ROSSINI RUSSO CARVALHO

INTELIGÊNCIA ARTIFICIAL GENERATIVA E DE BORDA: UM ESTUDO VOLTADO AO DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

LAVRAS – MG

2023

ROSSINI RUSSO CARVALHO

INTELIGÊNCIA ARTIFICIAL GENERATIVA E DE BORDA: UM ESTUDO VOLTADO AO DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para obtenção do título de Bacharel.

Prof. DSc. Arthur de Miranda Neto
Orientador

LAVRAS – MG

2023

Ficha catalográfica elaborada pela Coordenadoria de Processos Técnicos

da Biblioteca Universitária da UFLA

Rossini Russo Carvalho

Inteligência Artificial Generativa e de Borda : Um Estudo Voltado ao Desenvolvimento de Aplicações Móveis / . – Lavras : UFLA, 2023.

23 p. :

TCC(graduação)–Universidade Federal de Lavras, 2023.

Orientador: Prof. DSc. Arthur de Miranda Neto.

Bibliografia.

1. TCC. 2. Inteligência Artificial . 3. *Edge AI*. 4. IA Generativa. 5. Computação de nuvem. I. Universidade Federal de Lavras. II. Título.

CDD-808.066

ROSSINI RUSSO CARVALHO

INTELIGÊNCIA ARTIFICIAL GENERATIVA E DE BORDA: UM ESTUDO VOLTADO AO DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para obtenção do título de Bacharel.

APROVADA em 13 de Dezembro de 2023.

Prof. DSc. Danilo Alves de Lima	UFLA
MSc. Fernando Elias de Melo Borges	UFLA

Prof. DSc. Arthur de Miranda Neto
Orientador

LAVRAS – MG
2023

AGRADECIMENTOS

Agradeço a minha família, que me proporcionou suporte incodicional durante toda minha trajetória: ao meu pai, Rossini; à minha mãe, Jaqueline. Aos membros do núcleo da Adriene: Aline, Andre, Adriene, Artur, Eduardo, Jordany, Thiago A., Thiago H., Luiza e Giovanna; que estiveram do meu lado desde o ínicio da graduação. Um agradecimento especial aos amigos Samuel e Thiago que me ajudaram a superar diversos obstáculos. Expresso minha gratidão ao professor Arthur que me ajudou e orientou durante o trabalho e a graduação.

RESUMO

A Inteligência Artificial (IA) faz parte do conjunto de tecnologias habilitadoras, sendo cotada para gerar um impacto profundo nos modelos de negócio operacionais. Contudo, trata-se de um assunto complexo e tem sido fonte de discussões nas principais empresas do Brasil e do mundo. Uma das importantes questões tem relação com a utilização de recursos em nuvem e/ou na borda. Por essa razão, este estudo realiza uma análise comparativa de duas abordagens para o desenvolvimento de aplicações de IA. A primeira, utiliza uma arquitetura baseada em nuvem e a API Chat GPT-3.5 Turbo, destaca-se em respostas em linguagem natural, mas enfrenta desafios relacionados à conectividade e privacidade. Em contraste, a segunda, desenvolvida em *Edge AI*, demonstra uma classificação eficiente localmente, garantindo autonomia e segurança aprimorada. A comparação destaca implicações de custo; enquanto a abordagem baseada em nuvem gera despesas escaláveis com base no uso, a abordagem baseada em *Edge AI* envolve custos de desenvolvimento dependentes da complexidade da rede neural. Esta pesquisa oferece insights sobre o cenário em evolução do desenvolvimento de aplicações de IA, proporcionando uma perspectiva matizada para auxiliar na tomada de decisões.

Palavras-chave: Inteligência Artificial. Edge AI. IA Generativa. Computação de Nuvem. Linguagem Natural.

ABSTRACT

Artificial Intelligence (AI) is a key enabler of technologies and is poised to substantially impact operational business models. This complex subject is a point of discussion among leading companies in Brazil and worldwide, particularly concerning the use of cloud and/or edge resources. This study conducts a comparative analysis of two methods for developing AI applications. The first approach leverages a cloud-based architecture and the Chat GPT-3.5 Turbo API. This configuration excels in generating natural language responses but encounters challenges pertaining to connectivity and privacy. Conversely, the second approach utilizes *Edge AI*, which offers efficient local classification, ensuring better security and autonomy. A crucial aspect of the comparison is the cost implications. The cloud-based approach incurs scalable expenses based on usage, while the *Edge AI*-based approach involves development costs, which are contingent on the neural network's complexity. This research provides valuable insights into the evolving landscape of AI application development, offering a nuanced perspective to aid decisionmaking. It underscores the trade-offs between cloud AI and *Edge AI*, including processing power, latency, energy consumption, connectivity, security, and cost. Both approaches have their strengths and limitations, and the choice between them will depend on specific application requirements and business objectives.

Keywords: Artificial Intelligence. Generative AI. Edge AI. Cloud Computing.

1 INTRODUÇÃO

Nos últimos anos, a produção de conteúdo por meio da Inteligência Artificial (IA) emergiu como um campo de crescente interesse, não apenas entre os especialistas em ciência da computação, mas também capturando a atenção do público em geral. O interesse cresce em torno dos diversos produtos de geração de conteúdo desenvolvidos por gigantes da tecnologia, tais como o ChatGPT (ZHANG, 2023) e o DALL-E2 (RAMESH et al., 2022). O conteúdo gerado por IA diz respeito à criação de material por meio de técnicas avançadas de IA generativa, em contraposição à autoria humana, possibilitando a automação da produção de grandes volumes de conteúdo em curtos períodos de tempo (YUNJIU; WEI; ZHENG, 2022). O ChatGPT, um modelo de linguagem desenvolvido pela OpenAI, destaca-se na construção de sistemas de IA conversacional, demonstrando uma capacidade excepcional de compreender e responder de maneira coerente a entradas em linguagem humana. Da mesma forma, o DALL-E-2, também desenvolvido pela OpenAI, sobressai como um modelo de IA generativa de última geração capaz de gerar imagens únicas e de alta qualidade a partir de descrições textuais em questão de minutos.

