INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CURSO TÉCNICO EM INFORMÁTICA PARA INTERNET
CAMPUS NATAL - ZONA NORTE

BRUNO TAVARES DOS SANTOS
JOÃO PEDRO DE SOUZA E SILVA
MARCOS FILIPE GARCIA DANTAS

ZU: SISTEMA PARA AUXILIAR PESSOAS PORTADORAS DE NECESSIDADE ESPECIAL VISUAL NO RECONHECIMENTO DE OBJETOS

NATAL - RN
2018

BRUNO TAVARES DOS SANTOS
JOÃO PEDRO DE SOUZA E SILVA
MARCOS FILIPE GARCIA DANTAS

ZU: SISTEMA PARA AUXILIAR PESSOAS PORTADORAS DE NECESSIDADE ESPECIAL VISUAL NO RECONHECIMENTO DE OBJETOS

Relatório de Prática Profissional apresentado à Coordenação do Curso Técnico em Informática para Internet do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Técnico em Informática para Internet.

Orientador: Rodolfo da Silva Costa

Coorientador: Dr. Diego Silveira Costa Nascimento

NATAL - RN
2018

DIREITOS DE AUTOR

Esta produção está assegurada sob uma Licença *Creative Commons*.
O uso do conteúdo está declarado sob as seguintes condições:

Atribuição-Não Comercial-Sem Derivações
CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0/

BRUNO TAVARES DOS SANTOS
JOÃO PEDRO DE SOUZA E SILVA
MARCOS FILIPE GARCIA DANTAS

ZU: SISTEMA PARA AUXILIAR PESSOAS PORTADORAS DE NECESSIDADE ESPECIAL VISUAL NO RECONHECIMENTO DE OBJETOS

Relatório de Prática Profissional apresentado à Coordenação do Curso Técnico em Informática para Internet do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Técnico em Informática para Internet.

Relatório apresentado em ___/___/___, e avaliado pela seguinte Banca Examinadora:

BANCA EXAMINADORA

_____

Prof. Esp. Rodolfo da Silva Costa - Presidente
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Prof. Dr. Diego Silveira Costa Nascimento - Avaliador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Prof Dr. Francisco das Chagas da Silva Junior - Avaliador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

AGRADECIMENTOS

Agradecemos aos nossos orientadores, Rodolfo da Silva Costa e Dr. Diego Silveira Costa Nascimento, pelo suporte que nos têm providenciado no desenvolvimento do projeto, e por seus ensinamento em nossa trilha como pesquisadores.

Agradecemos também à nossa colega de turma e amiga Gabriela Melo Silva, bem como ao nosso ex-professor, Addson Araújo da Costa, pelo apoio que nos têm fornecido, seja este moral, a avaliação de nossos relatórios ou auxílio em nossas pesquisas.

E também aos nossos amigos, que estiveram sempre presentes em nossos momentos de disponibilidade e foram pacientes e compreensivos em nossos momentos de ausência.

“Deus Vult”

(A resposta dos cristãos à convocação da Primeira Cruzada pelo papa Urbano II, Concílio de Clermont, 1095)

RESUMO

Os integrantes da comunidade das pessoas portadoras de deficiência visual enfrentam recorrentes problemas relativos a locomoção e se mantém dependente de ferramentas majoritariamente limitadas (por fatores como altos preços, no caso de cães-guia) e analógicas (como é o caso das bengalas), e, mesmo com os crescentes investimentos em tecnologia e inovação, sistemas anteriormente desenvolvidos para solucionar estes problemas costumam possuir limitações de funcionalidade ou métodos ortodoxos e pouco acessíveis ou de usabilidade satisfatória para o público. Apresentamos, portanto, o Zu, um sistema para ambiente *mobile* que contará com uma integração de ferramentas para facilitar a locomoção de deficientes visuais e cegos, utilizando-se do sistema de reconhecimento de objetos YOLO (*You Only Look Once*) para detecção e classificação de obstáculos e pelo sistema padrão de sintetização de voz de celulares Android, numa interface que lê o ambiente ao redor do usuário coletando informação visual através da câmera e repassa a este todo e qualquer objeto ou obstáculo encontrado, através da interface de áudio. A arquitetura do sistema permite que a informação seja passada rapidamente ao usuário de maneira simples e eficiente, com atraso variante, porém baixo, devido ao fato de que o processo de detecção e classificação dos objetos e processamento de imagem serão realizados em um servidor externo, que estará passando as informações aos clientes, para que estes informem então o usuário.

Palavras-chave: YOLO. Redes neurais. Deficiência visual. Acessibilidade.

ABSTRACT

Integrants of the community of visually impaired people face constant problems with locomotion and have remained dependent on tools that are mostly limited (such as high prices for guide dogs) and analog (such as walking sticks), and even with increasing investments in technology and innovation, previously developed systems to solve these problems usually have limitations of functionality or orthodox methods and of unsatisfactory accessibility or usability for the public. We present, therefore, Zu, a system for *mobile* environment that will have an integration of tools to facilitate the locomotion of the blind and visually impaired, using the YOLO (*You Only Look Once*) object recognition system for detection and classification of obstacles and the standard Android phone voice synthesizer system in an interface that reads the environment around the user collecting visual information through the camera and gives the user data about any and every object or obstacle found through the audio interface. The system architecture allows the information to be passed quickly to the user in a simple and efficient way, with a variant but low delay, due to the fact that the process of detecting and classifying objects and image processing will be performed in an external server, that will send the information to the clients, so that they inform the user then.

Keywords: YOLO. Neural networks. Visual impairment. Accessibility.

LISTA DE ABREVIATURAS E SIGLAS

mAP	mean Average Precision
YOLO	You Only Look Once
IA	Inteligência Artificial
CPU	Central Processing Unit
GPU	Graphic Processing Unit
GPGPU	General-Purpose computing on Graphic Processing Unit
ML	Machine Learning
DL	Deep Learning
IDE	Integrated Development Environment
FPS	Frames Per Second
OS	Operating System
ANN	Artificial Neural Network
API	Application Programming Interface

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1. Introdução

Através da realização de pesquisas de campo, sendo estas compostas por entrevistas realizadas com pessoas portadoras de necessidade especial visual, foi possível perceber interesse por uma ferramenta móvel e acessível com tais propriedades e que fosse capaz de comunicar-se, sendo capaz de auxiliá-las em sua vida diária no que se refere a locomoção. Estes relataram não encontrar boas ferramentas para seu auxílio em tarefas gerais no dia-a-dia, sejam estas analógicas (como bengalas) ou virtuais (aplicativos, entre outros). Os entrevistados apontaram, também, como seu maior problema relativo a locomoção o reconhecimento de objetos. Todos os entrevistados demonstraram interesse em uma aplicação neste escopo, apontando-a como solução adequada e viável para seus problemas de locomoção.

Em paralelo, o desenvolvimento *mobile* e pesquisa em aprendizado de máquina têm estado em uma crescente de desenvolvimento nos últimos anos, e o desenvolvimento de inteligência artificial (IA) tem se tornado cada vez mais comum na indústria, e sua existência é perceptível em todo tipo de plataforma, com suas aplicações variando de tarefas aparentemente simples como gerenciamento em tarefas diárias de um usuário (tendo como exemplo funções de agendamento de compromissos, realização de compras e interação máquina-usuário em geral, como opera a popular IA assistente Siri, da Apple, ou a Cortana, da Microsoft) a desenvolvimento de tecnologias de alta complexidade, como mostram as notícias atuais de popularização de carros autônomos gerenciados por IA, para definição e prosseguimento de percursos, além de leitura do caminho e do ambiente, evitando por conta própria eventuais obstáculos.

Tais tipos de aplicações estão fortemente (quando não inteiramente) ligadas a conceitos de aprendizado de máquina, a técnica de desenvolver tecnologias capazes de “aprender” através de conjuntos iniciais de informação para referencial e processos de treinamento, pelo qual iniciam o refino de suas técnicas de predição baseada em padrões, como sugere seu nome. Dentro de tal escopo, estão as redes neurais, algoritmos que cumprem tal requisito através de estruturas virtuais que simulam o funcionamento de estruturas neurais biológicas, sendo capazes de aprender e desenvolver-se através de tentativa e erro, tendo, no entanto, o diferencial de ser necessário que se estruture e programe a rede para determinada

funcionalidade, enquanto as redes neurais biológicas são generalistas, capazes de identificar quaisquer tipos de padrões que recebam, contanto que consigam identificá-los e lê-los com clareza.

Não só no público-alvo, o interesse geral por estudos e aplicações tem crescido exponencialmente nas últimas décadas, como aponta o Legassic *et al.* (2017), segundo o qual, a quantidade de artigos publicados com temas que envolvem IA se multiplicou em mais de nove vezes desde 1996, e, desde então, a frequência de fundação de novas startups que trabalham com a tecnologia cresceu em mais de nove vezes, resultando em um aumento de mais de 14 vezes desde o ano 2000. Também segundo Legassic *et al.*, o “interesse público” pela tecnologia triplicou desde 2013.

Em vista de tais fatores, definiu-se como objetivo do projeto a criação do ZU, uma aplicação para Android (OS baseado em Linux, produzido pela Open Handset Alliance, que é utilizado em grande parte dos dispositivos *mobile* na atualidade), fazendo uso do YOLO (*You Only Look Once*, um sistema de reconhecimento de objetos em tempo real, que detecta e classifica-os em imagens, separando-os em “classes”, tais como “pessoa”, “cadeira”, “escada” ou “placa”), implementado em servidor remoto, fazendo uma troca constante de dados com os terminais *mobile*, que comprimem e enviam a imagem coletada pela câmera ao servidor, para então então receber uma resposta com os objetos detectados, sobre os quais o usuário será informado, através de uma terceira funcionalidade, dito que o aplicativo conta com o sistema de sintetização de voz nativo do sistema Android para comunicação com o usuário por via auditiva. A interface simplificada do aplicativo foi também adaptada para uso daqueles com necessidade especial visual e baixa visão.

1.1. Problemática

A redução temporária ou definitiva da capacidade visual de um indivíduo pode acarretar em diversas dificuldades na vivência deste, prejudicando-o nos âmbitos pessoal, social, profissional e diversos outros, e tornando perigosas tarefas consideradas banais, como o manuseio de objetos do dia-a-dia. Tais dificuldades, normalmente, devem-se a um problema comumente relatado por membros desta comunidade: a restrição na capacidade de coleta de informação espacial, seja esta relativa à estrutura do meio em que se inserem ou objetos próximos. Isto acarreta

em uma dependência por parte do portador de necessidade especial em terceiros ou ferramentas de auxílio à locomoção. Alguns estudos apontam tal problema como importantíssimo:

> Quando nós perguntamos a um colega cego no nosso instituto sobre o problema mais urgente relacionado a navegação dos cegos, ele respondeu que os principais problemas são determinar sua própria posição, determinar a direção para a qual estão voltados ou direção do movimento, e a falta de informação sobre objetos importantes próximos e distantes no ambiente. Nesse contexto, qualquer informação sobre as características de objetos podem ser importantes (HUB *et al.*, 2004, tradução dos autores)

Ademais, as ferramentas já desenvolvidas possuem, em maioria, algum tipo de limitação funcional ou de viabilidade, ao exemplo dos cães-guia, que possuem custo de aquisição elevado, aproximando-se dos R$ 35.000,00, segundo o Instituto IRIS (2016), representando uma barreira para a maior parte do público. As ferramentas mais comuns e baratas, no entanto, costumam possuir carência em eficiência, como as bengalas, que possuem área de atuação limitada ao espaço em frente aos pés do usuário, não fornecendo informação quanto a obstáculos aéreos ou distantes. Mesmo no escopo digital, as aplicações existentes dedicadas a lidar com tais dificuldades apresentam, ainda, problemas de funcionalidades, sejam estes relacionados a popularidade e alcance da tecnologia ou pura usabilidade e acessibilidade ao público.

