Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Ferramenta Orientada a Algoritmos de *Deep Learning* para Identificação e Categorização de Blocos de Informação em Faturas de Energia

Autor: Renan Welz Schadt

Orientador: Prof. Dr. Maurício Serrano

Brasília, DF

2023

Renan Welz Schadt

Ferramenta Orientada a Algoritmos de *Deep Learning* para Identificação e Categorização de Blocos de Informação em Faturas de Energia

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Maurício Serrano

Coorientadora: Profa. Dra. Milene Serrano

Brasília, DF

2023

Renan Welz Schadt

Ferramenta Orientada a Algoritmos de *Deep Learning* para Identificação e Categorização de Blocos de Informação em Faturas de Energia/ Renan Welz Schadt. – Brasília, DF, 2023-

110 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Maurício Serrano

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. RetinaNet. 2. Faturas. I. Prof. Dr. Maurício Serrano. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Ferramenta Orientada a Algoritmos de *Deep Learning* para Identificação e Categorização de Blocos de Informação em Faturas de Energia

CDU 02:141:005.6

Renan Welz Schadt

Ferramenta Orientada a Algoritmos de *Deep Learning* para Identificação e Categorização de Blocos de Informação em Faturas de Energia

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Brasília, DF, 18 de Dezembro de 2023:

Prof. Dr. Maurício Serrano
Orientador

Profa. Dra. Milene Serrano
Coorientadora

Prof. Dr. Tiago Alves da Fonseca
Convidado 1

Prof. Dr. Renato Coral Sampaio
Convidado 2

Brasília, DF
2023

Agradecimentos

Agradeço à minha família, por sempre me incentivar nos estudos desde muito cedo e me prover a melhor educação possível durante toda a vida.

Agradeço aos meus queridos amigos, pela companhia, risadas e incentivo durante toda minha jornada acadêmica.

Agradeço ao meu namorado, pelo incentivo durante a realização e ajuda no processo de revisão do trabalho.

Agradeço à empresa onde trabalho, por liberar os dados que permitem a realização deste TCC.

Agradeço aos meus orientadores, Maurício e Milene Serrano, por toda a dedicação, paciência e orientação providas para realizar o trabalho.

Resumo

Faturas são documentos que relatam uma transação comercial. Seu processamento em massa é uma realidade em diversas empresas, porém, a maior parte desse processamento ainda é feito de forma manual. Abordagens automáticas são raras, seja pela imprevisibilidade dos *layouts* nas faturas, ou pelo seu conteúdo com informações sensíveis. Dentro do mercado de energia brasileiro existem mais de 100 distribuidoras emitindo um conjunto incontável de *layouts* distintos de fatura. O presente trabalho tem como objetivo criar uma ferramenta para fazer a identificação e categorização de blocos de informação relevantes em faturas de energia. Para isso, o sistema utiliza uma rede neural para detecção de objetos (RetinaNet) em conjunto com uma ResNet-50 para extração dos mapas de características. O processo de treinamento ocorre em um conjunto de mais de 10 mil faturas de duas distribuidoras de energia, em um número limitado de blocos (tabela de faturamento, impostos, leitura, informações do cliente e número da instalação). Ao final da pesquisa foi possível analisar a eficiência da implementação, através de métricas como mAP e se a solução também é válida para outras distribuidoras de energia.

Palavras-chave: RetinaNet. fatura de energia. detecção de objeto. ResNet-50.

Abstract

Invoices are documents that report a business transaction. Its mass processing is a reality in several companies, however, most of this processing is still done manually. Automatic approaches are rare, either because of the unpredictability of layouts on invoices, or because of their content with sensitive information. Within the Brazilian energy market there are more than 100 distributors issuing an uncountable set of different invoice layouts. This work aims to create a tool to identify and categorize relevant blocks of information on energy bills. For this, the system use a neural network for object detection (RetinaNet) together with a ResNet-50 for feature map extraction. The training process take place in a set of more than 10 thousand invoices from two energy distributors, in a limited number of blocks (billing table, taxes, reading, customer information and installation number). At the end of the research, it was possible to analyze the efficiency of the implementation, through metrics such as mAP and whether the solution is also valid for other energy distributors.

Key-words: RetinaNet. energy invoice. object detection. ResNet-50.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ANEEL	Agência Nacional de Energia Elétrica
ANN	*Artificial Neural Network*
AP	*Average Precision*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
DL	*Deep Learning*
EGE	Empresa de Gerenciamento Energético
FN	*False Negative*
FP	*False Positive*
GPU	*Graphics Processing Unit*
mAP	*Mean Average Precision*
ML	*Machine Learning*
OCR	*Optical Character Recognition*
R-CNN	*Region-based Convolutional Neural Network*
TCC	Trabalho de Conclusão de Curso
TP	*True Positive*

Sumário

1 Introdução

Este capítulo apresenta a Contextualização da problemática, expondo ao longo do texto os principais aspectos relacionados à análise, ao reconhecimento e à extração de documentos, além de outros assuntos que circundam o tema deste trabalho. Adicionalmente, será exposta a Justificativa, bem como descrita a Questão de Pesquisa e apresentados os Objetivos, Geral e Específicos. Por fim, tem-se a Organização da Monografia em capítulos.

1.1 Contextualização

Documentos são meios de comunicação entre indivíduos ou instituições. Eles são criados para anunciar novos produtos, resumir acordos, publicar resultados de pesquisas, ou apenas despertar, prometer, informar, questionar, comandar e convencer outros. Com a intenção de auxiliar esses processos de trabalho, é necessário tornar os dados dos documentos úteis. A compreensão do documento desempenha um papel importante na extração eficaz de seus dados, que pode ser realizada de forma manual ou automática (DENGEL; SHAFAIT, 2014).

Document Analysis and Recognition (DAR) é um subcampo do processamento automático de imagens digitais que visa converter documentos para facilitar a sua modificação, o seu armazenamento e a sua transmissão de forma a extrair informações compreensíveis por humanos e codificá-las em formato legível por máquina (NAGY, 2000). Nesse contexto, uma das primeiras tecnologias a emergir foi o reconhecimento óptico de caracteres (*Optical Character Recognition* – OCR) (YAO et al., 2022). Na época de sua popularização, no começo dos anos 90, esperava-se que um computador pudesse ler qualquer documento. Entretanto, apesar da sua evolução constante desde a sua criação, ficou evidente que o progresso seria mais lento do que o esperado, e que a vasta diversidade de aplicações e necessidades tornaria impossível confiar apenas na identificação aprimorada para o reconhecimento de um único caractere (DOERMANN, 2014).

Com o avanço das pesquisas na área, o problema da análise e do reconhecimento de documentos foi dividido em três etapas, sendo: pré-processamento, análise de layout ou segmentação de objetos e reconhecimento de objetos (MARINAI, 2008).

1. O pré-processamento visa melhorar a qualidade das imagens;

2. A segmentação de objetos ou análise de layout permite identificar os objetos básicos no documento ou página. Divide a imagem do documento em regiões com um conteúdo homogêneo, e atribui um significado a essas regiões, e

3. O reconhecimento de objetos é o nível de processamento mais baixo, e lida com a identificação do conteúdo presente em cada uma das regiões encontradas.

Nas últimas décadas, a diminuição do custo de aquisição, armazenamento e processamento de documentos e a crescente utilização de algoritmos utilizando redes neurais deram origem a muitas soluções inovadoras para diferentes tarefas relacionadas ao processamento de documentos (MARINAI; GORI; SODA, 2005).

Os algoritmos de redes neurais (ex. redes neurais artificiais, redes neurais convolucionais, dentre outras) funcionam de maneira inspirada no cérebro humano. Sendo assim, utilizam uma estrutura que contém neurônios artificiais ligados uns aos outros e dispostos em camadas. As ligações entre os neurônios (pesos) são ajustadas de acordo com os exemplos que recebem, ou seja, as redes neurais aprendem sendo treinadas. Por esse motivo, esses algoritmos possuem a capacidade de realizar tarefas que antes só eram realizadas por pessoas, como classificar um documento e reconhecer visualmente suas informações importantes. A área de *Deep Learning* (DL) engloba a maior parte dos algoritmos de redes neurais. A palavra *deep* (profundo) refere-se à quantidade de camadas da rede (profundidade), já *learning* (aprendizado) compreende o processo de aprendizagem utilizando redes neurais (YAO et al., 2022), (MARINAI; GORI; SODA, 2005) e (ELGENDY, 2020).

Um documento apresenta-se de muitas formas, com layouts complexos; ou com tipografia e semântica inconsistentes; ou ainda ausentes de regras. É impossível treinar um sistema para lidar com todas essas variações. Tais situações levaram pesquisadores a se concentrarem em pequenos subconjuntos de problemas, de modo a ter uma solução focada em um determinado tipo de documento (DOERMANN, 2014).

Dessa forma, para o desenvolvimento do software atrelado à pesquisa deste trabalho, será utilizado somente um tipo de documento, no caso, faturas de energia. Uma fatura é um registro comercial usado para descrever e detalhar produtos vendidos ou serviços prestados. Pode ser disponibilizada em papel ou em um documento digital, como um PDF. Uma fatura também indicará o valor devido, a data de vencimento e outras informações relevantes, relacionadas ao ramo de negócio em que ela está inserida (ADOBE, 2022). No caso do ramo de energia, a fatura é carregada de informações específicas de seu contexto, demandadas pela Agência Nacional de Energia Elétrica (ANEEL) às distribuidoras (ANEEL, 2020).

1.2 Justificativa

O mercado de energia brasileiro é dividido em geração, transmissão e distribuição. Em suma, as geradoras produzem a energia, as transmissoras a transportam até as subestações nos grandes centros consumidores, e as distribuidoras levam a energia até as

unidades consumidoras (ANEEL, 2020). As faturas elétricas são expedidas pelas mais de cem distribuidoras que operam no país. Cada distribuidora, portanto, usa um layout de fatura, não havendo um documento padronizado e estruturado entre as diferentes distribuidoras no que se refere à expedição dessas faturas.

O processamento de faturas é uma tarefa diária na maioria das empresas. Nesse sentido, o volume em que elas são processadas depende muito do tipo de negócio de uma empresa. Quando existe uma carga grande de documentos, processá-los de forma manual exige um gasto humano elevado, além de ser uma atividade laboriosa e repetitiva (DENGEL; SHAFAIT, 2014).

Ao adotar uma solução automatizada, é possível extrair com rapidez e precisão as informações da fatura, de modo que o tempo de processamento é bastante reduzido com economia concomitante de trabalho (SHI; CUI; XIAO, 2020).

Considerando as justificativas apresentadas, a finalidade deste trabalho é o desenvolvimento de uma ferramenta que auxilie no tratamento de informações específicas de uma fatura de energia, atuando na parte de identificação dos diversos blocos de informação e categorização desses blocos com respeito ao conteúdo que carregam.

1.3 Questão de Pesquisa

Diante do exposto, este trabalho propõe-se a responder a seguinte questão de pesquisa: **É possível desenvolver uma ferramenta que identifica e categoriza automaticamente os blocos de informações relevantes dentro de uma fatura de energia?**

1.4 Objetivos

Visando responder à questão de pesquisa, este trabalho procurará atingir alguns objetivos, organizados nas seções a seguir em: Objetivo Geral e Objetivos Específicos.

1.4.1 Objetivo Geral

Desenvolvimento de uma ferramenta, cujos principais propósitos são a identificação e a categorização, de forma automática, dos blocos de informações relevantes em uma fatura de energia. Pretende-se orientar por algoritmos de *Deep Learning*, visando uma solução mais adequada ao problema.

1.4.2 Objetivos Específicos

No intuito de cumprir com o Objetivo Geral, foram estabelecidos alguns Objetivos Específicos, de menor escopo cada, sendo:

- Estudo de algoritmos relacionados aos tópicos de análise e classificação de documentos, dentro das etapas de pré-processamento, segmentação de objetos e classificação de objetos;
- Estudo de tópicos de pesquisa correlatos, tais como: Redes Neurais Artificiais e *Deep Learning*;
- Definição de informações a serem extraídas e os formatos aceitos para representação das mesmas;
- Organização da base de dados, visando o treinamento dos algoritmos relacionados, já com foco na área de *Deep Learning*;
- Documentação do desenvolvimento da ferramenta, usando como base os referenciais teóricos, tecnológicos e metodológicos estabelecidos no trabalho, e
- Condução de uma análise de resultados da ferramenta, considerando uma amostra de faturas para teste.

1.5 Organização da Monografia

Este trabalho de conclusão de curso está divido nos seguintes capítulos:

Capítulo 2 - Referencial Teórico: apresenta as áreas, os conceitos e algoritmos relacionados ao tema trabalhado;

Capítulo 3 - Referencial Tecnológico: aborda os principais recursos tecnológicos necessários para o desenvolvimento do produto, englobando desde as ferramentas utilizadas até as bibliotecas e *frameworks*;

Capítulo 4 - Metodologia: especifica as escolhas metodológicas quanto à condução do trabalho, englobando a elaboração da pesquisa, o desenvolvimento da ferramenta e a análise dos resultados obtidos;

Capítulo 5 - Ferramenta para Identificação e Categorização de Blocos de Informação em Faturas de Energia: trata a proposta, com foco no detalhamento da ferramenta desenvolvida neste trabalho, procurando apresentar os requisitos, arquitetura, descrição dos modelos e algoritmos utilizados;

Capítulo 6 - Análise de Resultados: apresenta os resultados obtidos, através de métricas pré-estabelecidas, da ferramenta construída para o projeto, e

Capítulo 7 - Conclusão: apresenta o estado final do trabalho, apontando os objetivos que foram concluídos, contribuições e limitações da ferramenta e possíveis evoluções.

2 Referencial Teórico

Com o intuito de compreender os algoritmos que serão utilizados na solução, este capítulo aborda definições e conceitos acerca dos tópicos de interesse desse trabalho sendo: Conceitos Gerais (Aprendizado de Máquina, *Deep Learning* e Visão Computacional), Redes Neurais Artificiais, Redes Neurais Convolucionais e Redes Neurais Convolucionais para Detecção de Objetos.

A intenção é conferir um adequado embasamento para a condução da pesquisa, bem como para a elaboração da ferramenta pretendida, orientando-se pela literatura especializada. Por fim, têm-se as Considerações Finais do Capítulo.

2.1 Conceitos Gerais

A seção de conceitos gerais explica os três conceitos fundamentais do trabalho, Aprendizado de Máquina, *Deep Learning* e Visão Computacional, apresentando suas definições, seus usos e como eles se relacionam. Ao fim dessa exposição, é apresentado um fluxo padrão de aplicação de visão computacional.

2.1.1 Aprendizado de Máquina

> Aprendizado de Máquina (*Machine Learning* - ML) é um campo da ciência da computação em que as máquinas aprendem a executar tarefas para as quais não foram explicitamente programadas. (SAMUEL, 1959).

Os algoritmos de Aprendizado de Máquina abordam problemas que não podem ser resolvidos facilmente com uma fórmula matemática ou um conjunto de regras claras a serem implementadas em código. Tratam-se de problemas que as pessoas resolvem de maneira "intuitiva", ou seja, com tarefas que elas aprendem a resolver com o tempo e a experiência, como identificar um modelo de carro ou uma espécie de animal dentro de uma imagem (GOODFELLOW; BENGIO; COURVILLE, 2016).

Dessa forma, para desempenhar suas funções, sistemas de Aprendizado de Máquina são treinados com um grande conjunto de dados, aprendendo a partir deles. Esse aprendizado pode se dar de forma supervisionada ou não supervisionada. No aprendizado supervisionado, as saídas/soluções desejadas são fornecidas em conjunto com os dados de treinamento. Já no aprendizado não supervisionado, as saídas esperadas não são fornecidas (AURELIEN, 2023).

Segundo Goodfellow, Bengio e Courville (2016), o desempenho de um algoritmo de Aprendizado de Máquina depende da representação de dados que ele utiliza. Por exemplo, para definir se um avião pode pousar em um determinado aeroporto, uma seleção de variáveis pertinentes pode ser analisada, como a direção e a velocidade do vento, o comprimento da pista, o peso do avião, entre outros. Dentro da área de Aprendizado de Máquina, essas informações analisadas são chamadas de *features*. As *features*, pedaços de informação relevantes para solução do problema, são selecionadas manualmente por uma pessoa nos algoritmos clássicos de Aprendizado de Máquina; ou são encontradas automaticamente em uma série de algoritmos pertencentes ao conjunto de *Deep Learning*.

2.1.2 *Deep Learning*

De acordo com Trask (2019) e Voulodimos et al. (2018), *Deep Learning* (DL) é um subconjunto de Aprendizado de Máquina, utilizado para resolver tarefas práticas em vários campos, como visão computacional (imagem), processamento de linguagem natural (texto) e reconhecimento automático de fala (áudio). Ele permite modelos computacionais com múltiplas camadas de processamento para aprender e representar dados, que simulam a maneira como o cérebro percebe e compreende informações.

A habilidade de aprendizado dos neurônios biológicos inspirou os cientistas a criarem uma rede de neurônios conectados uns aos outros. Dessa forma, quando um número suficiente de seus sinais de entrada é ativado, cada neurônio artificial dispara um sinal para todos os neurônios aos quais está conectado. Assim, os neurônios têm um mecanismo muito simples no nível individual, mas com milhões desses neurônios empilhados em camadas e conectados entre si, cada neurônio é conectado a milhares de outros neurônios, gerando um comportamento de aprendizado. O conhecimento envolvido na construção de uma rede neural profunda, com múltiplas camadas é chamado de *Deep Learning* (ELGENDY, 2020).

> DL é uma solução que permite aos computadores aprender com a experiência e compreender o mundo em termos de uma hierarquia de conceitos, com cada conceito definido por meio de sua relação com conceitos mais simples. A hierarquia de conceitos permite que o computador aprenda conceitos complicados construindo-os a partir de outros mais simples. (GOODFELLOW; BENGIO; COURVILLE, 2016).

Entre os fatores que contribuíram para o crescimento da área de *Deep Learning*, estão o surgimento de grandes bases de dados rotuladas, de alta qualidade e disponíveis publicamente, juntamente com o fortalecimento das unidades de processamento gráficas (*Graphics Processing Unit* - GPU) que aceleram o processo de treinamento (VOULODIMOS et al., 2018).

2.1.3 Visão Computacional

Dentro do contexto deste trabalho, a temática foco é a utilização dos algoritmos de *Deep Learning* para resolver um problema de Visão Computacional (*Computer Vision* - CV), um campo que permite que computadores e sistemas observem padrões e objetos por meio da visão ou alguma entrada visual, construindo um modelo físico do mundo para que um sistema de aprendizado possa tomar as ações apropriadas. Graças à dramática evolução do poder computacional e da quantidade de dados disponíveis, os algoritmos de *Deep Learning*, base da visão computacional moderna, conseguiram alcançar um desempenho sobre-humano em muitas tarefas complexas de percepção visual, tal qual sistemas de reconhecimento facial e sistemas de classificação de objetos, como em centros de reciclagem, para separar automaticamente os diferentes tipos de materiais, entre outras tarefas (ELGENDY, 2020).

A inspiração para fazer os sistemas de visão computacional surgiu da natureza. No sistema de visão humano, são necessários dois componentes principais: um dispositivo sensor para simular a função do olho, e um algoritmo para simular a função cerebral na interpretação e na classificação do conteúdo da imagem (ELGENDY, 2020).

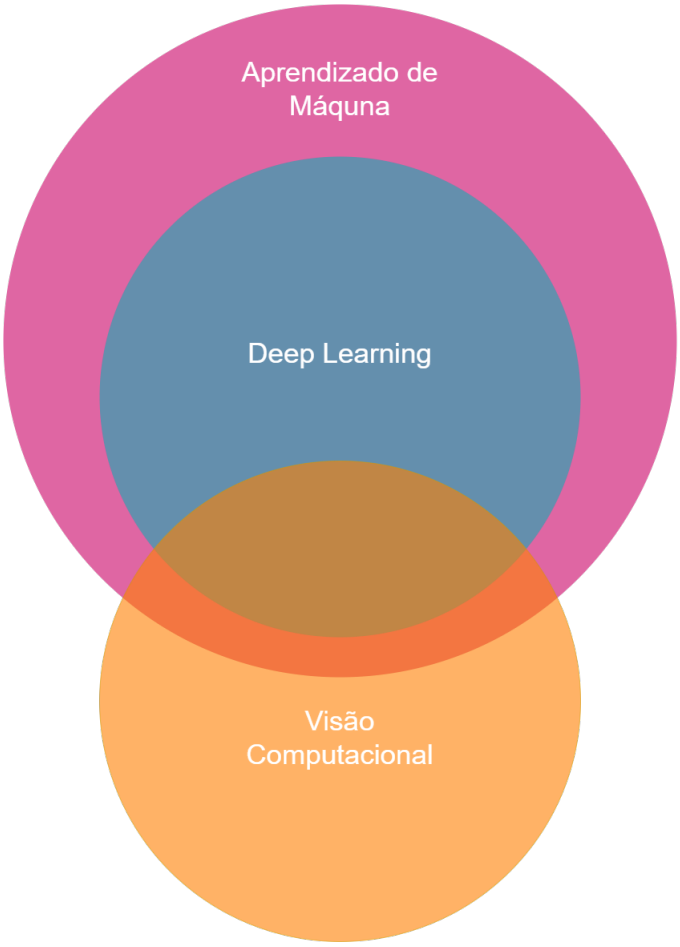
A Figura 1 mostra a relação entre os conceitos explicados nessa seção em forma de conjuntos utilizando um diagrama de Venn.

2.1.3.1 Fluxo Visão Computacional

Segundo Elgendy (2020), as aplicações de Visão Computacional variam, mas um sistema de visão típico usa uma sequência de etapas distintas para processar e analisar dados de imagem. Detalhes dessa sequência seguem nos passos a seguir:

1. **Entrada**: Um computador recebe entrada visual de um dispositivo como uma câmera. Essa entrada é normalmente capturada como uma imagem ou uma sequência de imagens, formando um vídeo. Uma imagem pode ser representada como uma função de duas variáveis x e y, que definem uma área bidimensional. Uma imagem digital é feita de uma grade de pixels. O pixel é o bloco de construção bruto de uma imagem. Toda imagem é composta por um conjunto de pixels em que seus valores representam a intensidade da luz que aparece em um determinado local da imagem. Os computadores veem imagens como matrizes. As imagens em tons de cinza podem ser representadas em uma matriz bidimensional, onde cada elemento representa a intensidade de brilho daquele determinado pixel. Uma imagem colorida tradicional tem três canais: vermelho, verde e azul. Ela é representada em uma matriz tridimensional, de forma que cada pixel possui seu nível de brilho em cada uma das três dimensões (vermelho, verde e azul).

Figura 1 – Diagrama de Venn relacionando conceitos



Fonte: Autor

2. **Pré-Processamento**: Cada imagem é enviada para algumas etapas de pré-processamento, cujo objetivo é tratá-la com a função de facilitar os processamentos posteriores. Esses tratamentos podem converter uma imagem colorida para escala de cinza, rotacionar, recortar, aumentar ou diminuir contraste, entre outros. Também faz parte do pré-processamento o uso de uma técnica chamada *data augmentation*, que permite criar mais imagens usando uma única como base, orientando-se pelos processos descritos anteriormente, com o objetivo de produzir mais imagens de treinamento para o aprendizado supervisionado.

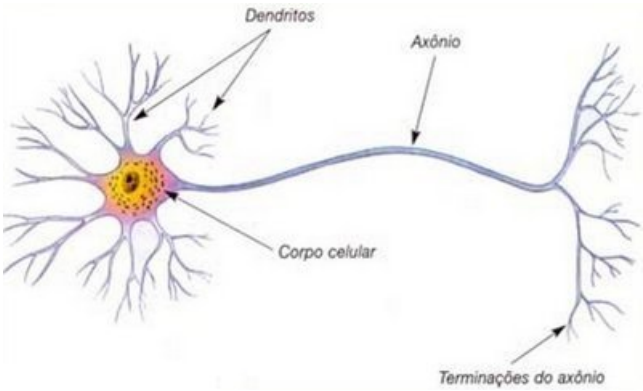
3. **Extração de Características (*features*)**: Captura os elementos que ajudam a definir os objetos. Geralmente, são informações que identificam linhas, arestas, formas, cor, entre outros. A saída desse processo é um vetor de *features*, que é uma lista de formas únicas que identificam o objeto.

4. **Classificação/Interpretação dos Resultados**: Analisa os recursos da etapa anterior para prever e classificar objetos.

2.2 Redes Neurais Artificiais

Um neurônio biológico tem três partes básicas: um corpo celular, e duas ramificações chamadas axônio e dendrito. Dentro do corpo celular, existe um núcleo, que controla as atividades da célula e contém seu material genético. O axônio se parece com uma longa cauda, e envia mensagens da célula (saída). Um dendrito se parece com o galho de uma árvore, e recebe mensagens para a célula (entrada). A Figura 2 mostra essas estruturas. Os neurônios comunicam-se quando a força total dos seus sinais de entrada excede um certo limite, enviando substâncias químicas chamadas neurotransmissores. A saída é então alimentada para outro neurônio, e assim por diante (ELGENDY, 2020).

Figura 2 – Representação de neurônio biológico



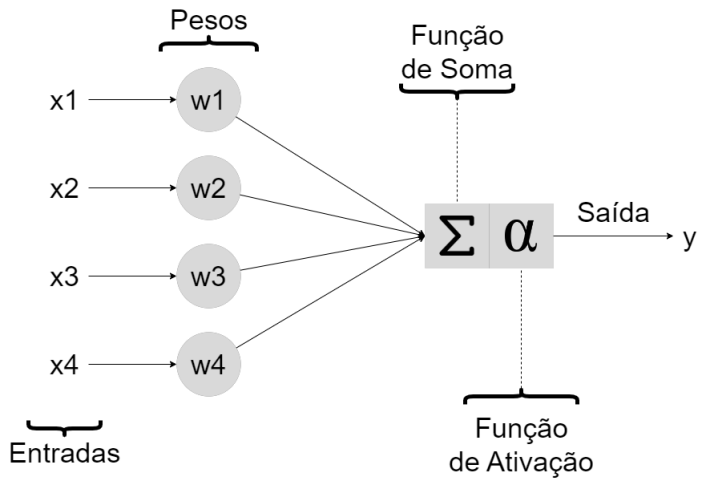
Fonte: (DUARTE, 2015)

Em uma rede neural artificial, os neurônios são alimentados com as entradas e com pesos designados para representar sua importância. Cálculos realizados dentro do neurônio utilizando as entradas, pesos e algumas funções adicionais definem a saída que será passada adiante na rede (ELGENDY, 2020).

Um dos modelos mais básicos de neurônio artificial é chamado de *perceptron*, conforme mostra a Figura 3. Ele contém os seguintes elementos:

- **Entradas**: Representação numérica das características extraídas;
- **Pesos**: Atribui um grau de importância a cada uma das entradas;
- **Função de Soma (combinação linear)**: Multiplica cada entrada com seu respectivo peso; faz uma soma de todos os resultados, e adiciona um novo termo (bias);
- **Saída**: Um *perceptron* contém uma única saída, que é definida de acordo com a função de ativação, podendo ser binária (0 ou 1); um conjunto pré-definido de números (1 a 10, 0 a 100, dentre outros), ou um número real, e
- **Função de Ativação**: Dado o resultado da função de soma, a função de ativação tem o objetivo de definir o formato da saída gerada.