Apesar das notáveis conquistas no campo do conteúdo gerado por IA, surgem desafios significativos no processamento dessas IAs generativas, principalmente quando se utiliza a computação em nuvem. Questões relacionadas à segurança, velocidade dos serviços e conexões lentas têm se tornado obstáculos frequentes, resultando em baixa largura de banda, alta latência e interferência (CHEN et al., 2016),(SINGH; KOVACS; KISS, 2022). Estes desafios se agravam com o contínuo aumento do número de dispositivos conectados à Internet, sejam móveis ou fixos (IFTIKHAR et al., 2023). Como resultado, propostas para reimaginar a própria natureza da nuvem ganham relevância, sugerindo a criação de uma "borda"distintiva separada do núcleo, onde o processamento e armazenamento em larga escala podem ocorrer em tempo real (ASLANPOUR et al., 2021).

O conceito de Edge AI destaca-se como a incorporação da IA em dispositivos do mundo real. Este modelo de IA envolve o processamento de informações próximo aos usuários, na borda da rede, em contraste com a abordagem centralizada de um centro de dados de um provedor de serviços em nuvem ou o armazenamento de dados privado de uma empresa (HU; HÉDÉ; PATEL, 2014). Neste contexto, o presente trabalho abordará o funcionamento de uma IA generativa e suas aplicações, apresentando um comparativo entre uma IA convencional e uma Edge AI.

Inteligência Artificial Generativa e de Borda: Um Estudo Voltado ao Desenvolvimento de Aplicações Móveis

Rossini Russo Carvalho*
Engenharia de Controle e Automação - UFLA
Lavras, Minas Gerais, BRA
rossini.carvalho@estudante.ufla.br

Arthur de Miranda Neto†
Departamento de Automática - UFLA
Lavras, Minas Gerais, BRA
arthur.miranda@ufla.br

Abstract

A Inteligência Artificial (IA) faz parte do conjunto de tecnologias habilitadoras, sendo cotada para gerar um impacto profundo nos modelos de negócio operacionais. Contudo, trata-se de um assunto complexo e tem sido fonte de discussões nas principais empresas do Brasil e do mundo. Uma das importantes questões tem relação com a utilização de recursos em nuvem e/ou na borda. Por essa razão, este estudo realiza uma análise comparativa de duas abordagens para o desenvolvimento de aplicações de IA. A primeira, utiliza uma arquitetura baseada em nuvem e a API Chat GPT-3.5 Turbo, destaca-se em respostas em linguagem natural, mas enfrenta desafios relacionados à conectividade e privacidade. Em contraste, a segunda, desenvolvida em *Edge AI*, demonstra uma classificação eficiente localmente, garantindo autonomia e segurança aprimorada. A comparação destaca implicações de custo; enquanto a abordagem baseada em nuvem gera des-pesas escaláveis com base no uso, a abordagem baseada em *Edge AI* envolve custos de desenvolvimento dependentes da complexidade da rede neural. Esta pesquisa oferece insights sobre o cenário em evolução do desenvolvimento de aplicações de IA, proporcionando uma perspectiva matizada para auxiliar na tomada de decisões.

CCS Concepts: • **IA Generativa**; • **Edge AI**; • **Inteligência Artificial**; • **Redes Neurais**; • **Nuvem**; • **Linguagem Natural**;

Keywords: Generative AI, Edge AI, Aplicação Android, Inteligência Artificial, Nuvem, Chat GPT-3.5, Linguagem Natural, Classificação de Imagens, Rede Neural.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Rossini Russo Carvalho and Arthur de Miranda Neto. 2024. Inteligência Artificial Generativa e de Borda: Um Estudo Voltado ao Desenvolvimento de Aplicações Móveis.

1 Introdução

Nos últimos anos, a produção de conteúdo por meio da Inteligência Artificial (IA) emergiu como um campo de crescente interesse, não apenas entre os especialistas em ciência da computação, mas também capturando a atenção do público em geral. O interesse cresce em torno dos diversos produtos de geração de conteúdo desenvolvidos por gigantes da tecnologia, tais como o ChatGPT [1] e o DALL-E2 [2]. O conteúdo gerado por IA diz respeito à criação de material por meio de técnicas avançadas de IA generativa, em contraposição à autoria humana, possibilitando a automação da produção de grandes volumes de conteúdo em curtos períodos de tempo [3]. O ChatGPT, um modelo de linguagem desenvolvido pela OpenAI, destaca-se na construção de sistemas de IA conversacional, demonstrando uma capacidade excepcional de compreender e responder de maneira coerente a entradas em linguagem humana. Da mesma forma, o DALL-E-2, também desenvolvido pela OpenAI, sobressai como um modelo de IA generativa de última geração capaz de gerar imagens únicas e de alta qualidade a partir de descrições textuais em questão de minutos.

Apesar das notáveis conquistas no campo do conteúdo gerado por IA, surgem desafios significativos no processamento dessas IAs generativas, principalmente quando se utiliza a computação em nuvem. Questões relacionadas à segurança, velocidade dos serviços e conexões lentas têm se tornado obstáculos frequentes, resultando em baixa largura de banda, alta latência e interferência [4],[5]. Esses desafios se agravam com o contínuo aumento do número de dispositivos conectados à Internet, sejam móveis ou fixos [6]. Como resultado, propostas para reimaginar a própria natureza da nuvem ganham relevância, sugerindo a criação de uma "borda" distintiva separada do núcleo, onde o processamento e armazenamento em larga escala podem ocorrer em tempo real [7].

O conceito de *Edge AI* destaca-se como a incorporação da IA em dispositivos do mundo real. Esse modelo de IA envolve o processamento de informações próximo aos usuários, na borda da rede, em contraste com a abordagem centralizada de um centro de dados de um provedor de serviços em nuvem ou o armazenamento de dados privado de uma empresa [8].

Neste contexto, o presente trabalho abordará o funcionamento de uma IA generativa e irá demonstrar sua usabilidade em uma aplicação android. Além disso, propõe-se um comparativo entre o desenvolvimento de uma aplicação que emprega uma IA convencional e outra que utiliza uma *Edge AI*.