> A navegação em ambientes fechados é altamente desafiadora para os severamente prejudicados visualmente, particularmente em espaços visitados pela primeira vez. Muitas soluções foram propostas para lidar com esse desafio. Apesar de algumas delas terem se mostrado úteis em cenários reais, estas envolvem um importante esforço de implementação ou uso de artefatos que não são naturais para usuários cegos. (GUERRERO *et al.*, 2012, tradução dos autores)

Ademais, os investimentos em pesquisa de ferramentas inovadoras para auxílio de deficientes visuais no Brasil mostram-se insuficientes, e revelam um problema crítico, dito que, segundo o Censo IBGE (2010), há, no Brasil, cerca de 520 mil pessoas cegas e 6 milhões de pessoas com baixa visão ou visão subnormal (apud Fundação Dorina Nowill, 2018).

1.2. Objetivos

O objetivo final do projeto é a projeção e construção de uma plataforma funcional para ambiente *mobile* que seja capaz de auxiliar deficientes visuais em geral em sua locomoção em qualquer tipo de ambiente, fornecendo, por meio auditivo, informações sobre o cenário e objetos que se encontrem neste, com foco em possíveis obstáculos como escadas e móveis, sendo também capaz de identificar portas e janelas, dando informações como distância e direção com relação ao usuário. Para tal, definiram-se os seguintes objetivos específicos:

- Realizar entrevistas com o público-alvo para aquisição de informações específicas quanto aos problemas enfrentados e suas proporções, além de sugestões de melhorias e incrementações futuras.
- Realizar a construção de uma aplicação servidora de processamento, controle e transporte de dados para os terminais *mobile*.
- Construção do aplicativo em ambiente *mobile* e otimização do servidor para
- Realizar testes de usabilidade para validar a funcionalidade do sistema e otimização.

2. Fundamentação Teórica

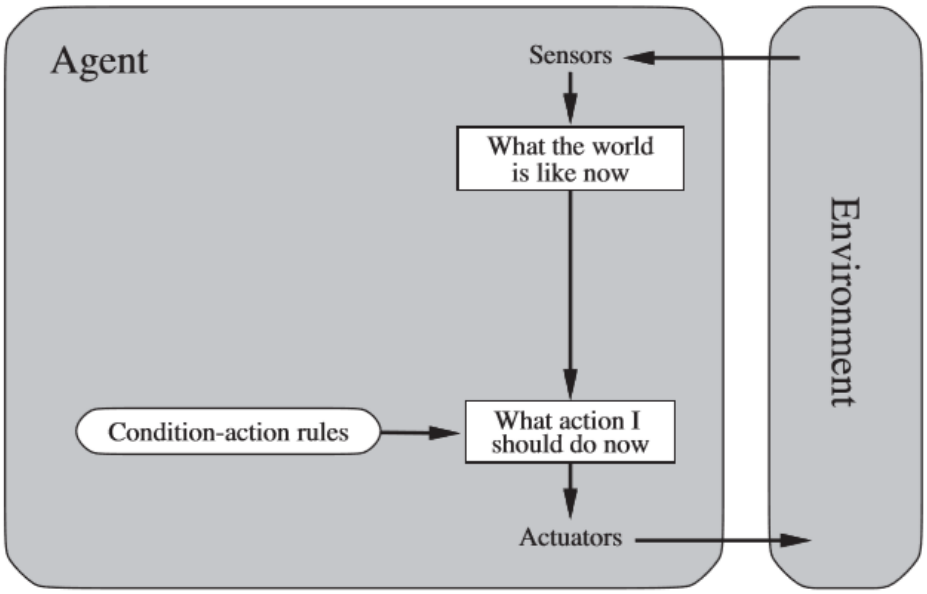
Nesta seção, serão apresentados os principais conceitos necessários para compreensão do projeto e do sistema desenvolvido. Optamos por separá-la em subseções, estas dedicadas, respectivamente, à inteligência artificial e sua definição; ao aprendizado de máquina, sua trajetória e usos; ao aprendizado profundo, seu surgimento e importância no desenvolvimento tecnológico atual; às redes neurais, suas estruturações e usos; à Darknet, *framework* no qual o YOLO foi desenvolvido; ao próprio YOLO, que é o sistema de reconhecimento de objetos utilizado no projeto; GPUs e suas funcionalidades; a GPGPU, suas práticas e possibilidades; a tecnologias como CUDA, que integram o YOLO; ao OS Android, que é o mais utilizado na atualidade; o MySQL, pelo qual gerenciamos os bancos de dados de usuários e outros tipos de informações; o Android Studio, o editor para aplicações android, no qual o aplicativo está sendo desenvolvido; o Microsoft Visual Studio 2015 e 2017, nos quais o servidor foi estruturado; ao reconhecimento de voz e à sintetização de fala, utilizados no projeto para comunicação com o usuário; ao *streaming* de vídeo, pelo qual o servidor comunica-se com o cliente; e, por fim, à tecnologia JSON, utilizada na transferência de dados.

2.1. Inteligência artificial

Segundo McCorduck (2004), em um *workshop* da Faculdade Dartmouth, no ano de 1956, o termo “inteligência artificial” (IA) viria a ser apresentado pela primeira vez, por John McCarthy. A área de IA, segundo Luger (2004), dedica-se a buscar métodos ou dispositivos que possuam ou multipliquem a capacidade do ser humano de resolver problemas.

Há ainda uma das concepções mais populares do estudo, que, segundo Russell e Norvig (2009), trata-se do desenvolvimento dos chamados “agentes inteligentes”, que seriam mecanismos capazes de perceber o ambiente à sua volta, analisá-lo e agir sobre este. Numa visão simplista, estes agentes seriam compostos por sensores (para percepção do ambiente e coleta de informações acerca dos elementos presentes neste), seus sistemas internos (para análise e categorização das informações e tomada de decisões), e “efetores” (responsáveis por realizar as ações, como, por exemplo, o que seriam os braços e pernas em um humano).

Figura 1: Diagrama esquemático de agente de “reflexo simples”



Fonte: Russel, Norvig (2009)

2.2 Aprendizado de Máquina

Na convenção de 1956, em que McCarthy cunhou o termo “inteligência artificial”, estava também Arthur Samuel, que viria a desenvolver a ideia e criar o conceito de “aprendizado de máquina” (ou ML, do inglês *machine learning*). Arthur (1959) definiu-a, em seu artigo, como um campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados, abrindo uma possibilidade para que uma máquina se tornasse melhor, em determinada tarefa, que a pessoa que a programou.

Assim como qualquer aplicação de IA, aquelas baseadas em princípios de ML seriam, então, algoritmos capazes de analisar dados e fazer previsões precisas baseadas em seus padrões, com o diferencial de serem capazes de evoluir em eficiência com o tempo, através da reunião de informações acerca de erros e acertos em previsões anteriores e do uso destas para suas novas previsões, com coleta constante de novos dados a cada tentativa.

2.3. Redes Neurais Artificiais

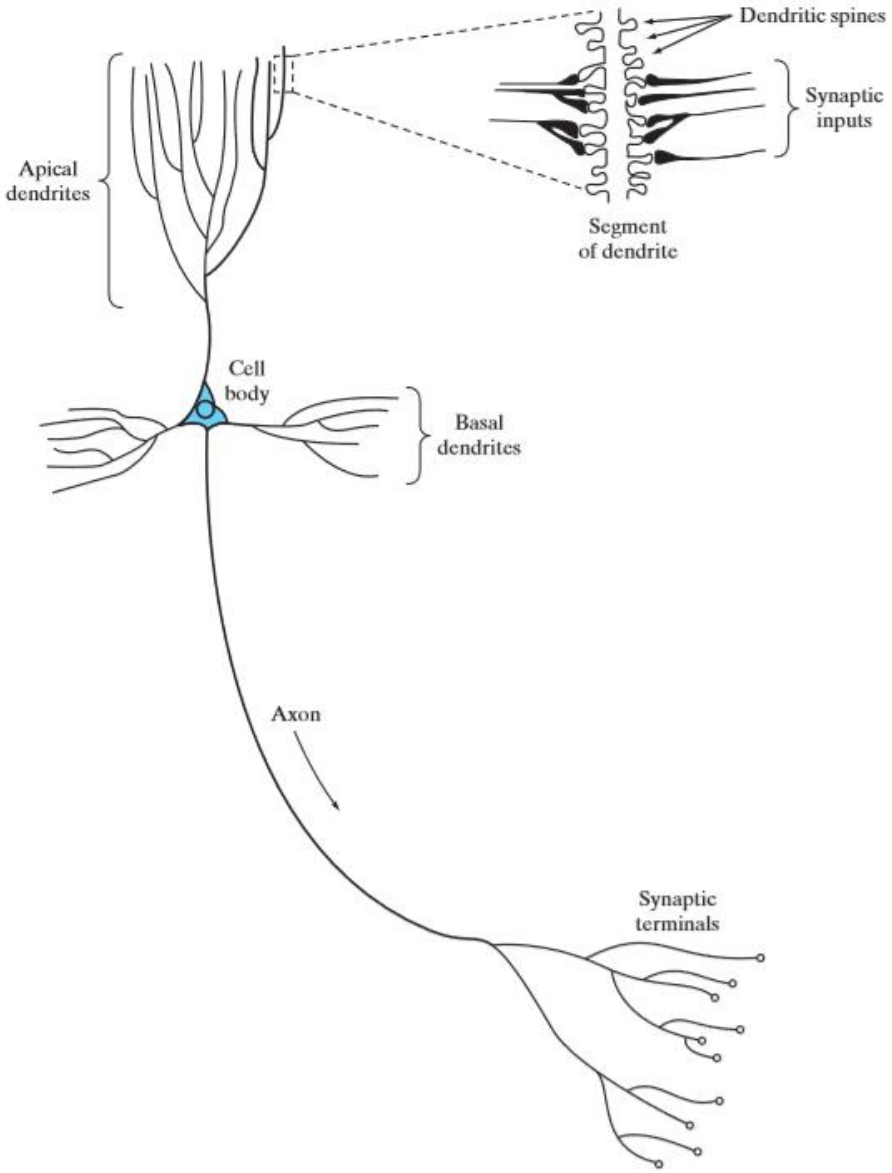
Dentro do contexto de evolução nas tecnologias de IA e desenvolvimento de novas abordagens para ML, a idéia de desenvolver “cérebros virtuais” sempre foi um objetivo dentro deste subcampo da ciência da computação, sendo amplamente visível nas artes relativas e ficção científica. Diversos estudos têm sido realizados

dentro da área de ML, para tais finalidades, nas últimas décadas, e modelos de sistemas neurais virtuais de princípio semelhante ao biológico foram desenvolvidos, sendo estes chamados de redes neurais artificiais (comumente chamados somente de “redes neurais”).