Figura 3 – *Perceptron*, um dos modelos de neurônio artificial



Fonte: Autor

Um *perceptron* é alimentado com uma instância de dado. Por vez, cada instância de dado produz um resultado, que pode estar próximo ou distante do resultado esperado. Essa distância é chamada de erro. A regra de aprendizado de um *perceptron* realiza ajustes nos pesos para reforçar conexões que ajudam a reduzir o erro (AURELIEN, 2023).

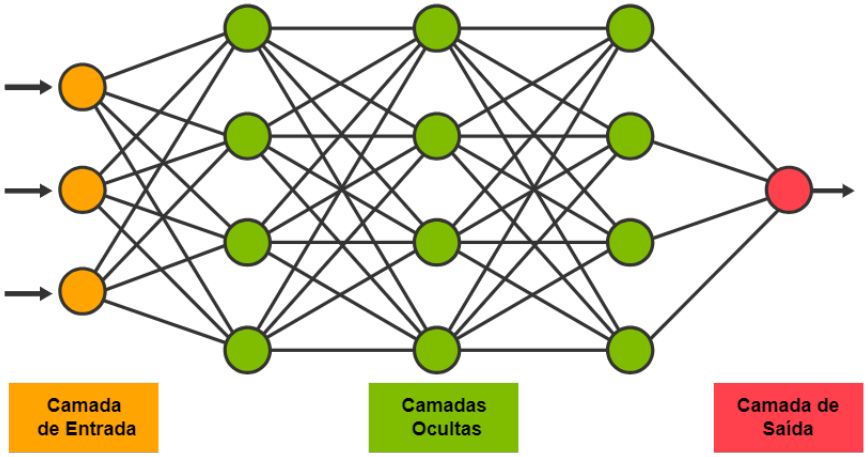
Um *perceptron* é uma função linear, de modo que uma única unidade dele não consegue categorizar ou prever algum valor de forma satisfatória em casos mais complexos. Para resolver esses problemas, existem as redes neurais artificais (*Artificial Neural Networks* - ANNs), que consistem em muitos *perceptrons* estruturados em camadas e conectados com pesos. Essa estrutura também é chamanda de *multilayer perceptron* (MLP), conforme ilustrado na Figura 4. As camadas de entrada e saída da ANN comportam-se de forma semelhante ao *perceptron*, enquanto as camadas ocultas extraem recursos cada vez mais abstratos a respeito da entrada fornecida (ELGENDY, 2020).

Essas camadas são chamadas de ocultas, porque seus valores não são fornecidos nos dados; em vez disso, o modelo deve determinar quais conceitos são úteis para explicar as relações nos dados observados. Uma característica da ANN é que ela é uma rede completamente conectada, ou seja, todos os neurônios (nós) de uma determinada camada têm pesos (arestas) conectados aos nós da próxima camada (GOODFELLOW; BENGIO; COURVILLE, 2016).

Segundo Chollet (2021), dentro da estrutura de um *perceptron*, a função de ativação restringe à saída gerada. Para o funcionamento correto da ANN, deve-se utilizar funções adequadas ao problema, tanto nas camadas ocultas, quanto na camada de saída.

Nas camadas ocultas, normalmente, são utilizadas as funções de tangente hiperbólica (*tanh*) ou a função de unidade linear retificada (*Rectified Linear Unit* - ReLU). Enquanto a primeira função gera valores entre -1 e 1, a segunda retorna a própria en-

Figura 4 – Esquemático de rede neural artificial



Fonte: TIBCO (2021), traduzido por autor

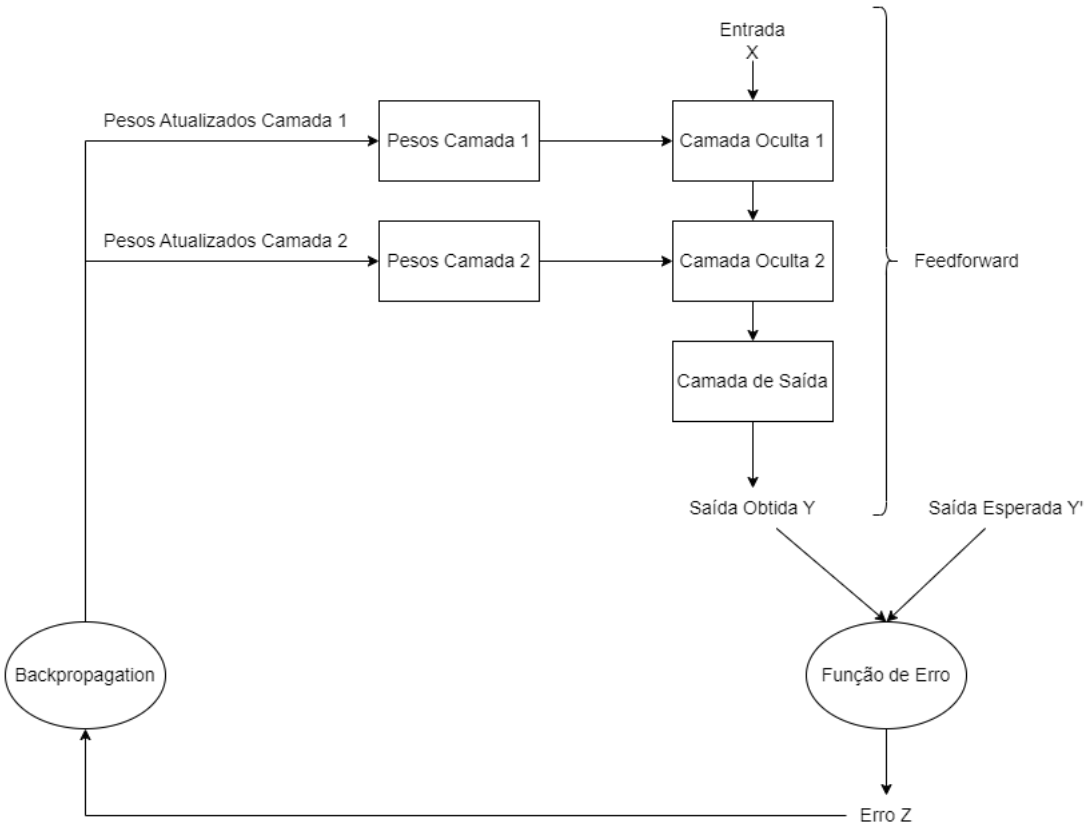
trada, caso ela seja positiva, e zero, caso ela seja negativa (ELGENDY, 2020).

A função de ativação da camada de saída reflete a natureza do problema. É possível escolher uma função que retorna zero ou um para problemas do tipo verdadeiro/falso (*Heaviside Step Function*); retorna uma probabilidade, ou seja, números de zero a um (*Sigmoid Function*), ou até retornar múltiplas probabilidades que somam um (100 porcento) em problemas de classificação multiclasse, em que é preciso apontar a probabilidade para cada classe (*Softmax Function*) (CHOLLET, 2021).

O processo do dado sair da camada de entrada, passar pelas camadas ocultas executando as funções de soma e ativação, e gerar um valor final na camada de saída é chamado de *feedforward*. Após esse processo, é calculado o erro, sendo a diferença entre a saída obtida e a saída esperada. Esse erro pode ser obtido com diferentes funções, dependendo do tipo de problema, como a função do erro quadrático médio (*Mean Squared Error* – MSE) ou a função logarítmica (*Cross-entropy*). Após o erro ser encontrado, os pesos devem ser ajustados de forma que em uma próxima iteração o resultado obtido seja mais próximo do resultado esperado. Esse processo é chamado de retropropagação (*backpropagation*). A Figura 5 ilustra a repetição dos processos de *feedforward*, visando encontrar o erro e a retropropagação que representam o treinamento de uma rede neural (CHOLLET, 2021).

Uma classe de algoritmos muito utilizada para se fazer o processo de retropropagação é a classe de algoritmos de gradiente descendente. Eles realizam a derivada do erro em relação a cada peso contido na rede. Dessa forma, é possível saber se é preciso aumentar ou diminuir o peso com o objetivo de reduzir o erro. Para utilizar o gradiente descendente, é preciso determinar a taxa de aprendizado (*learning rate*), sendo essa correspondente ao valor em que o peso será alterado a cada nova iteração (ELGENDY, 2020).

Figura 5 – Fluxo de treinamento de rede neural artificial



Fonte: Autor, inspirado em Chollet (2021)

> Inicialmente, os pesos da rede recebem valores aleatórios, de modo que o resultado obtido normalmente é distante do resultado esperado. Mas com cada exemplo que a rede processa, os pesos são ajustados pouco a pouco na direção correta, e o erro diminui. Este é o ciclo de treinamento, que, repetido um número suficiente de vezes (normalmente dezenas de iterações em milhares de exemplos), produz valores de peso que minimizam a função de erro (CHOLLET, 2021).

2.3 Redes Neurais Convolucionais

As redes neurais convolucionais (*Convolutional Neural Network* - CNN) são um tipo especializado de rede neural para processamento de dados, que possuem um formato multidimensional, como imagens, as quais podem ser consideradas como uma grade 2D de pixels (GOODFELLOW; BENGIO; COURVILLE, 2016). A influência para criação da rede neural convolucional vem da estrutura do córtex visual do cérebro humano, que tem como objetivo processar informações visuais. As CNNs têm sido bem-sucedidas em aplicações práticas de visão computacional (BARBOSA et al., 2021).

Segundo Liu et al. (2017) alguns aspectos do funcionamento de uma CNN assemelham-se a de uma ANN. Sendo assim, os pesos são iniciados aleatoriamente e ajustados durante

o treinamento da rede. Na sequência, são aplicadas funções de ativação e erro, e o processo de retropropagação atualiza os pesos gradativamente. Porém, a CNN possui duas grandes vantagens comparada à ANN em problemas relacionados à imagem:

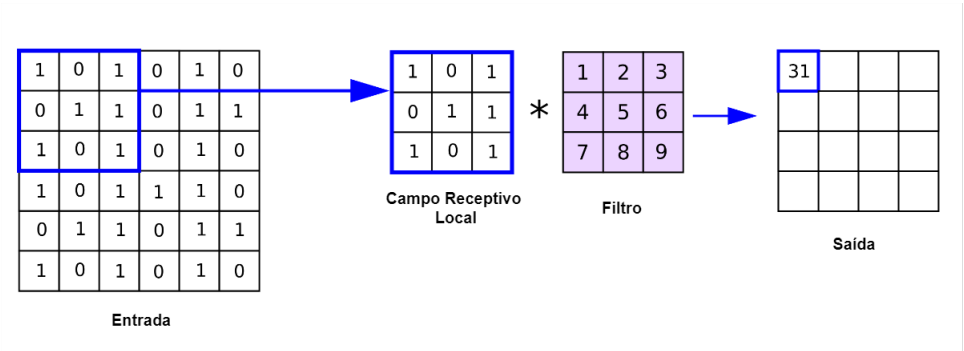
- O uso de uma camada convolucional para extrair as *features* (pedaços da imagem que ajudam a compreendê-la), em oposição a uma camada completamente conectada como nas ANNs. Desta forma, o número de pesos é diminuído, reduzindo assim a complexidade da rede, e
- É possível passar matrizes como entrada de uma CNN, enquanto uma ANN só permite vetores/listas, ou seja, em uma ANN, há uma perda considerável de informações por não relacionar de forma correta os pixels e suas posições.

Uma CNN contém tipicamente as seguintes camadas: convolucionais, de agrupamento (*pooling*), e completamente conectadas (densas). Cada camada tem sua função específica dentro da rede. Seguem outras considerações sobre essas camadas:

1. **Camada Convolucional**: A convolução é uma operação entre duas funções para produzir uma terceira função modificada. Utilizando uma matriz de entrada e filtros de convolução (*kernel*), ela produz mapas de características (*feature maps*) (ELGENDY, 2020).

   A Figura 6 mostra a entrada (que pode ser a própria matriz da imagem ou um mapa de características de alguma camada anterior); o campo receptivo local, que é o pedaço da entrada que será multiplicado ao filtro (produto escalar matricial), e a saída gerada. O filtro desloca-se de forma a navegar por toda a entrada, completando o *feature map* (saída).

Figura 6 – Processo de convolução



Fonte: Autor

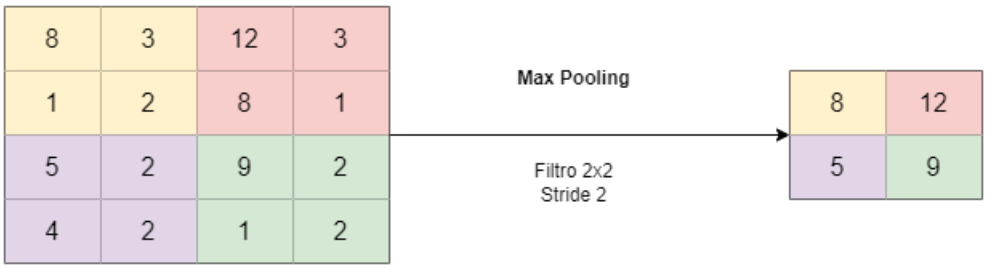
Os valores contidos no filtro são os pesos da camada convolucional, sendo iniciados aleatoriamente, e depois ajustados para melhorar o desempenho da rede no processo

de retropropagação. O tamanho do filtro (2x2, 3x3, dentre outros) é um parâmetro definido pelo desenvolvedor, juntamente com o *stride* (passo), que é o número de pixels que o filtro se desloca para formar o *feature map*. Quanto maiores os filtros e valores de *stride*, menores serão os *feature map* gerados (LIU et al., 2017).

Da mesma forma que as camadas ocultas de uma ANN podem conter vários neurônios, uma camada convolucional pode conter vários filtros. Cada filtro adicionado produz seu próprio *feature map*, a Figura 6 contém um filtro e produz uma matriz 4x4. Com dois filtros a saída gerada seria uma matriz 4x4x2 (ELGENDY, 2020).

2. **Camada de Agrupamento**: Executa a operação de redução de amostragem para diminuir a dimensão dos *feature maps* (altura e largura) que serão passados para a próxima camada. Normalmente, aplica-se de forma alternada com camadas de convolução. Um exemplo de função de agrupamento é representar os valores de uma determinada janela de agrupamento pelo seu valor máximo (*max pooling*), como mostra a Figura 7. Também utiliza os parâmetros de tamanho do filtro e *stride* (LOPEZ; MATTOS, 2021).

Figura 7 – Processo de agrupamento (*pooling*)

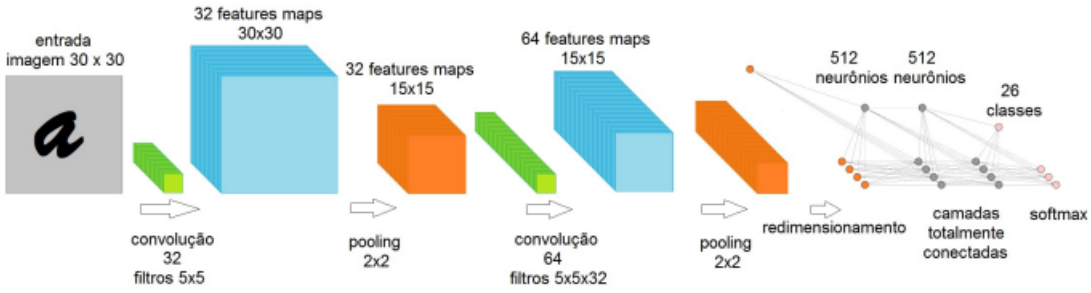


Fonte: Autor

3. **Camada Densa**: Conjunto de camadas completamente conectadas usadas para interpretar as *features* obtidas pelas camadas anteriores, e gerar uma ou múltiplas saídas, ou seja, uma ANN. Para que *features maps* obtidos na camada de convolução ou agrupamento sejam repassados como entrada, na camada densa, deve-se passar pelo processo de vetorização (*flattening*), que basicamente transforma uma estrutura multidimensional (2D, 3D, dentre outras) em um vetor unidimensional (EBERMAM; KROHLING, 2018).

A Figura 8 ilustra a arquitetura completa de uma CNN utilizada para reconhecer a letra contida na imagem. Ela contém duas camadas convolucionais (a primeira contendo 32 filtros 5x5, e a segunda 64 filtros 5x5); duas camadas de agrupamento (utilizando filtros 2x2), e três camadas densas (duas ocultas com 512 neurônios cada, e uma de saída com 26 neurônios, representando as letras do alfabeto) (EBERMAM; KROHLING, 2018).

Figura 8 – Rede Neural Convolucional



Fonte: (EBERMAM; KROHLING, 2018)

Segundo Chollet (2021), para realizar o processo de treinamento e testar a acurácia de uma rede neural, seja ela ANN ou CNN, é preciso dividir as entradas em conjuntos. Uma divisão comum é:

- **Conjunto de Treinamento**: Dados usados para treinar a rede neural;
- **Conjunto de Validação**: Dados usados para fornecer uma estimativa da acurácia da rede, enquanto são modificados os hiperparâmetros do modelo (taxa de aprendizado, tamanho dos filtros, número de filtros, *stride*, entre muitos outros parâmetros que podem ser personalizados na rede), e
- **Conjunto de Teste**: Dados usados para fornecer uma estimativa final imparcial da rede treinada.

Ao verificar a acurácia da rede (predições corretas / total de predições), é comum deparar-se com dois problemas. Um desses problemas é chamado *overfitting*, em que a rede tem boa acurácia na etapa de treinamento, mas falha em generalizar as entradas alimentadas a ela posteriormente, ou seja, a rede "decorou" os dados de treinamento e não aprendeu as regras subjacentes contidas neles (ELGENDY, 2020).

Uma solução existente para evitar o *overfitting* é a técnica de *dropout*. Essa técnica consiste em desligar aleatoriamente um percentual de neurônios da rede, ou seja, a cada nova iteração um conjunto distinto de neurônios será desligado. Uma rede não pode depender exclusivamente de um conjunto reduzido de seus neurônios para tomar uma decisão. Utilizando o *dropout*, é possível evitar que alguns neurônios tenham um grau de importância muito elevado para definir a saída gerada Ebermam e Krohling (2018).

Outro problema que pode ocorrer é o *underfitting*, onde a rede tem uma acurácia ruim, tanto nos dados de treinamento, como nos dados de validação. Sendo assim, a rede foi incapaz de aprender adequadamente as *features* que conectam entrada e saída. Esse tipo de problema pode ser sintoma de uma rede simplificada demais para resolver um problema complexo (ELGENDY, 2020).

2.4 Redes Neurais Convolucionais para Detecção de Objetos

As CNNs são excelentes na tarefa de classificação de objeto dentro de uma imagem, mas em muitas situações, em uma só imagem, existem vários objetos para se classificar. Adicionalmente, determinadas aplicações, como as utilizadas em veículos autônomos, não querem apenas obter quais objetos estão em uma imagem, mas também obter as posições ou coordenadas em que eles se encontram. Na visão computacional, refere-se a tais tarefas como detecção de objetos (ELGENDY, 2020).

Para detecção bem-sucedida de um objeto, ele precisa ser corretamente localizado (*object localization*) e corretamente classificado (*object classification*). A saída esperada de um algoritmo de detecção de objetos é uma caixa delimitadora (desenhada utilizando quatro coordenadas dadas pelo sistema) ao redor do objeto identificado com sua classe prevista (ELGENDY, 2020).

Segundo Elgendy (2020), um sistema de detecção de objetos contém os quatro componentes principais. A Figura 9 ilustra os passos a seguir:

1. **Geração de Regiões de Interesse (RoIs)**: Algoritmo ou modelo que define diversas caixas delimitadoras em torno de objetos de interesse na imagem. Cada caixa contém uma pontuação (*score*) que determina a probabilidade de haver um objeto de interesse nela. Caixas com uma pontuação abaixo de um determinado limiar (definido pelo desenvolvedor) são consideradas plano de fundo (*background*), e não passam para o próximo estágio. Caixas com pontuação acima do limiar são consideradas primeiro plano (*foreground*), e avançam para o estágio seguinte;

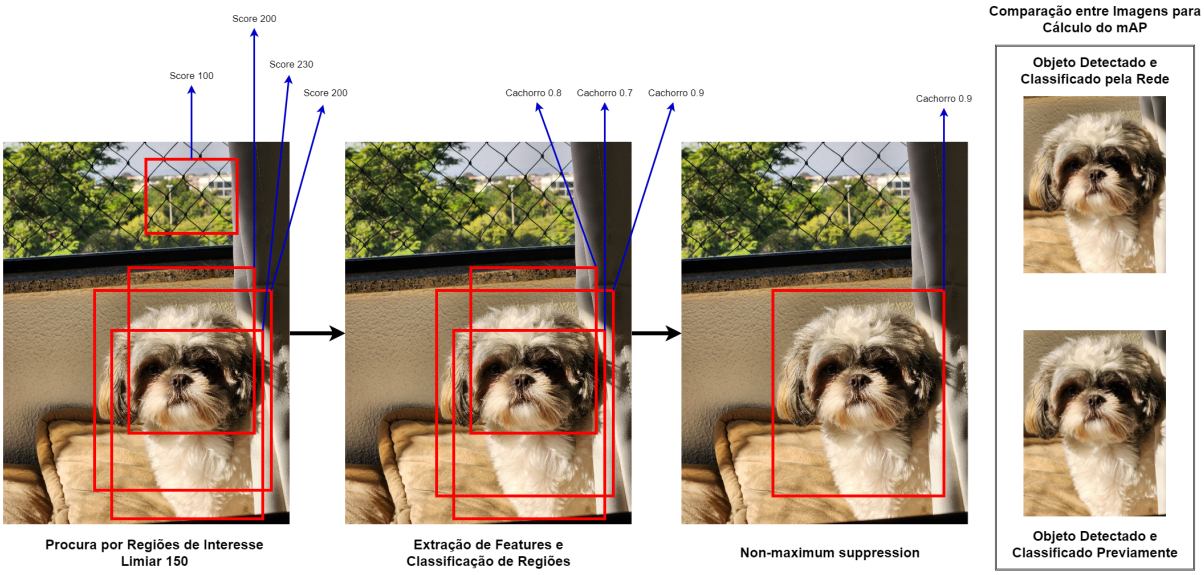
2. **Extração de *Features* e Previsões da Rede**: As regiões selecionadas na etapa anterior (caixas delimitadoras) passam por uma CNN para extrair suas *features* (traços, linhas, e outras formas intrinsecas àquela imagem), e realizar a classificação das imagens (detectar se um ou múltiplos objetos estão presentes e qual sua categoria);

3. ***Non-maximum Suppression* (NMS)**: O objetivo do algoritmo de NMS é selecionar a caixa delimitadora que se adequa de forma melhor ao objeto proposto, eliminando diversas caixas geradas na primeira etapa, e classificadas na segunda, que capturam o objeto de forma parcial ou de forma excessivamente ampla, e

4. **Métricas de Avaliação**: É possível avaliar a velocidade da rede através do número de quadros por segundo com que ela opera. Essa métrica é muito importante para sistemas de detecção por vídeo em tempo real. Também é possível avaliar a precisão da rede através de uma métrica chamada de *mean average precision* (mAP), que atribui à rede uma pontuação de 0 a 1, avaliando se as caixas delimitadoras estão corretas (comparando a delimitação esperada com a delimitação obtida no

algoritmo de intersecção sobre união), e verificando as métricas de *recall* e *precision* que comparam o número de verdadeiros positivos (caixa delimitadora dentro das medidas esperadas e correta identificação do seu conteúdo) com falsos positivos (caixa delimitadora não cobre efetivamente o objeto) e falsos negativos (objeto não localizado e classificado).

Figura 9 – Funcionamento de Rede Neural Convolucional para Detecção de Objetos



Fonte: Autor

Segundo Ren et al. (2017), uma das classes de algoritmos que resolve de forma mais adequada esse problema é a classe de *Region-based Convolutional Neural Networks* (R-CNNs). Uma das últimas evoluções dessa série de algoritmos é a Faster R-CNN, que obtém uma acurácia superior com tempo de treinamento e tempo de execução inferiores às propostas anteriores (R-CNN e Fast R-CNN). A arquitetura de uma Faster R-CNN compreende:

1. **Rede Convolucional para Extração de *Features***: Uma rede convolucional para extrair *features* a partir da imagem, gerando mapas de características (*feature maps*).

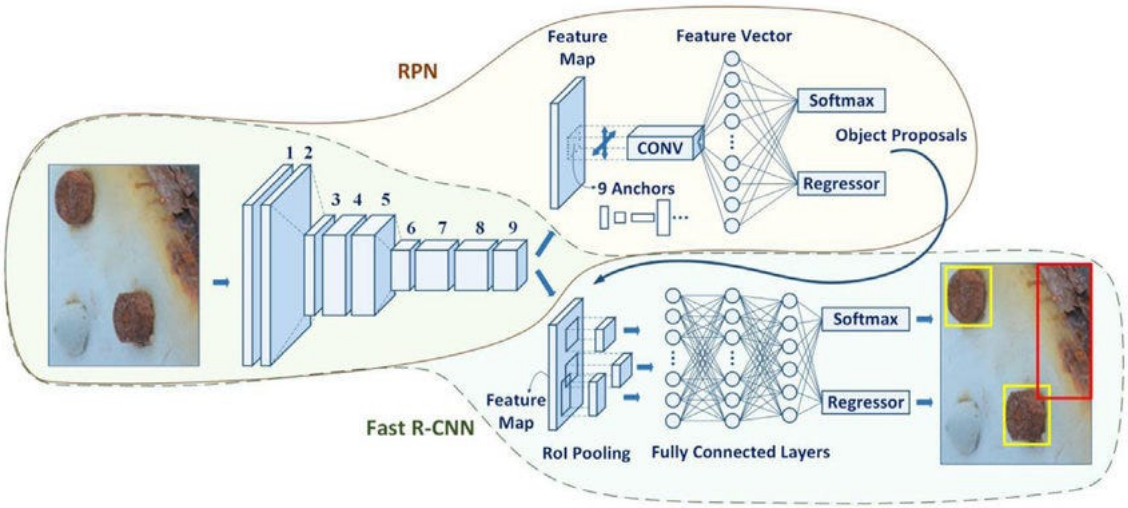
2. **Rede de Proposta de Regiões** (*Region Proposal Network* - RPN): Cria RoIs a partir dos mapas de características gerados anteriormente. Cada RoI, tem seu tamanho e sua posição definidos, juntamente a informação se há ou não há objeto dentro daquele RoI. Utiliza um sistema de caixas de ancoragem (*anchor box*), juntamente com um filtro que desliza sobre a imagem. Para cada região da imagem, várias caixas de ancoragem são criadas (de tamanhos e aspectos distintos), e seus valores são ajustados através de deltas (coordenadas x e y, além de altura e largura) de forma a se adequar melhor a um determinado objeto;

3. **Camada de Agrupamento RoI**: Gera caixas de tamanho fixo a partir dos mapas de características, antes de alimentar as RoIs para as camadas totalmente conectadas. Usa o *max pooling* para gerar um mapa de característica com tamanho fixo (altura × largura), e

4. **Camada de Saída Dupla**: Os RoIs de tamanho fixo são passados para um conjunto de camadas completamente conectadas, que se repartem em duas camadas de saída, sendo uma de classificação (*softmax*), que tem a função de reportar as probabilidades de categorias para cada RoI, e uma camada de regressão de caixa delimitadora, para prever deslocamentos em relação ao RoI original.

Uma das grandes vantagens de uma Faster R-CNN em relação às abordagens anteriores é o uso de redes neurais e *Deep Learning* de ponta-a-ponta no processamento, desde à proposta de regiões até as etapas finais de classificação. Além disso, há o uso de uma só rede convolucional que fornece insumos (*feature maps*) tanto para a RPN quanto para a camada de agrupamento RoI. A Figura 10 ilustra a arquitetura de uma Faster R-CNN, não foi encontrada uma imagem equivalente em português, porém, os termos apresentados nela foram mencionados no texto em tanto em português como em inglês, possibilitando a sua compreensão (REN et al., 2017).