2 Fundamentação Teórica

2.1 IA Generativa

Redes neurais e algoritmos de aprendizado profundo são peças essenciais para que a IA generativa chegue ao patamar que está hoje. São essas tecnologias que possibilitam a concepção do Modelo Transformer [9], uma arquitetura fundamental que impulsionou muitos dos modelos de última geração em diversas aplicações, como o GPT-3, DALL-E-2, Codex e Gopher. Foi concebido para superar as limitações dos modelos tradicionais, como as Redes Neurais Recorrentes (RNNs), na manipulação de sequências de comprimento variável e na compreensão contextual. Sua base está no mecanismo de autoatenção, permitindo que o modelo atente-se a diferentes partes de uma sequência de entrada.

Desde a introdução da arquitetura Transformer, essa tornou-se a escolha dominante no processamento de linguagem natural devido à sua capacidade de paralelismo e aprendizado. Esses modelos passam por uma fase inicial de pré-treinamento em grandes quantidades de texto, a fim de aprender padrões linguísticos e contextos diversos. Durante o pré-treinamento, o modelo absorve uma enorme quantidade de dados e aprende a associar palavras e frases, compreender relações gramaticais e semânticas e contextualizar informações. Essa etapa inicial de treinamento permite que o modelo desenvolva uma compreensão geral da linguagem. [9].

As principais evoluções no campo da produção de conteúdo por IA nos tempos recentes, em comparação com trabalhos anteriores, resultam do treinamento de modelos gerativos mais sofisticados em conjuntos de dados maiores e do acesso a recursos computacionais mais extensos. Por exemplo, a estrutura principal do GPT-3 permanece a mesma do GPT-2, mas o tamanho dos dados de pré-treinamento aumentou de WebText [10] (38GB) para CommonCrawl [11] (570GB após filtragem), e o tamanho do modelo fundamental cresceu de 1.5B para 175B. Portanto, o GPT-3 apresenta uma capacidade de generalização superior à do GPT-2 em diversas tarefas, como na extração de intenções humanas.

Após o pré-treinamento, o modelo pode ser adaptado ou melhorado para tarefas específicas por meio do que é chamado de fine-tuning. Nessa fase, o modelo é ajustado para uma tarefa particular, como tradução, sumarização de texto, geração de texto ou análise de sentimentos. Durante o fine-tuning, partes específicas do modelo são ajustadas para se adequarem à tarefa desejada, utilizando conjuntos menores de dados, de maneira a aprimorar a precisão e o desempenho naquela tarefa específica [12].

2.2 *Edge AI*

Além do desenvolvimento da IA, a ascensão da computação de borda tem sido fundamental para possibilitar avaliar, gerenciar e gerar dados imediatamente no local do evento. O termo "borda" se refere a um tipo de computação distribuída que coloca processamento, armazenamento de dados e geração de energia no local do acontecimento [8].

Houve um tempo em que dispositivos de borda não podiam ser usados para treinar e implantar modelos de *Machine Learning* (ML) e *Deep Learning*. No entanto, com a introdução de computação mais poderosa, a capacidade desses dispositivos de lidar com uma variedade de tarefas otimizadas para a IA cresceu substancialmente, o que possibilitou lidar com dados em tempo real, com um nível elevado de privacidade e segurança. Dessa maneira, graças à borda e à IA, máquinas e aparelhos podem entender, aprender e agir instantaneamente com dados e informações [13].

A coleta, preparação e análise de dados podem acontecer quase que em tempo real, sem a necessidade de esperar que a saída do modelo seja baixada da nuvem. Ao colocar modelos de IA na borda em vez de na nuvem, é possível que esses possam analisar os dados mais rapidamente, tomar decisões mais ágeis, aumentar a segurança do processamento de dados e melhorar a experiência do usuário [8].

O *TensorFlow Lite* (TFLite), por exemplo, é uma versão otimizada da biblioteca *TensorFlow*, projetada para facilitar a implementação eficiente de algoritmos de *ML* em dispositivos de borda. Essa plataforma é particularmente valiosa devido à sua capacidade de realizar inferência de modelos em ambientes com recursos computacionais limitados. O TFLite atinge esse objetivo por meio de otimizações, como quantização, que reduz a precisão numérica dos parâmetros do modelo, tornando-os mais adequados para dispositivos de baixo consumo energético. Além disso, suporta diversos tipos de modelos, oferece compatibilidade com o TensorFlow padrão e possui uma API de integração simples, facilitando o desenvolvimento de soluções para uma variedade de casos de uso em dispositivos de borda. Essa abordagem possibilita a execução eficiente de algoritmos em ambientes descentralizados, contribuindo para a disseminação da inteligência artificial em dispositivos com recursos limitados [14].

3 Metodologia

Neste estudo, foram elaborados dois aplicativos *Android* distintos: um destinado a evidenciar a eficácia de uma *IA Generativa*, fazendo uso da computação em nuvem, enquanto o

outro utiliza exclusivamente uma *Edge AI*. Posteriormente, uma análise comparativa é realizada com vistas a avaliar e destacar as diferenças fundamentais entre essas duas abordagens.

3.1 Aplicação integrada à nuvem

O propósito principal da aplicação desenvolvida [15] é fornecer respostas naturais às perguntas dos usuários, independentemente de serem feitas por texto ou voz. O fluxo da arquitetura está representado na Figura 1.

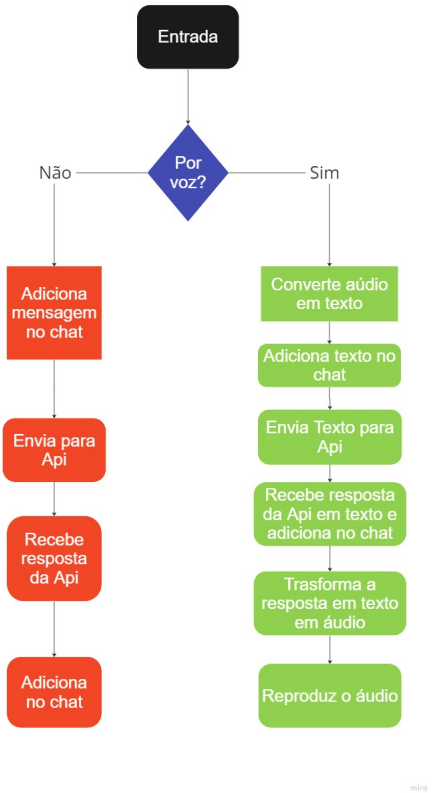

Figura 1 Fluxo de funcionamento do código.