Uma rede neural artificial (ou ANN, do inglês *artificial neural network*) é uma representação digital de um sistema neural biológico, sendo os neurônios representados pelas unidades de processamento simples, que reunidas e organizadas são capazes de formar sistemas complexos, interagindo de formas diferentes entre si, com diferentes “pesos” uns sobre os outros, sendo estes relativos ao tipo de dado processado por cada neurônio, assim como seus dados de saída. Dessa forma, Haykin (2008, p. 2, tradução dos autores) define “redes neurais”:

> Uma rede neural é um processador de distribuição massiva paralela, constituído por unidades de processamento simples que tem uma propensão natural para armazenar conhecimento experiência e disponibilizá-lo para uso. Assemelha-se ao cérebro em dois aspectos:
>
> 1. O conhecimento é adquirido pela rede a partir de seu ambiente por meio de um processo de aprendizado.
> 2. As forças de conexão interna, conhecidas como pesos sinápticos, são usadas para armazenar o conhecimento adquirido

Figura 2: Representação estrutural de um neurônio



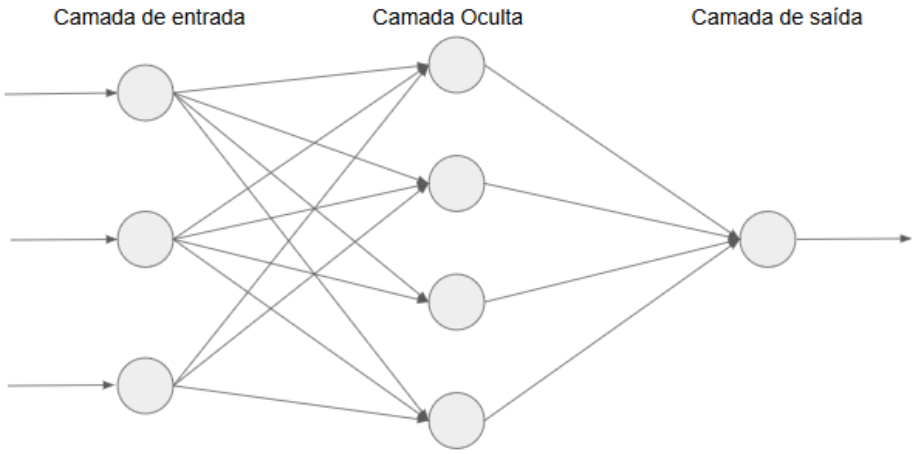
Fonte Simon Haykin (2009)

Segundo Lodish *et al.* (2003), em um sistema neural biológico, os neurônios são, de forma simplificada, compostos por quatro estruturas principais, segundo : o corpo da célula, o axônio, os terminais axônicos (ou terminais sinápticos) e os dendritos. Também conforme Lodish *et al.*, os dendritos, dispostos ao redor do corpo da célula, são responsáveis por receber os impulsos eletroquímicos enviados por outros neurônios e captados do ambiente, além da produção de algumas proteínas específicas. Lodish *et al.* afirmam também que no corpo da célula, encontram-se todas as informações e mecanismos necessários para assimilação de diversas proteínas, além de este ser uma espécie de “base de controle” do neurônio, com informações acerca do que deve ser feito com cada tipo de informação que se

recebe através dos dendritos. Também segundo Lodish *et al.*, o axônio é, no neurônio biológico, a estrutura responsável por passar tais impulsos adiante. Este se estende até o ponto em que se encontram os terminais sinápticos, que conectam-se aos dendritos de outras células por meio destes impulsos eletroquímicos.

Na representação virtual, os neurônios são similares aos biológicos em funcionalidade: possuem função de entrada e saída de dados, e realizam operações simples sobre estas, passando a informação a outros neurônios, que podem ou não ser ativados para passar os dados adiante, de acordo com o valor recebido e seu tipo. Em uma rede neural simplificada, a estrutura é de fácil entendimento, como exposto na figura 3:

Figura 3: modelo simplificado de rede neural



Fonte: elaborado pelos autores

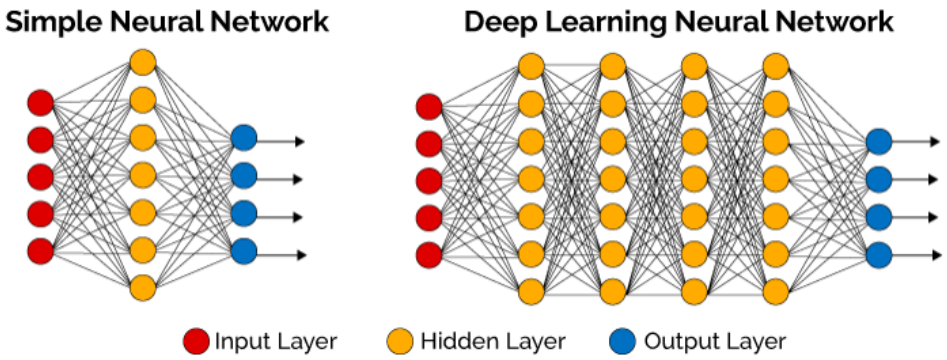
2.4. Aprendizado Profundo

Em contraste com os estudos realizados em ML e o desenvolvimento de novas tecnologias, o crescimento relativamente recente na capacidade de processamento dos computadores permitiu, nos sistemas atuais, o uso de estruturas mais e mais complexas, e tal mudança aplica-se também no escopo de IA e ML. Segundo Bengio (2013), aprendizagem profunda (ou DL, do inglês *deep learning*), é uma parte mais ampla do ML, baseada na aprendizagem de representações de dados, que pode ser supervisionada, semi-supervisionada e não supervisionada.

Conforme Deng (2014), baseia-se em uma série de algoritmos em múltiplos níveis, correspondendo a diversos níveis de abstração. Ou seja, trata-se de uma expansão dos métodos antigos para níveis de abstração e complexidade cada vez maiores, que requerem capacidades altíssimas de processamento. No contexto de

redes neurais, por exemplo, permite a estruturação de redes com múltiplas camadas internas de neurônios, fragmentando e repetindo cada vez mais o processamento de dados e identificando uma variedade cada vez maior de padrões e escopos.

Figura 4: Esquema de rede neural simples e rede neural de aprendizado profundo



Fonte: Flávio Vázquez (2017)

Desta forma, DL pode não ser considerada uma abordagem inovadora ou inventiva, mas somente o resultado do desenvolvimento tecnológico atual sobre os algoritmos já existentes. Ainda assim, a possibilitação de seu uso que as novas tecnologias tem provido abre possibilidades para diversos tipos de inovações e aprimoramento de tecnologias já existentes, ao exemplo das redes neurais, em suas aplicações práticas.

2.5. GPU

Segundo Weik (1961), CPUs são circuitos eletrônicos que cuidam das instruções de um programa, executando aritmética, lógica, inputs/outputs e outros controles específicos que possam existir nas instruções. Enquanto isso, em uma GPU, o funcionamento é diferente. Em 1999, a Nvidia foi a responsável por popularizar o termo “gpu”, através do lançamento da GeForce256, nomeada “a primeira GPU da história” (Nvidia, 2018). Apesar disso, o termo já era utilizado por volta da década de 80 (Hopgood; Hubbold; Duce, 1986).

Segundo Cheng, Grossman e McKercher (2014), GPUs possuem uma arquitetura que baseia-se em múltiplos núcleos, com especialidade em processamento paralelo de dados. Cheng, Grossman e McKercher (2014) afirmam também que a NVIDIA cunhou também o conceito *single instruction, multiple thread* (O *thread* é um fragmento do código que pode ser executado individualmente e

paralelamente) neste tipo de arquitetura, em referência à utilização, por parte da empresa, de tecnologias para manipulação simplificada dos diversos núcleos existentes em uma GPU, como visto em tecnologias como CUDA.

Assim, o diferencial entre uma CPU e uma GPU é a sua estruturação. Enquanto uma CPU lida com processos gerais de maneira extremamente rápida, com núcleos limitados em número e excelentes em potência individual, uma GPU é feita para realizar quantidades massivas de cálculos de maneira paralela, com inúmeros núcleos limitados em potência.

> Nas aplicações aceleradas por GPU, a parte sequencial da carga de trabalho é executada na CPU - que é otimizada para desempenho de encadeamento único - enquanto a parte de cálculo intensivo do aplicativo é executada em milhares de núcleos de GPU em paralelo (CUDA Zone, 2018, tradução dos autores)

2.6. GPGPU

Considerando-se a usabilidade das GPUs para processamento de quantidades massivas de informação, foi criado o conceito de GPGPU (ou *general-purpose computing on graphics processing unit*), que, segundo Ghorpade *et al.* (2012), é a utilização e manipulação dos componentes da GPU para processos gerais.

Esta tecnologia e suas aplicações permitem o uso das funcionalidades disponíveis em uma GPU para diversos tipos de aplicação, e têm melhorado e possibilitado diversas tecnologias nos últimos anos, especialmente no tocante a ML e ANNs, dito que um dos princípios de funcionamento das ANNs é processamento paralelo de dados entre seus múltiplos neurônios componentes. A partir de tais informações, é possível inferir que o uso de GPGPU é um dos mais vitais motivos para a crescente no uso de e interesse por ANNs em todas as partes da indústria e academia.

2.7. CUDA

Para propósitos de GPGPU e programação paralela, a Nvidia desenvolveu a API CUDA, que pode ser instalada em qualquer GPU que suporte a tecnologia (geralmente placas da Nvidia com seu *chipset*), segundo a própria, “uma plataforma de computação paralela e modelo de programação desenvolvida pela Nvidia para computação geral em GPUs” (CUDA Zone, 2018, tradução dos autores).

Nesta API, é fornecido um conjunto de comandos para manipulação dos núcleos da GPU e seus usos em aplicações. A plataforma foi projetada para ser versátil e de simples uso. Segundo a página oficial da Nvidia:

> Ao usar o CUDA, os desenvolvedores programam em linguagens populares como C, C++, Fortran, Python e MATLAB e expressam paralelismo através de extensões na forma de algumas palavras-chave básicas. (CUDA Zone, 2018, tradução dos autores)

2.8. Darknet

A Darknet foi desenvolvida por Joseph Redmon entre 2013 e 2016, e é “um *framework* para desenvolvimento de redes neurais, escrito em C e CUDA (...), e suporta computação em CPU e GPU” (Redmon, 2018, tradução dos autores). Também segundo Redmon (2018), esta possui código aberto e está disponível em um repositório no GitHub, sendo livre para pesquisas e modificações, desde que possua citação ao autor original. Em adição, conforme Redmon, a Darknet também hospeda a rede neural YOLO, assim como outras ferramentas, como a Nightmare e a DarkGO.

2.9. YOLO

Utilizando o *framework* Darknet, Redmon *et al.* (2016) viriam a desenvolver o YOLO, um sistema de reconhecimento de objetos de código aberto baseado na premissa de detectá-los em tempo real em fotos ou vídeos, analisando quadro a quadro e diferentes escopos para fazer a detectação do objeto, implementando-se como uma rede neural. Escrita em C++, esta rede utiliza-se das funcionalidades de CUDA disponíveis para GPGPU em placas da Nvidia.

A rede demonstra, em testes comparativos realizados, eficiência, velocidade de operação e mAP (*mean average precision*, ou “precisão média”, em tradução de autores) notavelmente superiores a outras redes (Redmon *et al.*, 2016). Esta rede foi desenvolvida também para ser capaz de “aprender” representações extremamente genéricas dos objetos (Redmon *et al.*, 2016), sendo aplicável em diversos cenários. Versões alternativas mais compactas e complexas da rede foram desenvolvidas, como o FastYOLO, que com uma estrutura mais simplista que foi capaz de elevar a velocidade de processamento da rede de 45 para 155 FPS (Redmon *et al.*, 2016).

2.10. Android

Segundo seu site oficial (2018), Android é um OS originalmente desenvolvido pela empresa de mesmo nome, e pertence atualmente à Google, baseado em uma versão modificada do Kernel Linux e outros *softwares* de código aberto, que tem como objetivo principal ser uma plataforma aberta e disponível para todos aqueles que queiram tornar suas ideias realidade, apresentando um produto que aprimore a experiência móvel de seus usuários.

Por ser um sistema de código aberto, qualquer um pode editar seu código-fonte. Essa personalização exagerada poderia ser um problema ao gerar implementações incompatíveis. Também segundo a Android (2018), para evitar isso, foi criado o Android Open Source Project, um projeto atualmente liderado pela Google, que especifica o significado e o que é necessário para ser “compatível com o Android”.

Segundo o site Androidpit (2018), O Android Open Source Project surge de uma colaboração da Google com uma série de fabricantes de smartphones para produzir um código open source que será a base dos nossos aparelhos. O projeto, também conhecido como "Android puro", é a plataforma do Google que é distribuída a fabricantes e desenvolvedores independentes, responsáveis por adaptá-lo para os seus dispositivos. É usado basicamente para descrever o Android antes das modificações feitas por terceiros.