Figura 10 – Arquitetura de uma Faster R-CNN



Fonte: (KHAZRI, 2021)

A Faster R-CNN é caracterizada como uma rede de detecção de objetos que funciona em dois estágios: O primeiro faz a proposta de regiões e o segundo faz a classificação e localização dos objetos dentro das regiões identificadas. As desvantagens destas redes são de que seu treinamento e execução de predições são lentos (em torno de 2 segundos por predição). Também existem as redes que operam em um estágio, realizando a classificação e a identificação das caixas delimitadoras sem ter gerado propostas de regiões

previamente, elas costumam trocar ter acurácia inferior aliada a uma velocidade superior às redes com dois estágios. Existem diversas redes desse tipo, como: YoLo, SSD e RetinaNet (LIN et al., 2017).

Entre essas redes a RetinaNet se destaca por oferecer uma performance equiparável e por vezes melhor que a Faster R-CNN, aliada a treinamento/detecção rápidos de uma rede de um só estágio. A Figura 11 ilustra o average precision (AP) da RetinaNet comparado a Faster R-CNN (LIN et al., 2017).

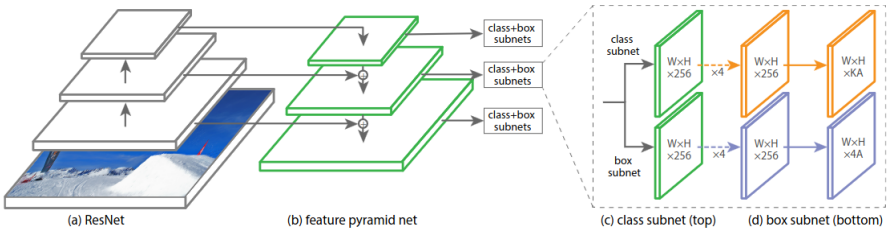
Figura 11 – Comparação RetinaNet e Faster R-CNN na base de dados CoCo

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [31]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [4]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [33]	Inception-ResNet-v2 [38]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [30]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [8]	DarkNet-19 [8]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9], [10]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [10]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Fonte: (LIN et al., 2017)

Segundo Lin et al. (2017), a estrutura de uma RetinaNet utiliza uma CNN (normalmente ResNet) que recebe as imagens e gera uma séries de convoluções sucessivas, gerando diversos *feature maps* em cada camada de convolução. Os *feature maps* de algumas camadas de convolução selecionadas são repassados a outra CNN, denominada Feature Pyramid Network (FPN), a partir de cada camada da FPN rodam duas outras sub-redes para identificar os objetos e encontrar suas caixas delimitadoras, como mostra a Figura 12.

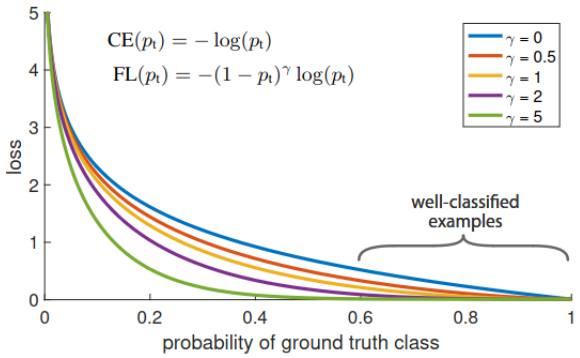
Figura 12 – Arquitetura da RetinaNet



Fonte: (LIN et al., 2017)

Segundo Lin et al. (2017), a principal inovação que da RetinaNet em relação a outros detectores de um estágio foi o uso de uma nova função de erro denominada Focal Loss, através dela objetos em que a rede tem um alto nível de confiança contribuem de forma muito pequena pra soma do erro total, diferente de funções de erro como a Cross-Entropy, em que a soma de milhares de itens de alta confiança acaba alterando muito o erro total, como mostra a Figura 13.

Figura 13 – Comparação de diferentes versões da Focal Loss com a Cross-Entropy Loss (em azul)



Fonte: (LIN et al., 2017)

2.5 Considerações Finais do Capítulo

Este capítulo teve a função de apresentar os conceitos e modelos que serão utilizados para o desenvolvimento da ferramenta proposta. Primeiramente, foram elucidados os conceitos de aprendizado de máquina, *deep learning* e visão computacional, para fornecer uma visão macro das áreas de interesse desse trabalho, antes de partir diretamente para os algoritmos e modelos existentes dentro dessas áreas.

Para compreender o funcionamento de uma RetinaNet, que será o modelo utilizado na ferramenta proposta, foi necessário entender os conceitos e funcionamento de neurônios artificiais, redes neurais artificias e redes neurais convolucionais. Foi possível compreender como é processo de aprendizado de uma rede neural, e como essas redes trabalham com imagens. O conhecimento dos tópicos anteriormente abordados também é indispensável, visto que eles são pequenos componentes que fazem parte da arquitetura final de uma rede para detecção de objetos.

3 Referencial Tecnológico

Este capítulo apresenta as ferramentas e tecnologias que foram e serão utilizadas no desenvolvimento deste trabalho. A seção de Suporte e Gestão aborda técnicas de gestão para condução deste trabalho, como o versionamento e a metodologia ágil. Por conseguinte, a seção da Elaboração da Monografia confere o ferramental utilizado para apoio à pesquisa e à escrita. Depois, tem-se a seção dedicada ao Desenvolvimento da Ferramenta, cujo viés é mais técnico, e destaca linguagem de programação e bibliotecas adicionais. Por fim, têm-se as Considerações Finais do Capítulo.

3.1 Suporte e Gestão ao Desenvolvimento

Para um bom controle e uma adequada organização na construção deste trabalho, foram utilizadas ferramentas de controle de versionamento, como Git e GitHub. Adicionalmente, tornam-se relevantes ferramentas e técnicas das metodologia Kanban e Scrum. Além de requisitos computacionais (de *hardware*) do projeto. Ressalta-se que uma visão metodológica mais abrangente será acordada no Capítulo 4 - Metodologia. A ideia, neste capítulo, é um olhar mais focado no arcabouço tecnológico.

3.1.1 Ferramentas de Versionamento - Git 2.41 e GitHub 3.9

Git é um sistema de controle de versão, distribuído, gratuito e disponível em diversas plataformas de desenvolvimento. Tem a funcionalidade, desempenho, segurança e flexibilidade que a maioria das equipes e desenvolvedores individuais precisam. Também conta com um ótimo suporte da comunidade e uma vasta base de usuários. A documentação inclui livros, tutoriais e sites dedicados (ATLASSIAN, 2016).

O GitHub é um software de controle de versão que utiliza Git, sendo, declaradamente, a maior comunidade de código aberto do mundo, hospedando mais de 372 milhões de repositórios (privados e públicos), incluindo códigos e suas documentações. Possui diversas funcionalidades, tais como: histórico visual de contribuições, rastreamento de problemas, notificações, painéis de status, ferramentas de documentação, integração e *deploy* (GITHUB, 2023).

3.1.2 Ferramenta de Apoio Metodológico - Trello 2.12.3

As metodologias ágeis representam a forma de modelagem e o gerenciamento no desenvolvimento de software. Neste trabalho, será adotada uma metodologia híbrida utilizando Kanban e Scrum, selecionando elementos do Scrum que são válidos para um time

de uma pessoa, como o conceito de *sprint* e mesclando com o Quadro Kanban que fornece uma ótima visualização do fluxo de trabalho (SUTHERLAND; SCHWABER, 2013).

O Kanban é um tipo de metodologia ágil simplificada que faz o uso de uma *dashboard* do trabalho e de seu fluxo, dividindo-o em subtarefas simultâneas, colocando-as no Quadro Kanban. A visualização é usada para representar cada etapa do processo de forma clara, para que toda a equipe compreenda a situação atual, permitindo aos membros maior participação. A divisão mais comum de quadros para armazenar as tarefas é entre: A Fazer, Fazendo e Feito (AHMAD; MARKKULA; OIVO, 2013).

No intuito de viabilizar um acompanhamento adequado, via Quadro Kanban, será utilizada a ferramenta Trello (TRELLO, 2023). Trello é uma ferramenta que permite especificar o Quadro Kanban, configurando-o conforme as necessidades de projeto. Popular e de fácil uso, o Trello ajudará o autor ao longo de toda a realização do trabalho.

3.1.3 Recursos Computacionais - Linux Mint 21.1 e GPU

Para realização do trabalho, será utilizado um computador com um sistema operacional baseado em Linux (LINUX, 2023), devido à ampla comunidade do mesmo, e pelo fato de ser de código aberto e conferir suporte às mais diversas ferramentas, linguagens e bibliotecas de desenvolvimento de software.

A distribuição utilizada será o Linux Mint (MINT, 2023) que fornece uma interface de fácil navegação e aprendizado, bom suporte a *drivers* (para garantir o funcionamento de periféricos, placas de rede, vídeo, entre outros), e uma central de atualizações de software controlada pelo usuário (garantindo maior estabilidade ao sistema). O Linux Mint utiliza o gerenciador de pacotes *Advanced Package Tool* (APT), na versão 2.4.9 (APT, 2023). Uma interface usada para instalar, atualizar e remover ferramentas, programas e bibliotecas.

O processo de treinamento de redes neurais demanda muito do *hardware* do computador. Por esse motivo, o computador utilizado no processo de treinamento terá 16GB de RAM, um processador (CPU), modelo AMD Ryzen 7 5700X, com 8 núcleos, 16 *threads*, e uma placa de vídeo (GPU) dedicada, modelo Nvidia RTX 4070, com 12GB de memória de vídeo (vRAM). Dessa forma, o processamento utilizando Python (PYTHON, 2023) e as bibliotecas TensorFlow (TENSORFLOW, 2023) e Keras (KERAS, 2023), descritas na seção Desenvolvimento da Ferramenta, será otimizado.

3.2 Elaboração da Monografia

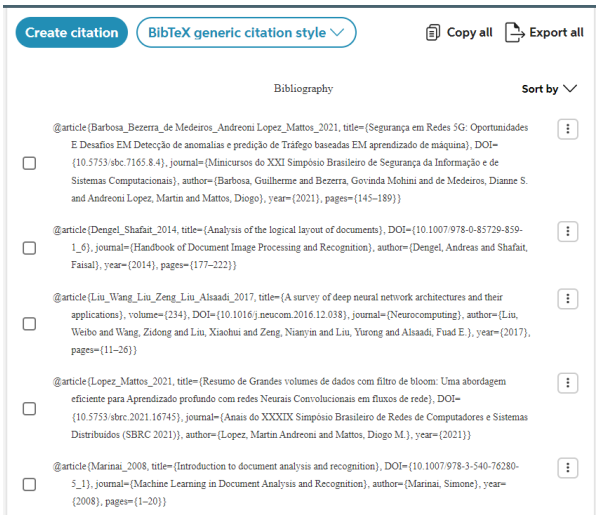
Esta seção aborda as ferramentas utilizadas em termos de pesquisa, bem como para a escrita desta monografia com os resultados obtidos até o momento.

3.2.1 Ferramenta de Auxílio Bibliográfico - BibMe

O BibMe (BIBME, 2023) é um gerador de referências e citações *online*. É possível referenciar livros, artigos, sites, entre outros recursos em diversos formatos, incluindo o formato BibTeX usado no LaTeX. A base de obras disponíveis no software é bem completa, e a procura pode ser feita pelo título, DOI, ISBN e outros parâmetros. O software também consegue armazenar uma lista de citações de um trabalho.

A Figura 14 mostra alguns artigos sendo mantidos na biblioteca sobre os tópicos de interesse deste trabalho.

Figura 14 – BibMe: Ferramenta usada para facilitar a construção da bibliografia



Fonte: Autor

3.2.2 Editor de Texto - Overleaf

O Overleaf (OVERLEAF, 2023) é uma ferramenta de escrita, revisão e publicação de texto, focada em escrita científica usando o formato LaTeX. O uso do Overleaf facilita a padronização do texto, gestão de bibliografia, hiperlinkagem com elementos não textuais (figuras, tabelas, dentre outras).

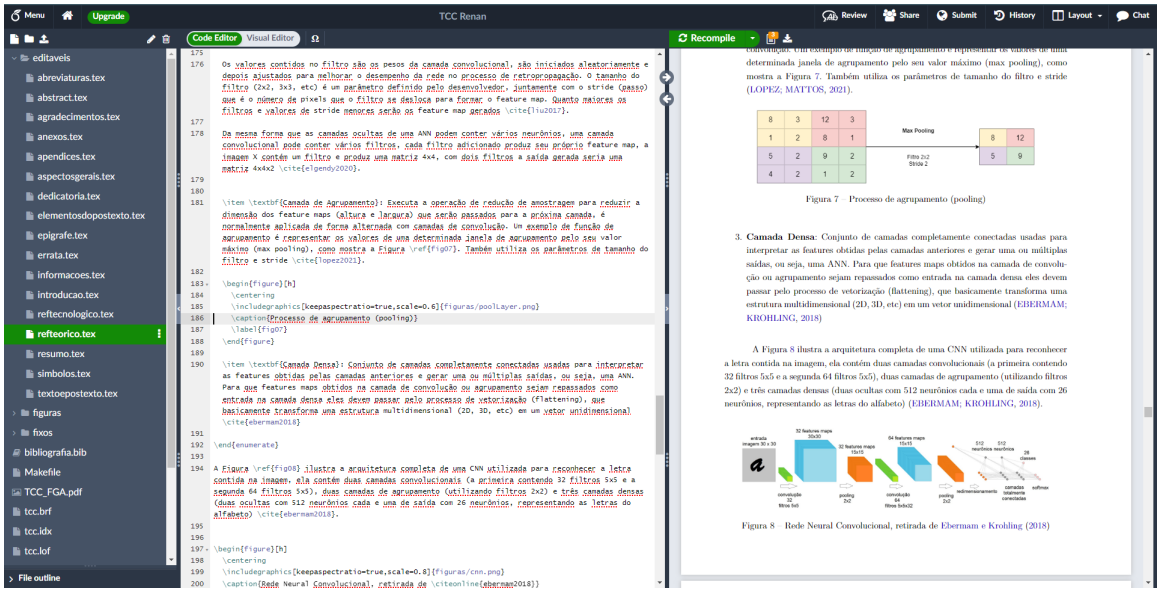
A Figura 15 ilustra a edição dessa monografia sendo realizada com apoio do Overleaf.

3.2.3 Ferramenta de Construção de Diagramas - Diagrams

O Diagrams é um software gratuito e de código aberto, usado para criar diversos tipos de esquemáticos, como diagramas de classe, de rede, de arquitetura, fluxogramas, mapas mentais, entre outros.

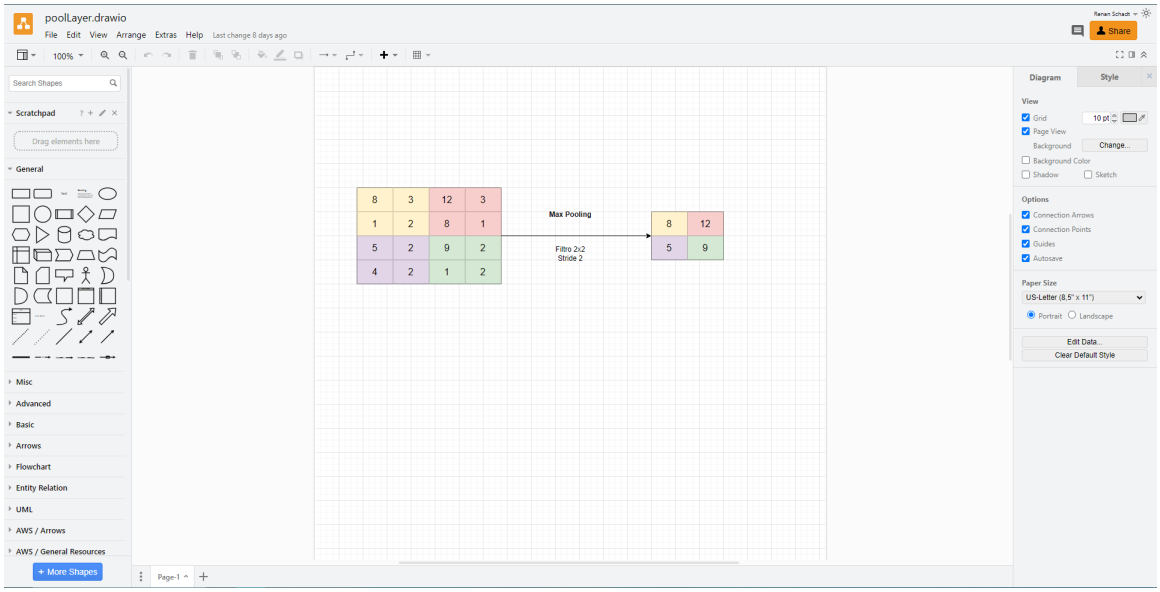
A Figura 16 apresenta o ambiente de elaboração de modelos sendo utilizado para modelagem de uma das imagens utilizada nesta monografia.

Figura 15 – Overleaf: Ferramenta de edição de texto LaTeX



Fonte: Autor

Figura 16 – Diagrams: Ferramenta de construção de esquemáticos e diagramas



Fonte: Autor

3.3 Desenvolvimento da Ferramenta

Esta seção aborda as tecnologias, as bibliotecas e os *frameworks* utilizados para o desenvolvimento da ferramenta proposta.

3.3.1 Linguagem de Programação - Python 3.10

Python (PYTHON, 2023) é uma linguagem de programação popular e de fácil aprendizado. Possui estruturas de dados eficientes e uma abordagem simples e eficaz para programação orientada a objetos. Oferece código conciso e legível para facilitar a compreensão e manutenção do software.

Os benefícios que tornam o Python uma das melhores opções para projetos que utilizam Aprendizado de Máquina incluem simplicidade e consistência; acesso a ótimas bibliotecas e estruturas de Aprendizado de Máquina; flexibilidade; independência de plataforma; boa documentação, e uma ampla comunidade de desenvolvedores para tirar dúvidas e consultar boas práticas (AURELIEN, 2023), (TRASK, 2019) e (CHOLLET, 2021).

3.3.2 Biblioteca Adicional - TensorFlow 2.13

O TensorFlow (TENSORFLOW, 2023) é uma plataforma de ponta-a-ponta para Aprendizado de Máquina, baseada em Python, desenvolvida principalmente pelo Google. Assim como o NumPy (NUMPY, 2023), o objetivo principal do TensorFlow é permitir que engenheiros e pesquisadores manipulem expressões matemáticas sobre matrizes ou tensores multidimensionais.

O TensorFlow é muito mais do que uma única biblioteca, é uma plataforma, que abriga um vasto ecossistema de componentes, alguns desenvolvidos pelo Google e alguns desenvolvidos por terceiros. O TensorFlow permite a execução de cálculos em vários tipos de dispositivos, incluindo CPU (*central processing unit*) e GPU (*graphics processing unit*). O processamento pela GPU é priorizado para acelerar os processos de treinamento (CHOLLET, 2021).

A documentação do TensorFlow (TENSORFLOW, 2023) define que Keras é a API de alto nível da plataforma. Ela fornece uma interface acessível e altamente produtiva para resolver problemas de Aprendizado de Máquina, com foco em *Deep Learning*. O Keras abrange todas as etapas do fluxo de trabalho envolvendo redes neurais, desde o processamento de dados e ajuste de hiperparâmetros, até a implantação. Foi desenvolvido com foco em permitir prototipagem e experimentação rápida. Todo usuário do TensorFlow deve usar as APIs Keras por padrão, são poucos casos de uso que exigem as APIs do TensorFlow de baixo nível.

3.3.3 Biblioteca Adicional - Keras 2.13

Keras é uma API de *Deep Learning* para Python, construída sobre o TensorFlow, que fornece uma maneira conveniente de definir e treinar vários tipos de modelos de *Deep*

Learning. Com o Keras, é possível ter acesso à escalabilidade e processamento otimizado em GPU e CPU do TensorFlow (KERAS, 2023).

Keras é conhecido por priorizar a experiência do desenvolvedor. Ele segue as melhores práticas para reduzir a carga de desenvolvimento: oferece fluxos de trabalho consistentes e simples, minimiza o número de ações necessárias para casos de uso comuns e fornece *feedback* claro e acionável em caso de erro do usuário. Isso torna o Keras fácil de aprender para um iniciante, e altamente produtivo para um especialista (CHOLLET, 2021).

A Figura 17 provê um exemplo básico de uso do Keras, os conceitos listados foram explicados no Capítulo 2 - Referencial Teórico. Os seguintes elementos estão contidos no exemplo:

1. **Base de Dados**: É utilizada a base de dados aberta MNIST, que contém 70 mil imagens (60 mil para treinamento e 10 mil para teste) de carecteres numéricos escritos a mão, o objetivo da rede neural é classificar corretamente esses caracteres em 10 categorias (zero até nove);

2. **Pré-processamento**: O pré-processamento feito nas imagens consiste em dividir os valores das matrizes (tamanho 28x28) por 255, para que os valores contidos nela fiquem entre 0 e 1;

3. **Definição da Rede Neural**: A rede neural criada (ANN) contém uma camada de vetorização (*flattening*) para transformar as matrizes da entrada em vetores; duas camadas completamente conectadas com funções de ativação para realizar a classificação das imagens e um *dropout* para evitar o *overfitting*;

4. **Compilação**: O comando *compile* prepara a rede para etapa de treinamento, definindo alguns hiperparâmetros aplicáveis a ela, como a função de erro, função de retropropagação e métricas utilizadas;

5. **Treinamento**: O comando *fit* realiza o treinamento da rede, recebendo as imagens e suas *labels* como entrada, além do número de *epochs* (iterações de *feedforward* e retropropagação que passam por todas instâncias de treinamento) em que a rede será treinada, e

6. **Avaliação**: Avalia o erro e métricas definidas pelo usuário (ex: acurácia, *recall*, *precision*, mAP, entre outros) utilizando os dados separados para teste.

O Keras é capaz de gerar uma representação textual da rede (através do comando *summary*), como mostra a Figura 18. Nela é possível visualizar as camadas da rede (convolucionais, de agrupamento e completamente conectadas); os procedimentos adicionais

Figura 17 – Criação de uma rede neural utilizando o Keras

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Fonte: (TENSORFLOW, 2023)

realizados (vetorização e *dropout*) e o total de parâmetros (pesos) que podem ser modificados pela rede em seu processo de treinamento.

Figura 18 – Representação de uma rede neural utilizando o Keras

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 16)        208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 32)        2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 64)          8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               512500
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5010
=================================================================
Total params: 528,054
Trainable params: 528,054
Non-trainable params: 0
_________________________________________________________________
```

Fonte: (ELGENDY, 2020)

Durante o processo de treinamento, como mostra a Figura 19, o Keras é capaz de exibir o tempo percorrido, o erro e métricas de avaliação personalizadas (definidas pelo usuário), além de outros parâmetros que o usuário pode configurar na função *fit*. Dessa forma, é possível ter um *feedback* rápido a respeito do desempenho da rede, antes mesmo dela finalizar o processo completo de treinamento.

Figura 19 – Processo de treinamento uma rede neural utilizando o Keras