3.1.1 Estrutura. A estrutura visual do aplicativo foi concebida para simular um chat, proporcionando aos usuários a sensação de interagir com uma entidade semelhante a uma pessoa.

3.1.2 Reconhecimento de voz. Foi utilizado o *SpeechRecognizer* para converter a fala em texto. Esta ferramenta permite capturar a fala do usuário por meio do microfone e traduzi-la em texto utilizável pelo aplicativo.

3.1.3 Conversão de texto em fala. Para converter texto em fala, foi empregado o *TextToSpeech*. Esta ferramenta possibilita transformar strings de texto em fala audível, facilitando a comunicação de informações por meio de áudio, através de uma voz robotizada.

3.1.4 Integração com a API. Durante o desenvolvimento do aplicativo, o primeiro passo consistiu na seleção do modelo de IA Generativa a ser utilizado. O escolhido foi o GPT-3.5 Turbo devido à sua ampla gama de recursos avançados em linguagem natural. Esse modelo proporciona respostas mais precisas e contextualizadas. Sua capacidade de compreender e gerar texto altamente coeso o torna ideal para projetos que exigem uma IA sofisticada e interações fluidas.

Para realizar requisições na API do GPT 3.5 Turbo, foi empregado o OkHttpClient, uma biblioteca de código aberto para Android, que é uma implementação do cliente HTTP para requisições de rede. Essa biblioteca oferece uma API eficaz para lidar com solicitações GET, POST, PUT, DELETE, entre outras. Um objeto JSON, representado na Figura 2, com informações relevantes para a comunicação, incluindo o número máximo de tokens, a chave secreta e a mensagem, é enviado para a API do GPT. Como retorno, um objeto Json que inclui um campo com a resposta elaborada pelo Chat Gpt.

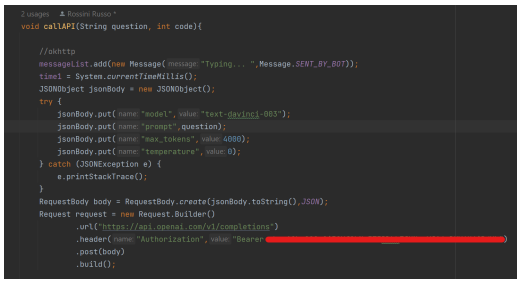

Figura 2 Montagem do Json que será enviado para a Api.

3.2 Aplicação utilizando *Edge AI*

Com o intuito de alcançar os objetivos delineados no trabalho, optou-se por criar um aplicativo de classificação de imagens. Essa escolha foi feita devido à complexidade de treinamento de um modelo de processamento de linguagem natural.

O processo de criação do aplicativo Android destinado à classificação de imagens por meio de uma *Edge AI* pode ser desdobrado em dois elementos essenciais: a formulação do modelo de ML da IA, sendo todo o processo de desenvolvimento/treinamento da IA feito no Google Colab, e a implementação do aplicativo Android propriamente dito, a interface pela qual o usuário interagirá, feito no *Android Studio*.

3.2.1 Base de dados. A etapa inicial para a concepção da *Edge AI* envolve a preparação de uma base de dados para o treinamento do modelo. Utilizou-se uma base de dados contendo diversas imagens de maçãs, bananas e laranjas, constituindo o substrato para o aprendizado da IA.

3.2.2 Configurando a rede neural. A arquitetura da rede neural em análise é composta por 10 camadas distintas, cada uma desempenhando um papel crucial no processo de aprendizado e extração de características. A seguir, a configuração de cada camada descrita:

1. Camada de Reescalonamento.

- A primeira camada da rede é uma camada de reescalonamento, cujo propósito é normalizar as imagens, ajustando os valores dos pixels para o intervalo de 0 a 1. Essa normalização contribui para um processamento mais eficiente e estável dos dados de entrada.

2. Camadas Convolutivas (3 camadas).

- Três camadas convolutivas foram implementadas, cada uma composta por 32 filtros (kernels) de tamanho 3x3. A ativação utilizada é a ReLU (*Rectified Linear Unit*). Essas camadas são projetadas para aprender padrões locais nas entradas, permitindo que a rede identifique características específicas em diferentes regiões dos dados de entrada.

3. Camadas de Agrupamento Máximo (3 camadas).

- Intercaladas com as camadas convolutivas, encontram-se três camadas de agrupamento máximo (*Max Pooling*). Essas camadas têm a finalidade de reduzir o tamanho dos mapas de características gerados pelas camadas convolutivas. Ao dividir a entrada em regiões (por exemplo, 2x2 ou 3x3) e reter apenas o valor máximo de cada região, essa etapa contribui para a preservação das características mais proeminentes, ao mesmo tempo em que reduz a dimensionalidade espacial. O funcionamento dessa camada esta exemplificado na Figura 3.

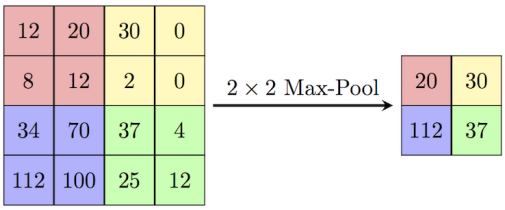

Figura 3 Exemplo de entrada e saída após passar pela camada de agrupamento máximo.

4. Camada de Achatamento (Flatten Layer).

- Após as camadas convolutivas e de agrupamento máximo, uma camada de achatamento é introduzida. Esta camada é responsável por transformar a entrada bidimensional em um vetor unidimensional, preparando os dados para a transição para as camadas densas subsequentes.