2.11. Android Studio

Segundo o site da Android (2018), o Android Studio, lançado em maio de 2013, é a IDE (Sistema de desenvolvimento integrado) oficial para desenvolvimento de aplicativos voltados para o sistema Android. Surgiu de uma parceria entre a Android e a JetBrains, utilizando como base o Intellij IDEA, um dos IDEs de Java mais avançados disponíveis. Por ser baseado no mesmo, o Android Studio oferece todas as ferramentas de desenvolvimento dele, além de muitos outros recursos que visam aumentar a produtividade da produção de aplicativos, como emuladores, ambiente unificado, compatibilidade com C++ e NDK, etc.

Segundo o Android Developers Blog (2018), o Android Studio possui um sistema de compilação baseado no Gradle (uma ferramenta para automatizar a construção de sistemas), o que fornece flexibilidade, versões personalizadas, resolução de dependência, etc. Este novo sistema de compilação permite que você

construa seus projetos na IDE, bem como em seus servidores de integração contínua. A combinação permite gerenciar facilmente configurações complexas de compilação nativamente, em todo o seu fluxo de trabalho e em todas as suas ferramentas.

2.14. Sintetização de Fala

Segundo o dicionário MacMillan (2018), síntese de fala é o processo onde um sistema Texto-Fala (*Text-To-Speech*) produz sons parecido com a voz humana, podendo este ser um software ou um hardware especializado. Rashad *et al.* (2010, tradução dos autores) afirmam também que “um sintetizador de fala é um sistema virtual capaz de ler qualquer texto em voz alta”.

Também segundo Rashad *et al*. (2010), a sintetização de fala ocorre em duas etapas, sendo a primeira a análise de texto e a segunda a formação de ondas sonoras. Na primeira, seria fornecida uma informação em formato de texto, para ser analisada e transformar-se em uma representação fonética, e na segunda, seria fornecida a saída auditiva baseada nos resultados das análises fonética e prosódica.

3. Metodologia

O projeto foi desenvolvido através de cinco fases: (i)pesquisa (bibliográfica, laboratorial e de campo), (ii)estruturação (definição da estrutura inicial do aplicativo desenvolvido), (iii)construção (implementação do modelo planejado), (iv)testes (confirmação de funcionalidade e coleta de dados diversos) e (v)análise de resultados (listagem dos resultados obtidos na fase anterior).

3.1. Pesquisa

Na fase de pesquisa, foram realizadas pesquisas laboratoriais (busca, coleta e testes de aplicações e tecnologias semelhantes ou úteis ao projeto, com classificação dos pontos positivos e negativos de cada uma), bibliográficas (para coleta de informação a respeito de novas técnicas e tecnologias relativas à área), e de campo (que ocorreu através da realização de entrevistas com deficientes visuais do campus Natal/Zona Norte, visando maior clareza acerca do produto desejado e de que tecnologias seriam apropriadas, além de informações sobre o contato com tecnologias semelhantes que tiveram anteriormente)

3.2. Estruturação

Posteriormente, na fase de estruturação, as informações coletadas na pesquisa foram analisadas para a definição de um objetivo final claro e a estruturação de um protótipo, além disso tecnologias mais apropriadas à finalidade definida foram selecionadas. Foi decidido então dividir a aplicação em servidor e clientes, o servidor ficou responsável por processar os dados e os clientes por iniciar a conexão e enviar e receber os dados do servidor, além da comunicação com o usuário.

3.3. Construção

Então a fase de construção do protótipo foi iniciada, com implementação da estrutura idealizada e utilização das ferramentas e sistemas selecionados. O YOLO foi utilizado na construção do servidor, sem as informações de suas classes nativas, sendo treinado para reconhecer nove objetos iniciais: placas, placas de banheiros, placas de saída, símbolos masculinos, símbolos femininos, símbolos de

acessibilidade, a palavra “masculino”, a palavra “feminino”, a palavra “acessível” e seus equivalentes, como “PCD” (pessoas com deficiência). Para a rede ser treinada, foram coletadas aproximadamente 1300 fotos ambientadas no IFRN - Campus Natal Zona Norte, cujos objetos foram manualmente selecionados e classificados para serem enviados ao sistema e integrados a sua coleção inicial de dados.

3.4. Testes

Após o treinamento inicial para o reconhecimento de objetos em tais classes, seguiu a fase de testes, na qual membros do grupo testaram continuamente o aplicativo em diversos ambientes e coletaram informações acerca da eficácia do sistema de detecção, além de possíveis erros, ocorrendo no ambiente do IFRN - Natal/Zona Norte. Foram testados também fatores como possibilidades de localização dos usuários dentro do IFRN com base na intensidade do sinal em diferentes roteadores.

3.5. Análise e refinamento

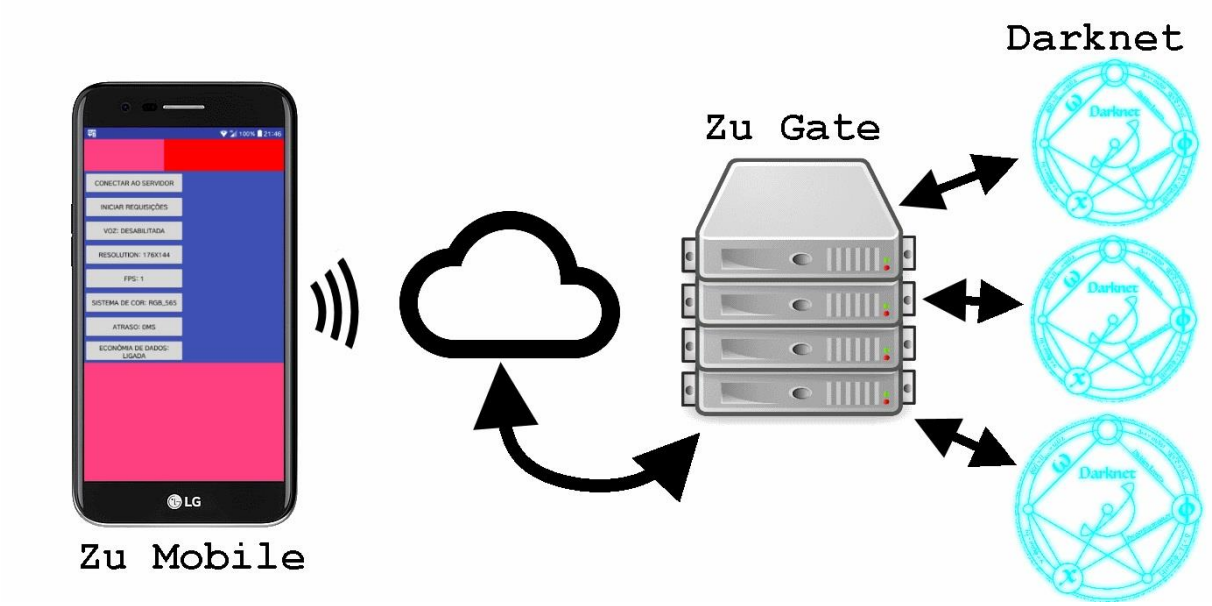
Com a conclusão da fase de testes, foi iniciada a primeira fase de análise de dados e aprimoramento do protótipo, na qual foi construída a segunda versão deste, com correção dos erros identificados na primeira e algumas melhorias no que se refere a velocidade e precisão, com a troca do arquivo de pesos criado no treinamento da etapa anterior por um já pronto, vindo do PASCAL VOC Challenge (um desafio anual, que disponibiliza pacotes de imagem a serem utilizadas para treinamento de classificadores de objetos, objetivando comparações de eficiência), além de melhorias na interface do usuário, com foco em aumento na acessibilidade. Foram realizadas análises de precisão, velocidade e conectividade.

3.6. Análise Final

Após, a segunda instância da fase de análise dos dados, na qual foi feita a compilação dos resultados gerais, e nesta fase final, realizou-se a sintetização das informações coletadas nos testes e pesquisas para a construção de um documento-guia contendo todos os dados e métodos utilizados no desenvolvimento do projeto, com informações acerca da construção dos protótipos, as técnicas utilizadas, a arquitetura e as modificações feitas.

4. Solução Proposta

Figura 5: O diagrama físico da aplicação



Fonte: elaborado pelos autores

A solução proposta consiste em permitir ao usuário a inscrição de imagens para a análise da darknet de maneira fácil e simples através da aplicação *mobile*, este transmite os dados pela wifi para o servidor do Zu e este seleciona a darknet para responder a requisição.

4.1. Zu Mobile

Como o objetivo principal é a realização de uma aplicação que possa ser levada com o usuário final durante o seu deslocamento no dia a dia, foi idealizada a ferramenta Zu Mobile, tendo a função de fazer a interface do sistema com o usuário e servir como ponto de entrada dos dados de imagem necessários para a interpretação. A versão Android na qual o protótipo foi desenvolvido é a 5.0 (API 21), que, segundo a seção de estatísticas do site oficial da Android (2018), oferece suporte a 88% dos aparelhos que utilizam o sistema operacional Android. Dada a necessidade especial do público alvo que foi proposto, a interação com o usuário vai ser dada através de respostas de áudio utilizando-se da biblioteca de sintetização de áudio que acompanha os smartphones modernos, apesar de ser viável a reprodução da voz sintetizada como transmissor de informações para o usuário o caminho reverso se tornaria mais complexo ao se tentar o reconhecimento de voz, que dependendo do local da aplicação pode render resultados não desejados como a má

compreensão do que se está sendo falado, logo se preferiu a criação de uma interface de botões simples para a utilização com as ferramentas de acessibilidade do próprio sistema como o *Talkback*, o qual realiza a leitura de pontos específicos da tela selecionados pelo usuário.

4.1.1. Estrutura do aplicativo

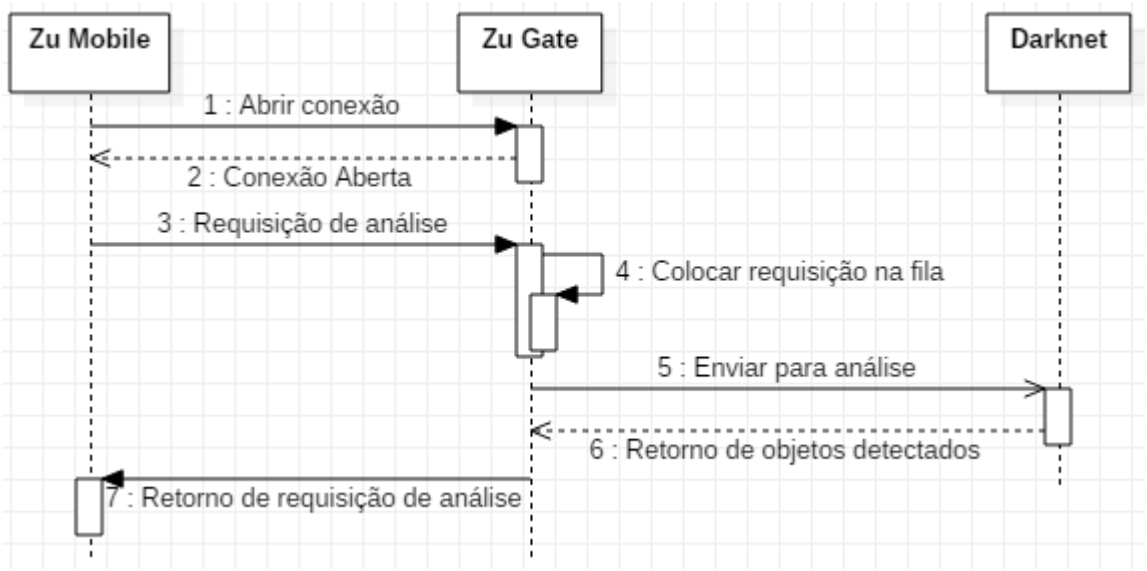
Na estruturação do código temos quatro classes principais, a primeira de todas é a que irá controlar o comportamento do aplicativo cuidando das requisições feitas pelo usuário através da interface de botões e tomará ações para com os demais componentes do sistema de acordo com as regras de negócio, para o controle do *TextToSpeech* e caso necessário da fila de requisições de fala requisitadas, como a utilização das funções de transmissão de dados para a rede não pode ser feita no *thread* principal da aplicação devido a este ser responsável pela a atualização da interface gráfica do aplicativo e a resposta das mensagens, logo uma classe que encapsula o seu próprio *thread* e que cuide das operações de escrita e leitura do soquete (soquete de rede é o nome que se dá ao ponto terminal na conexão entre dois processos que objetivam trocar algum tipo de informação, estando estes ligados por uma rede de computadores), por último para interagir de maneira mais fácil com a API de câmera do sistema no qual o aplicativo foi instalado. A sequência de processamento é simples ao iniciar a aplicação o usuário decide quando começará a requisitar análises ao servidor, registrando uma *callback* (função que atua como um “gatilho” para determinados tipos de eventos. Estas funções servem como ativadores de algoritmos em um sistema, sendo estes necessários quando o evento vinculado à *callback* ocorrer) no sistema para receber os frames capturados pelo sensor de imagem do dispositivo, cada quadro vem codificado de uma maneira de acordo com as configurações iniciais da requisição e das capacidades do *hardware*, após isso o dados da imagem são colocados em espera até serem transmitidos para o servidor.