```
Epoch 1/10
1875/1875 [==============================] - 5s 2ms/step - loss: 0.4960 - accuracy: 0.8246
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3721 - accuracy: 0.8663
Epoch 3/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3355 - accuracy: 0.8779
Epoch 4/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3085 - accuracy: 0.8874
Epoch 5/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2936 - accuracy: 0.8918
Epoch 6/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2758 - accuracy: 0.8978
```

Fonte: (TENSORFLOW, 2023)

3.3.4 Biblioteca Adicional - pdf2image 1.16.3

O pdf2image (PDF2IMAGE, 2023) é uma biblioteca disponível para Python, com o objetivo de transformar arquivos PDFs em imagens (JPEG, PNG, entre outros formatos). Utiliza como base outra biblioteca, chamada poppler (POPPLER, 2023), usada para ler, renderizar e manipular arquivos PDF.

O pdf2image será utilizado para converter as faturas (PDF) em imagens (JPEG), que depois de tratadas podem ser fornecidas à rede neural.

3.3.5 Biblioteca Adicional - Pillow 10.1.0

O Pillow (PILLOW, 2023) é uma biblioteca disponível para Python, para visualização e modificação de imagens. Ela permite redimensionamento, inversão de cores, recortes, troca de formato, mudança no brilho, contraste, entre outros.

O Pillow será utilizado nas etapas de pré-processamento das imagens, fazendo ajustes nas suas características (como tamanho, brilho e esquema de cores), de forma a padronizá-las, para que sejam utilizadas nas redes neurais.

3.3.6 Biblioteca Adicional - matplotlib 3.8.0

O matplotlib (MATPLOTLIB, 2023) é uma biblioteca disponível para Python, com objetivo de criar gráficos estáticos ou interativos. É possível definir uma série de parâmetros relativos ao formato do gráfico, legenda, cores, entre outros.

O matplotlib será utilizado para mostrar gráficos contendo métricas de desempenho da rede neural. Mostrando a evolução de métricas como erro, *precision*, *recall* e mAP no processo de treinamento.

3.4 Considerações Finais do Capítulo

Este capítulo buscou apresentar as ferramentas e tecnologias utilizadas para apoiar o desenvolvimento desta pesquisa, desde a realização da escrita, definição de metolodogia e documentação do projeto, até a linguagem e bibliotecas principais utilizadas para desenvolver a ferramenta proposta.

As ferramentas foram definidas com base em alguns critérios como: tamanho da comunidade; avaliação dos usuários; funcionalidades propostas e qualidade da documentação. Para trabalhar com redes neurais foi escolhido o Python como linguagem de programação e o TensorFlow/Keras como bibliotecas relevantes (CHOLLET, 2021).

A Tabela 1 apresenta as ferramentas utilizadas, suas respectivas versões, propósitos e referências (documentação ou site oficial). Um traço foi colocado em ferramentas onde não foi possível encontrar a numeração da versão atual.

Tabela 1 – Principais ferramentas utilizadas no trabalho

Ferramenta	**Versão**	**Propósito**	**Referência**
Git	2.41	Realizar versionamento do código	(GIT, 2023)
GitHub	3.9	Realizar versionamento do código	(GITHUB, 2023)
Trello	2.12.3	Acompanhar o projeto via Quadro Kanban	(TRELLO, 2023)
Linux Mint	21.1	Fornecer ambiente para o desenvolvimento da ferramenta proposta	(MINT, 2023)
BibMe	-	Armazenar e construir referências no formato BibTeX	(BIBME, 2023)
Overleaf	-	Escrever, editar e publicar monografia	(OVERLEAF, 2023)
Diagrams	21.6.2	Desenhar esquemáticos, fluxogramas e diagramas	(DIAGRAMS, 2023)
Python	3.10	Desenvolver ferramenta proposta	(PYTHON, 2023)
TensorFlow	2.13	Construir, treinar e avaliar rede neural	(TENSORFLOW, 2023)
Keras	2.13	Construir, treinar e avaliar rede neural	(KERAS, 2023)
matplotlib	3.8.0	Mostrar gráficos contendo métricas de desempenho	(MATPLOTLIB, 2023)
pdf2image	1.16.3	Transformar arquivos PDFs em imagens	(PDF2IMAGE, 2023)
Pillow	10.1.0	Realizar tratamento em imagens	(PILLOW, 2023)

Fonte: Autor

4 Metodologia

Este capítulo descreve a metodologia aplicada para a elaboração deste trabalho. Primeiramente, tem-se a seção de Classificação da Pesquisa, introduzindo o conceito de pesquisa e detalhando-a em relação a sua abordagem, natureza, objetivos e procedimentos. Depois, a Metodologia de Pesquisa Bibliográfica detalha como foi o processo de consulta da literatura especializada. Na sequência, tem-se a Metodologia de Desenvolvimento, relatando a metodologia híbrida utilizada para guiar o desenvolvimento da ferramenta. Já a Metodologia de Análise de Resultados, detalha o processo de pesquisa-ação e avaliação de resultados obtidos.

Posteriormente, tem-se o Fluxo de Atividades, detalhando as atividades realizadas no Trabalho de Conclusão de Curso. Por fim, têm-se as Considerações Finais do Capítulo.

4.1 Classificação da Pesquisa

Segundo Gerhardt e Silveira (2009) e Gil (2010), a metodologia é o estudo do método (caminho em direção a um objetivo), sua função é estabelecer um conjunto de regras e procedimentos para realizar uma pesquisa (procedimento racional e sistemático que tem como objetivo fornecer respostas aos mais diferentes tipos de problemas). A metodologia é fundamental para construção do conhecimento científico, que é um conhecimento objetivo e passível de replicação, usado para representar ou explicar determinados fenômenos. Quando em domínio público esse conhecimento pode evoluir através das contribuições de outros pesquisadores.

Segundo Gil (2010), a pesquisa científica é atividade nuclear da ciência, e ela pode ser classificada em relação a sua abordagem, natureza, objetivos e procedimentos. A pesquisa científica realizada neste trabalho busca obter o embasamento (conceitual e técnico) necessário para resolver o problema de identificação e classificação de áreas de interesse em faturas de energia, de modo que a pesquisa fundamenta a solução (ferramenta) proposta. Os itens a seguir procuram apresentar um resumo dessa classificação para o caso da pesquisa em andamento neste trabalho:

- **Abordagem**: A pesquisa é qualitativa, buscando aprofundar o conhecimento de um determinado fenômeno, sem se preocupar em quantificar valores ou com a representação numérica do que foi pesquisado;
- **Natureza**: A pesquisa é aplicada, pois tenta solucionar um problema existente dentro de empresas que precisam processar quaisquer tipos de faturas em larga

escala;

- **Objetivos**: A pesquisa realizada é do tipo exploratória, ou seja, busca descobrir novos conceitos e ideias para entender o fenômeno pesquisado. São úteis para diagnosticar situações ou explorar alternativas para resolução de um problema, e
- **Procedimentos**: Para desenvolver uma pesquisa é fundamental a escolha de um método de pesquisa. Este trabalho usa os métodos de pesquisa bibliográfica e pesquisa-ação. A pesquisa bibliográfica representa um levantamento feito principalmente através de livros, artigos e sites. A função desta pesquisa é analisar materiais novos e antigos para adquirir conhecimento científico no assunto. Já a pesquisa-ação caracteriza-se pela participação do pesquisador na situação investigada, coletando dados, propondo soluções e analisando resultados.

4.2 Metodologia de Pesquisa Bibliográfica

Conforme citado na seção Classificação da Pesquisa, um dos procedimentos utilizados para realização do trabalho foi a pesquisa bibliográfica. Uma vez que o tema e a questão de pesquisa foram definidos, o autor buscou melhorar seu conhecimento no assunto coletando artigos disponíveis na base de dados Scopus, que contém artigos nas mais diferentes áreas do conhecimento vindos de diferentes periódicos.

Para fazer a busca de artigos relevantes ao tópico em estudo é necessária a formulação de *strings* de busca. Uma *string* de busca é um termo construído com uma palavra ou um conjunto de palavras do assunto a ser analisado, que serão procuradas no título, resumo ou palavras-chave dos artigos disponíveis na base de dados.

Para construção da *string* de busca se usam os parênteses para definir ordem de precedência das operações, aspas duplas para definir que determinados termos devem ser encontrados de maneira sequencial e dois operadores lógicos, AND e OR. O operador lógico AND garante que os resultados apresentados vão conter os termos tanto à sua direita como à sua esquerda. Já o operador lógico OR garante que os resultados apresentados vão conter os termos à direita ou os termos à esquerda. As *strings* de busca, a quantidade de resultados obtidos inicialmente e o objetivo de cada *string* usada estão relatados a seguir:

- **document AND "automatic processing" - 225 resultados**: Usada para obter conhecimento a respeito da extração de informações de documentos em um contexto geral, sem especificar o tipo de documento. Através dessa *string* foram encontradas ótimas fontes como *Analysis of the Logical Layout of Documents* e *Handbook of Document Image Processing and Recognition*. Porém, os resultados ainda são muito generalistas por não especificar o tipo de documento (podem ser artigos, documentos históricos, livros, entre outros);

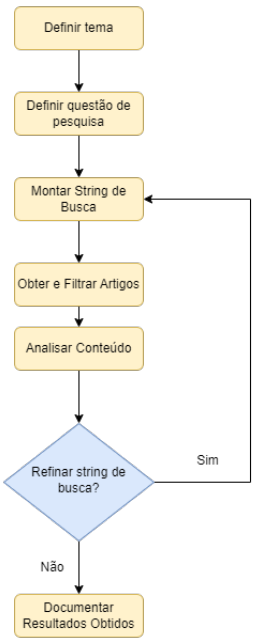
- **invoice AND classification - 154 resultados**: Usada para verificar as soluções para um dos contextos mais comuns de processamento automático de faturas, classificar seu domínio, ou simplesmente verificar se um documento é uma fatura;

- **invoice AND segmentation - 62 resultados**: Usada para verificar as soluções utilizadas quando há necessidade de analisar o conteúdo das faturas e encontrar informações relevantes como tabelas, imagens, datas, códigos identificadores, entre outros, e

- **invoice AND (segmentation OR classification) AND ("deep learning" OR "convolutional neural network") - 42 resultados**: Analisando os resultados obtidos nas duas *strings* anteriores verificou-se que os algoritmos de *Deep Learning* (principalmente algoritmos de redes neurais convolucionais) poderiam ser uma solução ao problema abordado pelo trabalho. Portanto, a última *string* de busca teve a função de encontrar trabalhos que utilizem esses algoritmos para realizar o processamento de faturas.

Os artigos foram submetidos a uma filtragem verificando sua data de publicação e quantas vezes eles foram referenciados. Foram selecionados para leitura do resumo apenas artigos posteriores a 2005 e com pelo menos um outro artigo o referenciando. A partir da leitura do resumo foram selecionados para leitura completa os artigos que poderiam contribuir com o desenvolvimento da pesquisa. A Figura 20 ilustra o fluxograma executado para obter artigos relevantes dentro da temática abordada.

Figura 20 – Fluxograma para obtenção de artigos e aprofundamento no tema



Fonte: Autor

Para melhorar o conhecimento do autor a respeito das redes neurais artificiais, seu funcionamento e diferentes tipos e usos, foram pesquisados livros dentro do contexto de *Deep Learning*, livros como *Deep Learning for Vision Systems* (ELGENDY, 2020) e *Deep Learning with Python* (CHOLLET, 2021) orientaram a construção do referencial teórico deste trabalho.

4.3 Metodologia de Desenvolvimento

Esta seção tem como objetivo apresentar a metodologia que será utilizada no desenvolvimento da ferramenta proposta. Ela irá guiar o ritmo de trabalho e o modo como o mesmo irá ocorrer. O autor selecionou duas metodologias, o Scrum e o Kanban, e mesclou elementos de ambos, conhecido na literatura como "scrumban" (LADAS, 2022).

4.3.1 Scrum

Segundo Sutherland e Schwaber (2013), "Scrum é um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível."

O Scrum consiste em um conjunto de papéis, eventos, artefatos e regras. Ele contém três pilares conceituais principais (SUTHERLAND; SCHWABER, 2013):

- **Transparência**: Permitir que todos os responsáveis tenham visão do fluxo, artefatos importantes e regras comuns ao projeto;
- **Inspeção**: Verificar os artefatos produzidos e o planejamento feito de forma a detectar inconsistências, e
- **Adaptação**: Encontradas inconsistências no processo de inspeção é possível adaptar o planejamento e os artefatos para entregar um melhor resultado final. Esse processo pode ocorrer tanto nas reuniões diárias, como nas reuniões de *sprint* e deve ser comunicado à equipe de forma a respeitar o primeiro princípio.

Os eventos do Scrum existem para alinhar toda a equipe de possíveis mudanças e permitir o compartilhamento de informações entre os desenvolvedores para evitar possíveis gargalos. Estes eventos estão listados a seguir (SUTHERLAND; SCHWABER, 2013):

- ***Sprint***: Ciclo de desenvolvimento no qual tarefas devem ser planejadas, feitas e entregues (de acordo com um conjunto de regras determinada definição de pronto), sua duração pode variar de uma semana até um mês dependendo da equipe. *Sprints* são construídas com um objetivo em mente, e ao fim dela o time deve ter agregado mais valor ao produto através de uma nova versão (incremento);

- ***Daily***: Reunião diária de alinhamento da equipe, em que cada desenvolvedor relata seu progresso no dia anterior e no dia atual. Usada para relatar dificuldades, realizar os processos de inspeção e adaptação, deve durar no máximo 15 minutos;

- **Planejamento da *Sprint***: Reunião usada para alinhar o planejamento da próxima *sprint* de acordo com o andamento do projeto. Nela são consultados o *Product Backlog* (documento com todas tarefas mapeadas para entregar o produto final) e o *Sprint Backlog* (distribuição das tarefas do *Product Backlog* em *sprints*) para definir quais tarefas serão priorizadas, e

- **Retrospectiva/*Review* da *Sprint***: Reunião para verificar o progresso do time em uma *sprint*. O que foi entregue em termos de tarefas, quais os pontos positivos e negativos observados durante o desenvolvimento, possíveis dificuldades, entre outros.

Os times do Scrum são auto-organizáveis e multifuncionais, os papéis podem ser rotacionados entre os membros e cada papel tem sua função dentro do processo de desenvolvimento. O time de desenvolvimento é responsável por entregar uma nova versão do produto ao fim de cada *sprint*. O *Product Owner* (PO) é responsável por maximizar o valor do produto, ele controla o *Product Backlog* e define as prioridades de tarefas, ele deve garantir que o time de desenvolvimento está compreendendo as tarefas e entregando valor ao fim de cada *sprint*. Por fim, o *Scrum Master*, é responsável por verificar se o Scrum está sendo corretamente entendido e aplicado por todos os membros da equipe, ajudando a equipe a entender quais rotinas e artefatos estão sendo bem utilizados e detectando pontos de melhoria (SUTHERLAND; SCHWABER, 2013).

4.3.2 Kanban

Um quadro Kanban é uma ferramenta ágil de gerenciamento de projetos. Ele é um método visual para projetar e gerenciar o fluxo de trabalho, para ajudar a visualizá-lo, limitar o número de tarefas em andamento e maximizar a eficiência. Um quadro Kanban é dividido em colunas, e as tarefas se movem entre as colunas de acordo com seu estado atual (AHMAD; MARKKULA; OIVO, 2013).

O Kanban permite ao time visualizar o fluxo de trabalho através das tarefas se movimentando pelo quadro, permitindo transparência (todos da equipe sabem quem está realizando e qual o estado de cada tarefa) e inspeção (é possível verificar possíveis gargalos em tarefas que estão há muito tempo em progresso ou um excesso de tarefas em progresso ao mesmo tempo). Reduzindo o número de tarefas em progresso paralelamente o Kanban ajuda a produzir um fluxo constante de tarefas concluídas (AHMAD; MARKKULA; OIVO, 2013).

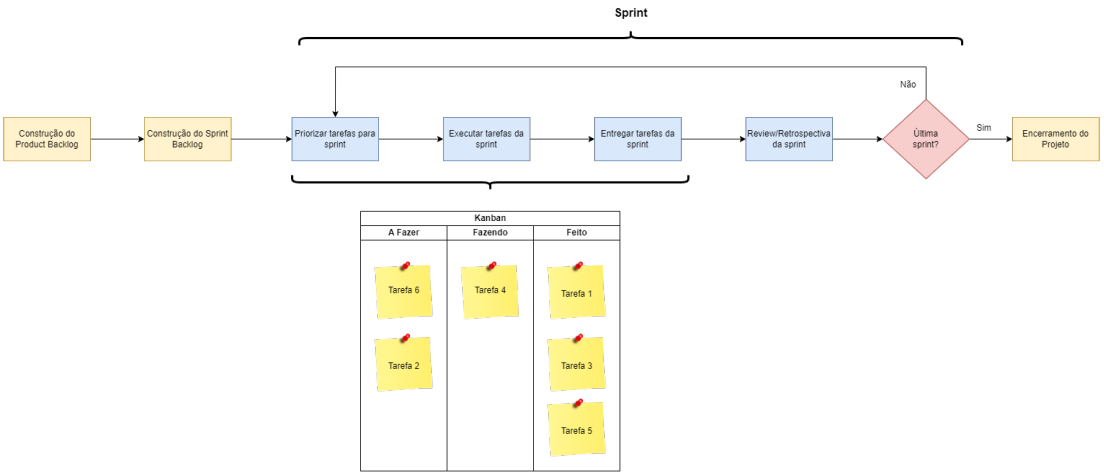
4.3.3 Metodologia Híbrida

Considerando que o time de desenvolvimento vai ter apenas uma pessoa, a metodologia híbrida proposta visa conciliar um ritmo saudável de desenvolvimento, aliado a um bom planejamento e organização.

Os elementos do Scrum que serão utilizados são: os três pilares conceituais (transparência, inspeção e adaptação); o conceito de *sprint*; reuniões de planejamento e retrospectiva/*review* da *sprint*; documentos de *Product Backlog* e *Sprint Backlog* e papéis de *Product Owner* e time de desenvovimento.

O elemento utilizado do método Kanban é o quadro de trabalho dividido nas colunas: A Fazer, Fazendo e Feito, de forma a ter uma boa visualização do fluxo de trabalho. A Figura 21 ilustra a metodologia proposta.

Figura 21 – Funcionamento da metodologia proposta



Fonte: Autor

4.4 Metodologia de Análise de Resultados

Para realizar a análise e melhoria do trabalho proposto, foi adotada a metodologia pesquisa-ação. Um processo reflexivo para constante aperfeiçoamento, que realiza a combinação de teoria e prática com foco na mudança (GERHARDT; SILVEIRA, 2009).

A pesquisa-ação permite ao pesquisador, além de aplicar a sua proposta, o acompanhamento em tempo real da solução e a implementação de novas melhorias. A estrutura da pesquisa-ação é cíclica, realizada repetidamente e compreende (GIL, 2010):

- **Diagnóstico**: Inicia o processo da pesquisa, com a identificação do contexto e do problema abordado;

- **Análise dos Dados**: Avalia o problema e os dados coletados, sendo feita a interpretação destes;
- **Elaboração do Plano de Ação**: Faz o planejamento de uma ação destinada a enfrentar o problema que foi objeto de investigação, e
- **Divulgação dos Resultados**: Realiza a documentação e a divulgação dos resultados obtidos no processo da pesquisa-ação.

O problema no qual foi feito o processo de diagnóstico é a dificuldade de identificar e categorizar informações importantes dentro de faturas de energia. A análise de dados compreende a pesquisa feita, procurando por soluções possíveis na literatura especializada. O plano de ação está especificado no capítulo de Proposta. Por fim, os resultados serão divulgados na parte 2 do Trabalho de Conclusão de Curso.

4.5 Fluxo de Atividades

O Trabalho de Conclusão de Curso (TCC), segundo as regras da Universidade de Brasília (UnB), é realizado em duas fases: uma chamada de TCC 01 e outra chamada de TCC 02. Durante o TCC 01, o foco deste trabalho foi fazer um levantamento bibliográfico referente à questão de pesquisa e realizar a documentação da pesquisa realizada de forma a embasar a proposta. Com base nas fundamentações teóricas e na pesquisa realizada ao longo do TCC 01, o trabalho realizado no TCC 02 estará focado em atingir o seguinte objetivo geral: Desenvolvimento de uma ferramenta, cujos principais propósitos são a identificação e a categorização, de forma automática, dos blocos de informações relevantes em uma fatura de energia, utilizando algoritmos de *Deep Learning*.

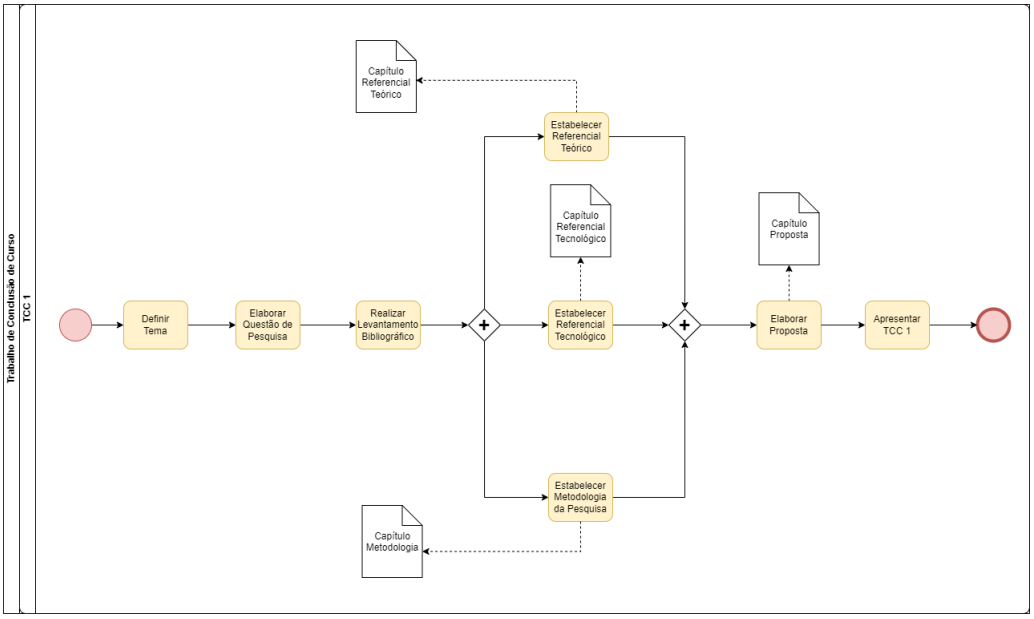
4.5.1 Primeira Fase - TCC 01

A primeira fase do TCC tem como objetivo realizar um levantamento bibliográfico referente à pesquisa, estabelecer metodologia utilizada, juntamente com os referenciais teórico e tecnológico e ao fim realizar uma proposta. A Figura 22 ilustra as atividades do TCC 1, em um fluxo utilizando o padrão BPMN.

O TCC 1 compreende 8 atividades principais, descritas a seguir:

- **Definir Tema**: Esta atividade engloba a escolha do contexto em que se deseja trabalhar e o refinamento deste contexto para se chegar a um tema;
- **Elaborar Questão de Pesquisa**: Esta atividade faz uso do tema selecionado para montar uma questão que será respondida na pesquisa;

Figura 22 – Fluxo de Atividades do TCC 01



Fonte: Autor

- **Realizar Levantamento Bibliográfico**: Esta atividade compreende a pesquisa por artigos e livros nas bases de conhecimento e o refinamento dessa pesquisa, com a procura por termos cada vez mais significativos ao trabalho em questão;

- **Estabelecer Referencial Teórico**: Esta atividade compreende o entendimento da teoria que circunda o tema escolhido, juntamente com a escrita desta, utilizando de referências na literatura especializada e recursos visuais para facilitar o entendimento, como imagens e tabelas;

- **Estabelecer Referencial Tecnológico**: Esta atividade tem por função descrever e dar uma breve explicação sobre as tecnologias que serão utilizadas na realização do trabalho, tanto no que tange à escrita da monografia, como ao desenvolvimento da proposta;

- **Estabelecer Metodologia da Pesquisa**: Esta atividade é responsável por definir as características metodológicas da pesquisa realizada no trabalho, como também detalhar como se dará o processo de desenvolvimento da ferramenta proposta;

- **Elaborar Proposta**: Esta atividade tem como função definir a proposta do trabalho, que será executada no TCC 2, ela deve fornecer contexto e detalhes da solução, e

- **Apresentar TCC 1**: Esta atividade tem como função apresentar os resultados obtidos até o momento para a banca examinadora.

A Tabela 2 ilustra o cronograma de atividades do TCC 1, especificando os meses de conclusão de cada uma delas.

Tabela 2 – Cronograma TCC 1

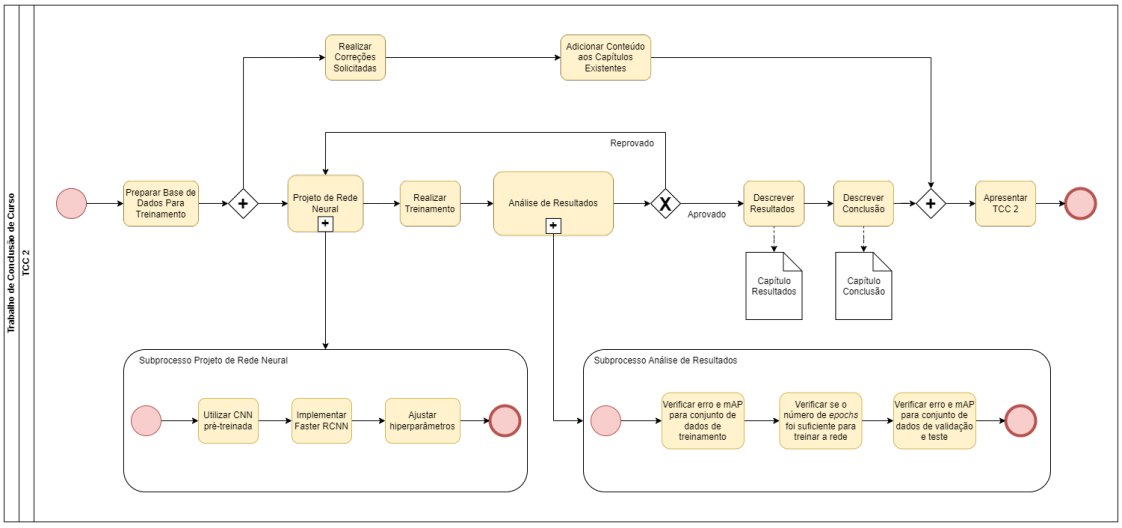
Atividade	**Março**	**Abril**	**Maio**	**Junho**	**Julho**
Definir Tema	X				
Elaborar Questão de Pesquisa	X	X			
Realizar Levantamento Bibliográfico		X	X		
Estabelecer Referencial Teórico		X	X		
Estabelecer Referencial Tecnológico		X	X		
Estabelecer Metodologia da Pesquisa			X	X	
Elaborar Proposta				X	X
Apresentar TCC 1					X

Fonte: Autor

4.5.2 Segunda Fase - TCC 02

A segunda fase do TCC tem como objetivo realizar o desenvolvimento da ferramenta proposta na primeira fase, analisar os resultados obtidos e fazer as correções necessárias no trabalho. A Figura 23 ilustra as atividades do TCC 2, em um fluxo utilizando o padrão BPMN.

Figura 23 – Fluxo de Atividades do TCC 02



Fonte: Autor

O TCC 2 compreende 7 atividades e 2 subprocessos, descritos a seguir:

- **Preparar Base de Dados para Treinamento**: Esta atividade envolve a coleta, transformação e tratamento de faturas, para serem fornecidas como insumo à rede neural;
- **Projeto de Rede Neural**: Este subprocesso compreende a implementação da rede neural proposta (CNN sendo utilizada como parte de uma RetinaNet) e o ajuste dos hiperparâmetros da rede;
