos, agrupados por caracteres, “getByDate” que faz a mesma consulta trazendo somente resultados dentro um período de tempo e “addConsult” adiciona uma nova linha ao diagnóstico.

3.1.3 Codificação

O treinamento da rede neural foi realizado com uma implementação em software, utilizando a linguagem Python e as bibliotecas Keras e Tensorflow, e implantado no Google Colab. Colab, abreviação de Colaboratory, é um produto da Google

Research. Com ele é possível executar códigos de Python através de um navegador, sendo desenvolvido especialmente para machine learning, análise de dados e educação. Fornece acesso gratuíto para diversos recursos computacionais, incluindo GPU (GOOGLE, 2020).

Python é uma linguagem de alto nível, que é considerada uma linguagem interpretada, porque programas em Python são executados por um interpretador. Ela pode ser usada em modo interativo ou modo por código (DOWNEY, 2008, p3).

Keras é uma biblioteca de código aberto escrita em Python para trabalhar com redes neurais. Ela pode ser usada junto com outras bibliotecas, como Tensorflow, Theano, PlaidML, além de outras. Em 2017, ela foi integrada no framework Tensorflow, desenvolvido pelo Google (KERAS, 2020).

Tensorflow é um framework de código aberto, desenvolvido pelos pesquisadores da Google para rodar códigos de machine learning, deep learning e outros associados com estatísticas e análises preditivas (ROUSE, M. 2018).

Para desenvolvimento da aplicação para celulares, foi utilizado o framework React Native, desenvolvido em Javascript, que é uma linguagem de programação normalmente utilizada para desenvolvimento web. Foi criada para adicionar elementos dinâmicos e interativos em páginas da internet. É uma linguagem executada no navegador do cliente, ao invés de em um servidor (CHRISTENSSON, 2014).

React Native é um framework baseado em React, que permite o desenvolvimento de aplicações para celulares, tanto para Android como para iOS, utilizando apenas JavaScript. Todo o código desenvolvido no React Native é convertido para a linguagem nativa do sistema operacional (BECKER, 2020).

3.2 Desenvolvimento da Rede Neural Profunda

Para o desenvolvimento de uma rede neural, é preciso antes preparar um dataset, certificando-se de que ele atende o necessário para o que a rede precisa prever. Então, esse dataset será usado como entrada para o treinamento da rede neural, gerando um arquivo com o modelo, que poderá ser utilizado para se fazer a inferência sobre imagens submetidas a ela. Para saber se esse arquivo com o

modelo treinado está funcionando como se espera, deve-se realizar testes com o mesmo. Após confirmado sua eficiência, esse modelo pode ser utilizado em produção.

3.2.1 Preparação do data set

O dataset utilizado neste trabalho é o ETL-9G, contendo 3036 caracteres compactados, sendo esses alguns dos principais kanjis utilizados no idioma japonês, além de todos os hiraganas. O dataset se encontra disponível no link http://etlcdb .db.aist.go.jp/specification-of-etl-9.

Dos 3036 caracteres no dataset, 46 são do Hiragana e 2290 são do Kanji, um número maior do que aqueles estabelecidos no Joyo Kanji, o que demonstra a possibilidade de ser utilizado para auxiliar em leituras. São 20 imagens de cada caractere escritos por 10 escritores, totalizando um total de 200 imagens por caracteres, o que totaliza um total de 607200 imagens no dataset. Cada imagem possui uma resolução de 127 x 128 pixels.

A Figura 6 mostra exemplos de kanjis, contidos no dataset utilizado, desenhados manualmente por diferente autores com os caracteres brancos e o fundo preto e armazenados no formato png. O dataset ainda contém, além das imagens, informações sobre a mesma, como o som e o código identificador.

Figura 6. Exemplo do Dataset

Fonte: Autor (2020)

O primeiro passo para o tratamento das imagens foi transformar de uma resolução 127 x 128 para uma resolução 32 x 32 e devolver um array correspondente às camadas RGB da imagem. Após essa etapa, foi separado 20% do número total de imagens para fazer a validação e 80% do dataset então foi usado para o treinamento.