4.1.2. Fluxo de dados

A figura 6 mostra o fluxo completo da informação no sistema, partindo da ação do usuário de iniciar a conexão com o servidor, durante todo o tempo de vida do ZuGate ele estará apto a receber novos clientes, na ação 3 da representada na mesma figura demonstra o início do processo de análise da imagem, onde o cliente

requisita ao Gate para a adição dela na fila de espera, demonstrada pela ação 4, tendo a disponibilidade de um processador a primeira requisição da lista é processada e futuramente será retornará para o *Gate* que irá apenas retransmitir para o usuário correto.

Figura 6: Fluxo de mensagens entre as aplicações que integram a aplicação



Fonte: elaborado pelos autores

O fluxo de dados no aplicativo segue o esquema descrito na figura 6, a cadeia de eventos tem início com a ação do cliente de iniciar a sua conexão com o servidor, descrita pelos pontos 1 e 2, é requisitado ao sistema Android um soquete livre, caso a operação seja realizada com sucesso o servidor ZuGate dá a resposta de que a conexão TCP está aberta, a comunicação é por meio de pacotes TCP, que é um protocolo orientado a conexão e assegura a entrega dos pacotes em ordem de transmissão, a decisão da utilização desse protocolo é devido a necessidade de oferecer uma conexão segura mesmo que esta seja estabelecida através de redes móveis como 3G ou 4G, por fim o servidor se põe a espera de requisições do cliente para interpretar. Caso o tempo de inatividade, tempo desde a última requisição ou da resposta do servidor, chegue a certo limite o servidor tentará notificar o cliente da sua ação e encerrará a conexão imediatamente para liberar recursos para mais clientes.

O próximo processo no fluxo dos dados é a requisição de uma análise, como mostra a ação 3, onde será enviado um pacote de dados contendo um cabeçalho com as informações pertinentes ao cliente que serão preenchidas pelo servidor, a resolução da imagem, sua codificação de *pixels* e um identificador único, já o corpo

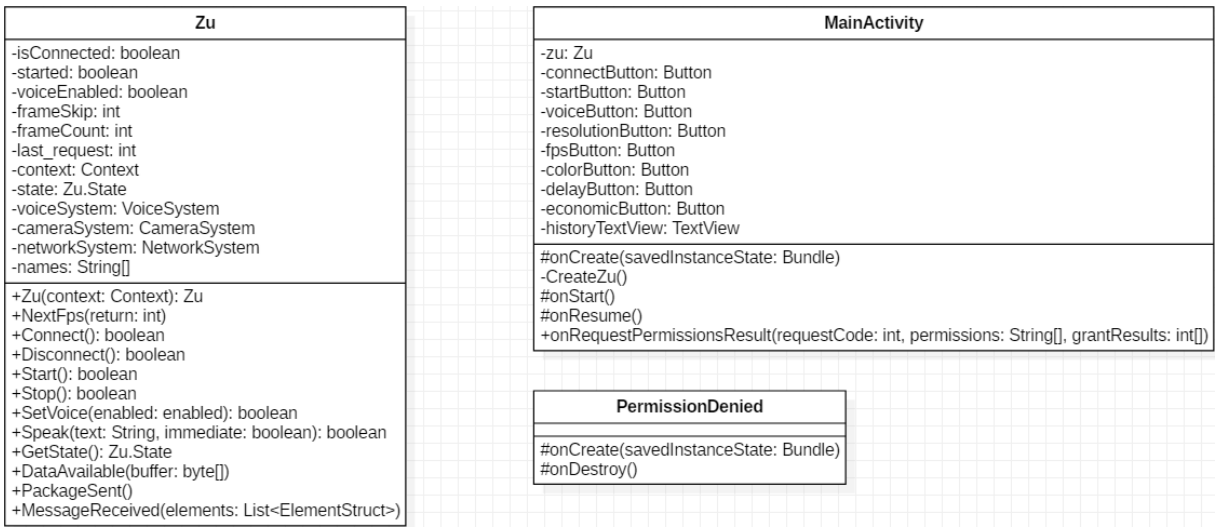
contém os dados da captura. Quando a transferência for concluída é iniciado um subprocesso no servidor para encaixá-la em uma posição na fila de análises, representado na figura pelo número 4, esse procedimento visa manter uma distribuição dos recursos para todos os clientes conectados.

O servidor checa o estado de todas as instâncias da Darknet conectadas para verificar uma disponibilidade, após isso ele envia a requisição que estiver no início da fila para o processamento, realizada a análise as classes detectadas são colocadas em um pacote de resposta que informa a sua posição relativa na imagem, esse pacote segue para o servidor onde é reencaminhado para o cliente, e isso finaliza o fluxo na sequência de ações 5 a 7. De acordo com a configuração o cliente pode escolher quais objetos são mais importantes no resultado e escolher comunicá-los ao usuário.

4.1.3. Implementação

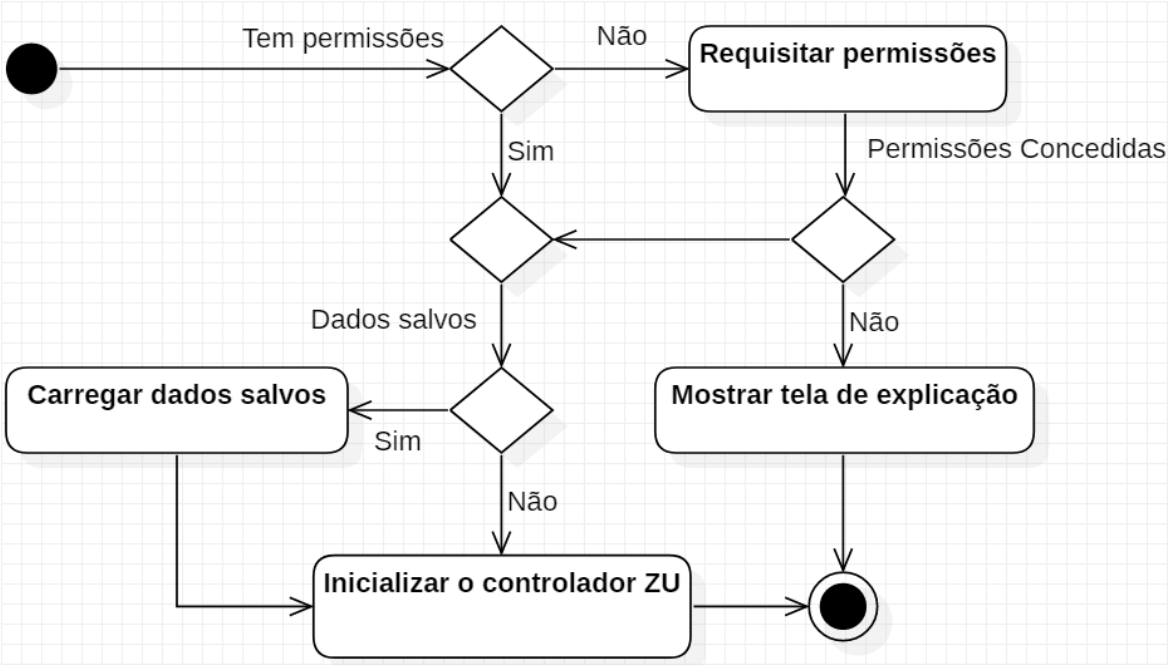
Toda atividade, que é basicamente como é chamada uma interface na aplicação Android, a qual é carregada pela classe *MainActivity* que funciona como ponto de entrada do programa, pelo ciclo de vida de atividades nos sistemas Android a primeira função a ser chamada é a *onCreate*, cuja recebe os estados salvos de execuções anteriores, logo em seguida verifica se o usuário já deu as permissões necessárias como o acesso a câmera do dispositivo e o acesso à rede wifi ou de dados móveis.

Figura 7: Representação das classes “Zu”, “MainActivity” e “PermissionDenied” do aplicativo android



Fonte: elaborado pelos autores

Figura 8: Fluxo pré inicialização do controlador Zu no aplicativo *mobile*

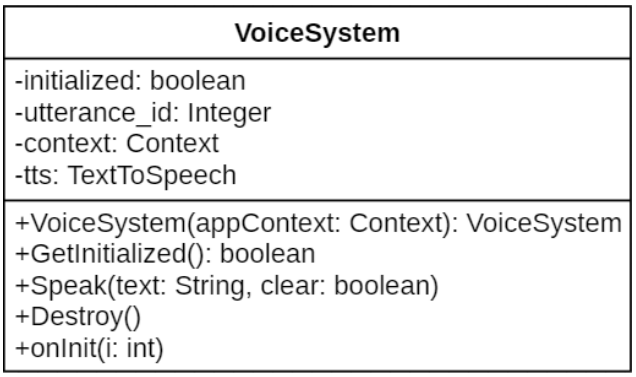


Fonte: elaborado pelos autores

O diagrama acima mostra o fluxo da atividade antes de carregar o controlador Zu, que ao comando do usuário irá estabelecer a conexão com o servidor adequado e iniciar requisições de interpretação ao mesmo. Caso as permissões não tenham sido concedidas logo na entrada do cenário acima, a rotina exibirá ao usuário a tela de diálogo padrão do sistema para requisição de permissões, por fim caso novamente o usuário recuse uma tela explicando a necessidade será exibida.

Por se tratar de um sistema muito dinâmico as tarefas que o sistema executa, como por exemplo a execução de um app, podem ser encerradas subitamente, o que, ocasiona a reinicialização da mesma quando o usuário retorne a ela, caso isso aconteça a aplicação vai tentar retornar ao estado anterior tomando como base os dados que a mesma havia guardado, desta forma o usuário tem acesso a uma utilização mais fluida já que não precisará repetir os passos que já perfez.

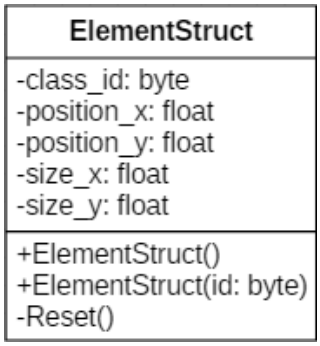
Figura 9: Representação da classe *VoiceSystem* no aplicativo *mobile*



Fonte: elaborado pelos autores

A figura 9 traz a classe responsável pela interação com os sistemas de sintetização de áudio disponíveis, já que existe a possibilidade de mais de um motor de sintetização estar presente no celular. Toda fala requisitada é colocada em uma fila de espera, até que a reprodução anterior cesse ou até que uma requisição emergencial apresente-se. Como a execução das requisições ainda permanece ativa mesmo quando o usuário deixa a aplicação, as leituras sobre os objetos detectados ainda são transmitidas através do áudio.