- **Realizar Treinamento**: Esta atividade compreende a realização do treinamento da rede neural nas faturas de energia selecionadas;
- **Análise de Resultados**: Este subprocesso estabelece quais métricas devem ser observadas durante os processos de treinamento e teste. Ele tem a função de verificar se a rede se comportou conforme o esperado, se há espaço para melhorias significativas, usando como base o erro e mAP obtidos a partir dos conjuntos de dados de treinamento, validação e teste;
- **Realizar Correções Solicitadas**: Esta atividade compreende a realização de correções solicitadas pela banca avaliadora, ou de pontos que o próprio autor identificou como incorretos;
- **Adicionar Conteúdo aos Capítulos Existentes**: Esta atividade tem como função a adição de informações aos capítulos escritos no TCC 1, com o objetivo de complementar alguma explicação ou adicionar uma nova informação;
- **Descrever Resultados**: Esta atividade compreende a descrição da experiência de desenvolvimento da ferramenta, juntamente com os resultados que ela apresentou;
- **Descrever Conclusão**: Esta atividade compreende a sintetização dos resultados obtidos, dificuldades encontradas e possíveis melhorias, fazendo um panorama geral do trabalho executado, e
- **Apresentar TCC 2**: Esta atividade tem como função apresentar os resultados finais para a banca examinadora.

A Tabela 3 ilustra o cronograma de atividades do TCC 2, especificando os meses de conclusão previstos por atividade.

4.6 Considerações Finais do Capítulo

Este capítulo apresentou as metodologias adotadas, desde a metodologia de pesquisa bibliográfica, até as metodologias de desenvolvimento e análise de resultados. Foram apresentadas as técnicas para levantamento bibliográfico, as *strings* de busca utilizadas,

Tabela 3 – Cronograma TCC 2

Atividade	**Agosto**	**Setembro**	**Outubro**	**Novembro**	**Dezembro**
Preparar Base de Dados para Treinamento	X	X			
Realizar Correções Solicitadas	X	X			
Adicionar Conteúdo aos Capítulos Existentes	X	X			
Projeto de Rede Neural		X	X		
Realizar Treinamento		X	X	X	
Análise de Resultados		X	X	X	
Descrever Resultados				X	X
Descrever Conclusão				X	X
Apresentar TCC 2					X

Fonte: Autor

processo de filtragem de artigos, entre outros. Em sequência, foi apresentada a metodologia de desenvolvimento, adaptada para um time de um único integrante, que utiliza elementos do Scrum e do Kanban. E a última metodologia apresentada foi a de análise de resultados, detalhando os passos da pesquisa-ação realizada no trabalho.

Por fim, foi feito um detalhamento de todas atividades desenvolvidas e planejadas para as duas fases do Trabalho de Conclusão de Curso, juntamente com seus cronogramas de execução.

5 Ferramenta para Identificação e Categorização de Blocos de Informação em Faturas de Energia

Este capítulo apresenta a ferramenta proposta para o projeto, detalhando os modelos de rede neural utilizados, funções adicionais, tratamento de dados, entre outros. A Contextualização retoma a problemática e os desafios relacionados à extração de faturas. Adicionalmente, em Solução Atual para Extração de Faturas detalha-se os problemas que ocorrem com um modelo de extrações a partir de posições absolutas, aplicado na empresa alvo do estudo deste trabalho. Em sequência, a Solução Proposta para Extração de Faturas explica como se dará a criação da rede neural para fazer a identicação e categorização de blocos importantes. Por fim, têm-se as Considerações Finais do Capítulo.

5.1 Contextualização

A ferramenta proposta no presente trabalho é baseada na experiência vivida pelo autor em uma empresa que atua no serviço de energia dentro do mercado brasileiro. A empresa será tratada com o nome fictício de Empresa de Gerenciamento Energético (EGE). O propósito da EGE é ajudar empresas a aumentar sua eficiência energética. Ela faz isso através de dois produtos: um é chamado de Dashboard e outro de Portal de Geração Distribuída.

Em relação à fonte de compra da energia é possível dividir as unidades consumidoras em dois grupos (ENERGIA, 2022):

- **Ambiente de Contratação Livre (ACL)**: Os consumidores livres compram energia diretamente dos geradores ou de comercializadores cadastrados na Câmara de Comercialização de Energia Elétrica (CCEE). O acordo é feito através de contratos bilaterais com condições livremente negociadas, como preço, prazo, volume, tipo, entre outros. Cada unidade consumidora paga uma fatura referente ao serviço de distribuição para a concessionária local (tarifa regulada) e uma ou mais faturas referentes à compra da energia (preço negociado de contrato), e

- **Ambiente de Contratação Regulada (ACR)**: Os consumidores cativos compram energia das distribuidoras às quais estão ligados. Cada unidade consumidora paga apenas uma fatura de energia por mês, incluindo o serviço de distribuição (Ta-

rifa por Uso do Sistema de Distribuição - TUSD) e a geração da energia (Tarifa de Energia - TE), as tarifas são reguladas pelo governo.

Em relação ao tipo de geração de energia também é possível dividir as unidades consumidoras em dois grupos (SOLAR, 2022):

- **Geração Centralizada**: A energia é gerada por usinas de grande porte, como grandes centrais hidrelétricas ou termoelétricas, que são conectadas ao Sistema Interligado Nacional (SIN), responsável por distribuir essa energia pelo Brasil, e
- **Geração Distribuída**: A energia é gerada por geradores de pequeno porte, no local ou próximo ao ponto de consumo. Normalmente utiliza fontes renováveis de energia, como solar ou eólica. A energia gerada pode ser distribuída entre diversas instalações usando o sistema de distribuição da distribuidora local. A energia gerada, distribuída e não consumida se torna crédito para ser utilizado posteriormente.

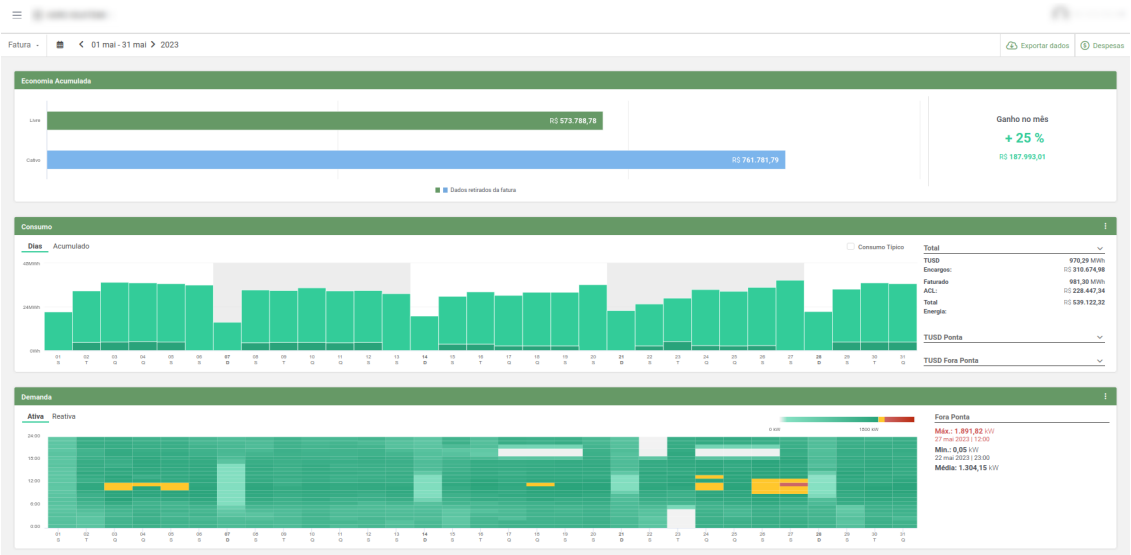
O Dashboard, ilustrado com uma tela de exemplo na Figura 24, é um produto focado em clientes comerciais e industriais com uma ou múltiplas unidades consumidoras, pertencentes ao mercado livre ou ao mercado cativo de energia, utilizando o sistema de geração centralizada. Para utilizar o Dashboard, a EGE faz a instalação de medidores de energia nas unidades consumidoras atendidas, estes medidores são capazes de fornecer dados como consumo, demanda e tensão em tempo real.

O produto em questão consegue auxiliar gestores a otimizar o consumo de energia de suas unidades consumidoras, a partir dos dados obtidos pelos medidores e pelas faturas de energia emitidas todo mês pelas distribuidoras. Este produto permite verificar anomalias como picos de consumo ou demanda, fazer uma comparação entre os modelos cativo e livre, verificar bandeiras, tarifas e impostos aplicados pela distribuidora, entre diversas outras funcionalidades.

O Portal de Geração Distribuída (PGD), ilustrado com uma tela de exemplo na Figura 25, é um produto focado em clientes comerciais e industriais com múltiplas unidades consumidoras, pertencentes ao mercado cativo de energia, utilizando o sistema de geração distribuída. A energia gerada por uma unidade consumidora é distribuída a diversas outras unidades através de um mecanismo de rateio (divisão), que é acordado e aplicado pela distribuidora mensalmente. Nessa divisão cada unidade consumidora receberá uma porcentagem da energia gerada, a Tabela 4 ilustra o fato relatado.

O PGD recebe como insumos faturas de energia e contratos de rateio firmados com as distribuidoras. Ele fornece ao cliente uma visualização em formatos de listas ou gráficos, contendo dados como: quantidade de energia gerada e consumida, saldo disponível, custo

Figura 24 – Dashboard da EGE mostrando informações detalhadas de consumo de um cliente



Fonte: Autor

Tabela 4 – Exemplo de tabela de rateio em um sistema de geração distribuída

Unidade	**Energia Gerada**	**Energia Consumida**	**Rateio**
Unidade Geradora	802 mWh	2 mWh	-
Unidade Consumidora 1	0 mWh	320 mWh	40 %
Unidade Consumidora 2	0 mWh	240 mWh	30 %
Unidade Consumidora 3	0 mWh	240 mWh	30 %

Fonte: Autor

de distribuição de energia, impostos, bandeiras tarifárias, gases não emitidos (CO2) por uso de energia limpa, economia de custos com a geração da própria energia, entre outros.

5.1.1 Faturas de Energia

Como foi verificado na Contextualização, as faturas de energia são insumos importantes para os dois produtos oferecidos na EGE, através das quais são extraídos dados fundamentais para funcionamento destes.

As faturas de energia no Brasil são disponibilizadas mensalmente pelas 53 concessionárias e 52 permissionárias habilitadas pela ANEEL a distribuir energia no Brasil. A palavra “distribuidoras” neste trabalho se refere ao conjunto de concessionárias e permissionárias. A Figura 26 mostra uma central de visualização da ANEEL, que ilustra área

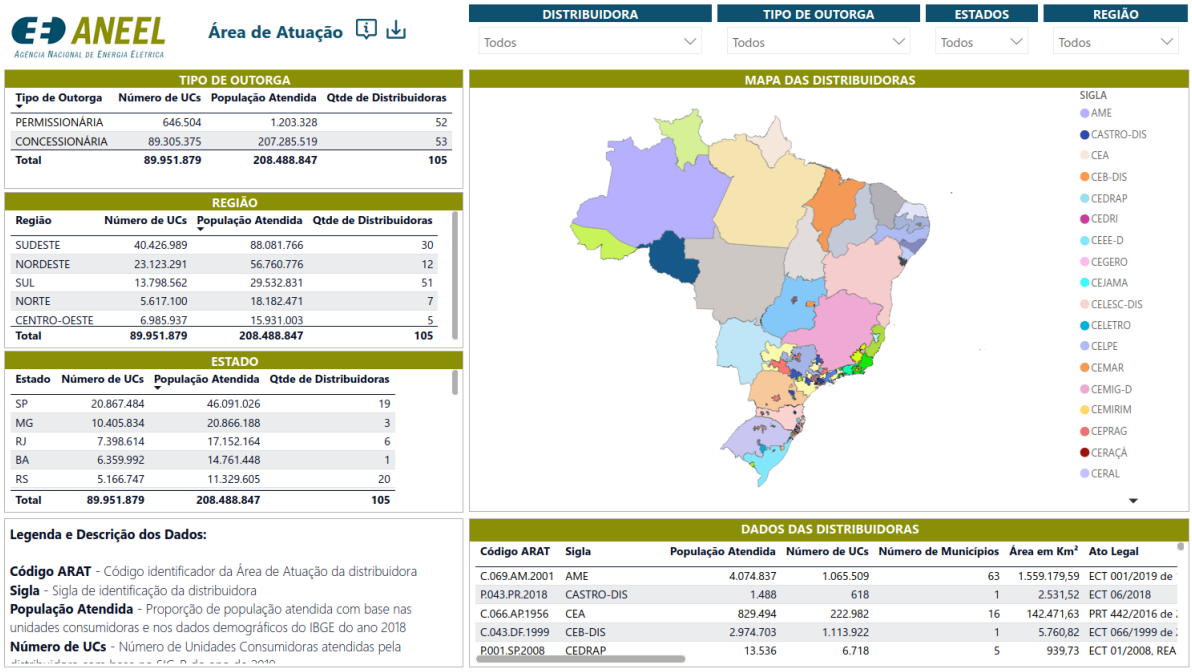
Figura 25 – Portal de Geração Distribuída da EGE mostrando distribuição e gasto da energia gerada entre diferentes unidades de um cliente

Nome	Consumo	Energia Contratada	Créditos Consumidos	Rateio Contratado	Rateio Aplicado pela Distribuidora	Previsão de Disponibilidade
[redacted]	2.077 kWh	--	1.590 kWh	0,2%	0,2%	
[redacted]	5.715 kWh	--	3.816 kWh	0,48%	0,48%	11 mai
[redacted]	23.200 kWh	--	23.200 kWh	4,06%	4,08%	28 mai
[redacted]	36.199 kWh	--	27.589 kWh	3,47%	3,45%	02 jun
[redacted]	3.959 kWh	--	3.959 kWh	0,54%	0,54%	11 mai
[redacted]	11.279 kWh	--	8.427 kWh	1,06%	1,05%	28 mai
[redacted]	6.320 kWh	--	6.320 kWh	0,93%	0,93%	01 jun
[redacted]	11.376 kWh	--	11.376 kWh	1,44%	1,43%	22 mai
[redacted]	5.825 kWh	--	5.825 kWh	0,74%	0,74%	02 jun
[redacted]	3.009 kWh	--	2.226 kWh	0,28%	0,28%	08 mai

Fonte: Autor

de cobertura, regiões e número de unidades consumidoras atendidas e outras informações a respeito das concessionárias e permissionárias no Brasil (ANEEL, 2022).

Figura 26 – Conjunto de visualizações da ANEEL mostrando informações sobre concessionárias e permissionárias de energia



Fonte: (ANEEL, 2022)

As unidades consumidoras de energia elétrica são classificadas em dois grupos em relação a tensão de recebimento da energia (CANALSOLAR, 2019):

- **Grupo A (alta e média tensão)**: É composto por unidades consumidoras que recebem energia em tensão igual ou superior a 2,3 kilovolts (kV) ou são atendidas a partir de sistema subterrâneo de distribuição em tensão secundária. Geralmente

se enquadram neste grupo as grandes indústrias e estabelecimentos comerciais de grande porte. São caracterizadas pela tarifa binômia (aplicada ao consumo e à demanda faturável). Podem fazer parte das modalidades tarifárias Azul e Verde, e

- **Grupo B (baixa tensão)**: É caracterizado por unidades consumidoras atendidas em tensão inferior a 2,3 kV. Contém uma tarifa monômia (aplicável apenas ao consumo). Geralmente se enquadram nessa categoria as residências, pequenas indústrias e pequenos estabelecimentos comerciais. Podem fazer parte das modalidades tarifárias Convencional e Branca.

Cada distribuidora de energia contém no mínimo dois modelos (*layouts*) distintos de fatura, sendo um para unidades consumidoras do grupo A e outro para unidades consumidoras do grupo B. Algumas distribuidoras também têm modelos distintos para consumidores do ambiente de contratação livre ou de modalidade tarifária Branca. A Figura 27 ilustra um exemplo de fatura da distribuidora CEMIG, pertencente ao grupo B e de modalidade tarifária Convencional.

As distribuidoras podem criar novos modelos, seja para promover uma melhor visualização para os clientes, para trocar a identidade visual ou para se enquadrar em mudanças promovidas pela ANEEL (como a inclusão de uma nova informação na fatura). Dessa forma, uma distribuidora em um período de três anos pode emitir faturas em diversos modelos distintos (por vezes mais de dez).

5.2 Solução Atual para Extração de Faturas

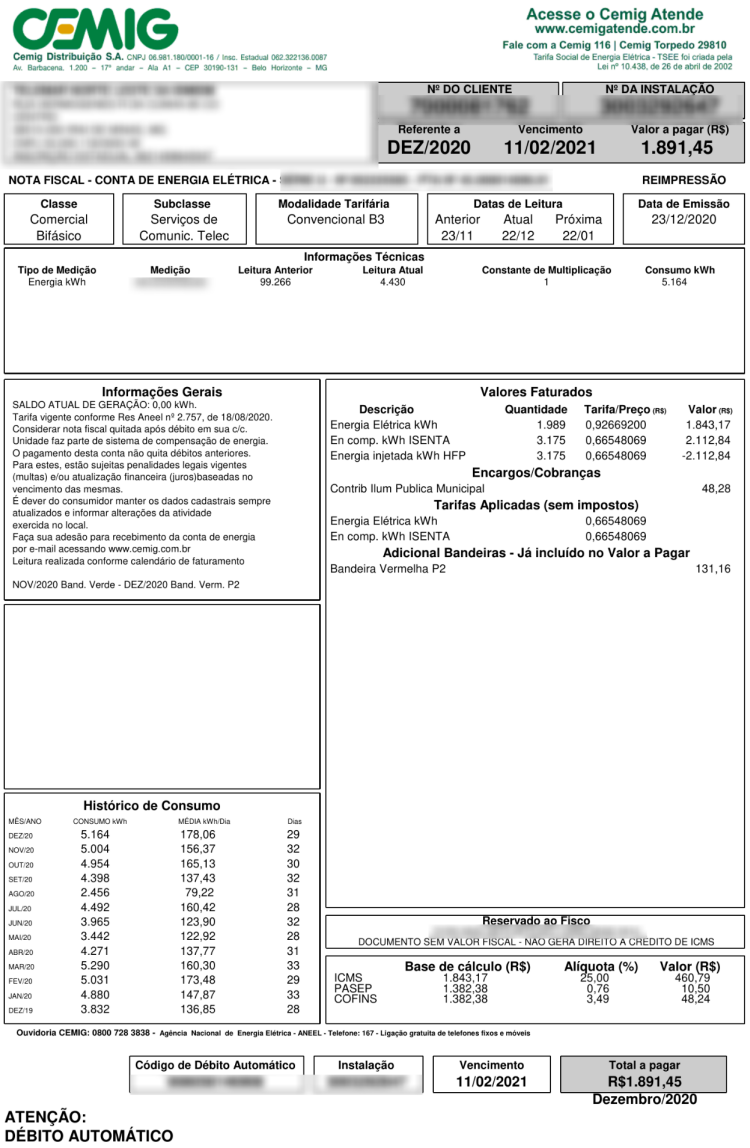
A EGE atualmente coleta informações de faturas de 36 distribuidoras de energia, com foco nas distribuidoras que atendem a um grande número de clientes, cobrindo todos os estados brasileiros com exceção de Rondônia e Amazonas. A extração dessas faturas pode se dar por um fluxo interno ou utilizando um serviço de terceiro especializado para extração.

O fluxo interno normalmente atende a distribuidoras em que a EGE possui um maior número de clientes, enquanto o fluxo externo (utilizando serviço de terceiro) atende os outros casos. A decisão de utilizar um serviço externo foi motivada pela difícil manutenção do serviço de extração, com um número grande de modelos, associado a um número grande de distribuidoras atendidas.

A arquitetura simplificada mostrando a conexão dos produtos da EGE com a *pipeline* de extração está ilustrada na Figura 28. Ela contém o seguintes elementos:

- **Serviços (Dashboard e Portal GD)**: Os dois produtos da EGE que são o destino final das informações de extração. O cliente pode submeter faturas para extração

Figura 27 – Exemplo de fatura com informações sensíveis borradas



CEMIG

Cemig Distribuição S.A. CNPJ 06.981.180/0001-16 / Insc. Estadual 062.322136.0087
Av. Barbacena, 1.200 – 17º andar – Ala A1 – CEP 30190-131 – Belo Horizonte – MG

Acesse o Cemig Atende
www.cemigatende.com.br
Fale com a Cemig 116 | Cemig Torpedo 29810
Tarifa Social de Energia Elétrica - TSEE foi criada pela Lei nº 10.438, de 26 de abril de 2002

Nº DO CLIENTE	Nº DA INSTALAÇÃO
[redacted]	[redacted]

Referente a	Vencimento	Valor a pagar (R$)
DEZ/2020	11/02/2021	1.891,45

NOTA FISCAL - CONTA DE ENERGIA ELÉTRICA - [redacted] REIMPRESSÃO

Classe	Subclasse	Modalidade Tarifária	Datas de Leitura			Data de Emissão
			Anterior	Atual	Próxima	
Comercial Bifásico	Serviços de Comunic. Telec	Convencional B3	23/11	22/12	22/01	23/12/2020

Informações Técnicas

Tipo de Medição	Medição	Leitura Anterior	Leitura Atual	Constante de Multiplicação	Consumo kWh
Energia kWh	[redacted]	99.266	4.430	1	5.164

Informações Gerais

SALDO ATUAL DE GERAÇÃO: 0,00 kWh.
Tarifa vigente conforme Res Aneel nº 2.757, de 18/08/2020.
Considerar nota fiscal quitada após débito em sua c/c.
Unidade faz parte de sistema de compensação de energia.
O pagamento desta conta não quita débitos anteriores.
Para estes, estão sujeitas penalidades legais vigentes (multas) e/ou atualização financeira (juros)baseadas no vencimento das mesmas.
É dever do consumidor manter os dados cadastrais sempre atualizados e informar alterações da atividade exercida no local.
Faça sua adesão para recebimento da conta de energia por e-mail acessando www.cemig.com.br
Leitura realizada conforme calendário de faturamento

NOV/2020 Band. Verde - DEZ/2020 Band. Verm. P2

Valores Faturados

Descrição	Quantidade	Tarifa/Preço (R$)	Valor (R$)
Energia Elétrica kWh	1.989	0,92669200	1.843,17
En comp. kWh ISENTA	3.175	0,66548069	2.112,84
Energia injetada kWh HFP	3.175	0,66548069	-2.112,84

Encargos/Cobranças

Contrib Ilum Publica Municipal	48,28

Tarifas Aplicadas (sem impostos)

Energia Elétrica kWh	0,66548069
En comp. kWh ISENTA	0,66548069

Adicional Bandeiras - Já incluído no Valor a Pagar

Bandeira Vermelha P2	131,16

Histórico de Consumo

MÊS/ANO	CONSUMO kWh	MÉDIA kWh/Dia	Dias
DEZ/20	5.164	178,06	29
NOV/20	5.004	156,37	32
OUT/20	4.954	165,13	30
SET/20	4.398	137,43	32
AGO/20	2.456	79,22	31
JUL/20	4.492	160,42	28
JUN/20	3.965	123,90	32
MAI/20	3.442	122,92	28
ABR/20	4.271	137,77	31
MAR/20	5.290	160,30	33
FEV/20	5.031	173,48	29
JAN/20	4.880	147,87	33
DEZ/19	3.832	136,85	28

Reservado ao Fisco

DOCUMENTO SEM VALOR FISCAL - NAO GERA DIREITO A CREDITO DE ICMS

	Base de cálculo (R$)	Alíquota (%)	Valor (R$)
ICMS	1.843,17	25,00	460,79
PASEP	1.382,38	0,76	10,50
COFINS	1.382,38	3,49	48,24

Ouvidoria CEMIG: 0800 728 3838 - Agência Nacional de Energia Elétrica - ANEEL - Telefone: 167 - Ligação gratuita de telefones fixos e móveis

Código de Débito Automático	Instalação	Vencimento	Total a pagar
[redacted]	[redacted]	11/02/2021	R$1.891,45

Dezembro/2020

ATENÇÃO:
DÉBITO AUTOMÁTICO

Fonte: Autor

pela interface ou cadastrar dados de acesso ao site da distribuidora para que faturas sejam automaticamente carregadas;

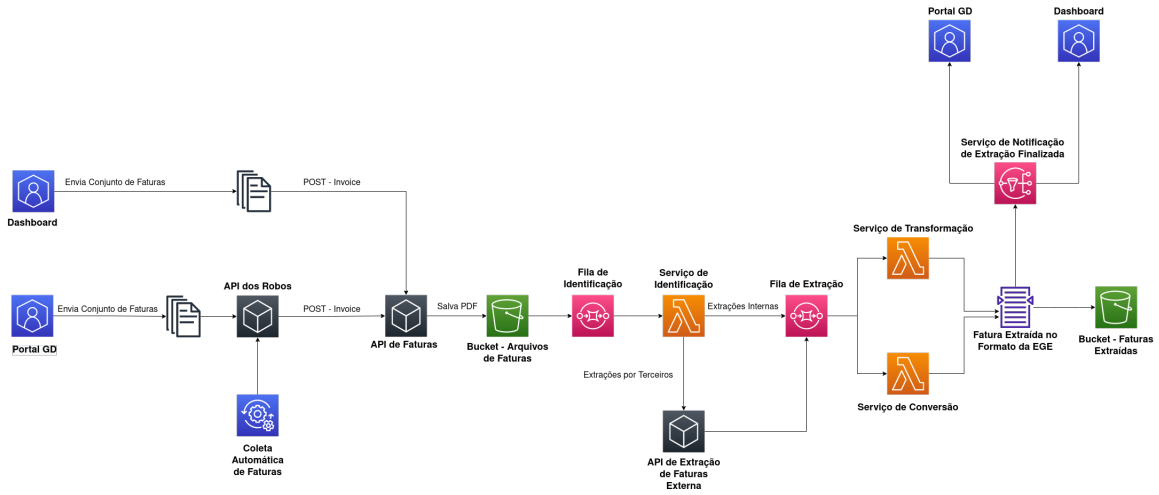
- **Coleta Automática**: Baixa faturas de energia dos sites das distribuidoras para clientes com informações de acesso;
- **API dos Robôs**: Interface para o controle de faturas baixadas pelos robôs ou vindas do Portal GD, se comunica com o banco dos robôs onde ficam cadastrados dados de acesso as distribuidoras, informações de unidades consumidoras, faturas baixadas, entre outros;
- **API de Faturas**: Interface responsável por obter extrações e direcionar novas fa-

turas para serem extraídas. É possível obter uma extração procurando por diversos filtros, como número da unidade consumidora, mês de referência, distribuidora, entre outros.

- ***Bucket* – Arquivos de Faturas**: Um *bucket* é um recurso de armazenamento em nuvem da suíte de ferramentas AWS (*Amazon Web Services*) para guardar uma grande quantidade de objetos (arquivos) de diferentes tipos. Este *bucket* armazena os arquivos PDFs de faturas de energia enviados à API de Faturas;
- **Serviço de Identificação**: É responsável por identificar se o arquivo recebido é uma fatura de energia, e caso seja, identificar a distribuidora e modelo da fatura. A identificação da distribuidora é feita por OCR, que procura por informações específicas de uma distribuidora na fatura, como seu CNPJ. Para identificar o modelo são consultados dicionários de dados que especificam cortes em posições absolutas e o conteúdo que deve estar presente dentro daquele corte. De forma que, uma fatura pertence a um modelo se uma palavra ou um conjunto de palavras está presente dentro de uma área delimitada da fatura, conforme indicado no dicionário;
- **Serviço de Transformação**: É responsável por fazer a extração de informações da fatura. Ao recebê-la, juntamente com as informações de distribuidora e modelo, este serviço realiza a extração e tratamento das informações contidas dentro de cada corte;
- **API Externa de Extração de Faturas**: Serviço de terceiro com a função de extrair faturas e retornar seus dados;
- **Serviço de Conversão**: Tem a função de transformar uma extração vinda de um serviço de terceiro no formato esperado pela EGE;
- ***Bucket* – Faturas Extraídas**: Armazena faturas extraídas em JSON (*JavaScript Object Notation*), com os campos esperados no formato EGE, e
- **Serviço de Notificação de Extração Finalizada**: Notifica o Dashboard ou o Portal GD quando uma extração é finalizada. Também informa alguns atributos de identificação da fatura, como instalação, mês de referência e um identificador único. É possível fazer uma requisição para API de Faturas para obter a extração completa.

O fluxo de extração da EGE utiliza de cortes retangulares em posições absolutas no PDF, em regiões onde há informação importante. De modo que, para cada modelo, um novo conjunto de cortes deve ser feito, indicando onde estão itens como: tabela de faturamento; campo de endereço e nome do cliente; tabela de leitura; informações de categorização da unidade (grupo, subgrupo, classe, modalidade tarifária) e outros. Como as

Figura 28 – Arquitetura de extração de faturas na EGE



Fonte: Autor

posições de cortes são absolutas ocorrem muitos casos de falha quando a distribuidora faz um deslocamento pequeno no modelo, as mesmas falhas ocorrem se a fatura é digitalizada. Se os cortes não estão perfeitamente posicionados, a extração não é realizada.

Um outro problema da solução da EGE para extração de faturas é o método de identificação de modelo, contido dentro do Serviço de Identificação. Ao criar um novo modelo de fatura é necessário encontrar diferenças para os modelos já existentes, dentro de uma mesma distribuidora, de forma a criar um corte que identifica unicamente o novo modelo. Dependendo da semelhança entre modelos antigos e novos, é preciso alterar os cortes do modelo antigo para o funcionamento correto do sistema.

5.3 Solução Proposta para Extração de Faturas

Tendo em vista os problemas dentro do Serviço de Identificação, foi desenvolvida uma ferramenta para identificar e categorizar blocos de informações importantes dentro de uma fatura de energia. A solução utiliza um algoritmo de rede neural convolucional para detecção de objetos, fundamentado dentro do capítulo de Referencial Teórico, na seção Redes Neurais Convolucionais para Detecção de Objetos. O link para acessar o repositório da ferramenta é: <https://github.com/renan601/TCC2-RedesNeuraisIdentificacaoFaturas>.

5.3.1 Construção da Base de Treinamento

Para compor a base de treinamento foram utilizadas faturas das seguintes distribuidoras: **CEMIG** e **Grupo CPFL** (CPFL Paulista, CPFL Piratininga, CPFL Santa Cruz e RGE). As diferentes distribuidoras do Grupo CPFL utilizam os mesmos modelos de faturas. A escolha destas distribuidoras foi feita com base na relevância delas em ter-

mos de volume de processamento pela EGE. Foram utilizadas 6 mil faturas da CEMIG e 4 mil faturas do Grupo CPFL. As faturas da CEMIG estão distribuídas em 5 modelos distintos, enquanto as faturas da CPFL estão distribuídas em 5 modelos distintos.

Para cada modelo a EGE tem um conjunto de cortes, que são as caixas delimitadoras mencionadas nas R-CNN, que delimitam o objeto de interesse. As faturas, juntamente com as posições dos cortes, são os dados de treinamento que foram passados à rede.

Dentro do escopo do TCC, apenas algumas informações das faturas foram previstas para serem extraídas, elas são as seguintes: **tabela de faturamento; tabela de leitura; tabela de impostos; informações da unidade consumidora (endereço, CEP, nome do cliente) e número de instalação da unidade consumidora**.

Faturas contendo múltiplas páginas foram divididas, de forma que cada página constitui uma nova fatura para a etapa de treinamento.

Todas faturas processadas estão inicialmente no formato PDF e foram emitidas pela distribuidora, não passaram por um processo de digitalização a partir de um celular ou qualquer outro dispositivo capaz de realizar essa atividade.