Além disso, foram criadas mais imagens aleatórias, rotacionando aleatoriamente algumas imagens entre -15º e 15º, e aplicando zoom aleatoriamente de até 20% comparado à imagem original. Essas imagens geradas são para considerar situações que a letra possa estar inclinada ou ter tamanhos diferentes do modelo treinado.

O processo de descompactar as imagens acaba exigindo um volume grande de armazenamento de dados. O processo com 3036 classes exige mais de 30 GB de RAM com as imagens descompactadas. Como o Google Colab libera no máximo 25 GB, se tornou impossível utilizá-lo para a tarefa. Portanto, foi primeiro criado uma rede neural utilizando somente 1000 classes, determinado seu funcionamento e utilizada outra máquina (Notebook G5-5590, com processador Intel Core i7, 16 GB de RAM, 1 TB de HD com 256 GB sendo SSD, e GPU NVIDIA GTX 1660Ti) para realizar o treinamento final com os 3036 caracteres. Para isso, o arquivo descompactado foi primeiro exportado e então carregado durante o treinamento, evitando assim a necessidade de uma máquina com uma quantidade elevada de memória RAM para o pré-processamento das imagens a cada treinamento.

3.2.2 Treinamento

A rede neural construída foi uma rede neural convolucional, que é a rede comumente utilizada para se fazer o treinamento com imagens. Após realizar a inferência, a rede devolve um valor, que identifica a classe da imagem. Cada classe representa um dos 3036 caracteres.

.A primeira camada da rede neural é uma camada convolucional de 32 neurônios. Depois os dados passam por uma camada de ativação que vai aplicar a função RELU, seguindo então para uma segunda camada convolucional de 32 neurônios, outra camada de ativação com relu. Depois passa por uma camada de pooling, que usa o espaço de valores capturado pela camada convolucional e transforma em um valor único. Por fim, uma camada Dropout, que aleatoriamente coloca valores 0 nas entradas, produzindo como efeito a desativação aleatória de alguns neurônios.

As próximas 6 camadas seguem o mesmo padrão acima, mas ao invés de 32 neurônios, as camadas convolucionais possuem 64 neurônios. Depois vem uma camada Flatter, que nivela os dados, uma camada densa, ou seja, que cada neurônio de entrada é conectado por um neurônio de saída, com 256 neurônios. Outra camada de ativação com relu, outra camada Dropout e uma camada de saída final densa com o número de neurônios do mesmo tamanho do número de classes, ou seja, 3036. Depois ainda passa por outra camada de ativação, que aplica a função softmax.

O treinamento foi realizado adotando 40 épocas, ou seja, o dataset passa pelo treinamento 40 vezes, com um tamanho de lote (*batch size*) de 512, significando que o treinamento foi feito utilizando partes do datasets com 512 exemplos do número total de imagens para treinamento por vez. O melhor resultado final obtido foi de 95% de acurácia (*accuracy*).

Como as imagens utilizadas no treinamento são com fundo preto e cor de caractere branco, na hora de utilizar o modelo é preciso se certificar que a imagem sendo utilizada para inferência também está nesse padrão. Caso contrário, é necessário realizar um pré-processamento gráfico da imagem.

3.2.3 Utilização do modelo treinado

Após o treinamento completo, a biblioteca Keras devolve o arquivo com o modelo, em formato JSON, representando a rede neural treinada. Passando uma imagem para inferência utilizando Keras ou Tensorflow, configurado com este arquivo de modelo, a rede neural treinada devolve um resultado referente ao número da classe, que corresponde ao resultado da predição. Consultando no dataset, é possível pesquisar qual código de caractere essa classe representa.

3.2.4 Testes

A rede neural teve 95% de acurácia na validação, utilizando 20% das imagens do dataset, como melhor resultado obtido no treinamento. Porém, considerando que as imagens do dataset são feitas pelos mesmos escritores, foram

realizados testes individuais para determinar o funcionamento da rede neural em caracteres que não foram escritos pelos mesmos escritores usados no treinamento. Para isso, foram selecionados 15 caracteres com complexidades variadas. Quando uma imagem falhou em ser reconhecida pela inferência, foi testado o mesmo caractere com uma grafia diferente. Todas os 15 caracteres passaram corretamente pela inferência, porém, algumas imagens falharam. Uma comparação entre a imagem que falhou e foi prevista corretamente do mesmo caractere oferece uma visão de como foi realizado o treinamento da rede neural.