5. Camadas Densas (2 camadas).

- Duas camadas densas foram incorporadas, a primeira composta por 128 neurônios e a segunda com 3 neurônios. A ativação utilizada é a ReLU. Estas camadas desempenham um papel crucial na realização de tarefas mais complexas de aprendizado, combinando e interpretando as características extraídas nas camadas anteriores. A última camada, com 3 neurônios, corresponde às classes que serão classificadas, tornando-se a camada de saída da rede.

```
model = tf.keras.Sequential(
    [
     tf.keras.layers.Rescaling(1./255),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Conv2D(32, 3, activation="relu"),
     tf.keras.layers.MaxPooling2D(),
     tf.keras.layers.Flatten(),
     tf.keras.layers.Dense(128, activation="relu"),
     tf.keras.layers.Dense(3)
    ]
)
```

Figura 4 Montagem da rede neural.

Essa configuração estratégica da rede neural (Figura 4) visa otimizar a capacidade de aprendizado, permitindo a identificação eficaz de padrões e características relevantes nos dados de entrada. O modelo é projetado para abordar tarefas de classificação, culminando na atribuição adequada das instâncias de entrada a uma das três classes definidas.

3.3 Desenvolvimento do aplicativo

O fluxo da arquitetura desenvolvida está representado na Figura 5.

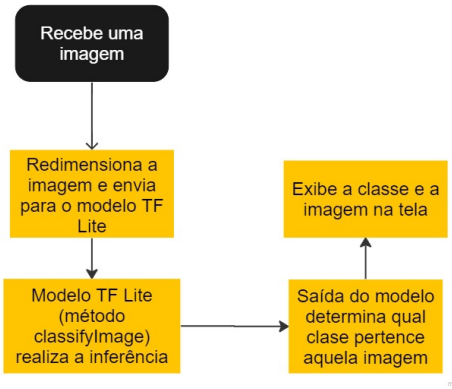

Figura 5 Fluxo do aplicativo de borda.

Na etapa inicial deste processo, a rede neural desenvolvida passa por uma conversão para o formato *TFLite*. O modelo resultante é então salvo em um arquivo designado, pronto para ser integrado posteriormente ao aplicativo Android.

A integração do arquivo contendo o modelo *TFLite* no aplicativo é simplificada pelo *Android Studio*, que oferece um campo específico para esse tipo de arquivo.

Para compreender o funcionamento do aplicativo Android, é crucial destacar dois métodos principais: `"onActivityResult()"` e `"classifyImage()"`, explicados a seguir.

Tratamento de Resultados da Atividade:

O método `"onActivityResult()"` entra em cena quando uma atividade iniciada aguarda um resultado. No contexto do aplicativo, esse método é acionado após a câmera capturar uma imagem ou quando o usuário seleciona uma imagem da galeria. O método extrai a imagem da intenção resultante, redimensiona-a para uma dimensão específica (`imageSize`), exibe a imagem na `ImageView` e, em seguida, invoca o método `"classifyImage()"` para realizar a classificação utilizando o modelo *TFLite*.

Classificação da Imagem:

O método `"classifyImage()"` recebe uma imagem como parâmetro e é encarregado de classificá-la com base no modelo *TFLite* pré-treinado. O processo lógico é detalhado da seguinte forma:

1. **Carregamento do Modelo:** No início, uma instância do modelo *TFLite* é criada utilizando a classe `Model`. Essa classe funciona como um invólucro gerado pelo *TFLite Model Maker* durante o treinamento do modelo.
2. **Criação dos Inputs para o Modelo:** Em seguida, um tensor de entrada (`inputFeature0`) é criado com um tamanho fixo de 1x32x32x3, indicando uma imagem de 32x32 pixels com 3 canais de cor (RGB). Além disso, um `ByteBuffer` é preparado para armazenar os valores dos pixels da imagem, e o tamanho desse *buffer* é calculado com base nas dimensões da imagem e seus canais de cor.
3. **Processamento da Imagem para o Modelo:** Os valores dos pixels da imagem são obtidos e convertidos para o formato RGB. Em seguida, esses valores são normalizados para o intervalo [0, 1] e inseridos no `ByteBuffer`.
4. **Carregamento dos Dados de Entrada no `TensorBuffer`:** Os dados normalizados do `ByteBuffer` são carregados para o tensor de entrada (`inputFeature0`).
5. **Execução da Inferência do Modelo:** A inferência do modelo é realizada utilizando os dados de entrada fornecidos, e a saída é obtida em um tensor (`outputFeature0`).
6. **Obtenção das Classes de Confiança:** As confianças atribuídas a cada classe pela rede neural são obtidas a partir do tensor de saída.
7. **Identificação da Classe com Maior Confiança:** Uma iteração sobre as confianças é realizada para encontrar a classe com a maior probabilidade.
8. **Resultado:** O resultado da classificação, ou seja, a classe com maior confiança, é exibido no `TextView` denominado "result". Além disso, os recursos associados ao modelo são liberados após a conclusão da classificação.

4 Resultados e Discussão

O aplicativo desenvolvido em nuvem (*Aplicação 1*) tem a capacidade de responder a perguntas formuladas, tanto em formato de texto quanto de voz, de maneira natural, desde que a aplicação esteja conectada à internet. Em contraste, o aplicativo baseado em *Edge AI* (*Aplicação 2*) destaca-se pela capacidade de classificação de imagens, contanto que essas imagens estejam classificadas dentre as 3 classes de imagens previamente treinadas, sendo elas maçã, banana e laranja.

É válido ressaltar que, embora os aplicativos desempenhem funções distintas, ambos seguem uma estrutura comum: uma entrada é fornecida e uma resposta é aguardada como resultado.

4.1 Arquitetura

Ao analisar as Figuras 1 e 5, torna-se evidente que a arquitetura de ambos os aplicativos não difere significativamente. Em ambos os casos, o processo inicia-se ao receber uma entrada, procedendo então ao tratamento dessa entrada para o formato apropriado, e finalmente encaminhando-a para um método específico para a obtenção da resposta. Na *Aplicação 1*, o método utilizado é a API do Chat GPT, uma fonte externa que opera de maneira remota. Por outro lado, na *Aplicação 2*, o método utilizado é o modelo *TFLite*, indicando que todo o processamento ocorre localmente no dispositivo, eliminando a necessidade de conexão externa.