5.3.2 Pré-processamento de Imagens

As faturas foram transformadas de PDF para PNG (formato de imagem), utilizando a biblioteca pdf2image (PDF2IMAGE, 2023). Comparado a outros formatos o PNG oferece uma compactação sem perdas, garantindo o nível de detalhe da imagem em informações sensíveis, como bordas finas que dividem os blocos das faturas (ADOBE, 2023).

Para facilitar o processo de treinamento da CNN as imagens utilizadas são do mesmo tamanho. Todas as imagens de treinamento e validação foram coloridas (3 canais de cor) e redimensionadas para que sua maior dimensão seja equivalente a 640 pixels. Como o projeto trata de faturas emitidas em PDFs, a dimensão maior é a altura. Após realizar o redimensionamento da imagem mantendo sua proporção altura/largura inicial, foram adicionados *paddings* (preenchimentos) nas bordas da imagem com o objetivo de que ela fique quadrada (640x640). As caixas delimitadoras também foram redimensionadas, para refletir a mudança na imagem.

Para aumentar o tamanho e a variedade das instâncias de treinamento foi utilizada a técnica de *data augmentation*, em que a partir de uma imagem original de treinamento são criadas variações desta. As imagens foram cortadas e reajustadas de volta ao tamanho de 640x640, juntamente com mudanças de brilho, saturação e contraste, de forma a também permitir a identificação de imagens fora do padrão emitido pela distribuidora (faturas que foram digitalizadas).

As imagens foram divididas em dois conjuntos para o treinamento da rede neural. Os conjuntos são:

- **Conjunto de Treinamento**: Contendo 85% do total de imagens, selecionadas de forma randômica, e
- **Conjunto de Validação**: Contendo 15% do total de imagens, selecionadas de forma randômica.

5.3.3 Redes Convolucionais - ResNet-50 e Faster R-CNN

Vários modelos arquiteturais de CNN foram desenvolvidos com o objetivo de atingir bons resultados em competições de classificação, como a CoCo (Microsoft *Common Objects in Context*) e a OpenImages. A ResNet (*Residual Neural Network*), um tipo de CNN, foi desenvolvida em 2015 por uma equipe de pesquisa da Microsoft. Eles introduziram uma nova arquitetura contendo blocos residuais com conexões de atalho (*skip connections*), para conseguir criar redes neurais com um número maior de camadas e com melhor capacidade de aprendizado (ELGENDY, 2020).

Um grande problema da arquitetura padrão de uma CNN é o desaparecimento do gradiente na retropropagação em redes com muitas camadas. Conforme elucidado no Referencial Teórico, o gradiente (derivada) do erro é calculado e retropropagado camada a camada por toda a rede neural, porém esse gradiente tende a zero em redes muito profundas (ELGENDY, 2020).

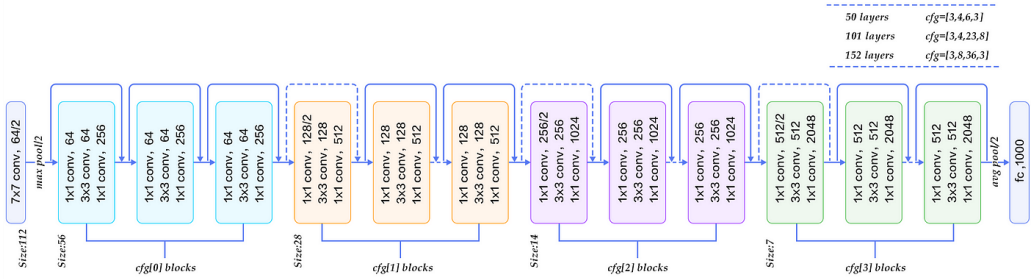
A solução da ResNet para esse problema são as conexões de atalho. Elas são usadas para propagar informações de camadas anteriores na rede para camadas posteriores, criando um caminho alternativo. Cada conexão de atalho consegue "pular" um bloco residual (conjunto de três camadas convolucionais) e se junta ao fluxo principal em uma função de ativação ReLU ao fim do bloco residual (ELGENDY, 2020).

A Figura 29 ilustra uma ResNet-50, onde é possível visualizar os blocos residuais e os atalhos que começam antes e terminam no fim de cada bloco. A ResNet-50 pode ser dividida da seguinte forma (ELGENDY, 2020):

- **Estágio 1**: Camada convolucional 7×7;
- **Estágio 2**: 3 blocos residuais, cada um contendo (uma camada convolucional 1x1, uma camada convolucional 3x3 e mais uma camada convolucional 1x1) = 9 camadas convolucionais;
- **Estágio 3**: 4 blocos residuais = total de 12 camadas convolucionais;
- **Estágio 4**: 6 blocos residuais = total de 18 camadas convolucionais;

- **Estágio 5**: 3 blocos residuais = total de 9 camadas convolucionais, e
- **Estágio 6**: Camada totalmente conectada utilizando a função *softmax*.

Figura 29 – Arquitetura de uma ResNet-50



Fonte: (RASTOGI, 2022)

Somando todas camadas previamente descritas (1+9+12+18+9+1), temos o total de 50 camadas, que dá o nome à rede.

A rede foi treinada inteiramente com a base de faturas sem usar um conjunto de pesos prévios, no treinamento foram utilizadas técnicas como o *learning rate decay* (diminui a taxa de aprendizado no momento em que a rede começa a oscilar os pesos para cima e para baixo do ponto de convergência) e *batch normalization* (normaliza os *feature maps* gerados pelas camadas convolucionais). As duas técnicas ajudam a garantir um bom desempenho da rede neural e sua aplicação é facilitada através do Keras.

A RetinaNet, elucidada no Referencial Teórico, foi a rede para detecção de objetos utilizada, no seu núcleo está uma rede neural convolucional (CNN) para extração de features. A CNN utilizada foi a ResNet-50, com a remoção da camada completamente conectada. Com a rede treinada a ferramenta é capaz de receber faturas nos formato PNG e retornar as caixas delimitadoras encontradas, juntamente com sua identificação.

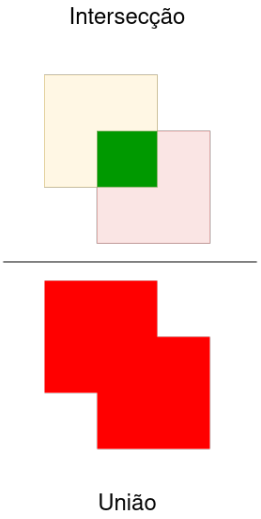
Ao realizar o treinamento da rede foi utilizado o conjunto de treinamento e o conjunto de validação. O conjunto de treinamento possui a função de atualizar os pesos da rede. Já o conjunto de validação não atualiza os pesos, mas mostra durante o processo de treinamento como a rede se comporta para dados que ela não conhece. Dessa forma, foi possível verificar casos de *overfitting* e *underfitting* e ajustes foram realizados a rede rapidamente.

5.3.4 Métricas de Avaliação

Existem diversas métricas para avaliar modelos de detecção de objetos, com o objetivo de avaliar a solução construída foram utilizados: IoU (*intersection over union*), TP (*true positive*), FN (*false negative*), FP (*false positive*), *recall*, *precision*, curva de *recall-precision*, AP (*average precision*) e mAP (*mean average precision*).

O IoU é uma métrica que avalia o quão próximas são duas caixas delimitadoras, seu valor varia entre 0 (não há intersecção) e 1 (as caixas tem medidas idênticas). Ele realiza a divisão entre a área de intersecção sobre a área de união de duas caixas, como ilustra a Figura 30.

Figura 30 – Ilustração do algoritmo de IoU



Fonte: Autor

Ao realizar uma detecção a rede anexa as caixas delimitadoras um valor de confiança (entre 0 e 1), de forma que, caixas com maior valor de confiança são mais prováveis de estarem corretas em relação a posição e classe identiticada. A partir dos valores de confiança gerados pela rede o usuário pode definir um limítrofe de confiança, no qual a rede só vai retornar predições com valor de confiança acima do limítrofe.

As métricas de TP, FN e FP dependem do IoU e do limítrofe de confiança definido pelo usuário. De forma que, TP é uma predição correta de uma caixa delimitadora, contendo IoU acima da porcentagem limite especificada e classificada com a classe correta. FN é uma caixa delimitadora que a rede não identificou corretamente. Enquanto FP é uma classificação incorreta da rede (seja por estar identificada com a classe errada, com IoU abaixo do limite ou por outra caixa já ter feito a classificação daquele objeto com um IoU superior) (SHARMA, 2022).

Ao rodar a base de dados de validação, as métricas de TP, FN e FP são obtidas para cada classe de objetos que a rede identifica. No caso da solução aplicada, ela identifica 5 tipos de objetos distintos (endereço, instalação, faturamento, impostos e leitura), cada uma das métricas descritas (TP, FN e FP) se aplicam a estas cinco classes.

A Figura 31 ilustra as identificações feitas em duas imagens, por redes diferentes e como realizar a contagem das métricas de TP, FN e FP. Dessa forma, é possível identificar que:

- No primeiro exemplo a rede identifica corretamente o objeto caracterizando um TP para a classe cachorro;
- No primeiro exemplo a rede identifica outra caixa como cachorro, mas não há cachorro ali, caracterizando um FP para a classe cachorro;
- No segundo exemplo a rede não identifica o objeto corretamente, caracterizando um FN para a classe cachorro;
- No segundo exemplo a rede identifica uma caixa como cachorro, mas ela não preenche a área limite determinada de IoU com a caixa verdadeira, caracterizando um FP para a classe cachorro, e
- No segundo exemplo a rede identifica outra caixa como gato, mas não há gato ali, caracterizando um FP para a classe gato.

A partir dos dados de TP, FN e FP é possível obter os valores de *recall* e *precision* da rede. A fórmula para calcular o *recall* é TP / (TP + FN), calcula a capacidade da rede de identificar corretamente os objetos nas imagens, a cada objeto não identificado o FN aumenta e por consequência o valor do *recall* diminui. A fórmula para calcular o *precision* é TP / (TP + FP), calcula o quão precisa a rede é, a cada nova predição incorreta o valor de FP aumenta e por consequência o valor do *precision* diminui. Os valores de *precision* e *recall* podem ser calculados por objeto ou se fazendo uma média entre todos os objetos e obtendo um valor final (SHARMA, 2022).

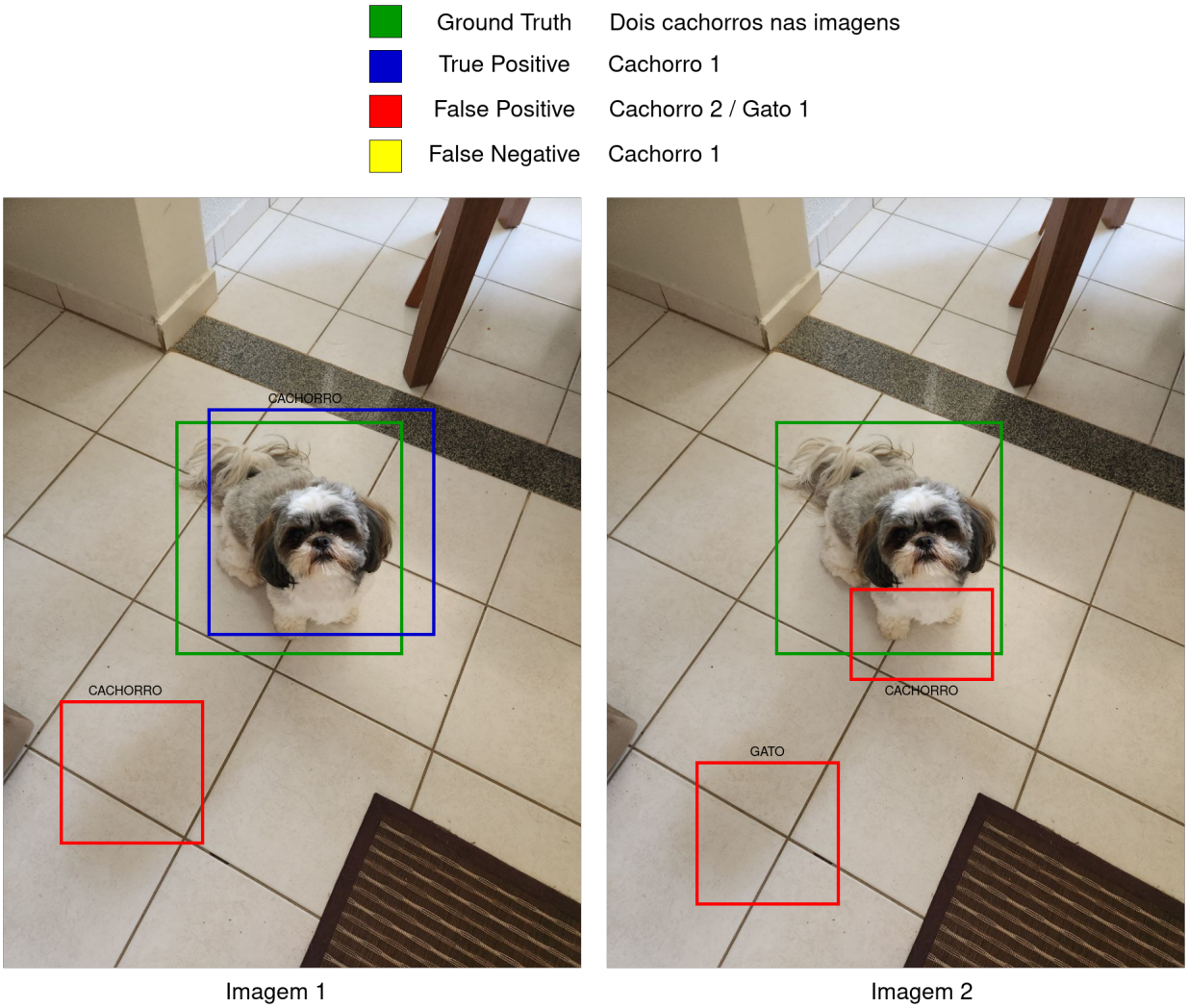
Para calcular a curva de *recall-precision* é necessário obter os valores de *recall* e *precision* dentro de um intervalo de confiança, definido pelo usuário. O parâmetro de confiança é utilizado pela rede para comunicar a "certeza" que ela tem a respeito de alguma predição. O *recall* é uma métrica que tende a aumentar conforme a confiança diminui, enquanto o *precision* tende a aumentar conforme a confiança aumenta, isso ocorre pois em limítrofes de confiança baixos a rede faz muitas predições, com isso há mais chances de acertar objetos, levando a um *recall* elevado, por outro lado são muitas caixas incorretas geradas pela rede, levando a um *precision* baixo.

O cálculo da área embaixo da curva de *recall-precision* corresponde ao AP (*average precision*) de um objeto, uma métrica que vai de 0 a 1 e agrupa todas as outras métricas mencionadas até então. A métrica final, mAP (*mean average precision*) é simplesmente uma média entre os APs de diferentes objetos (SHARMA, 2022).

5.4 Considerações Finais do Capítulo

Este capítulo especificou os modelos de redes neurais (ResNet-50 e RetinaNet), algoritmos (*learning rate decay*, *batch normalization*, entre outros) e métricas (mAP, AP,

Figura 31 – Identificação de True Positives, False Negatives e False Positives em imagens



Fonte: Autor

recall e *precision*) que foram utilizados na implementação da ferramenta proposta.

De início, foi apresentada uma contextualização a respeito da EGE, seus produtos e sua necessidade de obter dados provenientes de faturas. A solução atual foi apresentada, juntamente com os diversos problemas que ela possui. E por fim, foi apresentada a solução construída utilizando uma RetinaNet com uma ResNet-50 para identificar e categorizar blocos de faturas de energia, especificando também sobre como foi montada a base de dados, quais foram os tratamentos dados as imagens, os algoritmos utilizados, entre outros.

6 Análise de Resultados

Este capítulo tem por finalidade apresentar a análise sobre os resultados obtidos com a Ferramenta Orientada a Algoritmos de *Deep Learning* para Identificação e Categorização de Blocos de Informação em Faturas de Energia, disponível em <https://github.com/renan601/TCC2-RedesNeuraisIdentificacaoFaturas>. A Análise de Resultados foi conduzida com base na Metodologia de Análise de Resultados, já apresentada anteriormente. Essa estabelece um protocolo de Pesquisa-Ação, o qual compreende as fases de Diagnóstico, Coleta e Análise dos Dados, Elaboração do Plano de Ação e Divulgação dos Resultados.

6.1 Estrutura dos Ciclos de Teste

Conforme descrito na Metodologia de Análise de Resultados, este trabalho segue as fases da Pesquisa-Ação. Os ciclos de pesquisa-ação são aplicados dentro de quatro cenários formulados pelo autor, a cada novo cenário a ferramenta recebe mais dados de treinamento e precisa extrair mais informações. Os aprendizados obtidos através dos ciclos de pesquisa-ação em um cenário inicial são levados ao próximo cenário de mais complexidade.

As métricas analisadas ao fim de cada ciclo de pesquisa-ação são: *recall* (por objeto), *precision* (por objeto), *average precision* (por objeto) e *mean average precision* (geral). Juntamente as métricas serão apresentados gráficos ilustrando o avanço de aprendizado no processo de treinamento e exemplos das predições da rede.

6.2 Cenário de Teste - 1

O primeiro cenário de teste tem como objetivo verificar o comportamento inicial da rede, utilizando uma combinação reduzida de faturas e objetos a serem extraídos, permitindo um treinamento inicial mais veloz. Ele possui a seguinte composição:

- **Entrada (Treinamento)**: 500 imagens de faturas, todas elas pertencentes a um modelo de baixa tensão da distribuidora CEMIG;
- **Entrada (Validação)**: 75 imagens de faturas, todas elas pertencentes a um modelo de baixa tensão da distribuidora CEMIG;
- **Formato da Entrada**: Cada imagem tem tamanho 640 de altura, por 640 de largura e 3 canais de cores, utilizando o formato PNG;

- **Tratamento das Imagens na Entrada**: As imagens foram submetidas a um processo de redimensionamento prévio, mantendo sua proporção original altura x largura. De forma que, a largura é complementada com o processo de *padding*, adicionando bordas pretas na imagem fazendo com que ela fique quadrada (640x640);

- **Tratamento das Caixas Delimitadoras na Entrada**: As caixas delimitadoras são redimensionadas juntamente com as imagens, e seu valor em pixels é dividido pela sua dimensão, os valores de X são divididos pela largura, e os valores de Y pela altura. Dessa forma, o valor das coordenadas (x1,y1,x2,y2) de uma caixa é caracterizado como relativo (a altura e largura da imagem) e seu valor está entre 0 e 1;

- **Informações Extraídas**: As informações a serem identificadas para o primeiro cenário são: Endereço, Instalação e Tabela de Faturamento, e

- **Formato da Rede**: A rede utilizada para detecção de objetos foi uma RetinaNet em conjunto com uma rede convolucional ResNet-50.

6.2.1 Ciclo de Teste - 1

No primeiro ciclo de teste a rede utilizada para realizar o treinamento foi uma versão com um menor número de filtros da ResNet-50. O algoritmo de SGD (*stochastic gradient descent*) foi usado na função de otimização juntamente com a técnica de Piecewise Constant Decay que ajusta os pesos da rede de acordo com uma lista de iterações e uma lista de valores de *learning rate* informados diretamente no código.

O modelo foi treinado por 55 épocas e o comportamento do erro e do erro de validação durante o treinamento está ilustrado na Figura 32.

Resultados: A partir do cenário estabelecido e da construção da primeira versão da ferramenta foi possível realizar a execução do treinamento e coletar as métricas finais a partir das 75 imagens de validação (não utilizadas no treinamento). Utilizando limites de confiança de 0.5 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 75, Instalação: 75, Faturamento: 74;

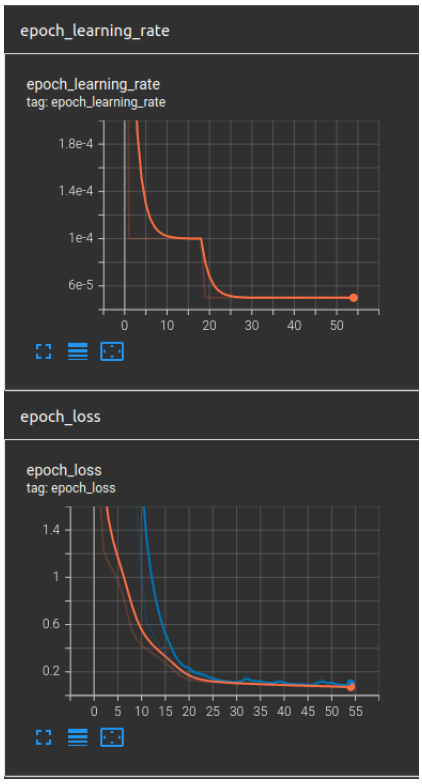
- **FN (False Negative)**: Endereço: 0, Instalação: 0, Faturamento: 1;

- **FP (False Positive)**: Endereço: 75, Instalação: 0, Faturamento 0;

- ***recall***: 0.99, e

- ***precision***: 0.83.

Figura 32 – (Cenário 1 - Ciclo 1) Comportamento do *learning rate* e dos erros de treinamento (linha laranja) e validação (linha azul) durante o processo de treinamento



Fonte: Autor

Utilizando limites de confiança de 0.85 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 65, Instalação: 41, Faturamento: 0;
- **FN (False Negative)**: Endereço: 10, Instalação: 34, Faturamento: 75;
- **FP (False Positive)**: Endereço: 0, Instalação: 0, Faturamento: 0;
- ***recall***: 0.47, e
- ***precision***: 0.67.

Utilizando limites de confiança de 0.25 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 75, Instalação: 75, Faturamento: 75;
- **FN (False Negative)**: Endereço: 0, Instalação: 0, Faturamento: 0;
- **FP (False Positive)**: Endereço: 212, Instalação: 0, Faturamento: 241;

- ***recall***: 1, e
- ***precision***: 0.5.

Por fim, calculando o *average precision* (AP) e o *mean average precision* (mAP) utilizando um intervalo de confiança de 0.3 até 0.99 e IoU de 0.75, os seguintes resultados foram obtidos:

- **AP Endereço**: 0.9734;
- **AP Instalação**: 0.9935;
- **AP Faturamento**: 0.967, e
- **mAP**: 0.978.

Análise dos Resultados: Analisando o comportamento da rede ao capturar objetos com índice de confiança acima de 0.5, quase todas as caixas delimitadoras receberam uma previsão correta (com IoU acima de 0.75), o que é um bom resultado aliado ao baixo número de FP para as categorias de Instalação e Faturamento. O FP foi elevado na categoria de Endereço indicando que outro pedaço da imagem confundiu a rede em relação a esse objeto.

Ao aumentar o nível de confiança exigido para 0.85 a rede perdeu uma boa porção das suas previsões corretas (demonstrando que as caixas delimitadoras não estão com um índice de confiança alto), gerando objetos sem identificação (FN), mas oferecendo uma boa precisão devido ao baixo valor de FP.

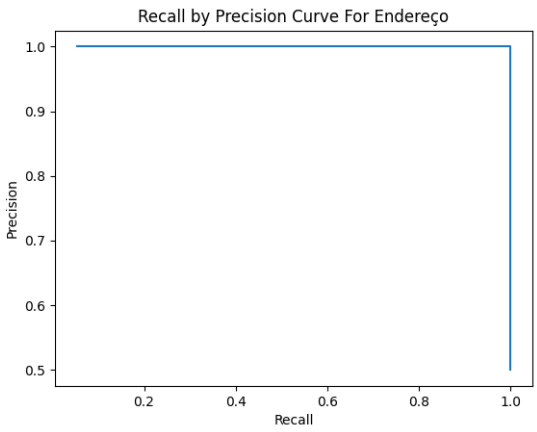
Ao diminuir o nível de confiança exigido para 0.25, a rede perde muito da sua precisão (aumento de FP), apresentando várias caixas identificadas incorretamente ou que não tem um bom nível de IoU com a caixa verdadeira.

Os valores de AP e mAP obtidos foram bem altos, em razão de uma curva balanceada entre *precision* e *recall*, como mostra a Figura 33, ilustrando a curva *recall-precision* para o objeto de Endereço.

Apesar do comportamento da rede dentro dos intervalos de confiança ser o esperado (aumento de FP em níveis de confiança baixos e aumento de FN em níveis de confiança altos), a base de dados utilizada com um só modelo é demasiada estática e previsível, principalmente porque dentro de um modelo específico, as posições dos itens são fixas e existe a possibilidade da rede estar decorando as posições dos objetos e não aprendendo os seus formatos.

Dois outros problemas notados foram o tempo necessário para execução de uma época e a necessidade de ficar monitorando constantemente o processo de treinamento, a fim de verificar se o erro de validação parou de cair (a rede parou de aprender).

Figura 33 – (Cenário 1 - Ciclo 1) Curva *recall-precision* para objeto Endereço



Fonte: Autor

Plano de Ação: Ainda dentro do cenário 1 o autor realizou outro ciclo de pesquisa-ação contendo as seguintes melhorias:

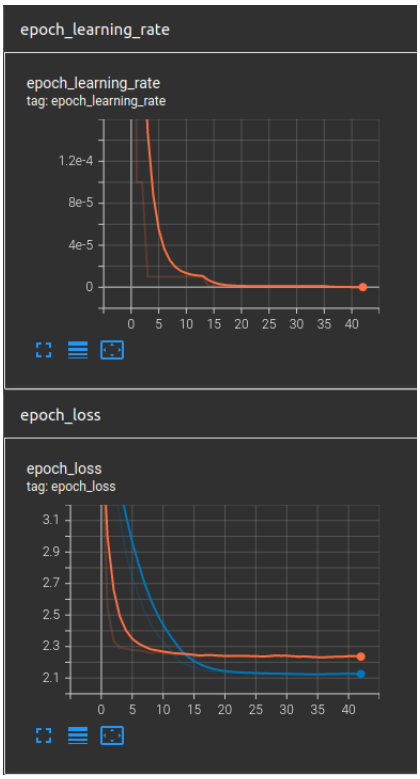
- ***Data Augmentation* na Entrada**: Modificar parte das imagens em relação a seu brilho, contraste e saturação. Além de fazer um corte aleatório em algum trecho da imagem (sem cortar uma caixa delimitadora) e por fim ajustar o *padding* para a imagem voltar ao tamanho de 640x640;
- **Implementação do Early Stopping**: Utilizar técnica para interromper treinamento ao sinal de que o erro de validação parou de cair, ajudando a previnir o *overfitting* (a rede treina tanto nos dados de entrada que pode acabar decorando-os), e
- **Aumento no *Batch Size***: Verificar *batch size* máximo possível que não exceda o limite de memória da placa de vídeo, de modo a diminuir o tempo de treinamento total.

6.2.2 Ciclo de Teste - 2

Para o segundo ciclo de teste foi implementada uma pipeline de *data augmentation* para impedir que a rede decore as posições dos itens, além de permitir que ela esteja melhor preparada a receber dados diferentes. Também foi utilizado o algoritmo de EarlyStopping para que a rede pare de ser treinada ao detectar que o aprendizado parou. E por fim, o aumento do *batch size* com objetivo de minimizar o tempo de treinamento.

O modelo foi treinado por 40 épocas, parando após 6 épocas consecutivas em que não houve diminuição do erro de validação. O comportamento do erro e do erro de validação durante o treinamento está ilustrado na Figura 34.

Figura 34 – (Cenário 1 - Ciclo 2) Comportamento do *learning rate* e dos erros de treinamento (linha laranja) e validação (linha azul) durante o processo de treinamento



Fonte: Autor

Resultados: A partir do cenário estabelecido e da modificação da ferramenta com as alterações propostas no ciclo anterior foi possível realizar a execução do treinamento e coletar as métricas finais a partir das 75 imagens de validação (não utilizadas no treinamento). Utilizando limites de confiança de 0.5 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 0, Instalação: 0, Faturamento: 0;
- **FN (False Negative)**: Endereço: 75, Instalação: 75, Faturamento 75;
- **FP (False Positive)**: Endereço: 0, Instalação: 0, Faturamento 0;
- ***recall***: 0, e
- ***precision***: 0.

Utilizando limites de confiança de 0.25 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 0, Instalação: 0, Faturamento: 0;

- **FN (False Negative)**: Endereço: 75, Instalação: 75, Faturamento 75;
- **FP (False Positive)**: Endereço: 247, Instalação: 0, 2: Faturamento 0;
- ***recall***: 0, e
- ***precision***: 0.

Por fim, calculando o AP e o mAP utilizando um intervalo de confiança de 0.3 até 0.99 e IoU de 0.5, os seguintes resultados foram obtidos:

- **AP Endereço**: 0;
- **AP Instalação**: 0;
- **AP Faturamento**: 0, e
- **mAP**: 0.

Análise dos Resultados: Analisando o comportamento da rede é possível verificar que ela não foi bem sucedida em identificar nenhum dos objetos, mesmo operando sob índices de confiança baixos. O valor final do erro de validação e do erro treinamento foram altos, indicando que a rede não conseguiu aprender os padrões de identificação dos objetos, um caso de *underfitting*, a rede tem um erro alto mesmo nos dados de treinamento.

O motivo principal do desempenho ruim nessa versão da rede está ligado a quantidade de dados treinamento disponíveis e sua variedade. O processo de *data augmentation* foi utilizado para modificar imagens existentes e não para gerar novas imagens, dessa forma, se diversificou muito a entrada da rede, mas ela continuou operando com poucos exemplos de treinamento (500).