A Figura 7 mostra três exemplos de caracteres iguais com imagens que a inferência acertou ou errou o caractere. O primeiro caractere representa a palavra “Derramar em”, o segundo a palavra “Amor” e o terceiro a palavra “Esposa”.

Figura 7. Falha de Inferência nas Imagens

Fonte: Autor (2020)

De acordo como o mostrado na Figura 7, é possível perceber que, no primeiro e no terceiro caractere em que a inferência falhou, a imagem não reconhecida possui uma grafia peculiar, diferente da grafia convencional do caractere, o que explica a dificuldade da rede neural em reconhecer. No segundo caractere, representando “Amor”, porém, a diferença está na espessura das linhas.

A Figura 8 mostra imagens com baixa resolução, que foram utilizadas para teste da inferência.

Figura 8. Imagens de Baixa Resolução

Fonte: Autor (2020)

Todas as imagens com baixa resolução não passaram por nenhum processo de melhorar sua qualidade e todas falharam na predição. Isso comprova a necessidade de otimizar a qualidade da imagem antes de passá-la para inferência, ou de realizar treinamento com estes tipos de imagem de baixa resolução.

3.3 Desenvolvimento da Aplicação Mobile

Nessa seção será mostrado como foi desenvolvida a aplicação para celular. A aplicação foi desenvolvida utilizando o framework javascript React Native, onde o desenvolvimento é feito usando javascript e depois convertido para linguagem nativa ao compilar. React Native exporta para Android e iOS, mas a aplicação só foi testada em dispositivo Android.

3.3.1 Desenvolvimento da Aplicação

A aplicação foi desenvolvida utilizando React Native. A lógica de programação é desenvolvida com javascript, utilizando componentes para acessar funcionalidades do telefone ou exibir na tela. Esses componentes estão em sua maioria em uma biblioteca chamada “react”. A navegação entre as telas é controlada por uma biblioteca chamada “react-navigation”, que define qual tela vai aparecer de acordo com a ação do usuário.

Os componentes que constroem a parte visual do sistema são escritos com tags, de uma forma que lembra HTML. A estrutura é parecida com a da linguagem

HTML, mas utilizando componentes próprios do react. O estilo dos componentes são definidos através de códigos CSS.

3.3.2 Utilizando a Rede Neural

Ao realizar o treinamento da rede neural, a biblioteca Keras em Python retorna um arquivo JSON, contendo as informações da rede neural para serem utilizadas. Essa rede precisa ser convertida para o uso dentro da aplicação mobile. Para isso, se utiliza uma biblioteca Python, chamada tensorflowjs, que converte o arquivo JSON entregue pela Keras em um arquivo JSON contendo as configurações da rede e um ou mais arquivos .bin contendo os valores dos pesos calculados na saída. Com 1000 caracteres, foi produzido somente um arquivo bin, mas com 3036 caracteres foram produzido dois. Nesse caso, foi necessário juntar os dois arquivos em um para ser utilizado no React Native.

Após convertida, a rede neural funciona no React Native através da biblioteca tensorflowjs. A utilização é da mesma forma de quando é utilizada no Python. Uma imagem pode ser enviada para inferência e será retornado um valor numérico referente à classe, a qual a imagem pertence.

3.3.3 Tratamento de Imagem

A biblioteca *react-native-image-picker* foi utilizada para se ter um componente para seleção de um pedaço da imagem, que o usuário usa para definir o caractere para enviar para inferência. Esse espaço definido pelo usuário é cortado e usado para se criar uma nova imagem a partir dele.

As imagens utilizadas no treinamento possuem fundo preto e cor de caractere branco, porém, quando realizando a leitura, esse padrão costuma ser inverso. Portanto, é preciso fazer a inversão de cores antes de passar para a inferência. Isso é feito utilizando a biblioteca *react-native-canvas* que utiliza o canvas do HTML para modificar a imagem.