4.2 Privacidade e Segurança

Conforme salientado no tópico anterior, a *Aplicação 2* opera de maneira independente, sem a necessidade de comunicação constante com uma fonte externa durante seu funcionamento. Essa autonomia reduz significativamente a exposição a potenciais vulnerabilidades de segurança, conferindo uma maior segurança ao processamento de dados sensíveis. Adicionalmente, o usuário fica imune a fatores externos, como a falta de conexão com a internet.

Ao contrário, a *Aplicação 1* requer conexão contínua à internet para seu pleno funcionamento. Nesse cenário, a necessidade de enviar dados para uma aplicação externa a fim de obter uma resposta implica em uma vulnerabilidade potencial. O usuário, ao compartilhar seus dados com uma aplicação externa, não pode garantir integralmente a segurança de seus dados, expondo-os ao risco de vazamento. Considerando esse ponto, seria prudente explorar a implementação de uma camada de criptografia. Essa medida adicionaria um nível a mais de segurança, mitigando os riscos associados ao possível vazamento de dados. No entanto, a adição dessa camada implica em uma maior complexidade da comunicação com a nuvem e também em maiores custos. Sendo assim, apesar de a comunicação com a API GPT-3.5 ser feita de forma segura, não temos controle sobre o destino desses dados compartilhados após a sua transmissão.

4.3 Custo do desenvolvimento

Um aspecto crucial a ser considerado é o custo envolvido no desenvolvimento de ambos os aplicativos. A utilização da API do Chat GPT implica em despesas financeiras, já que cada requisição feita demanda um pagamento. Para fornecer uma visão mais abrangente, foi realizada uma análise considerando as requisições de teste efetuadas ao longo do mês de outubro de 2023, representadas na Figura 6.

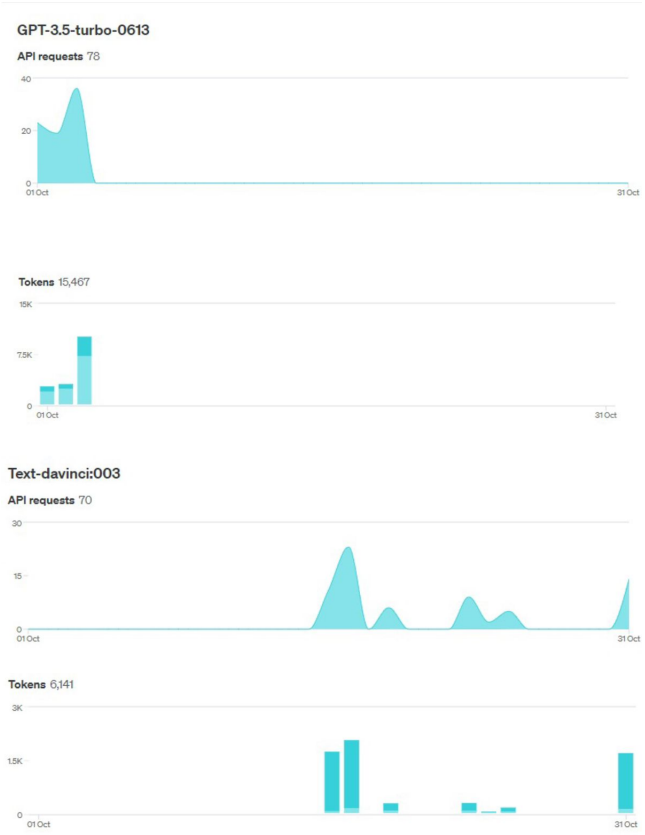

Figura 6 Requisições de teste efetuadas ao longo do mês de outubro de 2023.

Durante esse período, foram efetuadas 148 requisições à API, totalizando o envio de 21.688 tokens e gerando uma média de 146 tokens por requisição. O custo total associado a essas requisições foi de 0,15 centavos de dólar, o que equivale a aproximadamente 0,001 centavos de dólar por requisição. A princípio, esse valor pode parecer insignificante; entretanto, é crucial considerar que a finalidade de um aplicativo é atender a uma base diversificada de usuários. À medida que escalamos o número de usuários, esse custo tende a aumentar significativamente. Por exemplo, se considerarmos que o aplicativo possui 10.000 usuários em um mês e a média de requisições feitas por cada um deles é mantida em 148, ao final do mês, seria necessário desembolsar $1.480 para a OpenAI. Esse montante, embora inicialmente possa parecer modesto, torna-se uma consideração financeira substancial à medida que o aplicativo cresce em escala. Diferentemente, a *Aplicação 2* não incorre em custos durante sua execução, pois todo o processo ocorre na borda, no aparelho do usuário. Contudo, é crucial ressaltar, nesse ponto, o custo associado ao desenvolvimento e treinamento de uma inteligência artificial. A rede neural desenvolvida é relativamente simples, composta por apenas 10 camadas, e a quantidade de dados usada para seu treinamento é de 100MB. Portanto, não houve a necessidade de investimentos significativos em *hardware* especializado, e o treinamento pôde ser concluído em algumas horas.

Entretanto, treinar uma rede neural mais complexa, como uma IA generativa, envolve considerações críticas em termos de custo e tempo. Ambos os fatores estão intrinsecamente ligados à complexidade da arquitetura do modelo, ao tamanho do conjunto de dados de treinamento e aos recursos computacionais disponíveis. Modelos mais sofisticados e de maior escala geralmente demandam um investimento considerável em *hardware* especializado, como GPUs (*Graphics Processing Unit*) ou TPUs (*Tensor Processing Unit*), resultando em custos substanciais. Além disso, o tempo necessário para treinar uma IA generativa é diretamente proporcional à quantidade de dados e à complexidade do modelo, podendo variar desde horas até semanas, dependendo das circunstâncias.