Um outro problema verificado foi a função de otimização (SGD), junto ao algoritmo que é responsável pela queda do *learning rate* (Piecewise Constant Decay), que tornam a rede lenta pra convergir em um bom resultado. Os paramêtros do Piecewise Constant Decay para queda do *learning rate* são muito arbitrários e de difícil definição.

Plano de Ação: As melhorias levadas do último ciclo de pesquisa-ação do cenário 1 para os ciclos do cenário 2 são:

- **Adam**: As redes do primeiro ciclo foram treinadas em torno de 50 épocas cada, ao aumentar a quantidade de dados o autor utilizou o Adam como algoritmo de otimização, ele consegue realizar o aprendizado de forma mais rápida que o SGD e com resultados finais próximos, sem necessidade do autor fornecer múltiplos parâmetros e definições;

- ***Learning Rate* Fixo**: Definir um valor de *learning rate* fixo e parar de utilizar o Piecewise Constant Decay. Mesmo com um valor fixo bem baixo definido para o *learning rate*, o Adam é capaz de convergir em um bom resultado rapidamente, e
- **Aumento na Quantidade de Dados na Entrada**: A base de dados de treinamento deve ser maior que a do cenário 1 para permitir a utilização da técnica de *data augmentation*.

6.3 Cenário de Teste - 2

O segundo cenário de teste tem como objetivo aumentar a quantidade de modelos e imagens de treinamento, para verificar o aprendizado da rede lidando com maior quantidade de faturas e modelos distintos, de modo que também seja possível aplicar as técnicas e algoritmos mencionados nos ciclos de pesquisa-ação aplicados no cenário 1, como a técnica de *data augmentation* que não pode ser aplicada com sucesso com poucos dados. O cenário possui a seguinte composição:

- **Entrada (Treinamento)**: 4800 imagens de faturas, pertencentes a modelos de baixa e média tensão da distribuidora CEMIG;
- **Entrada (Validação)**: 720 imagens de faturas, pertencentes a modelos de baixa e média tensão da distribuidora CEMIG;
- **Formato da Entrada**: Cada imagem tem tamanho 640 de altura, por 640 de largura e 3 canais de cores, utilizando o formato PNG;
- **Tratamento das Imagens na Entrada**: As imagens foram submetidas a um processo de redimensionamento prévio, mantendo sua proporção original altura x largura. De forma que, a largura é complementada com o processo de *padding*, adicionando bordas pretas na imagem fazendo com que ela fique quadrada (640x640);
- **Transformação das Imagens em Tempo de Execução - *Data Augmentation***: Metade das imagens é modificada em relação a seu brilho, contraste e saturação. Além de sofrerem um corte aleatório e um reajuste de *padding* para a imagem voltar ao tamanho de 640x640;
- **Tratamento das Caixas Delimitadoras na Entrada**: As caixas delimitadoras são redimensionadas juntamente com as imagens, e seu valor em pixels é dividido pela sua dimensão, os valores de X são divididos pela largura, e os valores de Y pela altura. Dessa forma, o valor das coordenadas (x1,y1,x2,y2) de uma caixa é caracterizado como relativo (a altura e largura da imagem) e seu valor está entre 0 e 1;

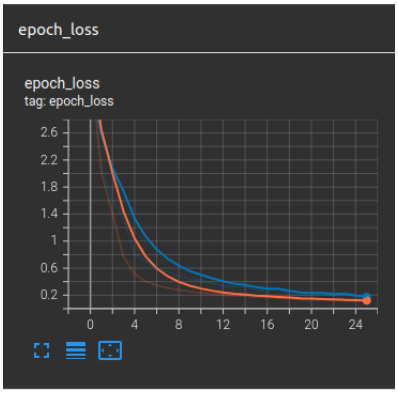
- **Informações Extraídas**: As informações identificadas para o segundo cenário são: Endereço, Instalação, Tabela de Faturamento, Tabela de Tributos e Tabela de Leitura;
- **Formato da Rede**: A rede utilizada para detecção de objetos foi uma RetinaNet em conjunto com uma rede convolucional ResNet-50, e
- **Hiperparâmetros da Rede**: A rede utilizou o Adam como função de erro para convergir em um resultado mais rapidamente, *batch size* de tamanho 16 para diminuir o tempo de treinamento e Early Stopping para evitar *overfitting*.

6.3.1 Ciclo de Teste - 1

Para o primeiro ciclo de teste do cenário 2, foram utilizadas todas as faturas disponíveis da CEMIG, de modo que haja um balanceamento entre os diferentes modelos. A rede utilizada é uma ResNet-50 com número de filtros reduzido, usando o Adam como algoritmo de otimização e aplicando *data augmentation*.

O modelo foi treinado por 25 épocas, parando após 6 épocas consecutivas em que não houve diminuição do erro de validação. O comportamento do erro e do erro de validação durante o treinamento está ilustrado na Figura 35.

Figura 35 – (Cenário 2 - Ciclo 1) Comportamento dos erros de treinamento (linha laranja) e validação (linha azul) durante o processo de treinamento



Fonte: Autor

Resultados: A partir do cenário estabelecido e da modificação da ferramenta com as alterações propostas no ciclo anterior foi possível realizar a execução do treinamento e coletar as métricas finais a partir das 720 imagens de validação (não utilizadas no treinamento). Utilizando limites de confiança de 0.5 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 503, Instalação: 404, Faturamento: 123, Impostos: 0, Leitura: 206;

- **FN (False Negative)**: Endereço: 3, Instalação: 102, Faturamento 383, Impostos: 506, Leitura: 300;
- **FP (False Positive)**: Endereço: 0, Instalação: 0, Faturamento 382, Impostos: 0, Leitura: 506;
- ***recall***: 0.488, e
- ***precision***: 0.506.

Utilizando limites de confiança de 0.5 e IoU de 0.9, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 339, Instalação: 366, Faturamento: 123, Impostos: 0, Leitura: 206;
- **FN (False Negative)**: Endereço: 167, Instalação: 140, Faturamento 383, Impostos: 506, Leitura: 300;
- **FP (False Positive)**: Endereço: 164, Instalação: 38, Faturamento 382, Impostos: 0, Leitura: 506;
- ***recall***: 0.408, e
- ***precision***: 0.422.

Por fim, calculando o AP e o mAP utilizando um intervalo de confiança de 0.3 até 0.99 e IoU de 0.75, os seguintes resultados foram obtidos:

- **AP Endereço**: 0.994;
- **AP Instalação**: 0.992;
- **AP Faturamento**: 0.208;
- **AP Impostos**: 0;
- **AP Leitura**: 0.312, e
- **mAP**: 0.5012.

Análise dos Resultados: Analisando o comportamento da rede ao capturar objetos com índice de confiança acima de 0.5 e IoU acima de 0.75, foi possível verificar um resultado excelente para o campo de Endereço, razoável para Instalação e ruim para os demais campos, principalmente o campo de Impostos que não foi identificado corretamente

nenhuma vez e a rede nem chegou a fazer tentativas. Em relação ao campo de Leitura, a rede está identificando-o corretamente em faturas de média tensão, mas não em faturas de baixa tensão, provavelmente devido a diferença profunda entre esses campos nos diferentes tipos de faturas. Além da diferença notável no formato há ainda outro agravante, faturas de média tensão tem uma estrutura para guardar informações adicionais que se parece muito com o campo de Leitura de uma fatura de baixa tensão, confundindo a rede.

Ao analisar o comportamento da rede com confiança de 0.5 e IoU de 0.9 algumas predições deixam de ser corretas, mas estão dentro do esperado para um IoU tão elevado. O campo de Faturamento parece estar com algum problema de generalização, onde uma parte pequena dos campos é identificada de forma precisa e a parte maior não é identificada.

Os valores de AP e mAP refletem o que foi mostrado nos resultados das outras métricas, com bons valores para Endereço e Instalação, e valores ruins para os outros campos.

Plano de Ação: As melhorias levadas do primeiro ciclo de pesquisa-ação do cenário 2 para os próximos ciclos são:

- **Ajuste de Caixa Delimitadora**: As caixas delimitadoras do campo de Faturamento foram ajustadas, pois foi identificado que elas não capturavam a totalidade do bloco que o campo estava inserido, isso pode estar prejudicando a generalização;

- **Diferenciar Leitura para Média e Baixa Tensão**: Criação de um novo tipo de objeto para leitura de média tensão, antigo objeto foi usado só para leituras de baixa tensão;

- ***Learning Rate* Adaptativo**: De modo a iniciar o treinamento com bom nível de aprendizado e convergir no melhor resultado possível foi utilizado o ReduceLROnPlateou do Keras, garantindo a queda do *learning rate* para 40% do seu valor original depois de três épocas sem evolução no erro de validação, e

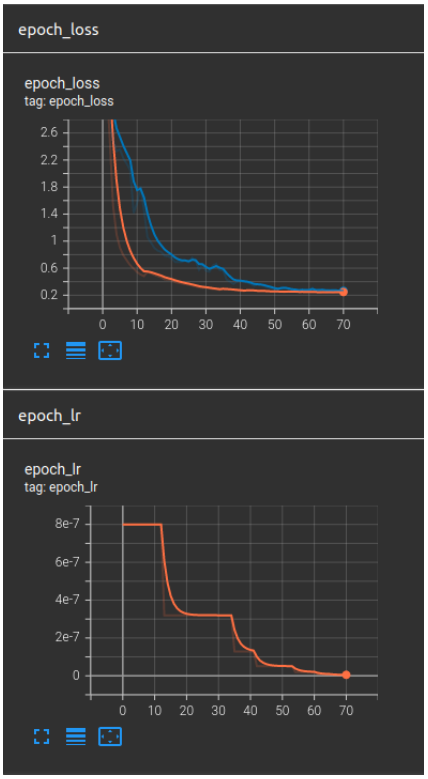
- **Mudança de Hiperparâmetros na RetinaNet**: No processo de treinamento a rede prevê diversas caixas em posições distintas nas imagens e compara essas caixas com as caixas delimitadoras fornecidas na entrada. Caso o nível de IoU seja acima de um limite pré-definido a rede vai prosseguir com essa caixa e tentar ajustar seu formato para melhor se encaixar no objeto. O limite no IoU para descarte de caixas foi reduzido para evitar o descarte de todas as caixas que tentam identificar o campo de Impostos.

6.3.2 Ciclo de Teste - 2

Para o segundo ciclo de teste do cenário 2, foram utilizadas todas as faturas disponíveis da CEMIG, de modo que haja um balanceamento entre os diferentes modelos. A rede utilizada é uma ResNet-50 com número de filtros reduzido, usando o Adam como algoritmo de otimização, reduzindo o *learning rate* quando não há aprendizado e aplicando *data augmentation*. Foram feitas mudanças de parâmetros da RetinaNet e algumas mudanças na identificação de classes e nas posições de caixas delimitadoras para tentar melhorar a generalização do modelo.

O modelo foi treinado por 70 épocas, parando após 8 épocas consecutivas em que não houve diminuição do erro de validação, demorou um total de 2 horas e 43 minutos para ser treinado. O comportamento do erro, erro de validação e *learning rate* durante o treinamento está ilustrado na Figura 36.

Figura 36 – (Cenário 2 - Ciclo 2) Comportamento do *learning rate* e dos erros de treinamento (linha laranja) e validação (linha azul) durante o processo de treinamento



Fonte: Autor

Resultados: A partir do cenário estabelecido e da modificação da ferramenta com as alterações propostas no ciclo anterior foi possível realizar a execução do treinamento e coletar as métricas finais a partir das 720 imagens de validação (não utilizadas no treinamento). Utilizando limites de confiança de 0.5 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 505, Instalação: 505, Faturamento: 464, Impostos: 480, Leitura: 0, LeituraMV: 206;
- **FN (False Negative)**: Endereço: 1, Instalação: 1, Faturamento 42, Impostos: 26, Leitura: 300, LeituraMV: 0;
- **FP (False Positive)**: Endereço: 16, Instalação: 2, Faturamento 75, Impostos: 23, Leitura: 0, LeituraMV: 512;
- ***recall***: 0.85375, e
- ***precision***: 0.77474.

Por fim, calculando o AP e o mAP utilizando um intervalo de confiança de 0.3 até 0.99 e IoU de 0.75, os seguintes resultados foram obtidos:

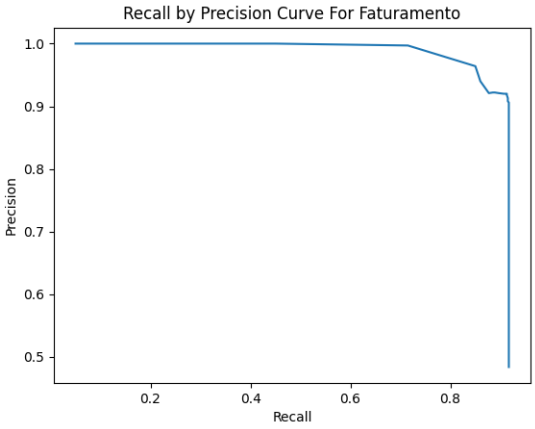
- **AP Endereço**: 0.96995;
- **AP Instalação**: 0.98592;
- **AP Faturamento**: 0.85911;
- **AP Impostos**: 0.9052;
- **AP Leitura**: 0.097;
- **AP Leitura Média Tensão**: 0.2225, e
- **mAP**: 0.6732.

Análise dos Resultados: Ao aplicar as mudanças propostas no plano de ação do ciclo anterior, aplicando um limite de IoU de 0.75, foi possível observar melhorias substanciais na detecção dos campos de Faturamento e Impostos, porém ainda há problemas com a detecção do campo de leitura em faturas de baixa e média tensão. A Figura 37 ilustra a curva *recall-precision* para o campo de Faturamento.

Analisando o resultado em índices de confiança acima de 0.5 foi possível verificar o baixo número de FN para a maioria dos objetos, indicando que a rede está sendo bem sucedida em detectar esses objetos, porém o número de FP ainda é elevado, principalmente em relação a LeituraMV, a rede fez várias predições que não deveriam ser feitas para esse objeto.

Plano de Ação: As melhorias levadas do segundo ciclo de pesquisa-ação do cenário 2 para os próximos ciclos são:

Figura 37 – (Cenário 2 - Ciclo 2) Curva *recall-precision* para objeto Faturamento



Fonte: Autor

- **Construção de Dois Modelos de Rede Distintos**: A confusão entre blocos semelhantes que guardam informações diferentes em faturas de baixa e média tensão leva a identificação ruim do campo de Leitura, a construção de duas redes cada uma contendo seu conjunto de pesos, uma para processar faturas de baixa tensão e outra para faturas de média tensão vai contornar esses casos.

6.4 Cenário de Teste - 3

O terceiro cenário de teste tem como objetivo avaliar a aplicação da rede em faturas de baixa tensão de distribuidoras distintas, de modo que também seja possível aplicar as técnicas e algoritmos mencionados nos ciclos de pesquisa-ação aplicados nos cenários 1 e 2, como a técnica de *data augmentation*, *learning rate* adaptativo, *early stopping*, entre outros. O cenário possui a seguinte composição:

- **Entrada (Treinamento)**: 3950 imagens de faturas, pertencentes a modelos de baixa tensão das distribuidoras CEMIG e CPFL;
- **Entrada (Validação)**: 590 imagens de faturas, pertencentes a modelos de baixa tensão das distribuidoras CEMIG e CPFL;
- **Formato da Entrada**: Cada imagem tem tamanho 640 de altura, por 640 de largura e 3 canais de cores, utilizando o formato PNG;
- **Tratamento das Imagens na Entrada**: As imagens foram submetidas a um processo de redimensionamento prévio, mantendo sua proporção original altura x largura. De forma que, a largura é complementada com o processo de *padding*, adicionando bordas pretas na imagem fazendo com que ela fique quadrada (640x640);

- **Transformação das Imagens em Tempo de Execução - *Data Augmentation***: Metade das imagens é modificada em relação a seu brilho, contraste e saturação. Além de sofrerem um corte aleatório e um reajuste de *padding* para a imagem voltar ao tamanho de 640x640;

- **Tratamento das Caixas Delimitadoras na Entrada**: As caixas delimitadoras são redimensionadas juntamente com as imagens, e seu valor em pixels é dividido pela sua dimensão, os valores de X são divididos pela largura, e os valores de Y pela altura. Dessa forma, o valor das coordenadas (x1,y1,x2,y2) de uma caixa é caracterizado como relativo (a altura e largura da imagem) e seu valor está entre 0 e 1;

- **Informações Extraídas**: As informações identificadas são: Endereço, Instalação, Tabela de Faturamento, Tabela de Tributos e Tabela de Leitura;

- **Formato da Rede**: A rede utilizada para detecção de objetos foi uma RetinaNet em conjunto com uma rede convolucional ResNet-50, e

- **Hiperparâmetros da Rede**: A rede utilizou o Adam como função de erro para convergir em um resultado mais rapidamente, *batch size* de tamanho 16 para diminuir o tempo de treinamento, Early Stopping para evitar *overfitting* além de alguns parâmetros personalizados para a RetinaNet de forma a identificar todos os campos.

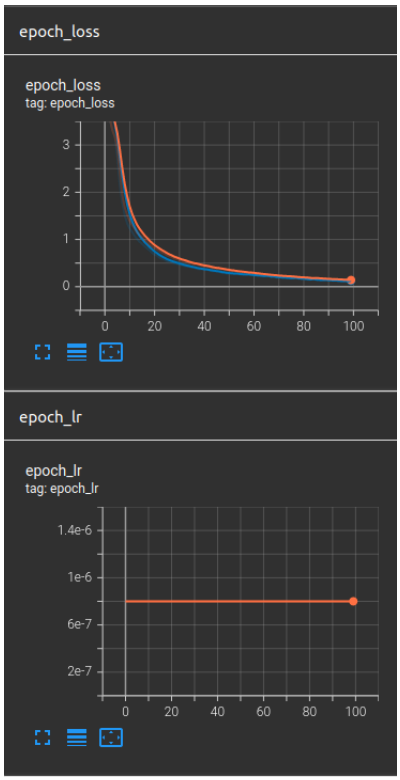
6.4.1 Divulgação dos Resultados - Rede Baixa Tensão

Para a construção de uma das redes finais da ferramenta foram utilizadas as faturas de baixa tensão da CEMIG e CPFL, de modo que haja um balanceamento entre os diferentes modelos. A rede utilizada é uma ResNet-50 com número de filtros reduzido, usando o Adam como algoritmo de otimização, reduzindo o *learning rate* quando não há aprendizado e aplicando *data augmentation*. Foram feitas mudanças de parâmetros da RetinaNet e algumas mudanças na identificação de classes e nas posições de caixas delimitadoras para tentar melhorar a generalização do modelo.

O modelo foi treinado por 100 épocas e foi interrompido a partir do momento em que o erro de validação passou a evoluir de forma bem lenta, demorou um total de 3 horas e 30 minutos para ser treinado. O comportamento do erro, erro de validação e *learning rate* durante o treinamento está ilustrado na Figura 38.

A partir do cenário estabelecido e da modificação da ferramenta com as alterações propostas nos ciclos anteriores foi possível realizar a execução do treinamento e coletar as métricas finais a partir das 590 imagens de validação (não utilizadas no treinamento). Utilizando limites de confiança de 0.5 e IoU de 0.75, os seguintes resultados foram obtidos:

Figura 38 – (Rede Final - Baixa Tensão) Comportamento do *learning rate* e dos erros de treinamento (linha laranja) e validação (linha azul) durante o processo de treinamento



Fonte: Autor

- **TP (True Positive)**: Endereço: 590, Instalação: 590, Faturamento: 584, Impostos: 300, Leitura: 590;
- **FN (False Negative)**: Endereço: 0, Instalação: 0, Faturamento: 6, Impostos: 0, Leitura: 0;
- **FP (False Positive)**: Endereço: 67, Instalação: 18, Faturamento: 18, Impostos: 0, Leitura: 42;
- ***recall***: 0.998, e
- ***precision***: 0.954.

Utilizando limites de confiança de 0.85 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 590, Instalação: 587, Faturamento: 583, Impostos: 298, Leitura: 589;
- **FN (False Negative)**: Endereço: 0, Instalação: 3, Faturamento: 7, Impostos: 2, Leitura: 1;

- **FP (False Positive)**: Endereço: 0, Instalação: 0, Faturamento: 7, Impostos: 2, Leitura: 1;
- ***recall***: 0.9948, e
- ***precision***: 0.9975.

Utilizando limites de confiança de 0.5 e IoU de 0.9, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 281, Instalação: 328, Faturamento: 325, Impostos: 46, Leitura: 326;
- **FN (False Negative)**: Endereço: 309, Instalação: 262, Faturamento: 265, Impostos: 254, Leitura: 264;
- **FP (False Positive)**: Endereço: 376, Instalação: 280, Faturamento: 277, Impostos: 254, Leitura: 306;
- ***recall***: 0.4578, e
- ***precision***: 0.4352.

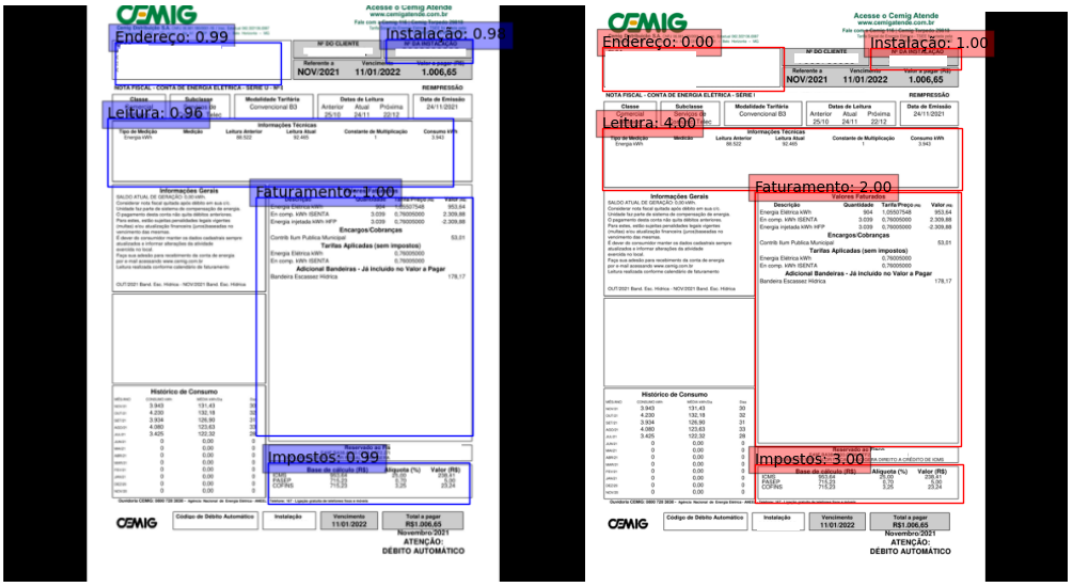
Por fim, calculando o AP e o mAP utilizando um intervalo de confiança de 0.3 até 0.99 e IoU de 0.75, os seguintes resultados foram obtidos:

- **AP Endereço**: 0.95;
- **AP Instalação**: 0.85;
- **AP Faturamento**: 0.88798;
- **AP Impostos**: 0.9;
- **AP Leitura**: 0.9, e
- **mAP**: 0.8975.

A Figura 39 ilustra um exemplo de fatura da CEMIG comparando as caixas delimitadoras de referência e as previstas pela rede.

A Figura 40 ilustra um exemplo de fatura da CPFL comparando as caixas delimitadoras de referência e as previstas pela rede.

Figura 39 – Comparação em fatura da CEMIG com caixas delimitadoras de referência em vermelho, caixas e índices de confiança previstos em azul



Fonte: Autor

Figura 40 – Comparação em fatura da CPFL com caixas delimitadoras de referência em vermelho, caixas e índices de confiança previstos em azul

Fonte: Autor

6.5 Cenário de Teste - 4

O quarto cenário de teste tem como objetivo avaliar a aplicação da rede em faturas de média tensão de distribuidoras distintas, de modo que também seja possível aplicar as técnicas e algoritmos mencionados nos ciclos de pesquisa-ação aplicados nos cenários 1, 2 e 3, como a técnica de *data augmentation*, *learning rate* adaptativo, early stopping, entre

outros. O cenário possui a seguinte composição:

- **Entrada (Treinamento)**: 4500 imagens de faturas, pertencentes a modelos de média tensão das distribuidoras CEMIG e CPFL;
- **Entrada (Validação)**: 675 imagens de faturas, pertencentes a modelos de média tensão das distribuidoras CEMIG e CPFL;
- **Formato da Entrada**: Cada imagem tem tamanho 640 de altura, por 640 de largura e 3 canais de cores, utilizando o formato PNG;
- **Tratamento das Imagens na Entrada**: As imagens foram submetidas a um processo de redimensionamento prévio, mantendo sua proporção original altura x largura. De forma que, a largura é complementada com o processo de *padding*, adicionando bordas pretas na imagem fazendo com que ela fique quadrada (640x640);
- **Transformação das Imagens em Tempo de Execução - *Data Augmentation***: Metade das imagens é modificada em relação a seu brilho, contraste e saturação. Além de sofrerem um corte aleatório e um reajuste de *padding* para a imagem voltar ao tamanho de 640x640;
- **Tratamento das Caixas Delimitadoras na Entrada**: As caixas delimitadoras são redimensionadas juntamente com as imagens, e seu valor em pixels é dividido pela sua dimensão, os valores de X são divididos pela largura, e os valores de Y pela altura. Dessa forma, o valor das coordenadas (x1,y1,x2,y2) de uma caixa é caracterizado como relativo (a altura e largura da imagem) e seu valor está entre 0 e 1;
- **Informações Extraídas**: As informações identificadas são: Endereço, Instalação, Tabela de Faturamento, Tabela de Tributos e Tabela de Leitura;
- **Formato da Rede**: A rede utilizada para detecção de objetos foi uma RetinaNet em conjunto com uma rede convolucional ResNet-50, e
- **Hiperparâmetros da Rede**: A rede utilizou o Adam como função de erro para convergir em um resultado mais rapidamente, *batch size* de tamanho 16 para diminuir o tempo de treinamento, Early Stopping para evitar *overfitting* além de alguns parâmetros personalizados para a RetinaNet de forma a identificar todos os campos.

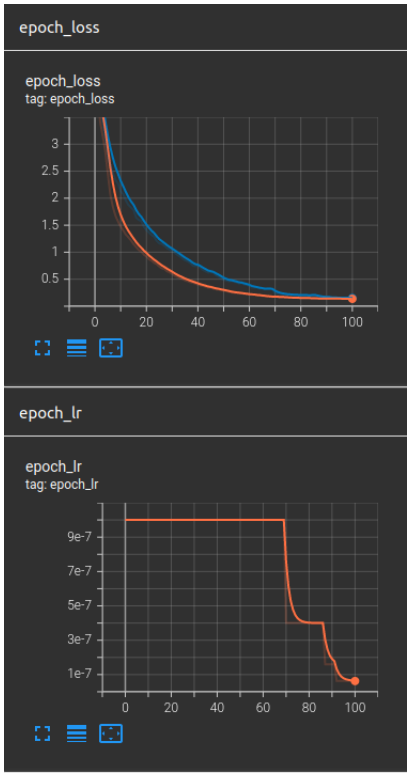
6.5.1 Divulgação dos Resultados - Rede Média Tensão

Para a construção de uma das redes finais da ferramenta foram utilizadas as faturas de média tensão da CEMIG e CPFL, de modo que haja um balanceamento entre os diferentes modelos. A rede utilizada é uma ResNet-50 com número de filtros reduzido,

usando o Adam como algoritmo de otimização, reduzindo o *learning rate* quando não há aprendizado e aplicando *data augmentation*. Foram feitas mudanças de parâmetros da RetinaNet e algumas mudanças na identificação de classes e nas posições de caixas delimitadoras para tentar melhorar a generalização do modelo.

O modelo foi treinado por 100 épocas, parando após 8 épocas consecutivas em que não houve diminuição do erro de validação, demorou um total de 3 horas e 50 minutos para ser treinado. O comportamento do erro, erro de validação e learning rate durante o treinamento está ilustrado na Figura 41.

Figura 41 – (Rede Final - Média Tensão) Comportamento do *learning rate* e dos erros de treinamento (linha laranja) e validação (linha azul) durante o processo de treinamento



Fonte: Autor