Por fim, ainda é feito um último tratamento na imagem. Como as imagens utilizadas para treinamento são monocromáticas e as imagens recortadas podem

apresentar padrões de cinza, são obtidos os valores dos canais RGB da imagem, dividido por 3 e somado 100. Essa soma vai garantir que, quando o valor é dividido novamente por 255, o valor máximo de um canal de cor terá valores arredondados 1 para branco e 0 para preto.

3.3.4 Banco de Dados

A última etapa da construção da aplicação de celular foi a de implementação do banco de dados. Como a rede neural devolve somente a classe do caractere na rede neural, essa informação não diz nada ao usuário. O banco de dados contém uma tabela com a classe, o som e o próprio caractere. Essas informações podem então serem exibidas ao usuário.

A outra tabela no banco de dados guarda quais caracteres foram passados por inferência e em qual data, possibilitando se montar um diagnóstico para que o usuário saiba os caracteres, em que enfrenta mais dificuldade para reconhecer.

A tradução do kanji não se encontra no banco de dados, sendo usada a API do Google Translate. Esse é, portanto, o único recurso não disponível quando não há conexão com a internet. O Cloud Translation, do Google, é utilizado para traduzir textos dinamicamente entre diversos pares de idiomas com o uso de inteligência artificial (Google, 2020).

3.3.5 Testes

Foram realizados testes unitários nas funções do aplicativo. Testes unitários acontecem quando as partes individuais do software são testadas com o propósito de avaliar que cada unidade do módulo funciona como esperado. Testes unitários são realizados durante o desenvolvimento da aplicação e isolam uma parte do código para verificar sua validade (Guru99, 2020). Para auxiliar nos testes foi utilizada a ferramenta jest. Jest é um framework para testes projetado para garantir o funcionamento correto de qualquer código Javascript (Jest.fun, 2020).

Os testes foram feitos nos métodos dos três componentes da aplicação, do componente principal, de inferência e diagnósticos. Foi criado um teste para verificar

se as conexões com o banco de dados estão trazendo os resultados esperados, assim como se as telas são visíveis de forma correta em aparelhos com outras resoluções através de snapshots.

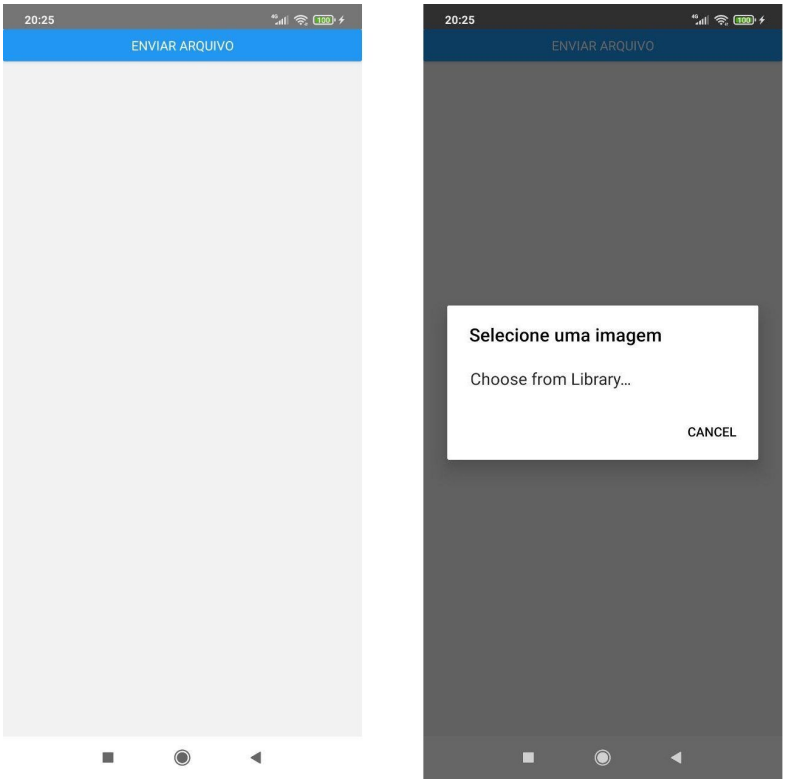
4 Experimentos

Essa seção mostra o uso da aplicação, com telas e explicações sobre o funcionamento tanto da tela de reconhecimento de caracteres, quanto da tela de diagnósticos.