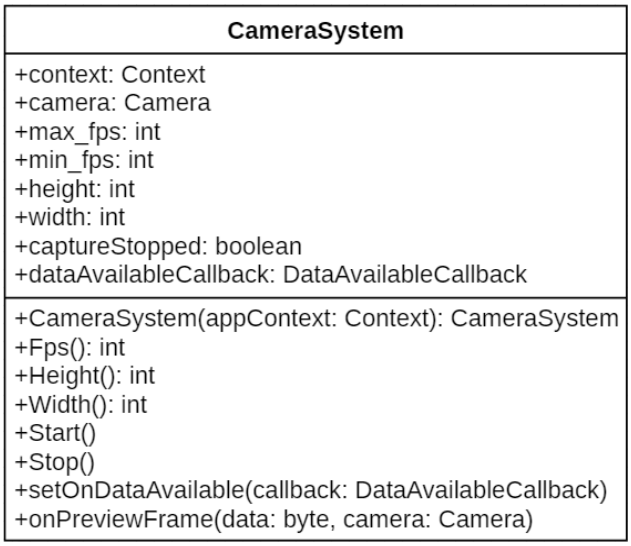
Figura 10: Estrutura utiliza para guardar informações sobre objetos detectados



Fonte: elaborado pelos autores

Toda leitura ou conexão ao servidor é feita em um *thread* separado, devido a necessidade da responsividade do aplicativo, sendo assim se torna primordial a existência de *buffers*, demonstrado na figura 10, para armazenar as informações sobre as requisições processadas até que o *thread* principal consiga lê-la e de acordo com a customização do usuário informá-lo.

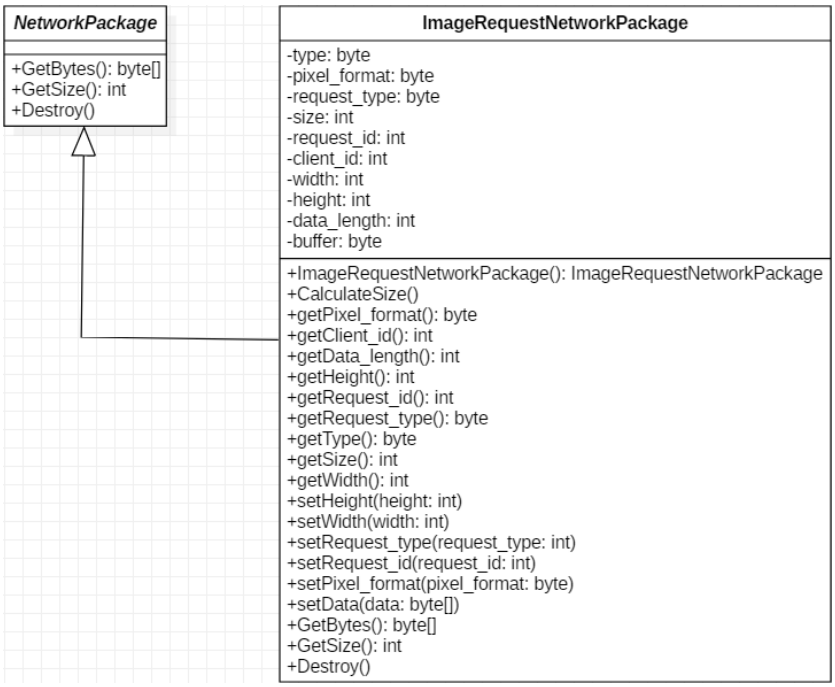
Figura 11: Classe *CameraSystem* no aplicativo *mobile*



Fonte: elaborado pelos autores

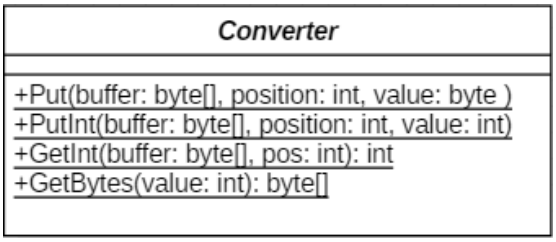
O sistema de câmera foi criado apenas para ser um adaptador, tornando a utilização do sistema com várias implementações da API gerenciadora de câmeras dos sistemas, para toda imagem capturada é gerado um *buffer* do quadro que então é passado para o controlador Zu em conjunto com as informações necessárias para a interpretação correta dos dados, como a quantidade de linhas e colunas da imagem, a disposição das cores de cada *pixel* e como o mesmo é representado.

Figura 12:Classe abstrata NetworkPackage e uma classe especializada ImageRequestNetworkPackage



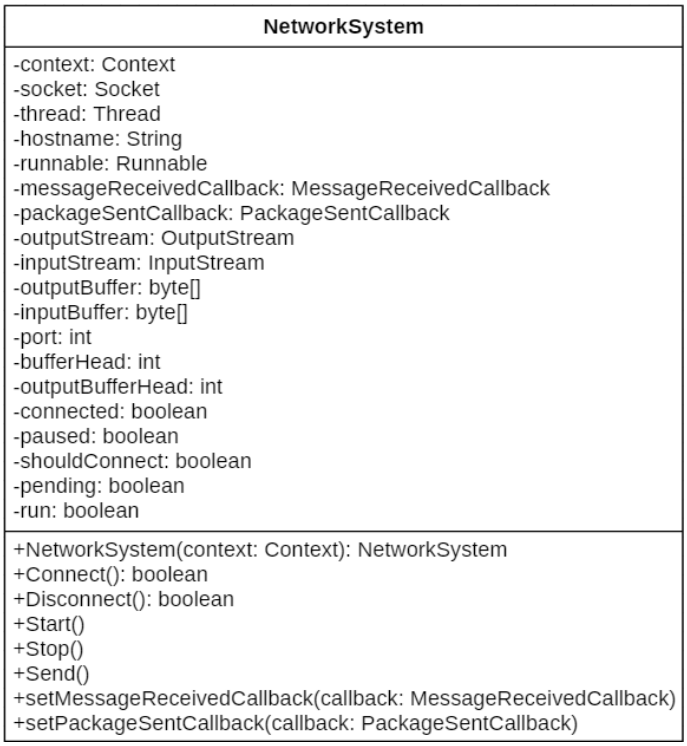
Fonte: elaborado pelos autores

Figura 13: Classe abstrata utilizada para facilitar a conversão de inteiros para bytes



Fonte: elaborado pelos autores

Figura 14: Classe responsável pelo controle de transmissão de dados no aplicativo *mobile*



Fonte: elaborado pelos autores

As figuras 12, 13 e 14 compõem o sistema de *network*, o pacote de internet é uma abstração para permitir a leitura do mesmo pelo *NetworkSystem* para posterior transmissão, a classe responsável por construir uma requisição de análise de imagem é a *ImageRequestNetworkPackage* que é uma especialização da anterior, as variáveis e métodos que esta possui visam facilitar tal tarefa. A classe abstrata *Convert*, figura 13, implementa métodos estáticos para a manipulação de vetores de *bytes*, para permitir uma posterior manipulação do *buffer*.

O sistema de *Network* se encarrega de manter a conexão com o servidor aberta, receber eventuais pacotes de comunicação e a resposta para cada pedido de análise feito.

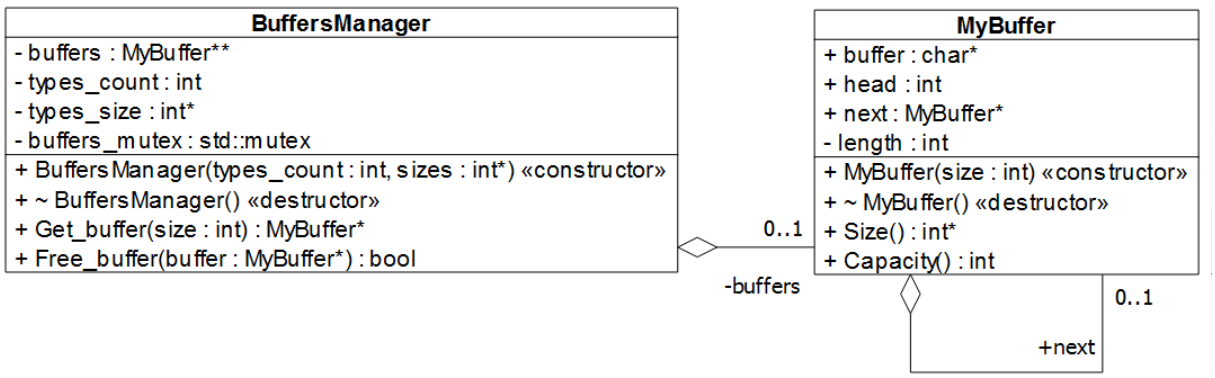
4.2. ZU Gate

Para facilitar a conexão do cliente e distribuição igualitária do trabalho gerado pelas requisições do cliente, se fez necessário a criação de uma aplicação que redireciona e ordena os atendimentos de acordo com parâmetros como tempo desde última requisição atendida. Como o processo da análise da imagem é consideravelmente pesado do ponto de vista computacional, o ZuGate também foi

pensado para ser um distribuidor de carga, possibilitando que mais de um computador possa realizar a interpretação da imagem.

4.2.2. Estrutura do código

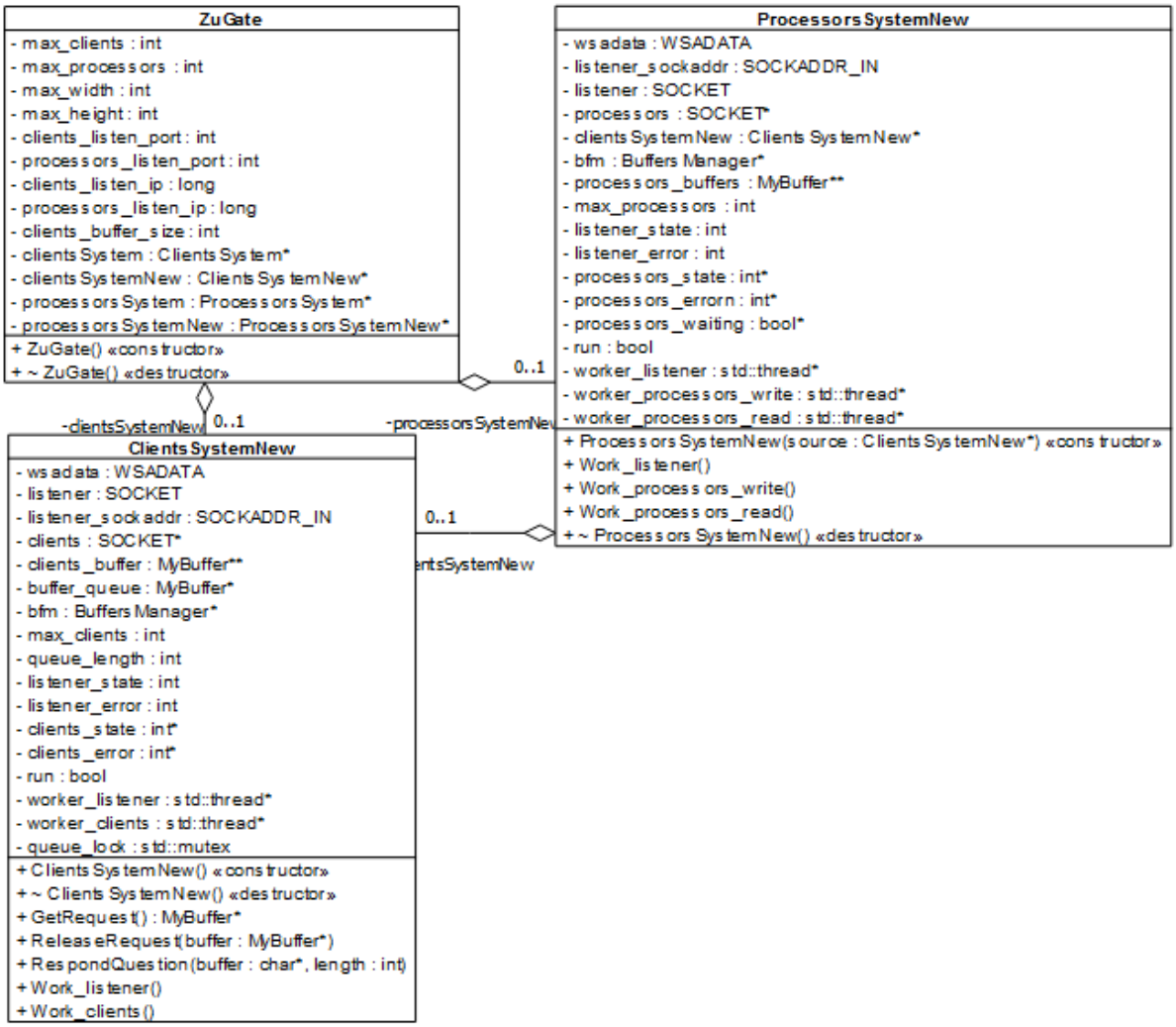
Figura 15: Classes de armazenamento dados