A partir do cenário estabelecido e da modificação da ferramenta com as alterações propostas nos ciclos anteriores foi possível realizar a execução do treinamento e coletar as métricas finais a partir das 675 imagens de validação (não utilizadas no treinamento). Utilizando limites de confiança de 0.5 e IoU de 0.75, os seguintes resultados foram obtidos:

- **TP (True Positive)**: Endereço: 314, Instalação: 372, Faturamento: 308, Impostos: 206, Leitura: 287;
- **FN (False Negative)**: Endereço: 69, Instalação: 11, Faturamento: 75, Impostos: 0, Leitura: 3;

- **FP (False Positive)**: Endereço: 93, Instalação: 6, Faturamento: 152, Impostos: 4, Leitura: 320;
- ***recall***: 0.903951, e
- ***precision***: 0,7211.

Utilizando limites de confiança de 0.85 e IoU de 0.75, os seguintes resultados foram obtidos:

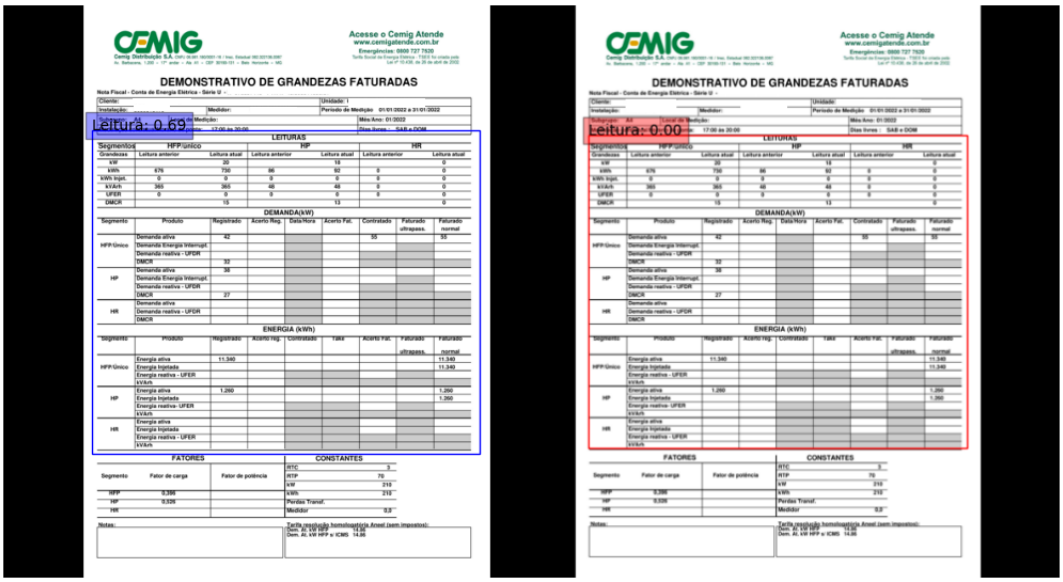
- **TP (True Positive)**: Endereço: 198, Instalação: 302, Faturamento: 296, Impostos: 204, Leitura: 56;
- **FN (False Negative)**: Endereço: 185, Instalação: 81, Faturamento: 87, Impostos: 2, Leitura: 234;
- **FP (False Positive)**: Endereço: 2, Instalação: 1, Faturamento: 52, Impostos: 0, Leitura: 101;
- ***recall***: 0.6419, e
- ***precision***: 0,8712.

Por fim, calculando o AP e o mAP utilizando um intervalo de confiança de 0.3 até 0.99 e IoU de 0.75, os seguintes resultados foram obtidos:

- **AP Endereço**: 0.7332;
- **AP Instalação**: 0.7737;
- **AP Faturamento**: 0.61056;
- **AP Impostos**: 0.9849;
- **AP Leitura**: 0.5184, e
- **mAP**: 0.72415.

A Figura 42 ilustra um exemplo de fatura da CEMIG comparando as caixas delimitadoras de referência e as previstas pela rede.

Figura 42 – Comparação em fatura da CEMIG com caixas delimitadoras de referência em vermelho, caixas e índices de confiança previstos em azul



Fonte: Autor

6.6 Considerações Finais do Capítulo

Este capítulo mostrou os resultados de algumas das redes construídas ao longo do trabalho, utilizando métricas como *recall*, *precision*, AP e mAP. A divisão em cenários permitiu trabalhar com diferentes bases de dados, que foram evoluindo de forma incremental ganhando novas distribuidoras e modelos.

Após os cenários 1 e 2 o autor verificou que a melhor opção para a detecção de objetos seriam duas redes, uma dedicada para faturas de baixa e outra para faturas de média tensão. A rede de baixa tensão atingiu valores excelentes de AP e mAP ao computar faturas da CEMIG e CPFL, já a rede de média tensão teve um desempenho razoável, pois ela conta com mais modelos e tem mais dificuldade em fazer as generalizações.

7 Conclusão

Este capítulo tem por finalidade apresentar as considerações finais em relação a ferramenta construída neste Trabalho de Conclusão de Curso. Inicialmente, apresenta-se o Contexto Geral de modo a relembrar as justificativas do trabalho. Posteriormente mostra-se o Estado do Trabalho, quais as atividades e objetivos que foram cumpridos. Além disso, em Contribuições e Limitações, serão destacadas as contribuições que este projeto oferece à comunidade, assim como suas principais limitações. A seção de Impressões do Autor em Relação ao Trabalho destaca as impressões do autor a respeito do projeto, quais foram os aprendizados e desafios encontrados. Por fim, com base nas limitações listadas, serão traçadas possíveis direções para Trabalhos Futuros.

7.1 Contexto Geral

O presente trabalho se situa dentro do problema de processamento em massa de documentos, com foco direcionado à faturas de energia. No Brasil, elas são emitidas por diferentes distribuidoras, em diferentes formatos e contém diversas informações específicas ao seu contexto, que não costumam aparecer em outros tipos de fatura.

Diversas soluções podem ser utilizadas para realizar a extração de informações dessas faturas, com diferentes níveis de automação. Em uma solução puramente manual uma pessoa precisa visualizar e coletar esses dados, sendo um trabalho laborioso, lento e até inviável com um número alto de faturas. Uma abordagem semi-automatizada baseada em "templates" seria identificar os diferentes modelos de faturas e traçar posições absolutas para se extrair informações. A solução funciona de maneira adequada para faturas "perfeitas", mas falha em itens que sofreram algum tipo de deslocamento de posição ou um processo de escaneamento.

O autor realizou uma busca na literatura por soluções para resolver o problema de como identificar e localizar dados em documentos e se deparou com soluções envolvendo *Deep Learning*, que vão desde a identificação de tabelas em documentos, até a identificação de cada tópico em um artigo científico, entre outros. Dentro da proposta, o autor aprofundou-se na pesquisa por redes para detecção de objetos, com o objetivo de localizar blocos com informações importantes em faturas de energia.

A pesquisa realizada permitiu ao autor escolher a rede para detecção de objetos RetinaNet, juntamente a rede convolucional ResNet-50, para serem treinadas em uma base de faturas de energia de duas diferentes distribuidoras. A construção da rede se deu usando bibliotecas que facilitam a construção e treinamento de redes neurais como

TensorFlow e Keras. Após testar cenários com diferentes volumes e variedade de dados o autor chegou a duas redes finais, uma dedicada a processar faturas de baixa tensão e outra dedicada a faturas de média tensão. As duas redes desempenharam um bom trabalho em identificar corretamente os blocos de informação em que elas foram treinadas, obtendo boas métricas como resultado.

7.2 Estado do Trabalho

O Objetivo Geral definido para o trabalho foi: "Desenvolvimento de uma ferramenta, cujos principais propósitos são a identificação e a categorização, de forma automática, dos blocos de informações relevantes em uma fatura de energia. Pretende-se orientar por algoritmos de *Deep Learning*, visando uma solução mais adequada ao problema."

O objetivo geral foi atendido e a ferramenta foi desenvolvida, porém, com uma diferença em relação ao planejamento, foram construídas duas redes, uma para identificar faturas de baixa tensão e outra para faturas de média tensão. O principal módulo construído foram duas redes personalizadas e treinadas em uma base de faturas de energia para extrair os campos de Endereço, Número de Instalação, Tabela de Faturamento, Tabela de Impostos e Tabela de Leitura. Ao inserir uma nova fatura e executar a ferramenta ela retorna ao usuário a identificação e localização dos blocos, junto com imagens cortadas nestas posições com o conteúdo de cada bloco.

No intuito de cumprir com o Objetivo Geral, foram estabelecidos alguns Objetivos Específicos, de menor escopo cada. Os objetivos específicos cumpridos estão listados a seguir:

- Estudo de algoritmos relacionados aos tópicos de análise e classificação de documentos, dentro das etapas de pré-processamento, segmentação de objetos e classificação de objetos;
- Estudo de tópicos de pesquisa correlatos, tais como: Redes Neurais Artificiais e *Deep Learning*;
- Definição de informações a serem extraídas e os formatos aceitos para representação das mesmas;
- Organização da base de dados, visando o treinamento dos algoritmos relacionados, já com foco na área de *Deep Learning*;
- Documentação do desenvolvimento da ferramenta, usando como base os referenciais teóricos, tecnológicos e metodológicos estabelecidos no trabalho, e

- Condução de uma análise de resultados da ferramenta, considerando uma amostra de faturas para teste.

A Questão de Pesquisa feita foi: "É possível desenvolver uma ferramenta que identifica e categoriza automaticamente os blocos de informações relevantes dentro de uma fatura de energia?"

Como foi esclarecido anteriormente, a solução é possível e foi desenvolvida, a única limitação encontrada em relação a pergunta foi o fato de que a ferramenta utiliza duas redes, uma para identificação de faturas de baixa tensão e outra para faturas de média tensão.

7.3 Contribuições e Limitações

A principal contribuição do trabalho está na base de dados construída para a realização do treinamento da rede, faturas de energia são documentos com uma quantidade alta de informações sensíveis. Dessa forma, são poucos trabalhos de referência que usam *Deep Learning* em faturas de um contexto específico. A maior parte dos trabalhos na área opta por ser genérico, capturando informações básicas como data de vencimento e valor total, mas falhando em informações únicas do mercado de energia, como histórico de consumo, número do medidor, valores de leitura, geração de energia, entre outros.

Até o momento, muito pela dificuldade de acesso à esses dados, um trabalho similar não havia sido feito. Dessa forma, uma contribuição relevante do trabalho é provar que é possível utilizar uma rede neural para identificar blocos de informações em faturas de energia no Brasil. O autor utilizou uma combinação de RetinaNet e ResNet-50, mas outras redes alternativas devem resolver o problema de forma similar.

Uma outra contribuição excelente da ferramenta construída, comparada a solução atual na empresa em que o autor trabalha, é a possibilidade de ler faturas não formatadas, ou seja, fotos de faturas com alterações pequenas de tamanho, brilho, contraste, entre outros. Isso é um feito que dificilmente uma abordagem que não utiliza *Deep Learning* consegue atingir.

A ferramenta construída também possui algumas limitações, são elas:

- Número limitado de distribuidoras atendidas, são somente duas, CEMIG e CPFL;
- Necessidade de grande volume de dados para fazer o treinamento e dar suporte a uma nova distribuidora;
- Número limitado de blocos de informações extraídos;

- Impossibilidade de fazer identificação por generalização para as distribuidoras ao qual não foi treinado;
- Possibilidade da rede gerar uma caixa delimitadora não tão ajustada ao objeto, gerando problemas em etapas seguintes de extração;
- Possibilidade pequena da rede não identificar o objeto, e
- Novos modelos de distribuidoras atendidas podem não ser corretamente identificados.

7.4 Impressões do Autor em Relação ao Trabalho

O presente Trabalho de Conclusão de Curso, tanto a primeira como a segunda parte, foram de grande aprendizado ao autor, que tinha pouca experiência prévia na área de Aprendizado de Máquina. A partir da pesquisa e construção do Referencial Teórico foi obtido um bom embasamento para conseguir realizar a construção da solução. O autor agora considera que tem uma boa proficiência sobre o tema, conseguindo compreender artigos da área e soluções construídas por terceiros.

O autor ficou satisfeito com os resultados obtidos, a evolução apresentada ao longo da Análise de Resultados e o comportamento da rede final. A Metodologia e Referencial Tecnológico construídos se adaptaram muito bem a necessidade do projeto. Tecnologias como TensorFlow e Keras foram a base da solução, enquanto outras bibliotecas como Pillow e MatPlotLib foram usadas de forma pontual. O planejamento e calendário estabecido na primeira parte do trabalho serviu de guia ao autor e a metodologia de análise de resultados se encaixou muito bem dentro do contexto de uma rede neural que busca melhorar seus resultados a cada novo processo de treinamento.

O desafio principal da primeira parte do trabalho foi obter o arcabouço de conhecimentos necessário para compreender o processo completo de identificação de objetos, o que exigiu do autor muita leitura e até releitura de material. Em relação aos desafios da segunda parte do trabalho é possível destacar:

- Dificuldade de realizar o *setup* do TensorFlow para utilizar a GPU dedicada;
- Troca da Faster R-CNN pela RetinaNet;
- Curva de aprendizado para aprender a trabalhar com os tensores do TensorFlow;
- Dificuldade na construção de um *pipeline* de dados personalizado para alimentar a rede neural, em oposição a *pipelines* utilizando bases de dados prontas encontradas com mais facilidade na internet;

- Necessidade de realizar um novo e demorado processo de treinamento para testar alguma mudança de hiperparâmetros, e
- Dificuldade no compreendimento e cálculo de algumas métricas. O Keras e TensorFlow em seus pacotes padrões tem poucas soluções para métricas de detecção de objetos, dessa forma, o autor preferiu realizar esse cálculo sem auxílio das bibliotecas principais do trabalho.

7.5 Trabalhos Futuros

Algumas sugestões para evolução do trabalho, dadas as limitações encontradas, são:

- Adição de novas distribuidoras e modelos de fatura;
- Treinar a rede com outros tamanhos de imagem;
- Utilizar uma versão com mais camadas da ResNet para melhorar capacidade de aprendizado, como a ResNet-101 e ResNet-152;
- Treinar uma CNN para identificar a distribuidora e o modelo previamente, de forma a também identificar previamente se a fatura é de baixa ou média tensão, e
- Realizar o treinamento com um conjunto de faturas mal-formatadas classificadas a mão.

Os pontos listados representam oportunidades de aprimoramento e podem ser abordados em futuros trabalhos relacionados a este projeto.

Referências

ADOBE. 2022. Disponível em: <https://www.adobe.com/sign/hub/document-types/invoice-vs-bill.html>. Acesso em: 05 de Abril de 2023. Citado na página 22.

ADOBE. *JPEG vs PNG*. 2023. Disponível em: <https://www.adobe.com/br/creativecloud/file-types/image/comparison/jpeg-vs-png.html>. Acesso em: 01 de Julho de 2023. Citado na página 73.

AHMAD, M. O.; MARKKULA, J.; OIVO, M. Kanban in software development: A systematic literature review. *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, 2013. Citado 2 vezes nas páginas 44 e 57.

ANEEL. *Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional – PRODIST*. [S.l.], 2020. Disponível em: <http://www.cerpro.com.br/publico/arquivos/prodist/Modulo11.pdf>. Citado 2 vezes nas páginas 22 e 23.

ANEEL. *Relatórios e indicadores relacionados ao setor de distribuição de energia elétrica*. 2022. Disponível em: <https://www.gov.br/aneel/pt-br/centrais-de-conteudos/relatorios-e-indicadores/distribuicao>. Acesso em: 01 de Julho de 2023. Citado na página 68.

APT. *APT Package*. 2023. Disponível em: <https://wiki.debian.org/Apt>. Acesso em: 15 de Maio de 2023. Citado na página 44.

ATLASSIAN. *What is Git: Atlassian Git Tutorial*. 2016. Disponível em: <https://www.atlassian.com/git/tutorials/what-is-git>. Citado na página 43.

AURELIEN, G. *Hands-on machine learning with scikit-learn, keras and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. [S.l.]: O'Reilly, 2023. Citado 3 vezes nas páginas 27, 32 e 47.

BARBOSA, G. et al. Segurança em redes 5g: Oportunidades e desafios em detecção de anomalias e predição de tráfego baseadas em aprendizado de máquina. *Minicursos do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, p. 145–189, 2021. Citado na página 34.

BIBME. *BibMe*. 2023. Disponível em: <https://www.bibme.org/>. Acesso em: 05 de Maio de 2023. Citado 2 vezes nas páginas 45 e 51.

CANALSOLAR. *Consumidores do grupo A faturados como grupo B*. 2019. Disponível em: <https://canalsolar.com.br/consumidores-do-grupo-a-faturados-como-grupo-b/>. Acesso em: 29 de Junho de 2023. Citado na página 68.

CHOLLET, F. *Deep learning with python*. [S.l.]: Manning Publications, 2021. Citado 8 vezes nas páginas 32, 33, 34, 37, 47, 48, 51 e 56.

DENGEL, A.; SHAFAIT, F. Analysis of the logical layout of documents. *Handbook of Document Image Processing and Recognition*, p. 177–222, 2014. Citado 2 vezes nas páginas 21 e 23.

DIAGRAMS. *Diagrams Official Website*. 2023. Disponível em: <https://app.diagrams.net/>. Acesso em: 10 de Maio de 2023. Citado na página 51.

DOERMANN, D. S. *Handbook of Document Image Processing and Recognition*. [S.l.]: Springer Reference, 2014. Citado 2 vezes nas páginas 21 e 22.

DUARTE, R. *Interface cérebro máquina híbrida utilizando amplificador EEG de baixo custo*. Tese (Doutorado), 2015. Citado na página 31.

EBERMAM, E.; KROHLING, R. Uma introdução compreensiva às redes neurais convolucionais: Um estudo de caso para reconhecimento de caracteres alfabéticos. Revista de Sistemas de Informação da FSMA, 2018. Disponível em: <https://www.fsma.edu.br/si/edicao21/FSMA_SI_2018_1_Principal_08.pdf>. Citado 2 vezes nas páginas 36 e 37.

ELGENDY, M. *Deep Learning for Vision Systems*. [S.l.]: Manning Publications Co., 2020. Citado 13 vezes nas páginas 22, 28, 29, 31, 32, 33, 35, 36, 37, 38, 49, 56 e 74.

ENERGIA, P. M. L. *Mercado Livre de Energia*. 2022. Disponível em: <https://www.mercadolivredeenergia.com.br/mercado-livre-de-energia/>. Acesso em: 29 de Junho de 2023. Citado na página 65.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de Pesquisa*. [S.l.]: Universidade Federal do Rio Grande do Sul, 2009. Citado 2 vezes nas páginas 53 e 58.

GIL, A. C. *Como elaborar Projetos de Pesquisa*. [S.l.]: Atlas, 2010. Citado 2 vezes nas páginas 53 e 58.

GIT. *Git Documentation*. 2023. Disponível em: <https://git-scm.com/>. Acesso em: 10 de Maio de 2023. Citado na página 51.

GITHUB. 2023. Disponível em: <https://docs.github.com/>. Acesso em: 15 de Maio de 2023. Citado 2 vezes nas páginas 43 e 51.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. Citado 4 vezes nas páginas 27, 28, 32 e 34.

KERAS. *Keras*. 2023. Disponível em: <https://keras.io/>. Acesso em: 15 de Maio de 2023. Citado 3 vezes nas páginas 44, 48 e 51.

KHAZRI, A. *Faster RCNN object detection*. Towards Data Science, 2021. Disponível em: <https://towardsdatascience.com/faster-rcnn-object-detection-f865e5ed7fc4>. Acesso em: 01 de Julho de 2023. Citado na página 40.

LADAS, C. *Scrumban*. 2022. Disponível em: <https://www.agilealliance.org/scrumban/>. Acesso em: 10 de Julho de 2023. Citado na página 56.

LIN, T.-Y. et al. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. Citado 2 vezes nas páginas 41 e 42.

LINUX. *The Linux Kernel Archives*. 2023. Disponível em: <https://www.kernel.org/>. Acesso em: 10 de Maio de 2023. Citado na página 44.

LIU, W. et al. A survey of deep neural network architectures and their applications. *Neurocomputing*, v. 234, p. 11–26, 2017. Citado 2 vezes nas páginas 34 e 36.

LOPEZ, M. A.; MATTOS, D. M. Resumo de grandes volumes de dados com filtro de bloom: Uma abordagem eficiente para aprendizado profundo com redes neurais convolucionais em fluxos de rede. *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2021)*, 2021. Citado na página 36.

MARINAI, S. Introduction to document analysis and recognition. *Machine Learning in Document Analysis and Recognition*, p. 1–20, 2008. Citado na página 21.

MARINAI, S.; GORI, M.; SODA, G. Artificial neural networks for document analysis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 27, n. 1, p. 23–35, 2005. Citado na página 22.

MATPLOTLIB. *matplotlib Documentation*. 2023. Disponível em: <https://pypi.org/project/matplotlib/>. Acesso em: 10 de Maio de 2023. Citado 2 vezes nas páginas 50 e 51.

MINT. *Linux Mint*. 2023. Disponível em: <https://linuxmint.com/>. Acesso em: 15 de Maio de 2023. Citado 2 vezes nas páginas 44 e 51.

NAGY, G. Twenty years of document image analysis in pami. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 1, p. 38–62, 2000. Citado na página 21.

NUMPY. *Numpy Documentation*. 2023. Disponível em: <https://pypi.org/project/numpy/>. Acesso em: 10 de Maio de 2023. Citado na página 47.

OVERLEAF. *Overleaf Official Website*. 2023. Disponível em: <https://www.overleaf.com/>. Acesso em: 10 de Maio de 2023. Citado 2 vezes nas páginas 45 e 51.

PDF2IMAGE. *pdf2image Documentation*. 2023. Disponível em: <https://pypi.org/project/pdf2image/>. Acesso em: 10 de Maio de 2023. Citado 3 vezes nas páginas 50, 51 e 73.

PILLOW. *Pillow Documentation*. 2023. Disponível em: <https://pypi.org/project/Pillow/>. Acesso em: 10 de Maio de 2023. Citado 2 vezes nas páginas 50 e 51.

POPPLER. *Poppler Documentation*. 2023. Disponível em: <https://pypi.org/project/python-poppler/>. Acesso em: 10 de Maio de 2023. Citado na página 50.

PYTHON. *Python*. 2023. Disponível em: <https://www.python.org/>. Acesso em: 15 de Maio de 2023. Citado 3 vezes nas páginas 44, 47 e 51.

RASTOGI, A. *ResNet-50*. 2022. Disponível em: <https://blog.devgenius.io/resnet50-6b42934db431>. Acesso em: 01 de Julho de 2023. Citado na página 75.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, n. 6, p. 1137–1149, 2017. Citado 2 vezes nas páginas 39 e 40.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, pp. 210–229, 1959. Citado na página 27.

SHARMA, A. 2022. Disponível em: <https://pyimagesearch.com/2022/05/02/mean-average-precision-map-using-the-coco-evaluator/>. Acesso em: 12 de Novembro de 2023. Citado 2 vezes nas páginas 76 e 77.

SHI, S.; CUI, C.; XIAO, Y. An invoice recognition system using deep learning. *2020 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, 2020. Citado na página 23.

SOLAR, P. *Geração distribuída de energia (GD): o que é, regras, benefícios e como fazer parte*. 2022. Disponível em: <https://www.portalsolar.com.br/geracao-distribuida-de-energia.html>. Acesso em: 29 de Junho de 2023. Citado na página 66.

SUTHERLAND, J.; SCHWABER, K. *Guia do Scrum*. 2013. Disponível em: <https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>. Acesso em: 05 de Maio de 2023. Citado 3 vezes nas páginas 44, 56 e 57.

TENSORFLOW. *TensorFlow*. 2023. Disponível em: <https://www.tensorflow.org/>. Acesso em: 15 de Maio de 2023. Citado 5 vezes nas páginas 44, 47, 49, 50 e 51.

TIBCO, S. *What is a neural network?* 2021. Disponível em: <https://www.tibco.com/reference-center/what-is-a-neural-network>. Acesso em: 25 de Abril de 2023. Citado na página 33.

TRASK, A. W. *Grokking Deep Learning*. [S.l.]: Manning Publications, 2019. Citado 2 vezes nas páginas 28 e 47.

TRELLO. *Trello Official Website*. 2023. Disponível em: <https://trello.com/>. Acesso em: 15 de Maio de 2023. Citado 2 vezes nas páginas 44 e 51.

VOULODIMOS, A. et al. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, p. 1–13, 2018. Citado na página 28.

YAO, X. et al. Invoice detection and recognition system based on deep learning. *Security and Communication Networks*, v. 2022, p. 1–10, 2022. Citado 2 vezes nas páginas 21 e 22.