Especialistas e tecnólogos estimam que o processo crítico de treinamento de modelos de linguagem extensos, como o GPT-3 da OpenAI, pode ultrapassar os US$4 milhões em custos. Modelos mais avançados podem exigir despesas superiores a "vários milhões de dólares". Um exemplo é o modelo LLaMA da Meta, lançado no início de 2023, que utilizou 2.048 GPUs Nvidia A100 para treinar em 1,4 trilhão de tokens, levando cerca de 21 dias. O processo de treinamento demandou aproximadamente 1 milhão de horas de GPU, resultando em um custo superior a US$ 2,4 milhões com preços dedicados da AWS. Apesar de possuir 65 bilhões de parâmetros, esse modelo é menor do que os atuais modelos GPT da OpenAI, como o ChatGPT-3, que possui 175 bilhões de parâmetros [16].

Dito isso, torna-se inviável para o usuário comum desenvolver uma inteligência artificial com o mesmo poder do GPT-3. No entanto, para tarefas mais simples, como a classificação de imagens dentro de um padrão, é possível utilizar alternativas como o *TFLite* para criar uma IA.

4.4 Complexidade no desenvolvimento

A escolha de adotar uma abordagem de IA centralizada para o desenvolvimento da *Aplicação 1* proporcionou notável simplicidade e agilidade no processo. Ficou evidente que não requer um profundo conhecimento em modelagem ou treinamento de IA. Essa eficiência é resultado da possibilidade de utilizar de diferentes APIs que disponibilizam acesso a modelos complexos e sofisticados, pré-treinados e prontos para integração. A escolha específica pela API 3.5 Turbo

mostrou-se particularmente vantajosa, exibindo eficácia em todas as tarefas designadas pelo aplicativo. Essa API permite a utilização de modelos mais robustos, aproveitando os consideráveis recursos computacionais disponíveis, resultando em desempenho otimizado.

Em contrapartida, a opção por adotar a tecnologia *Edge AI* no desenvolvimento da *Aplicação 2* revelou-se mais desafiadora e complexa. Essa abordagem demanda a realização de diversas etapas adicionais, incluindo a coleta de dados específicos, assim como o desenvolvimento e treinamento de modelos para desempenhar tarefas específicas na borda. Isso requer um conhecimento mais aprofundado em IA.

A limitação de recursos na borda apresenta um desafio adicional, pois os modelos devem ser otimizados para se tornarem mais simples e leves. A necessidade de otimização contínua, para assegurar a eficácia dentro dessas restrições, torna o processo mais intrincado. No entanto, essa complexidade ressalta a importância da eficiência energética e da capacidade de operar em ambientes com recursos limitados. Nesse cenário, embora a complexidade seja maior, a capacidade de adaptação a condições operacionais desafiadoras emerge como uma característica distintiva da *Edge AI*.

4.5 Latência, Conectividade e Tempo de Resposta

Esta análise envolveu a coleta de dados provenientes de 20 testes distintos realizados para cada uma das aplicações em questão, levando em consideração o tempo decorrido desde a entrada até a obtenção da resposta. Os testes foram realizados no simulador do *Android Studio*, o aparelho simulado foi o Pixel 2 e foram alocados 6 gigas de memória RAM para a simulação. A seguir, são apresentados dois gráficos (7, 8) que encapsulam os resultados obtidos.

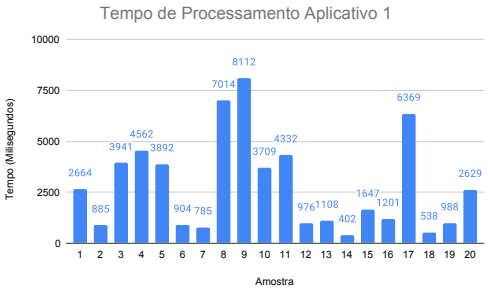

Figura 7 Tempo de processamento aplicação 1

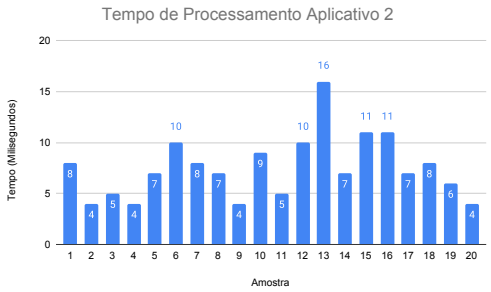

Figura 8 Tempo de processamento aplicação 2

Os dados revelam que, no caso da *Aplicação 1*, a média para a obtenção da resposta é de 2.833 milissegundos. Já para a *Aplicação 2*, essa média é notavelmente menor, registrando aproximadamente 7,5 milissegundos, representando uma diferença de cerca de 377 vezes em relação ao primeiro. Esta discrepância pode ser atribuída à abordagem adotada: enquanto a *Aplicação 2* realiza todo o processamento internamente no dispositivo, a *Aplicação 1* necessita enviar os dados a uma fonte externa para obter a resposta. Desta maneira, a *Aplicação 1* proporciona uma experiência mais estável aos usuários e oferece a possibilidade de aplicações com requisitos em tempo real.

5 Conclusão

Diante da evolução constante nas tecnologias de inteligência artificial, a presente pesquisa buscou explorar e comparar duas abordagens distintas no desenvolvimento de aplicações que utilizam IA. A aplicação fundamentada em uma arquitetura em nuvem, destaca-se pela capacidade de resposta natural a perguntas em texto e voz, por meio da API do Chat GPT-3.5 Turbo. Em contrapartida, a aplicação baseada em processamento de borda, *Edge AI*, demonstrou habilidade precisa na classificação de imagens localmente no dispositivo, sem depender de uma conexão contínua com a internet.

A análise comparativa revelou nuances significativas em aspectos cruciais, incluindo privacidade, custo, complexidade de desenvolvimento e desempenho operacional. A dependência de recursos externos à aplicação pelo uso de API incorre em despesas financeiras escaláveis com o aumento do número de usuários. Por outro lado, nas aplicações de borda, o treinamento de modelos de IA mais avançados são custosos. Requisitos de latência e tempo de resposta sublinham a eficiência do processamento na borda, *Edge AI*, com médias de tempo de resposta notavelmente inferiores em comparação com a abordagem em nuvem.

Em síntese, a escolha entre as abordagens analisadas dependerá das necessidades específicas relacionadas aos requisitos funcionais e não funcionais de aplicação. À medida que a tecnologia continua evoluindo, estas reflexões formam uma base sólida para a tomada de decisões informadas no

desenvolvimento de aplicações baseadas em inteligência artificial, com implicações significativas para os negócios do século 21.