4.1 Reconhecimento de caracteres

Ao abrir a aplicação no celular, o usuário é apresentado à tela inicial, onde pode selecionar uma imagem que possui o caractere, sobre o qual deseja fazer a inferência. A Figura 9 mostra a tela para o envio da imagem, onde se encontra o caractere para inferência.

Figura 9. Tela de Envio da Imagem



Fonte: Autor (2020)

O usuário deve tocar no botão “Enviar Arquivo” e o programa pedirá que ele escolha uma imagem para buscar o caractere desconhecido. O caractere pode então ser selecionado como mostra a Figura 10.

Figura 10. Inferência do Caractere

Fonte: Autor (2020)

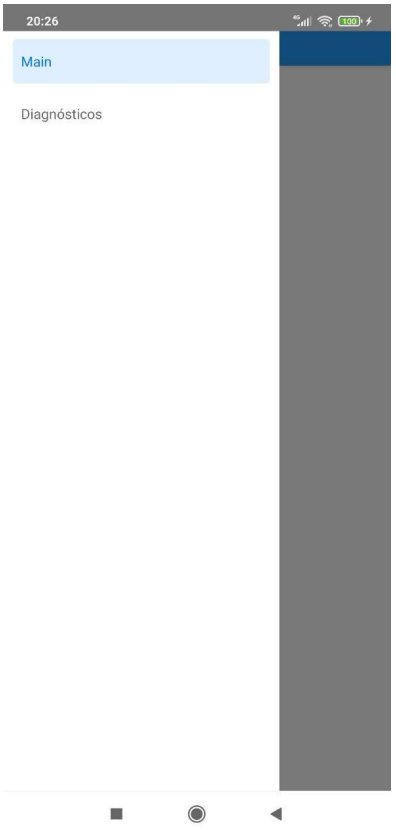
Como mostrado na Figura 10, a imagem selecionada é exibida na tela e o usuário pode mover um quadrado azul pela tela, mudando de tamanho quando necessário. O usuário deverá arrastar esse quadrado até o caractere que deseja

fazer a inferência, ajustar o tamanho para pegar o caractere e pressionar. A imagem terá a cor invertida, como demonstrado na imagem e será enviada para a inferência na rede neural, trazendo o resultado na tela com o caractere encontrado pela rede, a leitura do caractere e uma tradução da palavra. Nesse momento, o usuário poderá pressionar o botão verde para dizer que o caractere é esse mesmo, o que o gravará na tela de diagnósticos ou o botão vermelho para dizer que o caractere está errado, o que apagará as informações do caractere. É possível fazer a inferência de vários caracteres antes de confirmar e enviar ao banco de dados.

4.2 Diagnósticos

Além da tela de inferência, a aplicação conta com uma tela de diagnóstico, onde o usuário poderá ver um histórico dos caracteres que consultou utilizando o aplicativo. A Figura 11 mostra como, a partir da tela inicial, o usuário pode puxar um menu com as duas telas principais do aplicativo, puxando o lado esquerdo da tela.

Figura 11. Menu do aplicativo



Fonte: Autor (2020)

Como mostrado na FIgura 11, a primeira opção leva para a tela inicial onde é possível buscar uma imagem, enquanto a segunda opção leva para a tela de diagnósticos. A Figura 12 mostra a tela de diagnósticos para o estudante.

Figura 12. Tela de Diagnósticos

Fonte: Autor (2020)

Como o caractere pesquisado e a data pesquisada são salvos no banco, a tela mostra o número de vezes que um caractere precisou ser pesquisado, ordenando pelo caractere mais visto, conforme ilustrado na Figura 12. A tela exibe um resultado para o período total de uso do aplicativo, um período mensal, trazendo

as consultas feitas nos últimos 30 dias e um período diário, trazendo a consulta feita no dia que o diagnóstico foi aberto.

5 Trabalhos Correlatos

Essa seção apresenta outros sistemas similares ao apresentado nesse artigo, seja para desktop ou para celulares. Traz também uma comparação entre os sistemas apresentados e o desenvolvido para este trabalho.