Fonte: elaborado pelos autores

Foram criadas duas classes para realizarem o encadeamento e armazenamento das imagens, na figura 15 temos a estrutura da classe *BuffersManager* que agregará os objetos de *buffer* criados, gerando assim um ciclo de reutilização de objetos já instanciados. Devido aos múltiplos contextos dos *threads* do aplicativo a inserção e retirada de elementos da fila de *buffers*, é necessário a utilização de um função de sequenciamento para garantir que as operações não tenham execuções imprevisíveis, dado que mais de um *thread* pode tentar acessar-las ao mesmo tempo causando uma condição de corrida.

Figura 16: controlador Zu e as classes de sistema responsáveis pela conexão com os clientes e processadores



Fonte: elaborado pelos autores

Na figura 16 temos a classe principal do programa, nela são definidas as configurações padrões e número máximo de clientes que podem ser atendidos. Além disso ela inicializa as classes que se conectam com os clientes e os processadores.

Para a utilização dos soquetes na plataforma Windows precisamos inicializar a API WinSock2 e guardar os dados retornados em uma estrutura chamada WSADATA em cada sistema, após isso se sucede a operação de escuta em uma porta local na espera do contato de uma requisição de conexão. No sistema de clientes as requisições são recebidas para posteriormente serem consumidas pelo sistema de processadores, através do método *GetRequest* chamado pelo sistema de processadores através da referência da instância de *ClientsSystem* passada pelo controlador, retirando a requisição da fila de processamento e enviando-a para um

processador que não esteja em procedimento de análise de imagens. Dessa forma o serviço tem apenas como limitação, para quantos clientes se pode atender, a quantidade e capacidade dos processadores conectados, o que viabiliza um escalonamento vertical dos recursos.

4.3. Darknet

Considerando que o YOLO hospeda-se na Darknet (além de possuir entrada e saída de dados gerenciadas pelo *framework*), foi necessária a instalação da rede, no âmbito do servidor, para que fosse possível a comunicação com esta através do ZuGate.

4.3.2. Implementação

No protótipo, a Darknet foi instalada em um computador, para ser utilizada no lado do servidor do sistema. É necessário apontar que a versão utilizada da Darknet foi compilada para um OS diferente (Windows) ao do autor original (Linux), e pode ser encontrada em seu repositório (Yolo-for-windows-v2, 2018).

Foram realizadas pouquíssimas alterações na rede, sendo estas todas no arquivo “demo.cpp”. O arquivo foi editado para que o sistema retirasse os quadros de imagem recebidos do ZuGate.

A rede seria, na aplicação, responsável por fazer a interpretação da imagem recebida e conversão dos valores de seus *pixels* em *floats*, para que estes pudessem ser enviados ao YOLO e a análise para reconhecimento de objetos pudesse ocorrer. A rede foi modificada para ser capaz de ordenar *pixels* nos formatos necessários, sendo capaz de tratar os formatos RGB_565, RGB_888, entre outros.

4.4. YOLO

Na implementação da rede, esta precisa ser instalada com a Darknet e receber um arquivo de pesos de um treinamento antigo ou passar por um novo treinamento. Este arquivo contém os dados de imagens já analisadas, com informações acerca dos objetos, e é usado como uma “base de dados” para a rede. Uma vez criado e concluído o processo de treino, um arquivo de pesos não deve ser editado.

O YOLO foi instalado com as classes do desafio VOC, que foi disponibilizado por seu autor. Este arquivo contém as seguintes classes: avião, bicicleta, pássaro, barco, garrafa, ônibus, carro, gato, cadeira, vaca, mesa de jantar, cachorro, cavalo, motocicleta, pessoa, vaso de plantas, ovelha, sofá, trem, monitor.

Na implementação do Zu, a rede do YOLO não foi utilizada em unidade: o ZuGate está controlando diversas instâncias da rede, sendo uma geral e outras especializadas em determinados objetos, ou com pacotes específicos.

5. Resultados e Discussão

Com base na metodologia e na solução proposta foram gerados os seguintes resultados.

5.1. Análise dos resultados da pesquisa de campo

Foi desenvolvido um questionário, que foi aplicado em um total de 5 entrevistados, no qual deficientes visuais responderam algumas perguntas acerca das suas dificuldades de locomoção no seu dia-a-dia. O questionário teve como objetivo identificar os principais obstáculos enfrentados por eles, de forma a descobrir por onde começar na estruturação do protótipo.

Com as respostas destes, foi possível identificar uma dificuldade unânime quando nos referimos a obstáculos aéreos, como por exemplo árvores e orelhões, tendo os usuários atentado especialmente para a importância da identificação de placas e postes. Também foi observado que os entrevistados têm problemas com desníveis em seu caminho, como por exemplo batentes e degraus.

Uma vez constatadas essas dificuldades, foi perguntado aos usuários se eles já tiveram contato com aplicativos e sistemas que tenham por objetivo solucionar esse tipo de problema, e apenas dois deles tiveram experiência com tais tipos de tecnologias. Porém, ambos os usuários apresentaram reclamações relacionados, seja ao alto custo desta ferramenta, seja a problemas desempenho. Foi perguntado também a opinião dos usuários sobre o desenvolvimento desse sistema, e recebemos uma resposta unânime, onde todos o consideraram uma ferramenta de fundamental importância e que todos a utilizariam esta caso disponibilizada.

5.2. Aplicativo Móvel

O aplicativo foi desenvolvido utilizando a IDE Android Studio, a versão alvo do sistema Android utilizada foi a API 21, utilizando a linguagem de programação Java, nativa do Android, e também foi utilizado o XML como linguagem de marcação, que é o padrão utilizado para a geração de interfaces gráficas. O aplicativo permite o usuário a enviar imagens capturadas de maneira automática pela câmera para o servidor onde serão interpretadas.

Figura 17: tela única da aplicação móvel

Fonte: elaborado pelos autores

Ao entrar no aplicativo o usuário irá se deparar com uma tela que contém apenas botões, o visual simples da tela visa facilitar o uso de ferramentas de acessibilidade como o *Talkback*, uma vez que os elementos da tela são fáceis de se ler e interagir, ou permitir um escalonamento do tamanho dos elementos gráficos caso o usuário tenha baixa visão.

Para utilizar o aplicativo o usuário irá primeira clicar no botão “Conectar ao servidor”, após isso o aplicativo avisará-lo ao conseguir realizar a conexão e em seguida este pode alterar as configurações utilizando os outros botões, como por exemplo a taxa de quadros que deve ser utilizada para enviar requisições, quando estiver satisfeito ele pode iniciar as requisições com o botão “Iniciar requisições”, que iniciará a transmissão das imagens a uma taxa de 5 quadros por segundo; sendo assim, o tempo para o reconhecimento da imagem aproximadamente 200 milisegundos. Cada elemento que for detectado pelo servidor terá seu nome reproduzido para que o usuário consiga saber o que está na sua frente.

5.3. Zu gate

A partir dos diagramas explanados na solução proposta o ZuGate que irá fazer a ligação entre os clientes e os processadores da imagem, ele foi programado

na em C++ utilizando a IDE Visual Studio 2017, o programa aguarda as conexões dos clientes e guarda suas requisições na fila de processamento. Ao haver um processador que não esteja já interpretando uma imagem, uma requisição é retirada da fila e enviada ao mesmo, que após sua análise retornará ao servidor para ser redirecionada ao cliente.

5.4. Testes

Depois da prototipagem do servidor e do cliente a sistema como um todo foi testado foram obtidos parâmetros-base para o funcionamento do aplicativo, como taxa de frames, resolução de cada frame, latência, e verificação da capacidade de processamento. Todos os testes foram executados em um computador com um placa de vídeo GTX 1050 TI e um processador AMD FX-4300 4,1 GHz e com os dados sendo transmitidos por um access point a cerca de 10 metros do *smartphone*, um LG K10 Octacore 1,1 GHz . As classes de objetos do yolov2 testadas foram: pessoa, carro, teclado, mouse, copo, colher, televisão, bicicleta, mesa, cama, telefone e cadeira.

O primeiro parâmetro a ser definido foi a taxa de frames que deveria ser utilizada, e a placa utilizada conseguiu processar um vídeo local na resolução nativa de input (414x414) a cerca de 25 quadros por segundo. O desempenho foi testado no intervalo de 3 quadros por segundo até 20 quadros por segundo, devido a limitações do aplicativo cliente. O principal dado resultante deste teste é que 5 quadros por segundo seriam suficientes para que a funcionalidade de descrição dos objetos que surgissem diante a câmera avisasse o usuário a tempo de reagir no caso de um obstáculo.

O teste de resolução foi baseado em uma comparação da taxa de detecção na resolução máxima de 1280x720, que, devido ao escalonamento da primeira camada, será redimensionada para uma imagem 414x414 perdendo boa parte do detalhe, porém garantindo um resultado de controle. A menor taxa de detecção foi com a resolução de 176x144 de cerca de 70% em relação ao teste base, provavelmente devido a pouca quantidade de detalhes passados e que são necessários para objetos menores como um copo ou colher, as resoluções acima de 414x414 como, por exemplo, 640x480, não tiveram reduções significativas na precisão da detecção de objetos. Sendo assim, a resolução pode ser dada como um parâmetro personalizável pelo usuário final de acordo com sua vontade de

economizar a utilização de dados, pois um quadro na resolução de 176x144, utilizando o padrão RGB_565 utiliza, no máximo, 49.5 KB, um valor 12 vezes menor do que o tamanho máximo da resolução de 640x480, cerca de 600 KB.

Foi realizada uma simulação de latência (o tempo que a informação levará para chegar ao cliente, a partir do momento em que é requisitada) no servidor, e o sistema foi configurado para que os clientes registrassem a imagem como “recebida” contabilizando este atraso padrão. Dessa forma, toda latência que esteja abaixo de 400 milissegundos da transmissão até o recebimento da resposta gera uma experiência semelhante a um menor atraso como, por exemplo, 100 milissegundos. Isso é devido a interpretação das imagens ser constante, logo, não se percebe tanta diferença entre o que se está na frente da câmera e o que o aplicativo está descrevendo; porém, no caso de objetos que aparecem de repente, a experiência pode ser prejudicada pelo intervalo de tempo das respostas.