References

[1] "ChatGPT: Optimizing language models for dialogue," Nov. 2022.

[2] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," Apr. 2022. arXiv:2204.06125 [cs].

[3] L. Yunjiu, W. Wei, and Y. Zheng, "Artificial intelligence-generated and human expert-designed vocabulary tests: A comparative study," *SAGE Open*, vol. 12, no. 1, pp. 21582440221082130, 2022.

[4] X. Chen, L. Jiao, W. Li, X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," IEEE/ACM Trans. Netw. 24 (5) (2016) 2795–2808.

[5] R. Singh, J. Kovacs, T. Kiss, "To offload or not? an analysis of big data offloading strategies from edge to cloud," in: 2022 IEEE World AI IoT Congress (AIIoT), 2022, pp. 46–52.

[6] S. Iftikhar, et al., "AI-based Fog and Edge Computing: A Systematic Review, Taxonomy and Future Directions," Internet of Things, 2022, 100674.

[7] M.S. Aslanpour, A.N. Toosi, C. Cicconetti, B. Javadi, et al., "Serverless Edge Computing: Vision and Challenges," in: 2021 Australasian Computer Science Week Multiconference, 2021, pp. 1–10.

[8] M. Patel, *Mobile-edge computing – introductory technical white paper*, [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1, 2014.

[9] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, Alexander M. Rush, *Transformers: State-of-the-Art Natural Language Processing*, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, edited by Qun Liu and David Schlangen, October 2020, Online, published by the Association for Computational Linguistics, pages 38–45.

[10] A. Gokaslan and V. Cohen, "Openwebtext corpus", Abr, 2019.

[11] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[12] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, Hsiao-Wuen Hon, *Unified Language Model Pre-training for Natural Language Understanding and Generation*, in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett, volume 32, published by Curran Associates, Inc., year 2019, https://proceedings.neurips.cc/paper_files/paper/2019/file/c20bb2d9a50d5ac1f713f8b34d9aac5a-Paper.pdf.

[13] Y. Shi, K. Yang, T. Jiang, J. Zhang, K.B. Letaief, *Communication-efficient edge AI: algorithms and systems*, *IEEE Commun. Surv. Tutor.*, vol. 22, no. 4, pp. 2167–2191, 2020.

[14] Giorgos Demosthenous and Vassilis Vassiliades, *Continual Learning on the Edge with TFLite*, *CoRR*, Volume 2105.01946, 2021.

[15] Rossini Russo, *Tuesday AI*, [Online]. Disponível em: https://github.com/rossinirusso/ProjetoTCC, 2023.

[16] Jonathan Vanian, *ChatGPT and generative AI are booming, but the costs can be extraordinary*, *CNBC*, [Online]. Disponível em: https://www.cnbc.com/2023/03/13/chatgpt-and-generative-ai-are-booming-but-at-a-very-expensive-price.html, 2023.

3 CONCLUSÃO

Diante da evolução constante nas tecnologias de IA, a presente pesquisa buscou explorar e comparar duas abordagens distintas. A aplicação fundamentada em uma arquitetura em nuvem, destaca-se pela capacidade de resposta natural a perguntas em texto e voz, por meio da API do Chat GPT-3.5 Turbo. Em contrapartida, a aplicação baseada em processamento de borda, Edge AI, demonstrou habilidade precisa na classificação de imagens localmente no dispositivo, sem depender de uma conexão contínua com a internet.

A análise comparativa revela nuances significativas em aspectos cruciais, incluindo privacidade, custo, complexidade de desenvolvimento e desempenho operacional. A dependência de recursos externos à aplicação pelo uso de API incorre em despesas financeiras escaláveis com o aumento do número de usuários. Por outro lado, nas aplicações de borda, o treinamento de modelos de IA mais avançados são custosos. Requisitos de latência e tempo de resposta sublinham a eficiência do processamento na borda, Edge AI, com médias de tempo de resposta notavelmente inferiores em comparação com a abordagem em nuvem.

Em síntese, a escolha entre as abordagens analisadas depende das necessidades específicas relacionadas aos requisitos funcionais e não funcionais da aplicação. À medida que a tecnologia continua evoluindo, estas reflexões formam uma base sólida para a tomada de decisões informadas no desenvolvimento de aplicações baseadas em IA, com implicações significativas para os negócios do século 21.

REFERÊNCIAS

A. RAMESH, P. DHARIWAL, A. NICHOL, C. CHU, AND M. CHEN. **Hierarchical Text-Conditional Image Generation with CLIP Latents**.

ASLANPOUR, M. S. et al. **Serverless Edge Computing: Vision and Challenges**. 2021 Australasian Computer Science Week Multiconference. New York, NY, USA: ACM, 2021.

CHEN, X. et al. Efficient multi-user computation offloading for mobile-edge cloud computing. **ACM transactions on networking [a joint publication of the IEEE Communications Society, the IEEE Computer Society, and the ACM with its Special Interest Group on Data Communication]**, v. 24, n. 5, p. 2795–2808, 2016.

HU, Y.; HÉDÉ, P.; PATEL, M. **Mobile-Edge Computing – Introductory Technical White Paper**. [s.l: s.n.].

IFTIKHAR, S. et al. AI-based fog and edge computing: A systematic review, taxonomy and future directions. **Internet of Things**, v. 21, n. 100674, p. 100674, 2023.

RUSSO, R. **ProjetoTCC**. GitHub, 2023. Disponível em: <https://github.com/rossinirusso/ProjetoTCC>. Acesso em: 7 dec. 2023

SINGH, R.; KOVACS, J.; KISS, T. **To offload or not? An analysis of big data offloading strategies from edge to cloud**. 2022 IEEE World AI IoT Congress (AIIoT). IEEE, 2022.

YUNJIU, L.; WEI, W.; ZHENG, Y. Artificial Intelligence-Generated and Human Expert-Designed Vocabulary Tests: A Comparative Study. **SAGE Open**, v. 12, p. 215824402210821, 01 2022.

ZHANG, K. (KZ). **LLM: from Transformers to ChatGPT**.