5.1 KanjiTomo

O KanjiTomo é um programa desenvolvido para se identificar caracteres japoneses através de imagens, em que se pode apontar o mouse para qualquer imagem que está na tela do computador e consultar o dicionário ao mesmo tempo. O programa suporta tanto textos verticais, quanto horizontais. Foi desenvolvido utilizando a linguagem de programação Java (KanjiTomo, 2019).

O KanjiTomo funciona somente em um computador desktop e apresenta alguns recursos que o Nihongo Jouzu não possui, como uma visão computacional, que identifica o caractere automaticamente e um dicionário interno, dispensando a conexão com a internet para realizar a tradução. Além da falta de disponibilidade em celular, porém, o KanjiTomo também não possui uma ferramenta de diagnósticos dos caracteres, que são capturados pelo estudante. É possível salvar os caracteres em uma lista e exportá-la como texto, mas não há nenhuma forma de organização dentro da aplicação.

5.2 Kaku

Kaku é um dicionário japonês, que executa por cima dos outros aplicativos em um dispositivo Android. Ele utiliza a tecnologia OCR (*Optical Character Recognition*) para reconhecer os caracteres na tela do celular. Kaku é o equivalente ao KanjiTomo para Android (Kaku, 2019).

O Kaku, assim como o KanjiTomo, possui visão computacional e funciona sobre outros aplicativos, sendo possível ler direto da tela do celular. Assim como o KanjiTomo, porém, ele não oferece uma opção de guardar caracteres vistos, nem mesmo exportar uma lista, sendo uma ferramenta usada inteiramente para leitura. Falta, portanto, ferramentas que auxiliem o estudo.

6 Conclusão e Trabalho Futuro

Este artigo reportou o desenvolvimento de uma aplicação para celular, que reconhece caracteres japoneses em imagens com o uso de uma rede neural. Foi detalhado o desenvolvimento e treinamento da rede neural, o tratamento da imagem para inferência, assim como a construção do aplicativo e banco de dados para trazer informações sobre o caractere e montar o diagnóstico para auxiliar o estudante.

A construção do sistema trouxe conhecimentos sobre como funciona e como montar e treinar uma rede neural, sendo *deep learning* uma área importante da inteligência artificial, que está em uso crescente na academia e na indústria. Se aprofundar na área será importante para todo profissional no futuro. Além disso, o sistema trouxe outra área em expansão, a do desenvolvimento de aplicativos para celulares. Aplicações móveis já são mais utilizadas que aplicações *desktop*, e tão importante quanto aplicações *Web*. É importante que todo desenvolvedor tenha conhecimento de como desenvolver esse tipo de aplicações.

Será muito importante trabalhar melhor a parte de diagnósticos do aplicativo, por exemplo, deixando o usuário selecionar um período na hora de trazer a frequência que caracteres foram pesquisados. Também seria importante no auxílio do idioma, identificar radicais dos caracteres ou palavras que podem ser formadas com aqueles caracteres. A disponibilidade das palavras também será importante para se criar um dicionário interno, dispensando o uso de uma API externa para exibir na tela informação sobre o que aquele caractere representa, já que atualmente a única informação não dependente da internet é o som. Também auxiliaria no uso do aplicativo, desenvolver uma visão computacional que reconheça os caracteres em um espaço de imagem. Dessa forma, seria possível selecionar um pedaço da imagem com vários caracteres e o próprio sistema identificaria todos os caracteres

naquele espaço. Outra atualização para o futuro, é a possibilidade de tirar fotos de uma imagem, que contenha o caractere para fazer o reconhecimento, permitindo assim o uso para material físico, como livros.

REFERÊNCIAS

BENGIO, Y; COURVILLE A; GOODFELLOW I. **Deep Learning**. 1ª Edição. Estados Unidos: The Mit Press. 2016.

BECKER, L. **O Que É React Native?.** 2020. Disponível em: <https://www.organicadigital.com/blog/o-que-e-react-native/>. Acesso em 04/10/2020.

BURKOV, A. **The Hundred-Page Machine Learning Book**. 1ª Edição. Canadá: Andriy Burkov. 2019.

CHRISTENSSON, P. **JavaScript Definition.** 2014. Disponível em: <https://techterms .com/definition/javascript>. Acesso em 04/10/2020.

COPELAND, B. J. **Artificial intelligence.** 2020. Disponível em: <https://www .britannica.com/technology/artificial-intelligence> Acesso em 24/09/2020.