Como já foi exposto anteriormente, o processamento de imagens teve um limite no computador que foi testado de aproximadamente 25 quadros por segundo. Este fator, juntamente com a constatação que 5 quadros dariam uma experiência satisfatória possibilitou a ideia de criação de uma rotina de processamento de dados enviados pelos clientes, permitindo o atendimento “em tempo real” de mais de um cliente. Ainda estão sendo realizados testes sobre tal possibilidade e o desempenho da aplicação em tal cenário.

O segundo teste do aplicativo foi realizado através da internet, um host remoto atuou como servidor para um cliente dentro da rede do campus IFRN campus Natal Zona Norte, utilizamos diversas resoluções, a captura de quadros do aplicativo foi feita a uma taxa de 15 quadros por segundo, porém de acordo com a qualidade do sinal de *wifi* e do tamanho do pacote a ser transmitido, que escala proporcionalmente com a resolução e qualidade da imagem escolhida, o quantitativo de performance estabelecido foi a taxa de quadros real que o processador estava realizando.

Tabela 1: relação do tamanho da imagem com a taxa de quadros.

Resolução	Tamanho (KB)	Fps (q/s)
176x144	50,68	9,8
320x240	153,6	9,2
480x360	345,6	8,7
640x480	614,4	8,2
1280x720	1843,2	7,7

Fonte: elaborado pelos autores

Seguindo a tabela 1 o desempenho da aplicação começa a se degradar de acordo com a resolução de captura, devido a diversos fatores, sendo o mais deteriorante o tamanho de cada quadro, já que no protótipo não foi implementado nenhum mecanismo de compactação de dados.

6. Considerações Finais

Este projeto apresentou uma proposta auxiliar para pessoas com necessidade especial visual em sua locomoção. Objetivando isto, foi realizado um estudo de caso sobre as dificuldades enfrentadas por esta comunidade, bem como as tecnologias e aplicações em desenvolvimento e já desenvolvidas que visam solucioná-las. Foi então proposto e desenvolvido o sistema chamado Zu, que consiste em um aplicativo móvel que transmite imagens captadas pela câmera do *smartphone* para um servidor no qual roda um detector de objetos, que envia de volta a informação dos obstáculos reconhecidos em tempo real, para que estas sejam então passadas ao usuário por uma interface de áudio. Espera-se, com o projeto, criar uma ferramenta acessível e eficiente para auxílio de deficientes visuais em sua locomoção.

6.1. Principais Contribuições

- Detecção de obstáculos: envio de dados da câmera do *smartphone* para o servidor; detecção dos obstáculos na imagem em tempo real utilizando o YOLO; envio dessas informações para o aparelho do usuário em formato de texto e leitura feita através do sintetizador de voz nativo do Android
- Autonomia de deficientes visuais: com o desenvolvimento de uma aplicação voltada para a identificação de obstáculos, deficientes visuais podem desfrutar de uma maior autonomia na sua locomoção diária.

6.2. Trabalhos Futuros

Pretende-se adicionar algumas outras funcionalidades ao sistema no sentido de melhorar seu desempenho como provedor de autonomia para deficientes visuais, além do aprimoramento das já implementadas.

Deseja-se implementar uma integração ao sistema do Google Maps onde o usuário seja capaz de informar seu destino e, utilizando-se da localização atual e do GPS, o sistema ao mesmo tempo tenha a capacidade de guiá-lo pela melhor rota possível (assim como faz a função “Rotas” do Google Maps) e informá-lo dos obstáculos presentes no caminho, provendo assim uma maior independência de terceiros ao usuário.

É desejado também a implementação de uma ferramenta que indique a distância aproximada entre o Smartphone do usuário e os obstáculos identificados, objetivando assim uma maior situação do usuário no espaço em que se encontra.

Por fim, objetivando aumento de funcionalidade, pretende-se desenvolver um sistema de seleção de instâncias da Darknet especializadas em determinados objetos, baseado nas necessidades individuais de cada usuário, contando também com possibilidade de customização deste, que poderá escolher seus objetos de maior prioridade.

Referências

ANDROID Developers Blog. Disponível em: <https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html>. Acesso em: 05 set. 2018.

ANDROID Developers. Disponível em: <https://developer.android.com/studio/intro/?hl=pt-br>. Acesso em: 05 set. 2018.

ANDROID Pit. Disponível em: <https://www.androidpit.com.br/aosp-android-open-source-project>. Acesso em: 05 set. 2018.

BENGIO, Yoshua; COURVILLE, Aaron; VINCENT, Pascal. Representation Learning: A Review and New Perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v. 35, 2013. Disponível em: <http://www.iro.umontreal.ca/~lisa/pointeurs/TPAMISI-2012-04-0260-1.pdf>. Acesso em: 30 jul. 2018.

BENGIO, Yoshua; COURVILLE, Aaron; VINCENT, Pascal. *Representation Learning: A Review and New Perspectives.* ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, [S.l.], v. 35, 2013. Disponível em: <http://www.iro.umontreal.ca/~lisa/pointeurs/TPAMISI-2012-04-0260-1.pdf>. Acesso em: 30 jul. 2018.

CHENG, John; GROSSMAN, Max; MCKERCHER, Ty. ***Professional CUDA C Programming***. [S.l.]: Wrox Press, 2014. 528 p.

CUDA Zone. [201?]. Disponível em: <https://developer.nvidia.com/cuda-zone>. Acesso em: 30 jul. 2018.

DENG, Li; YU, Dong. ***Deep Leearning****: Methods and Aplications*. Boston: Now, 2014. 199 p.

ESTATÍSTICAS de instalação do Google Play. 2018. Disponível em: <https://developer.android.com/about/dashboards/>. Acesso em: 10 out. 2018.

FUNDAÇÃO Dorina Nowill para Cegos. [201?]. Disponível em: <https://www.fundacaodorina.org.br/blog/estatuto-da-fundacao-dorina/>. Acesso em: 30 jul. 2018.

GHORPADE, Jayshree et al. GPGPU *PROCESSING IN CUDA ARCHITECTURE.* ***Advanced Computing: An International Journal***, [S.l.], v. 3, n. 1, p. 105-120, jan. 2012. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1202/1202.4347.pdf>. Acesso em: 10 out. 2018.

GUERRERO, Luis A.; VASQUEZ , Francisco; OCHOA, Sergio F. ***An Indoor Navigation System for the Visually Impaired***. 2012. 23 p. Artigo (Computer Science and Informatics School, Department of Computer Science)- Universidad de Costa Rica, Universidad de Chile, Chile, 2012. Disponível em: <http://www.mdpi.com/1424-8220/12/6/8236/pdf>. Acesso em: 20 mar. 2018.

HAYKIN, Simon. ***Neural Networks and Learning Machines****.* [S.l.]: Pearson, 2008. 906 p. Disponível em: <http://dai.fmph.uniba.sk/courses/NN/haykin.neural-networks.3ed.2009.pdf>. Acesso em: 30 jul. 2018.

HOPGOOD, F.Robert A.; HUBBOLD, Roger J.; DUCE, David. ***Advances in Computer Graphics II****.* [S.l.]: Springer, 1986. 186 p.

HUB, Andreas; DIEPSTRATEN, Joachim; ERTL, Thomas. ***Design and Development of an Indoor Navigation and Object Identification System for the Blind***. [200?]. 6 p. Artigo (Visualization and Interactive Systems Institute)- University of Stuttgart, Alemanha, [200?]. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.533&rep=rep1&type=pdf>. Acesso em: 20 mar. 2018.

KOHAVI, Ron; PROVOST, Foster. ***Glossary of Terms****: Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*. 1998. Disponível em: <http://robotics.stanford.edu/~ronnyk/glossary.html>. Acesso em: 30 jul. 2018.

LEGASSICK, Calvin *et al*. ***Artificial Intelligence Index***. 2017. Disponível em: <http://cdn.aiindex.org/2017-report.pdf>. Acesso em: 30 jul. 2018.

LODISH, Harvey *et al*. ***Molecular Cell Biology***. [S.l.]: W. H. Freeman, 2003. 973 p. Disponível em: <http://www.mustafaaltinisik.org.uk/s-MolecularCellBiology.pdf>. Acesso em: 10 jun. 2018.

LUGER, George F. ***Artificial Intelligence****: Structures and Strategies for Complex Problem Solving*. 6. ed. [S.l.]: Pearson, [200?]. 754 p. Disponível em: <http://iips.icci.edu.iq/images/exam/artificial-intelligence-structures-and-strategies-for--complex-problem-solving.pdf>. Acesso em: 30 jul. 2018.

MACMILLAN Dictionary: Speech Synthesis. Disponível em: <https://www.macmillandictionary.com/dictionary/british/speech-synthesis>. Acesso em: 10 out. 2018.

MCCORDUCK, Pamela. ***Machines Who Think****.* 2. ed. Natick: A K Peters, 2004. 553 p. Disponível em: <https://monoskop.org/images/1/1e/McCorduck_Pamela_Machines_Who_Think_2nd_ed.pdf>. Acesso em: 30 jul. 2018.

NVIDIA Launches the World's First Graphics Processing Unit: GeForce 256. Disponível em: <https://www.nvidia.com/object/IO_20020111_5424.html>. Acesso em: 10 out. 2018.

RASHAD, M. Z. *et al*. *An Overview of Text-To-Speech Synthesis Techniques.* ***LATEST TRENDS on COMMUNICATIONS and INFORMATION TECHNOLOGY***, Ilha de Corfu, Grécia, p. 84-89, jul. 2010. Disponível em: <http://www.wseas.us/e-library/conferences/2010/Corfu/CIT/CIT-14.pdf>. Acesso em: 10 out. 2018.

REDMON, Joseph *et al*. ***You Only Look Once****: Unified, Real-Time Object Detection.* 2016. 10 p. Artigo (Ciência da Computação)- University of Washington, [S.l.], 2015. 5. Disponível em: <https://arxiv.org/pdf/1506.02640.pdf>. Acesso em: 19 mar. 2018.

REDMON, Joseph. ***Darknet****: Open Source Neural Networks in C.* Disponível em: <https://pjreddie.com/darknet/>. Acesso em: 10 out. 2018.

RUSSELL, Stuart J.; NORVIG, Peter. ***Artificial Intelligence****: A Modern Approach.* Nova Jersey: Prentice Hall, 1995. 932 p. Disponível em: <https://www.cin.ufpe.br/~tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf>. Acesso em: 04 set. 2018.

SAMUEL, Arthur L. *Some Studies in Machine Learning Using the Game of Checkers.* ***IBM Journal****,* [S.l.], v. 3, n. 3, p. 535-554, jul. 1959. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.368.2254&rep=rep1&type=pdf>. Acesso em: 10 out. 2018.

SIMON, Phil. ***Too Big to Ignore****: The Business Case for Big Data*. Carolina do Norte: Wiley, 2015. 225 p.

THE ANDROID Source Code. Disponível em: <https://source.android.com/setup/>. Acesso em: 05 set. 2018.

VÁZQUEZ, Favio. *Deep Learning made easy with Deep Cognition*. 2017. Disponível em: <https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351>. Acesso em: 10 out. 2018.

VEEN, FJODOR VAN. *THE NEURAL NETWORK ZOO.* 2016. Disponível em: <http://www.asimovinstitute.org/neural-network-zoo/>. Acesso em: 10 out. 2018.

WEIK, Martin H. A *Third Survey of Domestic Electronic Digital Computing Systems*. Universidade de Michigan: Ballistic Research Laboratories, 1961. 1131 p.

YOLO-FOR-WINDOWS-V2*.* Disponível em: <https://github.com/unsky/yolo-for-windows-v2>. Acesso em: 10 out. 2018.

INSTITUTO IRIS. Disponível em: <http://www.iris.org.br/faq>. Acesso em: 30 out. 2018.