DOWNEY, A.B. **Think Python**: **How To Think Like a Computer Scientist**. 1ª Edição. Needham MA, USA: Green Tea Press O'Reilly Media, Inc. 2008.

FABRI, J. A.; GONÇALVEZ, J.A.; L´ERARIO, A.; PAGOTTO, T. **Scrum Solo Processo de software para desenvolvimento individual.** 2016. Disponível em: <https://engenhariasoftware.files.wordpress.com/2016/04/scrum-solo.pdf>

GOOGLE. **Colaboratory.** Disponível em: <https://research.google.com/colaboratory/faq.html>. Acesso em 04/10/2020.

GOOGLE. **Documentação do Cloud Translation.** Disponível em: <https://cloud.google.com/translate/docs>. Acesso em 12/10/2020.

GURU99. **Unit Testing Tutorial: What is, Types, Tools, EXAMPLE.** Disponível em: <https://www.guru99.com/unit-testing-guide.html>. Acesso em 21/12/2020.

HEISIG, J.W. **Remembering the Kanji 1: A Complete Course on How Not To Forget the Meaning and Writing of Japanese Characters**. 4ª Edição. Tóquio, Japão: Japan Publications Trading Co., Ltd. 2001.

HATASA, Y.A; HATASA, K.;MAKINO, S. **Nakama 1**. 2ª Edição. Estados Unidos: Cengage Learning, 2010.

JACKSON, P. C. **Introduction to Artificial Intelligence**. 2ª Edição. Estados Unidos: Dover Publications, 1985.

JEST.FUN. **What is Jest?** Disponível em: <https://jest.fun/what-is-jest-test.html>. Acesso em 21/10/2020.

KAKU. **Japanese OCR Dictionary.** Disponível em: <https://kaku.fuwafuwa.ca/>. Acesso em 12/10/2020.

KANJITOMO. **Introduction**. Disponível em: <https://www.kanjitomo.net/>. Acesso em 16/09/2020.

KERAS. **What Is Keras?** Disponível em: <https://deepai.org/machine-learning-glossary-and-terms/keras>. Acesso em 04/10/2020.

KIM, T. **A Guide To Japanese Grammar: A Japanese Approach To Learning Japanese Grammar.** 1ª Edição. Estados Unidos: Createspace, 2014.

KOICHI. **Learn Hiragana: The Ultimate Guide.** Disponível em: <https://www.tofugu.com/japanese/learn-hiragana/> . Acesso em: 16/09/2020.

LECUN, Y. ;BOTTOU, L; BENGIO, Y.; HAFFNER, P. **Gradient-Based Learning Applied To Document Recognition.** 1998. Disponível em: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf. Acesso em 16/09/2020.

LOUBAC, A. L. **Como Funciona o Trello? Saiba tudo sobre programa para organizar objetos**. 2019. Disponível em: <https://www.techtudo.com.br/listas/2019/10/como-funciona-o-trello-saiba-tudo-sobre-programa-para-organizar-projetos.ghtml>. Acesso em 03/10/2020.

MITYE, Camila. **O Alfabeto Japonês**. Disponível em: <https://brasilescola.uol.com.br/japao/o-alfabeto-japones.htm>. Acesso em: 16/09/2020.

NIELSEN, M. A. **Neural Networks and Deep Learning.** Estados Unidos: Determination Press, 2015.

NORVIG, P; RUSSELL, S. J. **Inteligência Artificial.** Tradução de Regina Célia Simille. 3ª Edição. Rio de Janeiro. Elsevier, 2013.

RAFAEL, Luiz. **Alfabeto Japonês – Como funciona**. Disponível em :<https://www.aulasdejapones.com.br/alfabeto-japones/> . Acesso em: 16/09/2020.

ROUSE, M. **TensorFlow**. 2018. Disponível em: <https://searchdatamanagement .techtarget.com/definition/TensorFlow>. Acesso em: 04/10/2020.

Suki Desu. **Jōyō kanji: Os 2136 kanjis mais usados.** Disponível em: <https://skdesu.com/joyo-kanji-os-2136-kanjis-mais-usados/>. Acesso em 23/09/2020.